OPT2024: 16th Annual Workshop on Optimization for Machine Learning

A theoretical study of the (L_0, L_1)-smoothness condition in deep learning

Y. Cooper

Abstract

We study the (L_0, L_1)-smoothness condition introduced by Zhang-He-Sra-Jadbabai in 2020 in the setting of loss functions arising in deep learning. Theoretical work on (L_0, L_1)-smoothnes has focused on convergence guarantees for functions which satisfy this condition. In this paper we provide theoretical analysis of the condition in the setting of feedforward neural networks of depth at least 2, with either $L2$ or cross-entropy loss, and find the (L_0, L_1)-smoothness condition is not satisfied.

1. Introduction

The two properties most prized in the field of optimization are convexity and smoothness[1]. 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 [12] 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 [2] and [3], 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 [11] and Faw-Rout-Caramanis-Shakkottai in [4] 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 theoretical verification that deep neural networks satisfy the (L_0, L_1)-smoothness condition. In 2022 Patel-Zhang-Tian gave two examples of neural network

1. Smoothness is used in some communities to refer to a function that is infinitely differentiable, and in others to refer to a function whose second derivatives are uniformly bounded. In this paper, we use the word smooth in the second sense, as is prevalent in the optimization community. This is a perennial source of misunderstanding so we flag the issue here and review definitions in Section A.1 to minimize confusion.

loss functions L that are not (L_0, L_1)-smooth, in [10]. 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.

Building on their work, in this paper we examine the general case of deep feedforward neural networks. We show 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.

The high level picture is the following: if a function $L : \mathbb{R}^d \to \mathbb{R}$ satisfies the (L_0, L_1)-smoothness condition, then starting at the origin and going in any direction in the parameter space $\mathbb{R}^d$, the norm of the Hessian grows at most linearly in the norm of the gradient. In this paper, in the case of deep feedforward neural networks, we identify directions in the parameter space where the norm of the Hessian of the loss function L grows at least quadratically in the norm of the gradient. In this way, we show that the loss function of deep feedforward neural networks does not satisfy the (L_0, L_1)-smoothness condition.

We note that because on any compact set it is possible to choose L_0 and L_1 such that the loss function associated to a neural network of depth ≥ 3 satisfies the (L_0, L_1)-smoothness condition for that choice of L_0 and L_1, in order to prove our main results, by necessity we must analyze the relative magnitude of the gradient and hessian over a non-compact subset of the parameter space.

We also note that if for a chosen L_0 and L_1, the loss function fails to satisfy the (L_0, L_1)-smoothness condition at a point p in the parameter space, then by continuity of derivatives it will also fail to satisfy that (L_0, L_1)-smoothness condition on an open ball of positive radius centered at p. In particular, it will fail to satisfy the (L_0, L_1)-smoothness condition for that choice of L_0 and L_1 on a set of positive measure.

We now provide the statements of the main results Theorem 12 and Theorem 14 proved in Appendices C and D.

Our **first contribution** is to show that for deep feedforward networks with L2 loss, the (L_0, L_1) condition is not satisfied.

Theorem *[Theorem 12] In the setting of a regression problem, consider any feedforward neural network of depth $\ell \geq 2$, with twice differentiable activation function $\sigma : \mathbb{R} \to \mathbb{R}$ satisfying Assumption 1. The $L2$ loss function L of this network does not satisfy the (L_0, L_1)-smoothness condition, for any choice of L_0 and L_1.*

Our **second contribution** is to show that for deep feedforward networks with cross-entropy loss, the (L_0, L_1)condition is not satisfied.

Theorem *[Theorem 14] In the setting of a classification problem with b ≥ 2 classes, consider any feedforward neural network of depth $\ell \geq 2$, with twice differentiable activation function $\sigma : \mathbb{R} \to \mathbb{R}$ satisfying Assumption 1. The cross-entropy loss function L of this network does not satisfy the (L_0, L_1)-smoothness condition, for any choice of L_0 and L_1.*

Our **third contribution** is that in both cases not only do they not satisfy the (L_0, L_1) condition for any choice of L_0 and L_1, but they do not satisfy the generalized ℓ-smoothness condition of [6] for any subquadratic function ℓ. In other words, these loss functions lie exactly in the types of functions they showed can fail to have good convergence properties.

When there are no hidden layers, we show in Theorems 13 and 15 that for both $L2$ and cross-entropy loss, not only are there constants L_0 and L_1 such that the loss function L satisfies the (L_0, L_1)-smoothness condition, but in fact there is a constant m such that L is already $m-$smooth.

Thus for feedforward neural networks with any number of hidden layers other than one, we establish whether or not L satisfies the (L_0, L_1)-smoothness condition. The case of a feedforward neural network with one hidden layer is more challenging because certain derivatives become more complex in that case. Thus we restrict our analysis to the setting of feedforward architectures with zero or at least 2 hidden layers. The case of architectures with skip connections is even more challenging, for similar reasons. Thus we leave investigation of these cases to future work.

In the next section, we discuss related work. In Section 3 we give an overview of the strategy we will use to analyze this problem. In Appendix A we provide background and notation, in Appendix B we compute several derivatives essential to our analysis, and in Appendices C and D we provide proofs of our main results.

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 [12] . Their innovation is to allow the local smoothness constant to increase with the gradient norm.

Definition 1 (see [12]) *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 2 *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 [12], 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 [6], and [7], 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 [4].

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 [12], 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 [10]. 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 show that these examples are not isolated, but in fact representative of the general case. We prove that when the depth of the neural network is at least 3, the (L_0, L_1)-smoothness condition is not satisfied by feedforward neural networks.

2.1. Assumptions

Throughout this paper, we will assume that the activation function σ is twice differentiable. We will also often make the following assumption on the values of σ and σ'.

Assumption 1 *Assume that there is a constant c such that $\sigma(c) = 0$ but $\sigma'(c) \neq 0$.*

In the remainder of this paper the constant c will always denote the constant appearing in this assumption.

In the setting of cross-entropy loss, the y-values represent probabilities. So in this case we make the following assumption on the data.

Assumption 2

$$\sum_{j=1}^{d_{\text{out}}} y_{mj} = 1 \text{ for every } m = 1, \dots, n.$$

3. Roadmap

We provide an overview of our strategy for analyzing the (L_0, L_1)-smoothness of loss functions arising from deep neural networks. First, we note that given any choice of L_0 and L_1, in order to prove that a function L does not satisfy the (L_0, L_1)-smoothness condition, it suffices to exhibit a single point ρ at which the (L_0, L_1)-smoothness condition is not satisfied.

To show that for any choice of L_0 and L_1 there is a point $\rho_{(L_0,L_1)}$ at which L fails to be (L_0, L_1)-smooth for that choice of L_0 and L_1, we focus on a particular slice of the parameter space $\mathbb{R}^p$, along which we can produce such points.

Once we restrict attention to this slice, the relevant derivatives become manageable to compute and compare, and we complete the argument by directly computing them, and making judicious choices in constructing our points of interest.

We now describe the family at the center of this paper. Consider any feedforward neural network with 2 or more hidden layers. Within the parameter space $\mathbb{R}^d$ describing all possible choices of parameters for the network, we identify a one dimensional affine linear subspace which we call $\mathcal{R}$.

We define $\mathcal{R}$ to be the image of the following linear map $\rho : \mathbb{R} \to \mathbb{R}^d$. 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 c, the constant appearing in Assumption 1.

$$M^i = 0,\ b^i = \begin{pmatrix} c & \cdots & c \end{pmatrix}^T \qquad \text{if } 1 \le i \le \ell$$

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.

$$M = \begin{pmatrix} m_{11} & \dots & m_{1k} \\ \vdots & \ddots & \vdots \\ m_{b1} & \cdots & m_{bk} \end{pmatrix} \tag{2}$$

$$M^{\ell+1} = tM, \quad b^{\ell+1} = \mathbf{0} \tag{3}$$

Once we have chosen the loss function and made our choice for M, which fixes the direction $\mathcal{R}$, we bound how fast the norm of the gradient and the norm of the Hessian grows in this direction, as $t \to \infty$. This requires careful analysis of the derivatives of $f_\rho(x)$, which we carry out in Appendix B.

Once that has been completed, in Appendix C, we analyze the loss function L in the case of $L2$ loss. We choose the matrix M carefully, which determines a direction $\mathcal{R}$ that we focus on for $L2$ loss. We show that in this direction the norm of the gradient grows at most linearly in t, while the norm of the Hessian grows at least quadratically in t. Hence we are able to conclude that the norm of the Hessian grows at least quadratically in the norm of the gradient, making it impossible for the loss function L to satisfy the (L_0, L_1)-smoothness condition for any choice of constants L_0 and L_1.

In Appendix D we use a similar strategy to analyze the loss function L in the case of cross-entropy loss. We make a different choice for M in this setting, which determines a different direction $\mathcal{R}$ that we focus on for cross-entropy loss. We show that in this direction, as $t \to \infty$, the norm of the Hessian grows at least quadratically in the norm of the gradient, making it impossible for the loss function L to satisfy the (L_0, L_1)-smoothness condition for any choice of constants L_0 and L_1.

4. Conclusion

In this paper, we proved that for feedforward networks with two or more hidden layers, with either $L2$ or cross-entropy loss, the (L_0, L_1)-smoothness condition is not satisfied. To complement our analysis, we looked at an example feedforward network, sampled regions of its loss landscape, and verified that the relationship between the norms of the gradients and Hessians was consistent with our predictions.

This shows that the convergence guarantees that have been proved for (L_0, L_1)-smoothness cannot be applied directly to the loss functions arising from deep feedforward networks. Thus there is an opportunity to look to more sophisticated ways to relate techniques from optimization to deep learning.

In recent work, Li-Quian-Tian-Rakhlin-Jadbabaie [6] 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 proofs of Theorems 12 and 14 show that under the assumptions taken, 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 [6], 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 [8]. 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 practice in machine 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

[1] Alekh Agarwal, Sahand Negahban, and Martin 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.

[2] Ziyi Chen, Yi Zhou, Yingbin Liang, and Zhaosong Lu. Generalized-smooth nonconvex optimization is as efficient as smooth nonconvex optimization. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan 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.

[3] Michael Crawshaw, Mingrui Liu, Francesco Orabona, Wei Zhang, and Zhenxun 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.

[4] Matthew Faw, Litu Rout, Constantine Caramanis, and Sanjay Shakkottai. Beyond uniform smoothness: A stopped analysis of adaptive sgd. In Gergely Neu and Lorenzo 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.

[5] Elad Hazan, Kfir Levy, and Shai 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.

[6] Haochuan Li, Jian Qian, Yi Tian, Alexander Rakhlin, and Ali Jadbabaie. Convex and non-convex optimization under generalized smoothness. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=8aunGrXdkl.

[7] Haochuan Li, Alexander Rakhlin, and Ali 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., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/a3cc50126338b175e56bb3cad134db0b-Paper-Conference.pdf.

[8] Chaoyue Liu, Libin Zhu, and Mikhail 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.

[9] Haihao Lu, Robert M. Freund, and Yurii 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.

[10] Vivak Patel, Shushu Zhang, and Bowen 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.

[11] Bohan Wang, Yushun Zhang, Huishuai Zhang, Qi Meng, Zhi-Ming Ma, Tie-Yan Liu, and Wei Chen. Provable adaptivity in Adam, 2022. arXiv:2208.09900.

[12] Jingzhao Zhang, Tianxing He, Suvrit Sra, and Ali Jadbabaie. Why gradient clipping accelerates training: A theoretical justification for adaptivity. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJgnXpVYwS.

Appendix A. Background and notation

A.1. Weak smoothness

Definition 3 *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 4 (see [5]) *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 5 (see [1]) *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 6 (see [9]) *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 [8], 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 $\rho \in \mathbb{R}^p$.

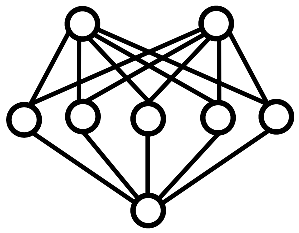

Figure 1: This is a diagram representing a feedforward neural network with one hidden layer, with the input width $d_{\text{in}} = 2$, the width of the hidden layer $k_1 = 5$, and output width $d_{\text{out}} = 1$.

Next, we choose an **activation function**

$$\sigma \colon \mathbb{R} \to \mathbb{R}. \tag{4}$$

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

$$\rho = (\mathbf{w}, \mathbf{b}) \in \mathbb{R}^p \tag{5}$$

in the parameter space, we define the function

$$f_\rho = f_{\mathbf{w},\mathbf{b}} \colon \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}} \tag{6}$$

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

$$\mathbb{R}^{d_{\text{in}}} \overset{M^1\mathbf{x}+\mathbf{b}^1}{\longrightarrow} \mathbb{R}^{c_1} \overset{\sigma}{\longrightarrow} \mathbb{R}^{c_1} \overset{M^2\mathbf{x}+\mathbf{b}^2}{\longrightarrow} \dots \overset{M^\ell\mathbf{x}+\mathbf{b}^\ell}{\longrightarrow} \mathbb{R}^{c_\ell} \overset{\sigma}{\longrightarrow} \mathbb{R}^{c_\ell} \overset{M^{\ell+1}\mathbf{x}+\mathbf{b}^{\ell+1}}{\longrightarrow} \mathbb{R}^{d_{\text{out}}}. \tag{7}$$

In this construction, the arrow $\mathbb{R}^{k_i} \overset{\sigma}{\longrightarrow} \mathbb{R}^{k_i}$ 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*

$$f_\alpha \colon x \mapsto w^2_{11}\big((w^1_{11}x + b^1_1)^2 + 1\big) + w^2_{12}\big((w^1_{21}x + b^1_2)^2 + 1\big) + w^2_{13}\big((w^1_{31}x + b^1_3)^2 + 1\big) + b^2_1. \tag{8}$$

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

$$f^1_\rho(\mathbf{x}) = M^1\mathbf{x} + \mathbf{b}^1 \tag{9}$$

$$f^i_\rho(\mathbf{x}) = M^i\sigma(f^{i-1}_\rho(\mathbf{x})) + \mathbf{b}^i, \tag{10}$$

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

$$L \colon \mathbb{R}^d \to \mathbb{R} \tag{11}$$

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

$$D := \big\{(\mathbf{x}_i, \mathbf{y}_i)\big\}_{i\in\{1,\ldots,n\}} \subset \mathbb{R}^{d_{\text{in}}} \times \mathbb{R}^{d_{\text{out}}}. \tag{12}$$

Definition 7 *The $L2$ loss is defined by:*

$$L(\rho) := \sum_{s=1}^{n} \big(f_\alpha(\mathbf{x}_s) - \mathbf{y}_s\big)^2. \tag{13}$$

Definition 8 *The cross-entropy loss is defined by:*

$$L(\rho) = -\sum_{m=1}^{n} \sum_{j=1}^{d_{\text{out}}} [y_m]_j \log \frac{e^{[f_\rho(x_m)]_j}}{\sum_{k=1}^{b} e^{[f_\rho(x_m)]_k}}. \tag{14}$$

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

Appendix B. Derivatives of f_ρ along our specialization

We begin by noting that in our direction $\mathcal{R}$ defined in Section 3, for any choice of t, for the corresponding parameter vector $\rho(t)$ the parameters of the first layer

$$(M^1, b^1)$$

satisfies

$$\sigma(M^1\mathbf{x} + \mathbf{b}^1) = 0,$$

for all inputs $\mathbf{x}$. Similarly, for all layers $2 \le i \le \ell$,

$$f_\rho^i(\mathbf{x})\Big|_{\rho=\rho(t)} = \boldsymbol{c} \tag{15}$$

for all choices of the matrix M.

Therefore,

$$f_\rho(\mathbf{x})\Big|_{\rho=\rho(t)} = \mathbf{0}. \tag{16}$$

Next, we compute the first derivatives of f_ρ with respect to the weights w^i_{jk}. In these formulas, we will use the notation that Δ^i_{jk} is the matrix with the dimensions of M^i, with all entries 0, except for a 1 in the j, k entry. Similarly, Δ^i_j is the vector with the same length as b_i, with all entries equal to 0, except for a 1 in the j^{th} entry.

These matrices and vectors appear repeatedly because

$$\frac{\partial}{\partial w^i_{jk}} M^i = \Delta^i_{jk}$$

and

$$\frac{\partial}{\partial b^i_j} b^i = \Delta^i_j.$$

In this section, we consider only the case of feedforward neural networks with at least two hidden layers, hence we will always assume that $\ell \geq 2$.

Lemma 9 *In our region $\mathcal{R}$, all first derivatives with respect to the weight parameters vanish:*

$$\frac{\partial}{\partial w^{i_1}_{j_1 k_1}} f_\rho(\mathbf{x})\Big|_{\rho=\rho(t)} = 0$$

Proof

To compute these derivatives, we work layer by layer, considering the derivatives of the intermediate functions f^i_ρ, as i increases.

We first prove the following:

Claim 1 *For any i, i_1,*

$$\frac{\partial}{\partial w^{i_1}_{j_1 k_1}} f^i_\rho(\mathbf{x})\Big|_{\rho=\rho(t)} = \begin{cases} \Delta^{i_1}_{j_1 k_1}\mathbf{x} & \textit{if } i_1 = i = 1 \\ 0 & \textit{otherwise} \end{cases} \tag{17}$$

Proof [Proof of Claim 1] Note that if $i_1 > i$, then

$$\frac{\partial}{\partial w^{i_1}_{j_1 k_1}} f^i_\rho(\mathbf{x})\Big|_{\rho=\rho(t)} = 0.$$

So for the remainder of this proof we consider the case that $i_1 \leq i$. We proceed with a proof by induction on i, considering i_1 to be fixed.

Base case $i = i_1$

When $i = i_1$, we take derivatives of equations 9 and 10 to obtain

$$\frac{\partial}{\partial w^{i_1}_{j_1 k_1}} f^{i_1}_\rho(\mathbf{x}) = \begin{cases} \Delta^{i_1}_{j_1 k_1}\mathbf{x} & \text{if } i_1 = 1 \\ \Delta^{i_1}_{j_1 k_1}\sigma(f^{i_1-1}_\rho(\mathbf{x})) & \text{otherwise.} \end{cases} \tag{18}$$

Once we specialize to our family $\rho(t)$, we obtain the simpler expression

$$\frac{\partial}{\partial w^{i_1}_{j_1 k_1}} f^{i_1}_\rho(\mathbf{x})\Big|_{\rho=\rho(t)} = \begin{cases} \Delta^{i_1}_{j_1 k_1}\mathbf{x} & \text{if } i_1 = 1 \\ 0 & \text{otherwise} \end{cases}$$

by Equation 15 and Assumption 1.

Inductive step

For the inductive step, $i > i_1$ and we use the chain rule and Equation 15 to see that under our specialization,

$$\frac{\partial}{\partial w^{i_1}_{j_1 k_1}} f^i_\rho(\mathbf{x})\Big|_{\rho=\rho(t)} = \sigma'(c)\left(M^i \frac{\partial}{\partial w^{i_1}_{j_1 k_1}} f^{i-1}_\rho(\mathbf{x})\right)\Big|_{\rho=\rho(t)} \tag{19}$$

Because in the specialization we have chosen, M^i is zero when $i \neq \ell+1$, this derivative vanishes unless $i = \ell + 1$. In other words,

$$\frac{\partial}{\partial w^{i_1}_{j_1k_1}} f^i_\rho(\mathbf{x})\Big|_{\rho=\rho(t)} = \begin{cases} \sigma'(c)tM \frac{\partial}{\partial w^{i_1}_{j_1k_1}} f^{i-1}_\rho(\mathbf{x})\Big|_{\rho=\rho(t)} & \text{if } i = \ell+1 \\ 0 & \text{otherwise} \end{cases}$$

By the inductive hypothesis

$$\frac{\partial}{\partial w^{i_1}_{j_1k_1}} f^{i-1}_\rho(\mathbf{x})\Big|_{\rho=\rho(t)} = 0$$

when $i - 1 = \ell > 1$, completing our inductive step, showing that the Claim 1 holds. △

Finally, we evaluate Equation 17 in the case that $i = \ell + 1$. Because we have assumed $\ell \geq 2$, the first case does not arise. Hence we see that

$$\frac{\partial}{\partial w^{i_1}_{j_1k_1}} f^{\ell+1}_\rho(\mathbf{x})\Big|_{\rho=\rho(t)} = 0$$

proving our desired result. ■

Next, we compute the first derivatives of f_ρ with respect to the biases b^i_j.

Lemma 10

$$\frac{\partial}{\partial b^{i_1}_{j_1}} f_\rho(\mathbf{x})\Big|_{\rho=\rho(t)} = \begin{cases} \Delta^{i_1}_{j_1} & \textit{if } i_1 = \ell+1 \\ \sigma'(c)tM\Delta^{i_1}_{j_1} & \textit{if } i_1 = \ell \\ 0 & \textit{otherwise} \end{cases}$$

Proof Again, we work layer by layer, considering the derivatives of the intermediate functions f^i_ρ, as i increases. We first prove:

Claim 2 *For any $i_1 \leq i$,*

$$\frac{\partial}{\partial b^{i_1}_{j_1}} f^i_\rho(\mathbf{x})\Big|_{\rho=\rho(t)} = \begin{cases} \Delta^{i_1}_{j_1} & \textit{if } i_1 = i \\ \sigma'(c)tM\Delta^{i_1}_{j_1} & \textit{if } i_1 + 1 = i = \ell+1 \\ 0 & \textit{otherwise} \end{cases} \tag{20}$$

Proof [Proof of Claim 2] As in Claim 1, if $i_1 > i$, we have

$$\frac{\partial}{\partial b^{i_1}_{j_1}} f^i_\rho(\mathbf{x})\Big|_{\rho=\rho(t)} = 0.$$

So we will restrict ourselves to the case that $i_1 \leq i$. We will again proceed with a proof by induction.

Base case $i = i_1$

First, suppose $i_1 = i$. Here, we compute the derivative to be

$$\frac{\partial}{\partial b^{i_1}_{j_1}} f^{i_1}_\rho(\mathbf{x}) = \Delta^{i_1}_{j_1}$$

This is consistent with Equation 20. This leaves the case when $i_1 < i$.

Inductive step

For the inductive step, we use the chain rule and Equation 15 to see that under our specialization,

$$\frac{\partial}{\partial b^{i_1}_{j_1}} f^i_\rho(\mathbf{x})\Big|_{\rho=\rho(t)} = \sigma'(c)\left(M^i \frac{\partial}{\partial b^{i_1}_{j_1}} f^{i-1}_\rho(\mathbf{x})\right)\Big|_{\rho=\rho(t)}$$

Because in the specialization we have chosen, M^i is zero when $i \neq \ell + 1$, this derivative vanishes unless $i = \ell + 1$. In other words,

$$\frac{\partial}{\partial b^{i_1}_{j_1}} f^i_\rho(\mathbf{x})\Big|_{\rho=\rho(t)} = \begin{cases} \sigma'(c)tM\frac{\partial}{\partial b^{i_1}_{j_1}} f^{i-1}_\rho(\mathbf{x})\Big|_{\rho=\rho(t)} & \text{if } i = \ell+1 \\ 0 & \text{otherwise} \end{cases}$$

By the inductive hypothesis, we see that

$$\frac{\partial}{\partial b^{i_1}_{j_1}} f^{i-1}_\rho(\mathbf{x})\Big|_{\rho=\rho(t)} = \begin{cases} \Delta^{i_1}_{j_1} & \text{if } i_1 = i-1 \\ 0 & \text{otherwise,} \end{cases}$$

completing our inductive step, and shows that Claim 2 holds. $\triangle$

To complete the proof of the lemma, we specialize Claim 2 to the case that $i = \ell + 1$ and see that

$$\frac{\partial}{\partial b^{i_1}_{j_1}} f^{\ell+1}_\rho(\mathbf{x})\Big|_{\rho=\rho(t)} = \begin{cases} \Delta^{i_1}_{j_1} & \text{if } i_1 = \ell+1 \\ \sigma'(c)tM\Delta^{i_1}_{j_1} & \text{if } i_1 = \ell \\ 0 & \text{otherwise} \end{cases}$$

proving our desired result. ■

Finally, we compute the second derivatives of f_ρ, with respect to any choice of weight and bias variables.

Lemma 11

The only nonzero second derivatives of f_ρ are

$$\frac{\partial}{\partial w^{i_2}_{j_2k_2}}\frac{\partial}{\partial w^{i_1}_{j_1k_1}} f_\rho(\mathbf{x})\Big|_{\rho=\rho(t)} = \begin{cases} (\sigma'(c))^2 tM\Delta^{i_2}_{j_2k_2}\Delta^{i_1}_{j_1k_1}\mathbf{x} & \text{if } i_1 = 1 \text{ and } i_2 = \ell = 2, \\ (\sigma'(c))^2 tM\Delta^{i_2}_{j_2k_2}\Delta^{i_1}_{j_1k_1}\mathbf{x} & \text{if } i_2 = 1 \text{ and } i_1 = \ell = 2, \end{cases}$$

$$\frac{\partial}{\partial w^{i_2}_{j_2k_2}}\frac{\partial}{\partial b^{i_1}_{j_1}} f_\rho(\mathbf{x})\Big|_{\rho=\rho(t)} = \begin{cases} \sigma'(c)\Delta^{i_2}_{j_2k_2}\Delta^{i_1}_{j_1} & \text{if } i_1+1 = i_2 = \ell+1 \\ (\sigma'(c))^2 tM\Delta^{i_2}_{j_2k_2}\Delta^{i_1}_{j_1} & \text{if } i_1+1 = i_2 = \ell, \end{cases}$$

and

$$\frac{\partial}{\partial b^{i_2}_{j_2}}\frac{\partial}{\partial b^{i_1}_{j_1}} f_\rho(\mathbf{x})\Big|_{\rho=\rho(t)} = \sigma''(c)tM(\Delta^{i_2}_{j_2} \odot \Delta^{i_1}_{j_1}) \qquad \text{if } i_1 = i_2 = \ell,$$

where $\odot$ denotes entry-wise multiplication.

Proof The second derivatives of f_ρ are more challenging to compute than the first derivatives. One difficulty is that we must compute both derivatives before specializing to our family $\rho(t)$.

As in the case of Lemmas 9 and 10, we proceed by computing the derivatives of the intermediate functions f_ρ^i. To streamline the argument, we introduce the notation that z^{i_1} is a variable from the $i_1{}^{th}$ layer, either $w^{i_1}_{j_1k_1}$, or $b^{i_1}_{j_1}$, for some j_1, k_1.

Note that the second derivative vanishes if any of the derivatives are with respect to weights or biases from a layer after the i^{th} layer. That is,

$$\frac{\partial}{\partial z^{i_1}}\frac{\partial}{\partial z^{i_2}} f^i_\rho(\mathbf{x})\Big|_{\rho=\rho(t)} = 0$$

if i is less than either i_1 or i_2. Hence for the rest of this proof, we assume that

$$i \geq i_2 \geq i_1.$$

We will prove the following statement by induction on i. Consider $i_2 \geq i_1$ fixed, and i will be increasing. The base case we take is when $i = i_2$.

Claim 3 *All second derivatives of the intermediate function f^i_ρ vanish except for the following:*

$$\frac{\partial}{\partial w^{i_2}_{j_2k_2}}\frac{\partial}{\partial w^{i_1}_{j_1k_1}} f^i_\rho(\mathbf{x})\Big|_{\rho=\rho(t)} = \begin{cases} \sigma'(c)\Delta^{i_2}_{j_2k_2}\Delta^{i_1}_{j_1k_1}\mathbf{x} & \text{if } i_1 = 1, i_2 = i = 2 \\ (\sigma'(c))^2 tM\Delta^{i_2}_{j_2k_2}\Delta^{i_1}_{j_1k_1}\mathbf{x} & \text{if } i_1 = 1, i_2 = 2, i = 3, \ell = 2, \end{cases} \tag{21}$$

$$\frac{\partial}{\partial w^{i_2}_{j_2k_2}}\frac{\partial}{\partial b^{i_1}_{j_1}} f^i_\rho(\mathbf{x})\Big|_{\rho=\rho(t)} = \begin{cases} \sigma'(c)\Delta^{i_2}_{j_2k_2}\Delta^{i_1}_{j_1} & \text{if } i_1 + 1 = i_2 = i \\ (\sigma'(c))^2 tM\Delta^{i_2}_{j_2k_2}\Delta^{i_1}_{j_1} & \text{if } i_1 + 1 = i_2 = i - 1 = \ell, \end{cases} \tag{22}$$

and

$$\frac{\partial}{\partial b^{i_2}_{j_2}}\frac{\partial}{\partial b^{i_1}_{j_1}} f^i_\rho(\mathbf{x})\Big|_{\rho=\rho(t)} = \sigma''(c)tM(\Delta^{i_2}_{j_2} \odot \Delta^{i_1}_{j_1}) \qquad \text{if } i_1 = i_2 = \ell, i = \ell + 1. \tag{23}$$

Proof [Proof of Claim 3]

Base case $i = i_2$

Case 1 The variable z^{i_1} is a weight or bias and z^{i_2} is a bias.

This derivative vanishes, because

$$\frac{\partial}{\partial z^{i_2}} f^{i_2}_\rho$$

is a constant function, so differentiating it again with respect to either a weight or a bias variable gives 0.

Case 2 Both variables z^{i_1} and z^{i_2} are weights.

Analogously to (18), we have

$$\frac{\partial}{\partial w^{i_2}_{j_2k_2}} f^{i_2}_\rho(\mathbf{x}) = \begin{cases} \Delta^{i_2}_{j_2k_2}\mathbf{x} & \text{if } i_2 = 1 \\ \Delta^{i_2}_{j_2k_2}\sigma(f^{i_2-1}_\rho(\mathbf{x})) & \text{if } i_2 \geq 2. \end{cases}$$

Further, taking $\frac{\partial}{\partial w^{i_1}_{j_1k_1}}$ on both sides, we obtain

$$\frac{\partial}{\partial w^{i_2}_{j_2k_2}} \frac{\partial}{\partial w^{i_1}_{j_1k_1}} f^{i_2}_\rho(\mathbf{x}) = \begin{cases} 0 & \text{if } i_2 = 1 \\ \Delta^{i_2}_{j_2k_2}\frac{\partial}{\partial w^{i_1}_{j_1k_1}}\sigma(f^{i_2-1}_\rho(\mathbf{x})) & \text{if } i_2 \geq 2. \end{cases}$$

This immediately implies the base case when $i_2 = 1$. When $i_2 \geq 2$, if we take the specialization, and simplify as in (19), we get

$$\frac{\partial}{\partial w^{i_2}_{j_2k_2}} \frac{\partial}{\partial w^{i_1}_{j_1k_1}} f^{i_2}_\rho(\mathbf{x})\Big|_{\rho=\rho(t)} = \sigma'(c)\Delta^{i_2}_{j_2k_2}\left(\frac{\partial}{\partial w^{i_1}_{j_1k_1}} f^{i_2-1}_\rho(\mathbf{x})\right)\Big|_{\rho=\rho(t)}. \tag{24}$$

Substituting $i_2 - 1$ for i in Equation (17) gives:

$$\frac{\partial}{\partial w^{i_1}_{j_1k_1}} f^{i_2-1}_\rho(\mathbf{x})\Big|_{\rho=\rho(t)} = \begin{cases} \Delta^{i_1}_{j_1k_1}\mathbf{x} & \text{if } i_1 = 1, i_2 = 2 \\ 0 & \text{otherwise} \end{cases}$$

So we conclude that

$$\frac{\partial}{\partial w^{i_2}_{j_2k_2}} \frac{\partial}{\partial w^{i_1}_{j_1k_1}} f^{i_2}_\rho(\mathbf{x})\Big|_{\rho=\rho(t)} = \begin{cases} \sigma'(c)\Delta^{i_2}_{j_2k_2}\Delta^{i_1}_{j_1k_1}\mathbf{x} & \text{if } i_1 = 1, i_2 = 2 \\ 0 & \text{otherwise} \end{cases}$$

Case 3 The variable z^{i_1} is a bias and z^{i_2} is a weight.

Analogously to the previous case, we prove that

$$\frac{\partial}{\partial w^{i_2}_{j_2k_2}} \frac{\partial}{\partial b^{i_1}_{j_1}} f^1_\rho(\mathbf{x}) = 0,$$

establishing the base case when $i_2 = 1$, and for $i_2 \geq 2$, we obtain analogously to (24) that

$$\frac{\partial}{\partial w^{i_2}_{j_2k_2}} \frac{\partial}{\partial b^{i_1}_{j_1}} f^{i_2}_\rho(\mathbf{x})\Big|_{\rho=\rho(t)} = \sigma'(c)\Delta^{i_2}_{j_2k_2}\left(\frac{\partial}{\partial b^{i_1}_{j_1}} f^{i_2-1}_\rho(\mathbf{x})\right)\Big|_{\rho=\rho(t)}$$

Rewriting (20) in the case $i = i_2 - 1$ gives:

$$\frac{\partial}{\partial b^{i_1}_{j_1}} f^{i_2-1}_\rho(\mathbf{x})\Big|_{\rho=\rho(t)} = \begin{cases} \Delta^{i_1}_{j_1} & \text{if } i_1 = i_2 - 1 \\ \sigma'(c)tM\Delta^{i_1}_{j_1} & \text{if } i_1 + 2 = i_2 = \ell + 2 \\ 0 & \text{otherwise} \end{cases}$$

We have only $\ell + 1$ affine maps in our neural network, so $i_2 \leq \ell + 1$. Hence the second case cannot occur.

We conclude that

$$\frac{\partial}{\partial b^{i_1}_{j_1}} f^{i_2-1}_{\rho}(\mathbf{x})\Big|_{\rho=\rho(t)} = \begin{cases} \Delta^{i_1}_{j_1} & \text{if } i_1 = i_2 - 1 \\ 0 & \text{otherwise} \end{cases}$$

and

$$\frac{\partial}{\partial w^{i_2}_{j_2 k_2}} \frac{\partial}{\partial b^{i_1}_{j_1}} f^{i_2}_{\rho}(\mathbf{x})\Big|_{\rho=\rho(t)} = \begin{cases} \sigma'(c)\Delta^{i_2}_{j_2 k_2}\Delta^{i_1}_{j_1} & \text{if } i_1 = i_2 - 1 \\ 0 & \text{otherwise} \end{cases}$$

Inductive step

Assume we have proved Claim 3 for $i-1$. Now we prove it for i.

By applying the chain rule twice, we obtain the equality

$$\frac{\partial}{\partial z^{i_2}} \frac{\partial}{\partial z^{i_1}} f^{i}_{\rho}(\mathbf{x})\Big|_{\rho=\rho(t)} = \sigma''(c)\left(M^i \frac{\partial}{\partial z^{i_2}} f^{i-1}_{\rho}(\mathbf{x}) \odot \frac{\partial}{\partial z^{i_1}} f^{i-1}_{\rho}(\mathbf{x})\right)\Big|_{\rho=\rho(t)} + \sigma'(c)\left(M^i \frac{\partial}{\partial z^{i_2}} \frac{\partial}{\partial z^{i_1}} f^{i-1}_{\rho}(\mathbf{x})\right)\Big|_{\rho=\rho(t)}. \quad (25)$$

When $i < \ell + 1$, both terms vanish because $M^i\Big|_{\rho=\rho(t)} = 0$, and this establishes the claim in this case, noting that $i > i_2$ in the inductive step. Therefore, from now on, we will assume $i = \ell + 1 \geq 3$.

We now analyze the two terms on the right hand side of Equation 25 separately. Note that in the range $i > i_2 \geq i_1$ we are considering, by Claim 1, the first term vanishes if any of the variables z^{i_1} and z^{i_2} is a weight. Finally, by Claim 2, noting that $i - 1 < \ell + 1$, we have

$$\sigma''(c)\left(M^i \frac{\partial}{\partial b^{i_2}_{j_2}} f^{i-1}_{\rho}(\mathbf{x}) \odot \frac{\partial}{\partial b^{i_1}_{j_1}} f^{i-1}_{\rho}(\mathbf{x})\right)\Big|_{\rho=\rho(t)} = \begin{cases} \sigma''(c) M^i (\Delta^{i_1}_{j_1} \odot \Delta^{i_2}_{j_2}) & \text{if } i_1 = i_2 = i - 1, \\ 0 & \text{otherwise.} \end{cases}$$

It remains to analyze the second term on the right hand side of Equation 25. For this, we again distinguish three cases.

Case 1 The variable z^{i_1} is a weight or bias and z^{i_2} is a bias.

In this case, the second term vanishes by the induction hypothesis, noting that $i - 1 < \ell + 1$.

Case 2 The variable z^{i_1} is a weight and z^{i_2} is a weight.

By the induction hypothesis, noting that $i - 1 < \ell + 1$, we have

$$\sigma'(c) M^i \left(\frac{\partial}{\partial z^{i_2}} \frac{\partial}{\partial z^{i_1}} f^{i-1}_{\rho}(\mathbf{x})\right)\Big|_{\rho=\rho(t)} = 0$$

unless $i_1 = 1$ and $i_2 = i - 1 = 2$, or more explicitly, $i_1 = 1$, $i_2 = 2$, $i = 3$ and $\ell = 2$ by our assumption $i = \ell + 1$. In this case, the second term equals

$$\sigma'(c) M^i \left(\frac{\partial}{\partial z^{i_2}} \frac{\partial}{\partial z^{i_1}} f^{i-1}_{\rho}(\mathbf{x})\right)\Big|_{\rho=\rho(t)} = \sigma'(c) t M \cdot \sigma'(c) \Delta^{i_2}_{j_2 k_2} \Delta^{i_1}_{j_1 k_1} \mathbf{x}$$

Case 3 The variable z^{i_1} is a bias and z^{i_2} is a weight.

By the induction hypothesis, again noting that $i - 1 < \ell + 1$,

$$\sigma'(c)M^i \left(\frac{\partial}{\partial z^{i_2}} \frac{\partial}{\partial z^{i_1}} f_\rho^{i-1}(\mathbf{x}) \right) \Big|_{\rho=\rho(t)} = 0$$

unless $i_1 + 1 = i_2 = i - 1$. In this case, noting that $i - 1 = \ell$, the second term equals

$$\sigma'(c)M^i \left(\frac{\partial}{\partial z^{i_2}} \frac{\partial}{\partial z^{i_1}} f_\rho^{i-1}(\mathbf{x}) \right) \Big|_{\rho=\rho(t)} = \sigma'(c)tM \cdot \sigma'(c)\Delta^{i_2}_{j_2k_2}\Delta^{i_1}_{j_1}$$

Adding together both terms on the right hand side of Equation 25 yields Claim 3. $\triangle$

Finally, we specialize Claim 3 to the case that $i = \ell + 1$. Because we have assumed $\ell \geq 2$, the first case in (21) does not arise. Taking into account that Claim 3 only lists nonvanishing derivatives for $i_2 \geq i_1$, we obtain the statement of Lemma 11.

■

Appendix C. (L_0, L_1)-smoothness, L2 loss

In this section, we study the (L_0, L_1) smoothness of the loss function L of a feedforward neural network with $L2$ loss.

To study this case, we choose an explicit member of the family we have been studying above, by fixing the weights of the last layer $M^{\ell+1}$ to be multiples of the especially simple matrix

$$\Omega = \begin{pmatrix} 1 & \dots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{pmatrix}. \tag{26}$$

Theorem 12 *Consider any feedforward neural network of depth $\ell \geq 2$, with twice differentiable activation function $\sigma : \mathbb{R} \to \mathbb{R}$ satisfying Assumption 1. The L2 loss function L of this network does not satisfy the (L_0, L_1)-smoothness condition, for any choice of L_0 and L_1.*

Proof

In the case of $L2$ loss (13), we choose $M^{\ell+1}$ to be the matrix

$$M^{\ell+1} = t\Omega = \begin{pmatrix} t & \dots & t \\ \vdots & \ddots & \vdots \\ t & \cdots & t \end{pmatrix} \tag{27}$$

and will consider the derivatives of L as $t \to \infty$.

Armed with the calculations in Section B of the first and second derivatives of the function $f_{\rho(t)}$, we consider the gradient and Hessian of $L(\rho)$, the main quantities of interest for us.

As noted in Remark 2, one can consider the (L_0, L_1)-condition for any matrix norm. In this proof, we will use the Frobenius norm. Proving the failure of the (L_0, L_1)-condition for this norm also proves the failure of the (L_0, L_1)-condition for any matrix norm.

We begin by calculating

$$\nabla L(\rho)$$

and then evaluate it along our space $\mathcal{R}$ of interest.

The gradient of L is given by

$$\nabla L(\rho) = 2\sum_{s=1}^{n} \big((\mathbf{D}f_\rho(\mathbf{x}_s))^T(f_\rho(\mathbf{x}_s) - \mathbf{y}_s)\big),$$

where $\mathbf{D}f_\rho$ is the Jacobian matrix of f_ρ.

Along our specialization, $f_{\rho(t)}$ is identically zero, so this simplifies to

$$\nabla L(\rho)\Big|_{\rho=\rho(t)} = -2\sum_{s=1}^{n} (\mathbf{D}f_\rho(\mathbf{x}_s))^T\mathbf{y}_s\Big|_{\rho=\rho(t)} \tag{28}$$

In any term of this sum, the summand $\mathbf{y}_m$ is constant in t. Meanwhile, by Lemmas 9 and 10, the entries of $\mathbf{D}f_\rho(\mathbf{x}_m)\Big|_{\rho=\rho(t)}$ are polynomial of degree 0 or 1 in t. Therefore,

$$\left\|\nabla L(\rho)\big|_{\rho=\rho(t)}\right\|^2$$

is a polynomial in t of degree at most 2, and

$$\left\|\nabla L(\rho)\big|_{\rho=\rho(t)}\right\|$$

is a function that grows at most linearly in t.

Next, we will show that $\|\mathbf{H}L(\rho)\|\|_{\rho=\rho(t)}$ grows at least quadratically in t. For this, it will suffice to prove that the absolute value of at least one entry of $\|\mathbf{H}L(\rho)|\rho = \rho(t)\|$ grows quadratically in t.

We compute

$$\mathbf{H}L(\rho) = 2\sum_{s=1}^{n} \big((\mathbf{H}f_\rho(\mathbf{x}_s))^T(f_\rho(\mathbf{x}_s) - \mathbf{y}_s)\big) + 2\sum_{s=1}^{n}(\mathbf{D}f_\rho(\mathbf{x}_s))^T\mathbf{D}f_\rho(\mathbf{x}_s),$$

where we view $\mathbf{H}f_\rho$ as a matrix-valued vector. Along our specialization, this simplfies to

$$\mathbf{H}L(\rho)\big|_{\rho=\rho(t)} = -2\sum_{s=1}^{n}(\mathbf{H}f_\rho(\mathbf{x}_s))^T\mathbf{y}_s\big|_{\rho=\rho(t)} + 2\sum_{s=1}^{n}(\mathbf{D}f_\rho(\mathbf{x}_s))^T\mathbf{D}f_\rho(\mathbf{x}_s)\big|_{\rho=\rho(t)}, \tag{29}$$

We start by considering the first term in Equation (29). The entries of

$$\mathbf{H}f_\rho(\mathbf{x}_s)\big|_{\rho=\rho(t)}$$

are linear in t. Hence, it will suffice to prove that at least one entry of the second term in Equation (29) is at least quadratic in t. The entry corresponding to the $\frac{\partial^2}{(\partial b_1^\ell)^2}$-derivative is given by

$$2\sum_{s=1}^{n}\left\|\frac{\partial}{\partial b_1^\ell} f_\rho(\mathbf{x}_s)\right\|^2\big|_{\rho=\rho(t)}.$$

By Lemma 10, this simplifies to

$$2\sum_{s=1}^{n} \|\sigma'(c)tM\Delta_1^\ell\|^2.$$

The matrix $M\Delta_1^\ell$ is the vector of length d_{out} whose entries are all equal to 1. Therefore $\|M\Delta_1^\ell\|^2 = d_{\text{out}}$, and we conclude that our expression reduces to

$$2(\sigma'(c))^2t^2\sum_{s=1}^{n} \|M\Delta_1^\ell\|^2 = 2(\sigma'(c))^2t^2\sum_{s=1}^{n} d_{\text{out}} = 2(\sigma'(c))^2t^2 \cdot n \cdot d_{\text{out}}.$$

Thus, at least one of the entries of the second term in (29) grows at least quadratically in t. We conclude that $\|\mathbf{H}L(\rho)|\rho = \rho(t)\|$ also grows at least quadratically in t.

We may then conclude the proof by noting that along our specialization $\rho(t)$, if we take $t \to \infty$, the L^2 norm of the gradient of L grows linearly while the norm of the Hessian grows at least quadratically. There is no choice of scalars $L0$ and $L1$ that would uniformly for all t ensure that

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

Therefore, L is not (L_0, L_1)-smooth for any choice of L_0 and L_1. ■

In Theorem 12, we saw that the feedforward networks of depth $\ell \geq 2$, the loss function is not (L_0, L_1)-smooth for any choice of L_0 and L_1. In the next theorem, we see that when $\ell = 0$, the loss function not only satisfies the (L_0, L_1) condition, but there exist values of m for which it is m-smooth. This leaves the case of depth $\ell = 1$ open for future work.

Theorem 13

Consider any feedforward neural network of depth $\ell = 0$, in other words with no hidden layers. Suppose the activation function σ satisfies Assumption 1. Then not only does the $L2$ loss function L of this network satisfy the (L0,L1) condition, but L satisfies the stronger condition that there exist values m such that L is m-smooth.

Proof In this case, because there are no hidden layers, f_ρ is a linear function in the weights and biases of the single layer. The loss function L then is a quadratic function in the weights and biases. Thus the Hessian of L is a constant matrix, with norm bounded by some constant L_0. Hence L is L_0-smooth for that value of L_0, and (L_0, L_1) smooth for that value of L_0 and any value of L_1. ■

Appendix D. Cross-Entropy Loss

Now we consider the case of cross-entropy loss, proving the following theorem.

Theorem 14 *In the setting of a classification problem with $b \geq 2$ classes, consider any feedforward neural network of depth $\ell \geq 2$, with twice differentiable activation function $\sigma : \mathbb{R} \to \mathbb{R}$ satisfying Assumption 1. The cross-entropy loss function L of this network does not satisfy the (L_0, L_1)-smoothness condition, for any choice of L_0 and L_1.*

Proof In this setting, we choose $M^{\ell+1}$ to be the scalar matrix

$$M^{\ell+1} = tI, \tag{30}$$

where I denotes the identity matrix extended by zeros to the right or bottom, and we will consider the derivatives of L as $t \to \infty$.

We begin by rewriting the Formula (14) for cross-entropy loss as

$$L(\rho) = -\sum_{m=1}^{n}\left(\sum_{j=1}^{d_{\text{out}}}[\mathbf{y}_m]_j[f_\rho(\mathbf{x}_m)]_j - \big(\sum_{j=1}^{d_{\text{out}}}[\mathbf{y}_m]_j\big)\big(\log\sum_{k=1}^{d_{\text{out}}}e^{[f_\rho(\mathbf{x}_m)]_k}\big)\right)$$

Because the vector $\mathbf{y}_m$ represents probabilities, in data for cross-entropy loss, the sum of the entries of $\mathbf{y}_m$ is 1, simplifying the second term to

$$L(\rho) = -\sum_{m=1}^{n}\left(\sum_{j=1}^{d_{\text{out}}}[\mathbf{y}_m]_j[f_\rho(\mathbf{x}_m)]_j\right) + \sum_{m=1}^{n}\left(\log\sum_{k=1}^{d_{\text{out}}}e^{[f_\rho(\mathbf{x}_m)]_k}\right).$$

We compute the gradient of L, in terms of the gradients of f, as

$$.\nabla L(\rho) = -\sum_{m=1}^{n}\left(\sum_{j=1}^{d_{\text{out}}}[\mathbf{y}_m]_j\nabla[f_\rho(\mathbf{x}_m)]_j\right) + \sum_{m=1}^{n}\frac{\sum_{k=1}^{d_{\text{out}}}e^{[f_\rho(\mathbf{x}_m)]_k}\nabla[f_\rho(\mathbf{x}_m)]_k}{\sum_{k=1}^{d_{\text{out}}}e^{[f_\rho(\mathbf{x}_m)]_k}} \tag{31}$$

Next, we compute the Hessian of L, in terms of derivatives of f, as

$$\begin{aligned}\mathbf{H}L(\rho) = &-\sum_{m=1}^{n}\left(\sum_{j=1}^{d_{\text{out}}}[\mathbf{y}_m]_j\mathbf{H}[f_\rho(\mathbf{x}_m)]_j\right) + \sum_{m=1}^{n}\frac{\sum_{k=1}^{d_{\text{out}}}e^{[f_\rho(\mathbf{x}_m)]_k}\mathbf{H}[f_\rho(\mathbf{x}_m)]_k}{\sum_{k=1}^{d_{\text{out}}}e^{[f_\rho(\mathbf{x}_m)]_k}}\\ &+\sum_{m=1}^{n}\frac{\sum_{k=1}^{d_{\text{out}}}e^{[f_\rho(\mathbf{x}_m)]_k}\nabla^T[f_\rho(\mathbf{x}_m)]_k\nabla[f_\rho(\mathbf{x}_m)]_k}{\sum_{k=1}^{d_{\text{out}}}e^{[f_\rho(\mathbf{x}_m)]_k}}\\ &-\sum_{m=1}^{n}\frac{\left(\sum_{k=1}^{d_{\text{out}}}e^{[f_\rho(\mathbf{x}_m)]_k}\nabla^T[f_\rho(\mathbf{x}_m)]_k\right)\left(\sum_{k=1}^{d_{\text{out}}}e^{[f_\rho(\mathbf{x}_m)]_k}\nabla[f_\rho(\mathbf{x}_m)]_k\right)}{\left(\sum_{k=1}^{d_{\text{out}}}e^{[f_\rho(\mathbf{x}_m)]_k}\right)^2}\end{aligned}$$

Now, we specialize to our space $\mathcal{R}$ and analyze the dependence of ∇L on t when restricted to $\mathcal{R}$. Along our specialization, $f_{\rho(t)}$ is identically zero, simplifying the above expressions to

$$\nabla L(\rho)\big|_{\rho=\rho(t)} = -\sum_{m=1}^{n}\left(\sum_{j=1}^{d_{\text{out}}}[\mathbf{y}_m]_j\nabla[f_\rho(\mathbf{x}_m)]_j\big|_{\rho=\rho(t)}\right) + \sum_{m=1}^{n}\frac{\sum_{k=1}^{d_{\text{out}}}\nabla[f_\rho(\mathbf{x}_m)]_k\big|_{\rho=\rho(t)}}{d_{\text{out}}}$$

and

$$\begin{aligned}\mathbf{H}L(\rho) = &-\sum_{m=1}^{n}\left(\sum_{j=1}^{d_{\text{out}}}[\mathbf{y}_m]_j\mathbf{H}[f_\rho(\mathbf{x}_m)]_j\Big|_{\rho=\rho(t)}\right) + \sum_{m=1}^{n}\frac{\sum_{k=1}^{d_{\text{out}}}\mathbf{H}[f_\rho(\mathbf{x}_m)]_k\Big|_{\rho=\rho(t)}}{d_{\text{out}}} \\ &+\sum_{m=1}^{n}\frac{\sum_{k=1}^{d_{\text{out}}}\nabla^T[f_\rho(\mathbf{x}_m)]_k\Big|_{\rho=\rho(t)}\nabla[f_\rho(\mathbf{x}_m)]_k\Big|_{\rho=\rho(t)}}{d_{\text{out}}} \\ &-\sum_{m=1}^{n}\frac{\left(\sum_{k=1}^{d_{\text{out}}}\nabla^T[f_\rho(\mathbf{x}_m)]_k\Big|_{\rho=\rho(t)}\right)\left(\sum_{k=1}^{d_{\text{out}}}\nabla[f_\rho(\mathbf{x}_m)]_k\Big|_{\rho=\rho(t)}\right)}{{d_{\text{out}}}^2}\end{aligned} \tag{32}$$

We first consider t-dependence in $\nabla L\big|_{\rho=\rho(t)}$. By Lemmas 9 and 10, the entries of $\nabla[f_\rho(\mathbf{x}_m)]_k\big|_{\rho=\rho(t)}$ are polynomials of degree 0 or 1 in t. Hence along our specialization $\mathcal{R}$,

$$\left\|\nabla L(\rho)\big|_{\rho=\rho(t)}\right\|^2$$

is a polynomial in t of degree at most 2, and

$$\left\|\nabla L(\rho)\big|_{\rho=\rho(t)}\right\|$$

is a function that grows at most linearly in t.

We next show that $\|\mathbf{H}L\|\big|_{\rho=\rho(t)}$ grows at least quadratically in t. For this, it will suffice to prove that at least one entry grows at least quadratically in t. First, consider the first two of the four terms in (32). The entries of

$$\mathbf{H}[f_\rho(\mathbf{x}_s)]_k\big|_{\rho=\rho(t)}$$

are linear in t, and hence the entries of these two terms are at most linear in t.

We now consider the remaining two terms in (32) together. We may simplify the entry corresponding to the $\frac{\partial^2}{(\partial b_1^\ell)^2}$ derivative via Lemma 10:

$$\begin{aligned}&\sum_{m=1}^{n}\frac{\sum_{k=1}^{d_{\text{out}}}\sigma'(c)t[M\Delta_1^\ell]_k\sigma'(c)t[M\Delta_1^\ell]_k}{d_{\text{out}}} \\ -&\sum_{m=1}^{n}\frac{\left(\sum_{k=1}^{d_{\text{out}}}\sigma'(c)t[M\Delta_1^\ell]_k\right)\left(\sum_{k=1}^{d_{\text{out}}}\sigma'(c)t[M\Delta_1^\ell]_k\right)}{{d_{\text{out}}}^2}\end{aligned}$$

After further simplification, we may rewrite this as:

$$(\sigma'(c))^2nt^2\left(\frac{\sum_{k=1}^{d_{\text{out}}}\left([M\Delta_1^\ell]_k\right)^2}{d_{\text{out}}}-\frac{\left(\sum_{k=1}^{d_{\text{out}}}[M\Delta_1^\ell]_k\right)^2}{{d_{\text{out}}}^2}\right).$$

Given our choice of M in (30), we have

$$[M\Delta_1^\ell]_k = [\Delta_1^\ell]_k = \begin{cases}1 & \text{if } k=1; \\ 0 & \text{if } k\neq 1.\end{cases}$$

So, we further simplify the entry to:

$$(\sigma'(c))^2 n \left(\frac{1}{d_{\text{out}}} - \frac{1}{{d_{\text{out}}}^2} \right) t^2 \neq 0.$$

Thus, we have established an entry of $\mathbf{H}L|_{\rho=\rho(t)}$ whose absolute value grows at least quadratically in t.

As in the L^2-loss case, we have established that along our specialization $\rho(t)$, if we take $t \to \infty$, of the gradient of L grows linearly while the norm of the Hessian grows at least quadratically. There is no choice of scalars $L0$ and $L1$ that would uniformly for all t ensure that

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

Therefore, L is not (L_0, L_1)-smooth for any choice of L_0 and L_1.

■

In Theorem 14, we saw that the feedforward networks of depth $\ell \geq 2$, the loss function is not (L_0, L_1)-smooth for any choice of L_0 and L_1. As in the case of $L2$ loss, we will now see that when $\ell = 0$, the loss function not only satisfies the (L_0, L_1) condition, but there exist values of m for which it is m-smooth. This leaves the case of depth $\ell = 1$ open for future work.

Theorem 15 *Suppose that L is the loss function for a feedforward neural network with cross-entropy loss, with no hidden layers, and smooth (not necessarily monotonic) activation function $\sigma : \mathbb{R} \to \mathbb{R}$. Then not only does L satisfy the (L0,L1) condition, but L is m-smooth for some m.*

Proof When $\ell = 0$, the functions $[f_\rho(\mathbf{x}_m)]_j$ are linear in the weights and biases. Hence, $\mathbf{H}[f_\rho(\mathbf{x}_m)]_j = 0$ and $\nabla[f_\rho(\mathbf{x}_m)]_j$ is a constant vector. The result then follows from (31), (32) and the following lemma. ■

Lemma 16 *Let $\ell_1, \ldots, \ell_k$ be a collection of linear forms in n variables $z_1, \ldots, z_n$. Then for any coefficients $c_1, \ldots, c_k$, the function*

$$\frac{\sum_k c_k e^{\ell_k(z)}}{\sum_k e^{\ell_k(z)}}$$

is bounded.

Proof Since $e^{\ell_j} \leq \sum_k e^{\ell_k}$ for any j, we know that

$$0 \leq \frac{e^{\ell_j(z)}}{\sum_k e^{\ell_k(z)}} \leq 1$$

for any j. Thus, the lemma follows from the triangle inequality. ■