An empirical study of the (L_0, L_1)-smoothness condition in deep learning

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

The (L_0, L_1)-smoothness condition was introduced by Zhang-He-Sra-Jadbabai in 2020, who both proved convergence bounds under this assumption and provided empirical evidence it is satisfied in deep learning. Since then, many groups have proven convergence guarantees for functions which satisfy this condition, motivated by the expectation that loss functions arising in deep learning satisfy it. In this paper we provide further empirical study of this condition in the setting of feedforward neural networks of depth at least 2, with $L2$ or cross entropy loss. The results suggest that the (L_0, L_1)-smoothness condition is not satisfied in this setting.

1 Introduction

The two properties most prized in the field of optimization are convexity and smoothness Many desirable convergence results have been proved for functions satisfying both of these conditions.

Meanwhile, in the past decade, deep learning has astonished experts and laypeople alike with its power and versatility. Understanding how deep neural networks work is of great interest, and it would be very desirable to apply the tools of optimization theory to understand the convergence properties of deep neural networks.

At present this is usually not possible. This is because many powerful results in optimization theory assume the function being optimized is smooth and convex, while it is well established that the loss function of deep neural networks is neither smooth nor convex. Therefore, it would be useful to develop weaker notions of smoothness and convexity, that both hold for deep neural networks and for which good convergence guarantees can be proved.

This motivated Zhang-He-Sra-Jadbabai Zhang et al. [2020b] to introduce the (L_0, L_1)-smoothness condition in 2020. In this promising approach, they succeeded both to provide empirical evidence that this form of smoothness is satisfied by deep neural networks, as well as later prove good convergence bounds for functions that satisfy this condition.

Motivated by the expectation that deep neural networks satisfy this condition, expressed for example in Chen et al. [2023] and Crawshaw et al. [2022], a considerable amount of work has been done proving convergence bounds under the (L_0, L_1) condition, for several different optimizers. Groups including Wang-Zhang-Zhang-Meng-Ma-Chen in Wang et al. [2022] and Faw-Rout-Caramanis-Shakkottai in Faw et al. [2023] have extended the work of Zhang-He-Sra-Jadbabai, and succeeded in proving good convergence guarantees for functions that satisfy this condition.

However there has been limited study of whether deep neural networks satisfy the (L_0, L_1)-smoothness condition. In 2022 Patel-Zhang-Tian gave two examples of neural network loss functions L that are not (L_0, L_1)-smooth, in Patel et al. [2022]. Their work showed it is not the case that all loss functions arising in deep learning satisfy the (L_0, L_1) condition, but left open the question of whether the networks they found were exceptional, or reflected a general property of loss functions arising in deep learning.

In this paper we undertake further empirical study of (L_0, L_1)-smoothness in the setting of deep feedforward neural networks. Our experiments suggest that for deep feedforward neural networks, the failure of the (L_0, L_1)-smoothness condition that Patel-Zhang-Tian observed in their examples is not the exception but the rule.

Our **first contribution** is to compute the magnitudes of the gradient and hessian of L along a fixed line $\mathcal{R}$, with L the loss function of a feedforward neural network. Under either $L2$ loss or cross entropy loss, we observe a quadratic relationship, suggesting a failure of the (L_0, L_1)-smoothness condition.

Our **second contribution** is again to look at the loss function L of a feedforward neural network with $L2$ loss. We sample a number of randomly generated initializations, and compute the the magnitudes of the gradient and hessian of L along a radial line segment through those lines. We observe a range of behaviors, some consistent with the (L_0, L_1)-smoothness condition, others suggesting a failure of the (L_0, L_1)-smoothness condition near initialization.

In the next section, we discuss related work. In Section 3 we describe several empirical results. In Appendix A we provide background and notation, and in Appendix B we provide supporting materials.

2 Related work

To date, none of the weaker smoothness conditions discussed in Section A.1 have been successfully verified for deep neural networks. In response, in 2020 Zhang-He-Sra-Jadbabai proposed a novel relaxation of the classical notion of smoothness, along with reasons one may hope that this condition is satisfied by deep neural networks, in Zhang et al. [2020b] . Their innovation is to allow the local smoothness constant to increase with the gradient norm.

Definition 1 (see Zhang et al. [2020b])**.** Given two real numbers L_0 and L_1, a twice differentiable function $L : \mathbb{R}^d \to \mathbb{R}$ is (L_0, L_1)-smooth if the Hessian of L, denoted $\mathbf{H}L$, satisfies the inequality

$$\|\mathbf{H}L(\rho)\| \leq L_0 + L_1\|\nabla L(\rho)\| \tag{1}$$

for all $\rho \in \mathbb{R}^d$, where the norm on the left is taken to be the operator norm of the matrix and the norm on the right is the L^2 norm of the vector.

Remark 1. All matrix norms are equivalent up to constants, that is, for any two matrix norms $||\cdot||_A$ and $||\cdot||_B$, there exist constants q and r such that

$$q||M||_A \leq ||M||_B \leq r||M||_A,$$

for all matrices M. Therefore, one can equivalently use any matrix norm on the left hand side of Equation 1, up to rescaling the constants L_0 and L_1. In this paper, we will take the Frobenius norm on the left side and the L^2 norm on the right side of Equation 1.

After introducing this condition, Zhang-He-Sra-Jadbabai went on to prove upper and lower bounds convergence for functions L that satisfy the (L_0, L_1)-smoothness condition for some choice of L_0 and L_1, for several different optimizers. They consider gradient descent, clipped gradient descent, as well as stochastic versions of both.

They go on to provide empirical evidence that this condition is satisfied by deep neural networks. In experiments on a variety of architectures and tasks, including image recognition and language generation, they consider the (L_0, L_1)-smoothness condition in the regions of the loss landscape traversed during training and find evidence that it is satisfied.

Because a condition which both allows us to prove convergence results and is satisfied by deep neural networks has been long desired, this work is very appealing and a number of works have expanded on this seminal work.

Following Zhang et al. [2020b], several groups have gone on to provide analyses of the convergence properties of additional optimizers, and under a wider range of assumptions, for functions that satisfy the (L_0, L_1)-smoothness condition. In Li et al. [2023a], and Li et al. [2023b], Li-Qian-Tian-Rakhlin-Jadbabaie proved convergence results for additional optimizers, such as Adam, and under a wider range of assumptions, including a generalization of the (L_0, L_1)-smoothness condition. In related work, Faw-Rout-Caramanis-Shakkottai developed new techniques allowing them to derive convergence bounds for SGD without assuming uniform bounds on the noise support in Faw et al. [2023].

There has been less attention on studying this new condition in the specific context of deep neural networks. Since the (L_0, L_1)-smoothness condition was introduced in Zhang et al. [2020b], fewer groups have analyzed the motivating hope that loss functions arising from deep neural networks satisfy the (L_0, L_1)-smoothness condition.

One group that did consider this question is Patel-Zhang-Tian, who gave a theoretical analysis of the geometry of several loss functions and in doing so produced two examples of loss functions L that do not satisfy the (L_0, L_1)-smoothness condition in Patel et al. [2022]. The first is a very simple feedforward network with three linear layers and a single nonlinear layer, learning the simple dataset input 0 output 0 with probability 1/2 and input 1 output 1 with probability 1/2.

The second is a 1-dimensional linear recurrent neural network, learning a similar dataset. In both cases, they give a complete mathematical analysis of the smoothness of the resulting loss function and conclude not only that the loss functions are not m-smooth for any m, but also do not satisfy the (L_0, L_1)-smoothness condition for any choice of L_0 and L_1.

In this paper, we find that these examples are not isolated. Our experiments suggest that they are in fact representative of the general case.

3 Experiments

3.1 Fixed line

In the first experiment, we fix a line motivated by the work of Patel et al. [2022], compute the norms of the Hessian and gradient along the line, and observe the relative growth rates. We made the following computations in Mathematica, any other programming platform that can compute neural networks can be used as well.

We begin with the case of L2 loss. We begin by initializing a feedforward neural network of layer widths (1, 4, 7, 1). We take the activation function σ to be $\tanh$. Next, we choose 20 data points, with x and y drawn uniformly at random from the interval $[-1, 1]$. Having made these choices, we can compute the corresponding loss function.

In the case of L2 loss, the direction in the loss landscape we sample from is the region near the line $\mathcal{R}$ specified in Appendix B. We draw points at random from a tubular neighborhood of this line, by taking points on the line for values of t between 100 and 700, incrementing by 3 each time, and adding noise drawn uniformly at random to each point, with width $\epsilon(t) = 1/t^2$. Finally, we compute the norms of the Hessian and gradient and the sampled points.

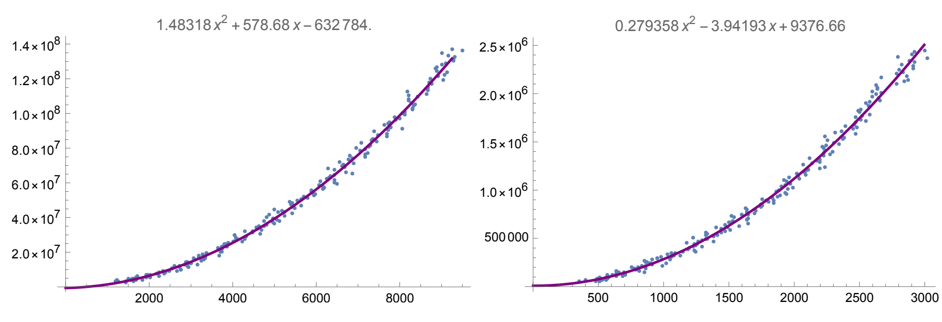

Figure 1: Left: our calculations for the example with L2 loss are shown in a scatter plot of the pairs (norm of gradient, norm of Hessian). Right: the same, for the example with cross-entropy loss.

For the case of cross-entropy loss, we proceed similarly. We begin by initializing a feedforward neural network of layer widths (1, 3, 3, 2). We take the activation function σ to be $\tanh$. Next, we choose 10 data points, at random, with x drawn uniformly at random from the interval $[-1, 1]$ and y assigned to be $(1, 0)$ for the first 8 points, and $(0, 1)$ for the remaining. (The width of the tubular neighborhood is chosen based on the proportions of each label, so fixing the proportions is easiest.) Having made these choices, we can compute the corresponding loss function.

In the case of cross-entropy loss, the direction in the loss landscape we sample from is the region near the line $\mathcal{R}$ described in the proof. We draw points at random from a tubular neighborhood of this line, by taking points on the line for values of t between 100 and 700, incrementing by 3 each time, and adding noise drawn uniformly at random to each point, with width $\epsilon(t) = 20/t^2$. Finally, we compute the norms of the Hessian and gradient and the sampled points.

The resulting plots are shown in Figure 1, together with the degree 2 polynomial of best fit for each.

At first glance, these plots look different than the ones shown in Zhang et al. [2020b]. Here we note that the two sets of plots are consistent, as the scatter plots in Figure 1 are plotted directly, while the scatter plots shown in the body of Zhang et al. [2020b] are shown on a log-log plot.

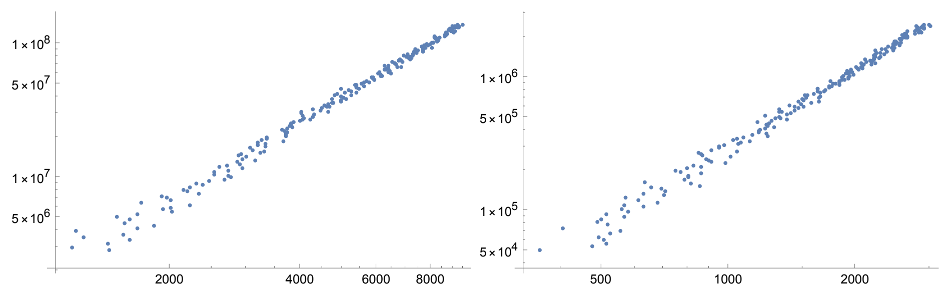

Figure 2: Left: our calculations for the example with L2 loss are shown in a log-log scatter plot of the pairs (norm of gradient, norm of Hessian). Right: the same, for the example with cross-entropy loss.

Note that a polynomial of any degree shown on a log-log plot will look linear. In Figure 2, we redisplay the information on Figure 1 on log-log plots. In this format, the figures look similar to the figures in Zhang et al. [2020b].

3.2 Random segments

In the previous experiment, we studied the relationship between the magnitudes of the gradient and the Hessian of L along a fixed line. One might ask what plots of the magnitude of the gradient against the magnitude of the Hessian look like in regions encountered when training a neural network. One place to look is at initialization.

In this experiment, we again compare the gradient and the Hessian of L, this time near points initialized according to Kaiming initialization, along random line segments through those points.

We used the same architectures as in Subsection 3.1. Namely, we consider a feedforward network with layer widths [1, 4, 7, 1], L2 loss, and using tanh for the activation function. We then generated 20 data points at random to define the loss function L. We record the randomly chosen data in Appendix B.

We then generated 15 random initialization parameters $p_1, ..., p_{15}$ using the Kaiming initialization procedure. We chose a radial line segment S_i through each parameter p_i, and computed the magnitudes of the gradient and hessian at 50 equidistributed points along each S_i. In 6 cases the relationship appeared approximately linear. In 5 cases the relationship appeared superlinear. In 4 cases, other nonlinear graphs were observed.

In the following, we display some of these graphs, numbered in the order of appearance, not the order they were generated in. In Appendix B we record the endpoints of the line segments S_i, which we call $start_i$ and end_i.

At the randomly initialized point p_1 on the left in Figure 3, the resulting graph appears approximately linear to begin, then approximately like an upward-facing semicircle. At the randomly initialized point p_2 on the right, the resulting graph appears approximately linear for the first quarter, then approximately linear to begin, then approximately linear but with a steeper slope, then approximately linear with an even steeper slope in the last stretch.

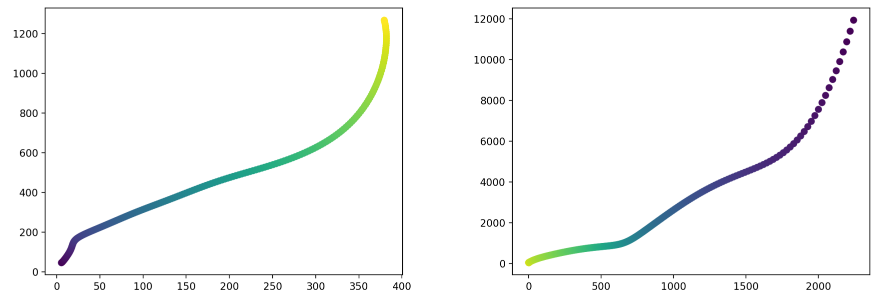

Figure 3: We show $|\nabla L|$ along the x-axis and the $|hess(L)|$ along the y-axis.

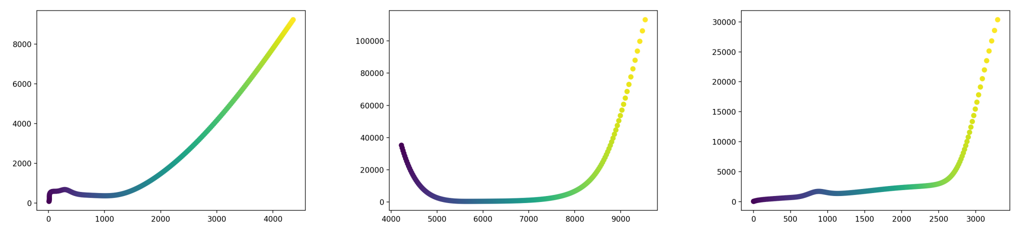

Figure 4: We show $|\nabla L|$ along the x-axis and the $|hess(L)|$ along the y-axis.

At the randomly initialized point p_3 on the left in Figure 4, the resulting graph appears approximately quadratic. At the randomly initialized point p_4 in the center, the resulting graph looks U-shaped. Finally at the randomly initialized point p_5 on the right, the resulting graph grows quickly at the end.

The last two calculations show examples of initializations near which, in the radial direction, the magnitude of the hessian does appear to be bounded by a linear function of the magnitude of the gradient, as seen in Figure 5.

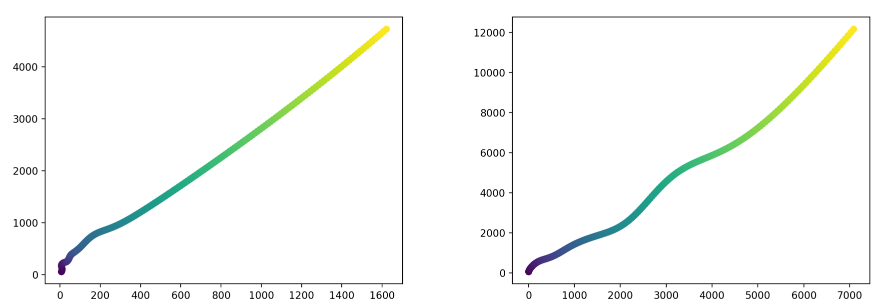

Figure 5: We show $|\nabla L|$ along the x-axis and the $|hess(L)|$ along the y-axis.

We note that empirical measurements cannot prove if the loss function L satisfies an (L_0, L_1)-smoothness condition or not. Indeed, on any compact set such as this spherical shell we are studying in this experiment the loss function L not only will satisfy a (L_0, L_1)-smoothness condition for some choice of L_0 and L_1 but will satisfy $m-$smoothness for some choice of m.

That being said, on a compact region one may ask if the hessian of L appears bounded by a linear function of the gradient of L. Near some of the random initializations we generated the hessian of L does appear bounded by a linear function of the gradient of L, such as the plots in Figure 5.

However, near other random initializations we generated, the answer appears to be no, such as in the plots in Figure 4. This provides empirical evidence that in the region near random initializations, the loss function does not satisfy a (L_0, L_1)-smoothness condition.

4 Conclusion

In this paper, we made an empirical study of the (L_0, L_1)-smoothness condition in the setting of feedforward networks, with either $L2$ or cross-entropy loss. The results suggest that the (L_0, L_1)-smoothness condition is not in general satisfied.

Thus the convergence guarantees that have been proved for (L_0, L_1)-smoothness might not be directly applicable to the loss functions arising from deep feedforward networks. Though we take a different conclusion from Zhang-He-Sra-Jadbabai, our results are not in contradiction with the empirical studies in the original paper by Zhang et al. [2020a]. Note that in their work, they compute the magnitude of the gradient of L and the magnitude of the hessian of L along gradient trajectories, but do not compute those quantities in transverse directions.

In contrast, we compute the magnitude of the gradient of L and the magnitude of the hessian of L in radial directions. So it is not contradictory that we observe different relationships. We note that the geometry near a gradient trajectory, in directions transverse to the trajectory, are relevant in theoretical bounds on convergence. So the additional empirical study here provides useful further information.

In recent work, Li-Quian-Tian-Rakhlin-Jadbabaie Li et al. [2023a] introduce a class of conditions generalizing the (L_0, L_1)-smoothness condition, which they call ℓ-smoothness conditions, for any function ℓ. The (L_0, L_1)-smoothness condition is recovered in the special case that ℓ is an affine linear function.

The rates of growth of the magnitude of the hessian as a function of the magnitude of the gradients we observe suggest that not only does the loss function L of a deep neural network not satisfy the (L_0, L_1)-smoothness condition, that is ℓ-smoothness for a linear function, but that L also does not satisfy ℓ-smoothness for any subquadratic function ℓ. This is worth noting because in Li et al. [2023a], convergence guarantees proven in cases when ℓ is subquadratic, and in the thorough analysis given, examples are also provided illustrating that similar guarantees are not possible in cases when ℓ is quadratic or superquadratic. Our work shows that the loss functions of deep feedforward networks lie in this more challenging setting.

Our work suggests that in order to develop similar convergence arguments that can be applied directly to the loss functions arising in deep learning, different generalizations of the (L_0, L_1)-smoothness condition may be needed.

One could also study (L_0, L_1)-smoothness with the approach used to study weak convexity in the setting of deep networks by Liu-Zhu-Belkin Liu et al. [2022]. It may be that while the (L_0, L_1)-smoothness condition does not hold uniformly over the loss landscapes of deep feedforward networks, that it is possible to identify regions of the loss landscape on which (L_0, L_1)-smoothness holds.

This work is an invitation to interesting directions for future work. The study of (L_0, L_1)-smoothness is an exciting nexus where new techniques in optimization are being developed with inspiration from the geometries that arise in deep learning.

5 Broader Impacts

This work focuses on the mathematical understanding of a technical aspect of deep learning. While this may feel removed from the machine learning systems that are beginning to be integrated into our daily lives, advances in this and similar papers are expected to improve the performance of machine learning systems over time. Therefore this work may have greater societal impact than is initially apparent.

As the authors of this work, we have a responsibility to make our technical advancements understandable to the broadest range of people, to promote the beneficial uses of these technologies, and to work to mitigate the risks of these technologies.

References

A. Agarwal, S. Negahban, and M. J. Wainwright. Fast global convergence of gradient methods for high-dimensional statistical recovery. *The Annals of Statistics*, 40(5):2452 – 2482, 2012. doi: 10.1214/12-AOS1032. URL https://doi.org/10.1214/12-AOS1032.

Z. Chen, Y. Zhou, Y. Liang, and Z. Lu. Generalized-smooth nonconvex optimization is as efficient as smooth nonconvex optimization. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 5396–5427. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/chen23ar.html.

M. Crawshaw, M. Liu, F. Orabona, W. Zhang, and Z. Zhuang. Robustness to unbounded smoothness of generalized signsgd. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 9955–9968. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/40924475a9bf768bdac3725e67745283-Paper-Conference.pdf.

M. Faw, L. Rout, C. Caramanis, and S. Shakkottai. Beyond uniform smoothness: A stopped analysis of adaptive sgd. In G. Neu and L. Rosasco, editors, *Proceedings of Thirty Sixth Conference on Learning Theory*, volume 195 of *Proceedings of Machine Learning Research*, pages 89–160. PMLR, 12–15 Jul 2023. URL https://proceedings.mlr.press/v195/faw23a.html.

E. Hazan, K. Levy, and S. Shalev-Shwartz. Beyond convexity: Stochastic quasi-convex optimization. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/934815ad542a4a7c5e8a2dfa04fea9f5-Paper.pdf.

H. Li, J. Qian, Y. Tian, A. Rakhlin, and A. Jadbabaie. Convex and non-convex optimization under generalized smoothness. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023a. URL https://openreview.net/forum?id=8aunGrXdkl.

H. Li, A. Rakhlin, and A. Jadbabaie. Convergence of adam under relaxed assumptions. In A. Oh, T. Neumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 52166–52196. Curran Associates, Inc., 2023b. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/a3cc50126338b175e56bb3cad134db0b-Paper-Conference.pdf.

C. Liu, L. Zhu, and M. Belkin. Loss landscapes and optimization in over-parameterized non-linear systems and neural networks. *Applied and Computational Harmonic Analysis*, 59:85–116, 2022. ISSN 1063-5203. doi: https://doi.org/10.1016/j.acha.2021.12.009. URL https://www.sciencedirect.com/science/article/pii/S106352032100110X. Special Issue on Harmonic Analysis and Machine Learning.

H. Lu, R. M. Freund, and Y. Nesterov. Relatively smooth convex optimization by first-order methods, and applications. *SIAM Journal on Optimization*, 28(1):333–354, 2018. doi: 10.1137/16M1099546. URL https://doi.org/10.1137/16M1099546.

V. Patel, S. Zhang, and B. Tian. Global convergence and stability of stochastic gradient descent. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 36014–36025. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/ea05e4fc0299c27648c9985266abad47-Paper-Conference.pdf.

B. Wang, Y. Zhang, H. Zhang, Q. Meng, Z.-M. Ma, T.-Y. Liu, and W. Chen. Provable adaptivity in Adam, 2022. arXiv:2208.09900.

B. Zhang, J. Jin, C. Fang, and L. Wang. Improved analysis of clipping algorithms for non-convex optimization. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 15511–15521. Curran Associates, Inc., 2020a. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/b282d1735283e8eea45bce393cefe265-Paper.pdf.

J. Zhang, T. He, S. Sra, and A. Jadbabaie. Why gradient clipping accelerates training: A theoretical justification for adaptivity. In *International Conference on Learning Representations*, 2020b. URL https://openreview.net/forum?id=BJgnXpVYwS.

A Background and notation

A.1 Weak smoothness

Definition 2. Let $m > 0$. A function $L\colon \mathbb{R}^d \to \mathbb{R}$ is m-smooth if for every $\alpha, \beta \in \mathbb{R}^d$, we have

$$\|\nabla L(\beta) - \nabla L(\alpha)\| \leq m\|\alpha - \beta\|.$$

When L is twice differentiable, this can alternatively be stated as the condition that the magnitude of the second derivative of L is uniformly bounded by m.

Because many functions that one would like to minimize are not m-smooth for any m, researchers have long proposed weaker notions of smoothness and tried to prove convergence results under such alternate definitions of smoothness, in an effort to expand the range of functions that we can confidently optimize. We begin by noting a few of the popular definitions.

Definition 3 (see Hazan et al. [2015])**.** Let $\tau, \epsilon > 0$, $\gamma \in \mathbb{R}^d$. A function $L\colon \mathbb{R}^d \to \mathbb{R}$ is (τ, ϵ, γ)-locally-smooth if for every $\alpha, \beta \in \mathbb{R}^d$ such that $\|\alpha - \gamma\| \leq \epsilon$ and $\|\beta - \gamma\| \leq \epsilon$, we have

$$|L(\beta) - L(\alpha) - \langle \nabla L(\beta), \alpha - \beta\rangle| \leq \frac{\tau}{2}\|\alpha - \beta\|^2.$$

Definition 4 (see Agarwal et al. [2012])**.** Let $\tau, \epsilon > 0$ and let $R\colon \mathbb{R}^d \to \mathbb{R}^+$ be a regularizer. A function $L\colon \mathbb{R}^d \to \mathbb{R}$ satisfies restricted smoothness with respect to R with parameters (τ, ϵ) if for every $\alpha, \beta \in \mathbb{R}^d$, we have

$$|L(\beta) - L(\alpha) - \langle \nabla L(\beta), \alpha - \beta\rangle| \leq \frac{\tau}{2}\|\alpha - \beta\|^2 + \epsilon R^2(\alpha - \beta).$$

Definition 5 (see Lu et al. [2018])**.** Let $h\colon \mathbb{R}^d \to \mathbb{R}$ be a differentiable convex "reference function", and m a positive real number. A function $L\colon \mathbb{R}^d \to \mathbb{R}$ is m-smooth relative to h if for any $\alpha, \beta \in \mathbb{R}^d$ we have

$$L(\beta) \leq L(\alpha) + \langle \nabla L(\alpha), \beta - \alpha\rangle + m(h(\beta) - h(\alpha) - \langle \nabla h(\alpha), \beta - \alpha\rangle).$$

A.2 Weak convexity

In the Introduction, we noted that there is interest both in weaker notions of smoothness and weaker notions of convexity, and in determining whether the loss functions arising from deep neural networks satisfy any of them.

While in this paper we will focus on smoothness conditions, here we point to work considering these questions for convexity conditions.

In Liu et al. [2022], Liu-Zhu-Belkin considered the PL* condition, a condition related to the classical Polyak-Lojasiewicz condition. They showed that for the loss function of neural networks, if the network satisfies some conditions including that they are sufficiently wide, one can construct many balls within the parameter space $\mathbb{R}^d$ on which the PL* condition holds.

While it is known that the loss functions arising in deep learning are not convex, this result shows that there is a weaker type of convexity that is satisfied in some regions for some neural networks.

At this time, we do not know of analogous results for relaxed smoothness conditions. In this work, we will give a negative result, for most neural networks, it is not the case that the (L_0, L_1)-smoothness condition holds over the entire parameter space $\mathbb{R}^d$. Perhaps an analog of Liu-Zhu-Belkin's result for the PL* convexity condition is possible - perhaps there are regions in the parameter space $\mathbb{R}^d$ on which the (L_0, L_1)-smoothness condition holds. We leave that question to future work.

A.3 Notation

A.3.1 Fully connected feedforward neural networks

To define a fully connected feedforward neural network, we begin by specifying the number of layers ℓ of the network, and the widths $d_{\text{in}}, c_1, \ldots, c_\ell, d_{\text{out}}$ of the layers, ordered from "earliest" to "latest". For each adjacent pair of layers $i, i+1$, we will have the space of affine linear maps from $\mathbb{R}^{c_i}$ to $\mathbb{R}^{c_{i+1}}$. Such a map is given by the choice of a $c_i \times c_{i+1}$ matrix we will call M^i, and a vector in $\mathbb{R}^{c_{i+1}}$ we will call b^i. The entries of M^i we call weights, the entries of b^i we call biases, and the choice of weights and biases for all the layers we call the choice of a parameter vector $\rho \in \mathbb{R}^p$.

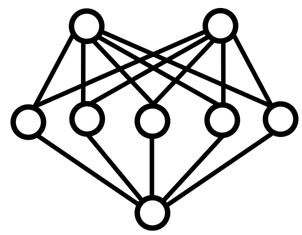

Figure 6: This is a diagram representing a feedforward neural network with one hidden layer, with the input width $d_{\text{in}} = 2$, the width of the hidden layer $k_1 = 5$, and output width $d_{\text{out}} = 1$.

Next, we choose an **activation function**

$$\sigma \colon \mathbb{R} \to \mathbb{R}. \tag{2}$$

In this paper, we assume that σ is twice differentiable.

Given the choices above of an architecture and σ, this neural network provides a way to input a set ρ of weights and biases for the network and output a function $f_\rho : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}}$.

Given a vector

$$\rho = (\mathbf{w}, \mathbf{b}) \in \mathbb{R}^p \tag{3}$$

in the parameter space, we define the function

$$f_\rho = f_{\mathbf{w},\mathbf{b}} \colon \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}} \tag{4}$$

by composing the following sequence of maps specified by the neural network and the choice of the weights and biases in all the layers $\mathbf{w}, \mathbf{b}$:

$$\mathbb{R}^{d_{\text{in}}} \xrightarrow{M^1\mathbf{x}+\mathbf{b}^1} \mathbb{R}^{c_1} \xrightarrow{\sigma} \mathbb{R}^{c_1} \xrightarrow{M^2\mathbf{x}+\mathbf{b}^2} \cdots \xrightarrow{M^\ell\mathbf{x}+\mathbf{b}^\ell} \mathbb{R}^{c_\ell} \xrightarrow{\sigma} \mathbb{R}^{c_\ell} \xrightarrow{M^{\ell+1}\mathbf{x}+\mathbf{b}^{\ell+1}} \mathbb{R}^{d_{\text{out}}}. \tag{5}$$

In this construction, the arrow $\mathbb{R}^{k_i} \xrightarrow{\sigma} \mathbb{R}^{k_i}$ indicates that we apply σ componentwise.

Example 1. Consider a fully connected feedforward graph with layers of widths 1, 3, 1 and $\sigma = (u \mapsto u^2 + 1)$. The corresponding function space consists of those functions of the form

$$f_\alpha \colon x \mapsto w^2_{11}\big((w^1_{11}x + b^1_1)^2 + 1\big) + w^2_{12}\big((w^1_{21}x + b^1_2)^2 + 1\big) + w^2_{13}\big((w^1_{31}x + b^1_3)^2 + 1\big) + b^2_1. \tag{6}$$

In our calculations we will find it useful to have the following notation for the stages of the neural network. We define recursively

$$f^1_\rho(\mathbf{x}) = M^1\mathbf{x} + \mathbf{b}^1 \tag{7}$$

$$f^i_\rho(\mathbf{x}) = M^i\sigma(f^{i-1}_\rho(\mathbf{x})) + \mathbf{b}^i, \tag{8}$$

so that the previously defined function $f_\rho(\mathbf{x})$ equals $f^{\ell+1}_\rho(\mathbf{x})$.

A.3.2 The loss function L

In deep learning, one starts with a data set, chooses an architecture for a neural network, and then wishes to find a parameter vector, in other words a set of weights and biases for the network, such that with that choice, the function expressed by the network predicts well on similar data.

To find such a parameter vector, a key step is to define a loss function

$$L\colon \mathbb{R}^d \to \mathbb{R} \tag{9}$$

from the set of all parameters to the real numbers. This function L is constructed in such a way that parameter vectors ρ on which the loss function achieves a low value are good choices for the network. In today's implementations, a gradient descent based algorithm is used to find such ρ that minimize L.

In this paper, we consider two ways of constructing L, and we define each in this section. In both cases, we fix

- a neural network,
- a choice of activation function σ,
- and a data set
$$D := \big\{(\mathbf{x}_i, \mathbf{y}_i)\big\}_{i\in\{1,\ldots,n\}} \subset \mathbb{R}^{d_{\text{in}}} \times \mathbb{R}^{d_{\text{out}}}. \tag{10}$$

Definition 6. The $L2$ loss is defined by:

$$L(\rho) := \sum_{s=1}^{n} \big(f_\alpha(\mathbf{x}_s) - \mathbf{y}_s\big)^2. \tag{11}$$

Definition 7. The cross-entropy loss is defined by:

$$L(\rho) = -\sum_{m=1}^{n} \sum_{j=1}^{d_{\text{out}}} [y_m]_j \log \frac{e^{[f_\rho(x_m)]_j}}{\sum_{k=1}^{b} e^{[f_\rho(x_m)]_k}}. \tag{12}$$

where $[v]_j$ denotes the j^{th} entry of a vector v.

B Supporting material

First, the description of the line $\mathcal{R}$ appearing in Section 3.1.

We define $\mathcal{R}$ to be the image of the following linear map $\rho : \mathbb{R} \to \mathbb{R}^d$, where $\mathbb{R}^d$ is the parameter space of the neural network. Given a real number $t \in \mathbb{R}$, $\rho(t)$ is the following choice of weights and biases.

For all but the last layer, we take the weights to be zero and the biases all equal to zero.

$$M^i = 0,\ b^i = (c \quad \cdots \quad c)^T \qquad\qquad \text{if } 1 \le i \le \ell$$

In the last layer, we take $M^{\ell+1}$ to be t times a constant matrix M. We choose M carefully, depending on the loss. Finally, we take all the biases equal to 0.

$$M = \begin{pmatrix} m_{11} & \dots & m_{1k} \\ \vdots & \ddots & \vdots \\ m_{b1} & \cdots & m_{bk} \end{pmatrix} \tag{13}$$

$$M^{\ell+1} = tM, \quad b^{\ell+1} = \vec{0} \tag{14}$$

Now, the randomly chosen data x and y for the experiment in Section 3.2.

$$\vec{x} = \begin{pmatrix} -0.51 & -0.32 & 0.45 & -0.42 & -0.51 & -0.30 & 0.50 & -0.98 & -0.22 & -0.51 \\ 0.40 & 0.79 & 0.20 & 0.96 & 0.90 & -0.04 & 0.60 & -0.27 & 0.01 & 0.41 \end{pmatrix}$$

$$\vec{y} = \begin{pmatrix} -0.70 & -0.86 & -0.20 & 0.88 & 0.88 & -0.33 & -0.92 & 0.06 & 0.89 & 0.21 \\ -0.35 & 0.32 & 0.27 & 0.97 & 0.86 & -0.20 & 0.49 & -0.05 & -0.75 & 0.90 \end{pmatrix}$$

$start_1 =$

$$\begin{pmatrix} 0.048 & 0.026 & -0.0063 & 0.025 & -0.014 & -0.0014 & -0.059 & 0.024 \\ -0.026 & 0.0021 & -0.022 & 0.013 & -0.012 & -0.0093 & -0.021 & 0.0052 \\ -0.026 & -0.03 & -0.022 & -0.014 & -0.011 & 0.018 & 0.00088 & -0.024 \\ -0.023 & -0.021 & 0.026 & 0.019 & -0.019 & 0.025 & 0.0017 & -0.028 \\ 0.012 & 0.014 & -0.0012 & 0.012 & -0.017 & -0.019 & 0.022 & 0.022 \\ 0.009 & 0.015 & -0.027 & 0.02 & -0.01 & -0.0078 & -0.0032 & -0.016 \\ -0.0082 & -0.0049 & -0.0077 \end{pmatrix}$$

$end_1 =$

$$\begin{pmatrix} 9.6 & 5.2 & -1.3 & 4.9 & -2.8 & -0.29 & -12 & 4.8 & -5.3 & 0.43 \\ -4.3 & 2.7 & -2.5 & -1.9 & -4.2 & 1 & -5.2 & -6 & -4.4 & -2.8 \\ -2.2 & 3.6 & 0.18 & -4.9 & -4.6 & -4.3 & 5.3 & 3.8 & -3.8 & 5.1 \\ 0.34 & -5.7 & 2.3 & 2.9 & -0.25 & 2.3 & -3.3 & -3.8 & 4.5 & 4.5 \\ 1.8 & 2.9 & -5.4 & 4 & -2.1 & -1.6 & -0.65 & -3.3 & -1.7 & -0.98 & -1.6 \end{pmatrix}$$

$start_2 =$

$$\begin{pmatrix} 0.068 & -0.059 & -0.035 & -0.034 & 0.096 & -0.055 & 0.055 & -0.02 \\ -0.03 & -0.007 & 0.036 & 0.048 & -0.027 & -0.04 & -0.029 & 0.0052 \\ -0.041 & -0.014 & -0.019 & 0.025 & 0.017 & 0.042 & 0.017 & -0.018 \\ -0.044 & -0.022 & 0.046 & 0.041 & 0.021 & 0.046 & 0.049 & -0.041 \\ 0.016 & 0.022 & 0.024 & 0.017 & 0.036 & -0.0045 & 0.0011 & 0.048 \\ -0.033 & -0.05 & 0.041 & -0.013 & -0.026 & 0.018 & 0.031 & 0.013 \\ -0.024 & 0.019 & 0.015 \end{pmatrix}$$

$end_2 =$

$$\begin{pmatrix} 14 & -12 & -7.1 & -6.8 & 19 & -11 & 11 & -4 & -6 & -1.4 \\ 7.2 & 9.6 & -5.4 & -8 & -5.8 & 1 & -8.2 & -2.8 & -3.9 & 5 \\ 3.5 & 8.4 & 3.4 & -3.6 & -8.9 & -4.5 & 9.3 & 8.2 & 4.3 & 9.2 \\ 9.9 & -8.1 & 3.3 & 4.4 & 4.9 & 3.5 & 7.2 & -0.91 & 0.23 & 9.7 \\ -6.6 & -10 & 8.3 & -2.7 & -5.3 & 3.6 & 6.2 & 2.6 & -4.8 & 3.9 & 3 \end{pmatrix}$$

361 $start_3 =$

$$\begin{aligned}
\big(&0.085 \quad 0.052 \quad 0.4 \quad 0.021 \quad 0.24 \quad 0.23 \quad 0.29 \quad -0.064 \\
&-0.098 \quad 0.024 \quad 0.079 \quad 0.12 \quad -0.14 \quad 0.12 \quad 0.13 \quad 0.042 \\
&-0.083 \quad -0.076 \quad -0.047 \quad -0.055 \quad 0.13 \quad 0.02 \quad 0.097 \quad -0.0038 \\
&-0.19 \quad -0.015 \quad 0.088 \quad -0.048 \quad 0.089 \quad -0.16 \quad -0.13 \quad 0.013 \\
&-0.12 \quad -0.011 \quad 0.2 \quad 0.13 \quad 0.053 \quad -0.2 \quad 0.062 \quad -0.014 \\
&0.15 \quad -0.013 \quad -0.035 \quad 0.042 \quad -0.0064 \quad -0.14 \quad -0.087 \quad -0.14 \\
&-0.099 \quad 0.082 \quad -0.024\big)
\end{aligned}$$

362 $end_3 =$

$$\begin{aligned}
\big(&17 \quad 10 \quad 80 \quad 4.3 \quad 49 \quad 46 \quad 59 \quad -13 \quad -20 \quad 4.7 \\
&16 \quad 24 \quad -28 \quad 24 \quad 26 \quad 8.5 \quad -17 \quad -15 \quad -9.5 \quad -11 \\
&25 \quad 4 \quad 20 \quad -0.76 \quad -37 \quad -3 \quad 18 \quad -9.7 \quad 18 \quad -32 \\
&-27 \quad 2.7 \quad -24 \quad -2.3 \quad 40 \quad 26 \quad 11 \quad -39 \quad 12 \quad -2.9 \\
&30 \quad -2.6 \quad -6.9 \quad 8.5 \quad -1.3 \quad -28 \quad -17 \quad -28 \quad -20 \quad 17 \quad -4.8\big)
\end{aligned}$$

363 $start_4 =$

$$\begin{aligned}
\big(&-16 \quad 27 \quad 5.7 \quad 0.032 \quad 15 \quad 42 \quad -35 \quad -22 \\
&-5.9 \quad 20 \quad 19 \quad 0.22 \quad -13 \quad -13 \quad -5.4 \quad -15 \\
&21 \quad 3.8 \quad 10 \quad 17 \quad -6 \quad -21 \quad 17 \quad -11 \\
&11 \quad -15 \quad 4.4 \quad 13 \quad 6.5 \quad -15 \quad -11 \quad 0.92 \\
&-22 \quad -16 \quad 3.1 \quad -0.93 \quad 16 \quad 0.43 \quad 19 \quad -12 \\
&-16 \quad -11 \quad 15 \quad 17 \quad -13 \quad 8.3 \quad -0.28 \quad -12 \\
&11 \quad 12 \quad 13\big)
\end{aligned}$$

364 $end_4 =$

$$\begin{aligned}
\big(&-35 \quad 60 \quad 13 \quad 0.071 \quad 32 \quad 92 \quad -78 \quad -49 \quad -13 \quad 45 \\
&42 \quad 0.5 \quad -28 \quad -28 \quad -12 \quad -34 \quad 46 \quad 8.4 \quad 23 \quad 38 \\
&-13 \quad -48 \quad 37 \quad -24 \quad 25 \quad -32 \quad 9.8 \quad 28 \quad 14 \quad -34 \\
&-24 \quad 2 \quad -48 \quad -35 \quad 6.8 \quad -2.1 \quad 36 \quad 0.96 \quad 42 \quad -27 \\
&-36 \quad -25 \quad 32 \quad 38 \quad -29 \quad 18 \quad -0.62 \quad -26 \quad 24 \quad 27 \quad 28\big)
\end{aligned}$$

365 $start_5 =$

$$\begin{aligned}
\big(&0.37 \quad -0.29 \quad 0.045 \quad 0.35 \quad -0.16 \quad 0.2 \quad 0.16 \quad 0.14 \\
&-0.047 \quad -0.11 \quad 0.16 \quad -0.15 \quad 0.066 \quad 0.11 \quad -0.22 \quad -0.095 \\
&0.053 \quad 0.028 \quad 0.068 \quad 0.22 \quad -0.13 \quad -0.22 \quad -0.092 \quad 0.023 \\
&-0.097 \quad -0.11 \quad -0.22 \quad -0.19 \quad 0.17 \quad 0.014 \quad 0.18 \quad 0.18 \\
&0.011 \quad 0.015 \quad -0.13 \quad -0.17 \quad 0.22 \quad 0.051 \quad 0.17 \quad 0.14 \\
&0.13 \quad -0.071 \quad 0.22 \quad 0.054 \quad -0.084 \quad 0.036 \quad 0.037 \quad 0.035 \\
&-0.1 \quad -0.16 \quad 0.14\big)
\end{aligned}$$

366 $end_5 =$

$$\begin{aligned}\Big(&-33 \quad 5.1 \quad 39 \quad -18 \quad 22 \quad 18 \quad 16 \quad -5.3 \quad -12 \quad 18\\ &-17 \quad 7.5 \quad 13 \quad -25 \quad -11 \quad 6 \quad 3.1 \quad 7.7 \quad 25 \quad -15\\ &-25 \quad -10 \quad 2.6 \quad -11 \quad -12 \quad -25 \quad -21 \quad 19 \quad 1.6 \quad 20\\ &21 \quad 1.3 \quad 1.7 \quad -15 \quad -20 \quad 25 \quad 5.8 \quad 19 \quad 16 \quad 14\\ &-8 \quad 25 \quad 6.2 \quad -9.6 \quad 4.1 \quad 4.2 \quad 4 \quad -11 \quad -18 \quad 16\Big)\end{aligned}$$

367 $start_6 =$

$$\begin{aligned}\Big(&0.27 \quad -0.38 \quad 0.37 \quad 0.35 \quad -0.31 \quad -0.055 \quad 0.19 \quad 0.12\\ &0.031 \quad 0.16 \quad 0.1 \quad -0.00026 \quad 0.11 \quad 0.18 \quad 0.16 \quad 0.0043\\ &0.14 \quad 0.073 \quad -0.13 \quad -0.0036 \quad -0.17 \quad 0.055 \quad -0.14 \quad 0.11\\ &-0.13 \quad -0.21 \quad -0.027 \quad 0.0049 \quad -0.14 \quad 0.068 \quad 0.12 \quad -0.12\\ &-0.093 \quad -0.083 \quad -0.086 \quad 0.11 \quad -0.2 \quad 0.029 \quad 0.2 \quad 0.22\\ &-0.12 \quad -0.2 \quad -0.093 \quad -0.074 \quad 0.0041 \quad -0.094 \quad -0.017 \quad 0.016\\ &-0.032 \quad -0.12 \quad -0.053\Big)\end{aligned}$$

368 $end_6 =$

$$\begin{aligned}\Big(&30 \quad -42 \quad 41 \quad 39 \quad -35 \quad -6.1 \quad 21 \quad 14 \quad 3.5 \quad 18\\ &11 \quad -0.029 \quad 12 \quad 20 \quad 18 \quad 0.48 \quad 16 \quad 8.2 \quad -14 \quad -0.4\\ &-19 \quad 6.1 \quad -16 \quad 12 \quad -15 \quad -24 \quad -3 \quad 0.55 \quad -16 \quad 7.7\\ &14 \quad -13 \quad -10 \quad -9.3 \quad -9.7 \quad 12 \quad -22 \quad 3.3 \quad 22 \quad 25\\ &-14 \quad -22 \quad -10 \quad -8.3 \quad 0.46 \quad -11 \quad -1.9 \quad 1.8 \quad -3.6 \quad -13 \quad -5.9\Big)\end{aligned}$$

369 $start_7 =$

$$\begin{aligned}\Big(&0.34 \quad 0.09 \quad -0.37 \quad -0.33 \quad 0.21 \quad -0.21 \quad 0.17 \quad -0.38\\ &0.13 \quad -0.1 \quad -0.021 \quad -0.099 \quad -0.12 \quad -0.16 \quad 0.14 \quad 0.15\\ &0.11 \quad -0.0072 \quad 0.073 \quad -0.12 \quad -0.039 \quad 0.18 \quad 0.18 \quad -0.15\\ &-0.16 \quad -0.18 \quad 0.12 \quad -0.18 \quad -0.016 \quad -0.049 \quad 0.024 \quad -0.01\\ &0.044 \quad 0.13 \quad 0.12 \quad -0.073 \quad -0.12 \quad -0.077 \quad -0.19 \quad -0.11\\ &0.15 \quad -0.023 \quad -0.12 \quad -0.12 \quad 0.089 \quad 0.15 \quad 0.038 \quad -0.061\\ &-0.096 \quad -0.0099 \quad 0.082\Big)\end{aligned}$$

370 $end_7 =$

$$\begin{aligned}\Big(&33 \quad 8.9 \quad -37 \quad -32 \quad 21 \quad -21 \quad 17 \quad -38 \quad 13 \quad -10\\ &-2 \quad -9.7 \quad -12 \quad -15 \quad 14 \quad 15 \quad 11 \quad -0.71 \quad 7.2 \quad -11\\ &-3.9 \quad 18 \quad 18 \quad -15 \quad -16 \quad -18 \quad 12 \quad -18 \quad -1.6 \quad -4.8\\ &2.3 \quad -1 \quad 4.4 \quad 12 \quad 11 \quad -7.2 \quad -12 \quad -7.6 \quad -19 \quad -11\\ &15 \quad -2.2 \quad -11 \quad -12 \quad 8.7 \quad 14 \quad 3.8 \quad -6 \quad -9.5 \quad -0.98 \quad 8.1\Big)\end{aligned}$$

The random vector

$$w_2 = \Big(-0.87 \quad -0.67 \quad 0.5 \quad -0.29 \quad -0.93 \quad 0.83 \quad -0.004 \quad 0.64$$
$$0.5 \quad 0.46 \quad -0.46 \quad 0.47 \quad 0.046 \quad -0.29 \quad -0.023 \quad 0.38$$
$$-0.22 \quad -0.27 \quad 0.41 \quad -0.1 \quad 0.032 \quad -0.14 \quad -0.28 \quad -0.31$$
$$0.23 \quad 0.18 \quad -0.12 \quad -0.066 \quad 0.2 \quad 0.058 \quad -0.36 \quad 0.37$$
$$0.31 \quad 0.22 \quad -0.17 \quad -0.27 \quad -0.11 \quad -0.34 \quad 0.095 \quad 0.34$$
$$-0.31 \quad -0.26 \quad 0.18 \quad 0.24 \quad -0.29 \quad 0.23 \quad 0.022 \quad -0.012$$
$$-0.28 \quad 0.28 \quad 0.36\Big)$$

The random vector

$$w_4 = \Big(-0.72 \quad 0.74 \quad -0.86 \quad -0.19 \quad -0.15 \quad -0.094 \quad -0.23 \quad 0.41$$
$$-0.32 \quad 0.35 \quad -0.095 \quad 0.46 \quad 0.39 \quad 0.1 \quad 0.037 \quad -0.38$$
$$-0.3 \quad -0.39 \quad -0.44 \quad 0.1 \quad -0.21 \quad 0.084 \quad -0.5 \quad 0.12$$
$$0.31 \quad 0.22 \quad -0.41 \quad -0.29 \quad 0.057 \quad -0.39 \quad -0.048 \quad 0.091$$
$$-0.094 \quad -0.079 \quad -0.44 \quad 0.31 \quad 0.27 \quad 0.16 \quad 0.35 \quad 0.48$$
$$-0.22 \quad 0.23 \quad -0.12 \quad 0.3 \quad -0.078 \quad -0.31 \quad -0.13 \quad 0.31$$
$$0.011 \quad -0.2 \quad 0.34\Big)$$

The random vector

$$w_6 = \Big(0.64 \quad -0.26 \quad -0.76 \quad 0.37 \quad 0.45 \quad 0.82 \quad 0.6 \quad 0.2$$
$$-0.34 \quad -0.4 \quad -0.23 \quad -0.2 \quad -0.08 \quad -0.099 \quad 0.017 \quad 0.35$$
$$-0.36 \quad -0.3 \quad 0.18 \quad -0.13 \quad -0.062 \quad 0.33 \quad 0.37 \quad -0.37$$
$$0.23 \quad -0.13 \quad -0.16 \quad -0.23 \quad 0.37 \quad 0.31 \quad 0.0082 \quad -0.047$$
$$0.37 \quad -0.18 \quad -0.32 \quad 0.037 \quad -0.25 \quad 0.44 \quad 0.088 \quad 0.2$$
$$-0.048 \quad 0.17 \quad 0.44 \quad 0.12 \quad -0.11 \quad 0.23 \quad 0.11 \quad 0.22$$
$$-0.063 \quad 0.14 \quad 0.12\Big)$$