THE UNIVERSITY OF CHICAGO

TROPICAL ALGEBRA AND ALGEBRAIC TOPOLOGY OF DEEP NEURAL NETWORKS

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCES
IN CANDIDACY FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

DEPARTMENT OF STATISTICS

BY
GREGORY NAISAT

CHICAGO, ILLINOIS
JUNE 2020

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to thank my academic adviser Professor Lek-Heng Lim for his immense support and guidance throughout this five years journey. To thank him for all the small and large things that he helped me with, for opening so many doors for me, and that he was always there when I needed help and guidance.

I would also like to specially thank my defense committee members, the statistics department and the university staff whose help and assistance helped me during my degree.

Finally, many thanks to my family, without their support this journey would not be possible.

ABSTRACT

We present a theoretical and empirical study of feedforward neural networks using tropical algebra and topological data analysis. We show how examining neural networks through the lens of these two disciplines yields insights into their operation and efficacy. This work is divided into two parts: *Topology of Deep Neural Networks* and *Tropical Geometry of Deep Neural Networks*. Each part is respectively a self-contained analysis of deep neural networks from the perspectives of algebraic topology and of tropical algebra. There is a noteworthy connection between the two parts: One of our conclusions from the first part is that it is important to bound the topological complexity of decision boundaries; the work in the second part, among other things, provides such a bound in terms of the number of linear regions.

The first part of this thesis is joint work with Liwen Zhang and Lek-Heng Lim and has appeared as [89]. The second part is joint work with Andrey Zhitnikov and Lek-Heng Lim and has been submitted for publication.

CHAPTER 1

INTRODUCTION

Deep neural networks have recently received much limelight for their enormous success in a variety of applications across many different areas of artificial intelligence, computer vision, speech recognition, and natural language processing [48, 38, 46, 4, 42]. Nevertheless, it is also well-known that our theoretical understanding of their efficacy remains incomplete.

There have been several attempts to analyze deep neural networks from different perspectives. Notably, earlier studies have suggested that a deep architecture could use parameters more efficiently and requires exponentially fewer parameters to express certain families of functions than a shallow architecture [20, 7, 63, 25, 76, 2]. Recent work [88] showed that several successful neural networks possess a high representation power and can easily shatter random data. However, they also generalize well to data unseen during training stage, suggesting that such networks may have some implicit regularization. Traditional measures of complexity such as VC-dimension and Rademacher complexity fail to explain this phenomenon. Understanding this implicit regularization that begets the generalization power of deep neural networks remains a challenge.

The goal of our work is to approach the study of neural networks from two nonconventional angles. We investigate their somewhat mysterious inner workings and explain their seemingly unreasonable effectiveness from the perspectives of *algebraic topology* and *tropical geometry*. Apart from shedding light on a problem of tremendous current interests, our study also provides a set of hitherto unexploited tools to analyze existing neural networks and to design new ones. Although not part of this thesis, some of these aforementioned topological insights have led to a commercial application [53]; this is based on work that we did [66] but the patent is held by the SAS Institute.

In Chapter 2, we study how the topology of a data set $M = M_a \cup M_b \subseteq \mathbb{R}^d$, representing two classes of objects a and b in a binary classification problem, changes as it passes through the layers of a well-trained neural network, i.e., one with perfect accuracy on its training

set and a near-zero generalization error ($\approx 0.01\%$). The goal is to shed light on two well-known mysteries in deep neural networks: (i) a nonsmooth activation function like ReLU outperforms a smooth one like hyperbolic tangent; (ii) successful neural network architectures rely on having many layers, despite the fact that a shallow network is able to approximate any function arbitrary well. We performed extensive experiments on the persistent homology of a wide range of point cloud data sets, both real and simulated. The results consistently demonstrate the following: (1) Neural networks operate by changing topology, transforming a topologically complicated data set into a topologically simple one as it passes through the layers. No matter how complicated the topology of M we begin with, when passed through a well-trained neural network $f : \mathbb{R}^d \to \mathbb{R}^p$, there is invariably a vast reduction in the Betti numbers of both components M_a and M_b; in fact they nearly always reduce to their lowest possible values: $\beta_k\big(f(M_i)\big) = 0$ for $k \geq 1$ and $\beta_0\big(f(M_i)\big) = 1$, $i = a, b$. Furthermore, (2) the reduction in Betti numbers is significantly faster for ReLU activation compared to hyperbolic tangent activation as the former defines nonhomeomorphic maps that change topology, whereas the latter defines homeomorphic maps that preserve topology. Lastly, (3) shallow and deep networks transform the same data set somewhat differently — a shallow network operates mainly through changing geometry and changes topology only in its final layers, a deep one spreads topological changes more evenly across all layers.

In Chapter 3, we establish, for the first time, explicit connections between feedforward neural networks with ReLU activation and tropical geometry — we show that the family of such neural networks is equivalent to the family of tropical rational maps. Among other things, we deduce that feedforward ReLU neural networks with one hidden layer can be characterized by zonotopes, which serve as building blocks for deeper networks; we relate decision boundaries of such neural networks to tropical hypersurfaces, a major object of study in tropical geometry; and we prove that linear regions of such neural networks correspond to vertices of polytopes associated with tropical rational functions. The number of linear regions in the graph of neural network is one of the most widely adopted measures the neural

network's expressivity. Building on our result that a feedforward ReLU neural network is a tropical rational function, i.e., a tropical ratio of two tropical polynomials, we show that the linear regions are closely related to the tropical zeros of these two tropical polynomials. This in turn allows us to use tools from tropical geometry to quantify the number of linear regions as a function of the parameters of a neural network. An insight from our tropical formulation is that a deeper network is exponentially more expressive than a shallow network.

One unifying thread that runs through both Chapters 2 and 3 concerns the decision boundaries of neural networks in a binary classification problem. In Chapter 2, we show that it is important to study the topological complexity, i.e., the sum of all Betti numbers, of the decision boundary as it passes through the layers of a well-trained neural network. This is evidently a challenging problem and it is not clear if the topological techniques in Chapter 2 apply in this situation — we used these techniques to study the topology of the data sets but not the decision boundary. Our tropical description of neural networks in Chapter 2 on the other hand provides combinatorial means to bound the topological complexity of decision boundaries in terms of the number of linear regions.

The first part of this thesis is joint work with Liwen Zhang and Lek-Heng Lim and has appeared as [89]. The second part is joint work with Andrey Zhitnikov and Lek-Heng Lim and has been submitted for publication.

CHAPTER 2

TOPOLOGY OF DEEP NEURAL NETWORKS

2.1 Introduction

A key insight of topological data analysis is that "*data has shape*" [13, 14]. That data sets often have nontrivial topologies, which may be exploited in their analysis, is now a widely accepted principle with abundant examples across multiple disciplines: dynamical systems [43], medicine [50, 69], genomics [74], neuroscience [29], time series [75], etc. An early striking example came from computer vision, where [15] showed that naturally occurring image patches reside on a low-dimensional manifold that has the topology of a Klein bottle.

We will study how modern deep neural networks transform topologies of data sets, with the goal of shedding light on their breathtaking yet somewhat mysterious effectiveness. Indeed, we seek to show that neural networks operate by changing the topology (i.e., shape) of data. The relative efficacy of ReLU activation over traditional sigmoidal activations can be explained by the different speeds with which they change topology — a ReLU-activated neural network (which is not a homeomorphism) is able to sharply reduce Betti numbers but not a sigmoidal-activated one (which is a homeomorphism). Also, the higher the topological complexity of the data, the greater the depth of the network required to reduce it, explaining the need to have an adequate number of layers.

We would like to point out that the idea of changing the topology of space to facilitate a machine learning goal is not as esoteric as one might imagine. For example, it is implicit in kernel methods [81] — a data set with two components inseparable by a hyperplane is embedded in a higher-dimensional space where the embedded images of the components are separable by a hyperplane. Note that *dimension* is a topological invariant, changing dimension is changing topology. We will see that a ReLU-activated neural network with many layers effects topological changes primarily through changing Betti numbers, another topological invariant.

Our study differs from current approaches in two important ways. Many existing studies either (i) analyze neural networks in an asymptotic or extreme regime, where the number of neurons in each layer or the number of layers becomes unbounded or infinite, leading to conclusions that really pertain to neural networks of somewhat unrealistic architectures; or (ii) they focus on what a neural network does to a single object, e.g., an image of a cat, and examine how that object changes as it passes through the layers. While we do not dispute the value of such approaches, we would like to contrast them with ours: We study what a neural network with a realistic architecture does to an entire class of objects. It is common to find expositions (especially in the mass media) of deep neural networks that purport to show their workings by showing how an image of a cat is transformed as it passes through the layers. We think this is misguided — one should be looking at the entire manifold of cat images, not a single point on that manifold (i.e., a single cat image). This is the approach we undertake in this thesis.

Figure 2.1 illustrates what we mean by 'changing topology'. The two subfigures are caricatures of real results (see Figures 2.2, 2.11, 2.12, 2.13, for the true versions obtained via actual Betti numbers computations and projections to the principal components.)

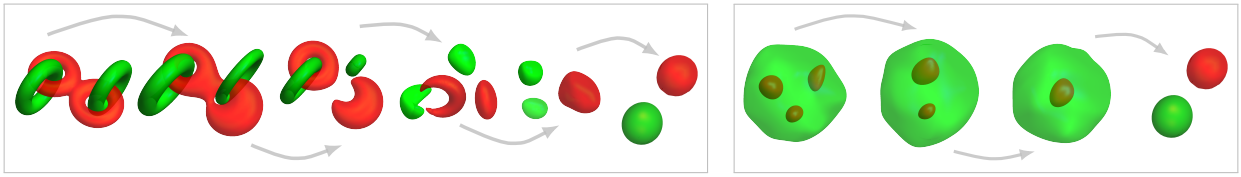


Figure 2.1: Progression of Betti numbers $\beta(X) = (\beta_0(X), \beta_1(X), \beta_2(X))$. *Left*: β(red): $(1,2,0) \to (1,2,0) \to (2,1,0) \to (2,0,0) \to (1,0,0) \to (1,0,0)$; β(green): $(2,2,0) \to (2,2,0) \to (2,1,0) \to (2,0,0) \to (2,0,0) \to (1,0,0)$. *Right*: β(red): $(3,0,0) \to (2,0,0) \to (1,0,0) \to (1,0,0)$; β(green): $(1,0,3) \to (1,0,2) \to (1,0,1) \to (1,0,0)$.

In both subfigures, we begin with a three-dimensional manifold $M = M_a \cup M_b$, comprising two disjoint submanifolds M_a (green) and M_b (red) entangled in a topologically nontrivial manner, and track its progressive transformation into a topologically simple manifold comprising a green ball and a red ball. In the left box, M is initially the union of the two green solid tori M_a, interlocked with the red solid figure-eight M_b. In the right box, M is initially

a union of M_a, the green solid ball with three voids inside, and M_b, three red balls each placed within one of the three voids of M_a. The topological simplification in both boxes are achieved via a reduction in the Betti numbers of both M_a and M_b so that eventually we have $\beta_k(M_i) = 0$ for $k \geq 1$ and $\beta_0(M_i) = 1$, $i = a, b$. Our main goal is to provide (what we hope is) incontrovertible evidence that this picture captures the actual workings of a *well-trained*[1] neural network in a binary classification problem where M_a and M_b represent the two classes.

In reality, the manifold $M = M_a \cup M_b$ would have to be replaced by a point cloud data set, i.e., a finite set of points sampled with noise from M. The notion of *persistent homology* allows us to give meaning to the topology of point cloud data and estimate the Betti numbers of its underlying manifold.

Key findings. This work is primarily an empirical study — we performed more than 10,000 experiments on real and simulated data sets of varying topological complexities and have made our codes available for reader's further experimentations.[2] We summarize our most salient observations and discuss their implications:

(i) For a fixed data set and fixed network architecture, topological changes effected by a well-trained network are robust across different training instances and follow a similar profile.

(ii) Using smooth activations like hyperbolic tangent results in a slow down of topological simplification compared to nonsmooth activations like ReLU or Leaky ReLU.

(iii) The initial layers mostly induce only *geometric* changes, it is in deeper layers that *topological* changes take place. Moreover, as we reduce network depth, the burden of producing topological simplification is not spread uniformly across layers but remains

1. One with near zero generalization error.

2. `https://github.com/topnn/topnn_framework`.

concentrated in the last layers. The last layers see a greater reduction in topological complexity than the initial layers.

Observation (ii) provides a plausible answer to a widely asked question [65, 55, 30]: What makes rectified activations such as ReLU and its variants perform better than smooth sigmoidal activations? We posit that it is not a matter of smooth versus nonsmooth but that a neural network with sigmoid activation is a *homeomorphic* map that preserves topology whereas one with ReLU activation is a *nonhomeomorphic* map that can change topology. It is much harder to change topology with homeomorphisms; in fact, mathematically it is impossible; but maps like the hyperbolic tangent achieve it in practice via rounding errors. Note that in IEEE finite-precision arithmetic, the hyperbolic tangent is effectively a piecewise linear step function:

$$\tanh_\delta(x) = \begin{cases} +1 & \text{if } \mathrm{fl}(\tanh(x)) > 1-\delta, \\ \mathrm{fl}(\tanh(x)) & \text{if } -1+\delta \leq \mathrm{fl}(\tanh(x)) \leq 1-\delta, \\ -1 & \text{if } \mathrm{fl}(\tanh(x)) < -1+\delta, \end{cases}$$

where $\mathrm{fl}(x)$ denotes floating point representation of $x \in \mathbb{R}$, and $\delta > 0$ is the *unit roundoff*, i.e., $\delta = \epsilon/2$ with $\epsilon = \inf\{x > 0 : \mathrm{fl}(1+x) \neq 1\}$ the *machine epsilon* [72]. Applied coordinatewise to a vector, $\tanh : \mathbb{R}^n \to (-1,1)^n$ is a homeomorphism of $\mathbb{R}^n$ to $(-1,1)^n$ and necessarily preserves topology; but $\tanh_\delta : \mathbb{R}^n \to [-1,1]^n$ is not a homeomorphism and thus has the ability to change topology. We also observe that lowering the floating point precision increases the value of δ (e.g., for double precision $\delta = 2^{-54}$, for half precision[3] $\delta = 2^{-9}$), which has the effect of coarsening $\tanh_\delta$, making it even further from a homeomorphism and thus more effective at changing topology. We suspect that this may account for the paradoxical superior performance of lower precision arithmetic in deep neural networks [18, 34, 40].

3. Assuming the BFloat16 floating-point format used in TensorFlow.

The ReLU activation, on the other hand, is far from a homeomorphism (for starters, it is not injective) even in exact arithmetic. Indeed, if changing topology is the goal, then a composition of an affine map with ReLU activation, $\nu : \mathbb{R}^n \to \mathbb{R}^n$, $x \mapsto \max(Ax + b, 0)$, is a quintessential tool for achieving it — any topologically complicated part of $M \subseteq \mathbb{R}^n$ can be affinely moved outside the nonnegative orthant and collapsed to a single point by the rectifier. We see this in action in Figure 2.2, which unlike Figure 2.1, is a genuine example of a ReLU neural network trained to perfect accuracy on a two-dimensional manifold data set, where M_a comprises five red disks in a square M and $M_b = M \setminus M_a$ is the remaining green portion with the five disks removed. The 'folding' transformations in Figure 2.2 clearly require many-to-one maps and can never be achieved by any homeomorphism.

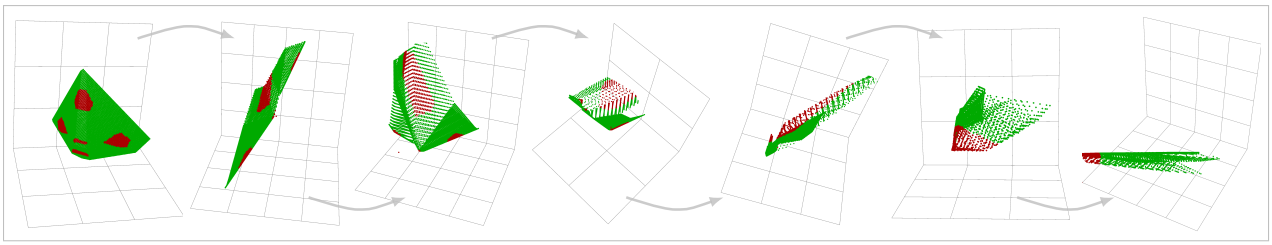


Figure 2.2: How the data set is transformed after passing through layers $2, 3, \ldots, 8$ of a ReLU network with three neurons in each layer, well-trained to detect five disks in a square. β(red): $(5, 0) \to (4, 0) \to (4, 0) \to (4, 0) \to (2, 0) \to (1, 0) \to (1, 0)$.

The effectiveness of ReLU activation over sigmoidal activations is often attributed to the former's avoidance of the vanishing/exploding gradient problem. Our results in Section 2.7 indicate that this does not give the full explanation. Leaky ReLU and ReLU both avoid vanishing/exploding gradients, yet they transform data sets in markedly different manners — for one, ReLU reduces topology faster than Leaky ReLU. The sharpness of the gradients is clearly not what matters most; on the other hand, the topological perspective perfectly explains why.

Observation (iii) addresses another perennial paradox [46, 24, 82]: Why does a neural network with more layers work better, despite the well-known universal approximation property that any function can be approximated arbitrarily well by a two-layer one? We

posit that the traditional approximation-theoretic view of neural networks is insufficient here; instead, the proper perspective is that of a topologically complicated input getting progressively simplified as it passes through the layers of a neural network. Observation (iii) accounts for the role of the additional layers — topological changes are minor in the first few layers and occur mainly in the later layers, thus a complicated data set requires many more layers to simplify.

We emphasize that our goal is to explain the mechanics of what happens from one layer to the next, and to see what role each attribute of the network's architecture — depth, width, activation — serves in changing topology. Note that we are not merely stating that a neural network is a blackbox that collapses each class to a component but how that is achieved, i.e., what goes on inside the blackbox.

Relations with and departures from prior works. As in topological data analysis, we make use of persistent homology and quantify topological complexity in terms of Betti numbers; we track how these numbers change as a point cloud data set passes through the layers of a neural network. But that is the full extent of any similarity with topological data analysis. In fact, from our perspective, topological data analysis and neural networks have opposite goals — the former is largely concerned with *reading* the shape of data, whereas the latter is about *writing* the shape of data; not unlike the relation between computer vision and computer graphics, wherein one is interested the inverse problems of the other. Incidentally, this shows that a well-trained neural network applied in reverse can be used as a tool for labeling components of a complex data set and their interrelation, serving a role similar to *mapper* [83] in topological data analysis. This idea has been explored in [73, 66].

To the best of our knowledge, our approach towards elucidating the inner workings of a neural network by studying how the topology, as quantified by persistent Betti numbers, of a point cloud data set changes as it passes through the layers has never been done before. The key conclusion of these studies, namely, that the role of a neural network is primarily

as a topology-changing map, is also novel as far as we know. Nevertheless, we would like to acknowledge a Google Brain blog post [71] that inspired our work — it speculated on how neural networks may act as homeomorphisms that distort *geometry*, but stopped short of making the leap to topology-changing maps.

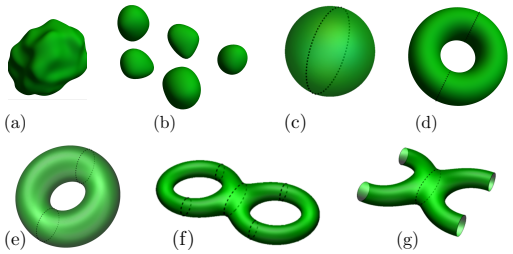
There are other works that employ Betti numbers in the analysis of neural networks. [8] did a purely theoretical study of upper bounds on the topological complexity (i.e., sum of Betti numbers) of the decision boundaries of neural networks with smooth sigmoidal activations; [78] did a similar study with a different measure of topological complexity. [35] studied the empirical relation between the topological complexity of a data set and the minimal network capacity required to classify it. [79] used persistent homology to monitor changes in the weights of neural network during training and proposed an early stopping criteria based on persistent homology.

Outline. In Section 2.2 we introduce, in an informal way, the main topological concepts used throughout this chapter of the thesis. This is supplemented by a more careful and detailed treatment in Section 2.3, which provides a self-contained exposition of simplicial homology and persistent homology tailored to our needs. Section 2.4 contains a precise formulation of the problem we study, specifies what is tracked empirically, and addresses some caveats. Section 2.6 introduces our methodology for tracking topological changes and implementation details. We present the results from our empirical studies with discussions in Section 2.7 and Section 2.8 and conclude this part of the thesis in Section 2.9.

2.2 Quantifying topology

In this part of the thesis, we rely entirely on *Betti numbers* $\beta_k(M)$ to quantify topology as they are the simplest topological invariants that capture the shape of a space $M \subseteq \mathbb{R}^d$, have intuitive interpretations, and are readily computable within the framework of persistent homology for a point cloud data set sampled from M. The zeroth Betti number, $\beta_0(M)$,

counts the number of connected components in M; the kth Betti number, $\beta_k(M)$, $k \geq 1$, is informally the number of k-dimensional holes in M. In particular, $\beta_k(M) = 0$ when $k > d$ as there is no $(d+1)$-dimensional holes in d-dimensional space. So for $M \subseteq \mathbb{R}^d$, we write $\beta(M) \coloneqq (\beta_0(M), \beta_1(M), \ldots, \beta_d(M))$ — these numbers capture the shape or topology of M, as one can surmise from Figure 2.3. So whenever we refer to 'topology' in this thesis, we implicitly mean $\beta(M)$.



	Manifold $M \subseteq \mathbb{R}^3$	$\beta(M)$
(a)	Single contractible manifold	$(1,0,0)$
(b)	Five contractible manifolds	$(5,0,0)$
(c)	Sphere	$(1,0,1)$
(d)	Solid torus (filled)	$(1,1,0)$
(e)	Surface of torus (hollow)	$(1,2,1)$
(f)	Genus two surface (hollow)	$(1,4,1)$
(g)	Torso surface (hollow)	$(1,3,0)$

Figure 2.3: Manifolds in $\mathbb{R}^3$ and their Betti numbers.

If M has no holes and can be continuously (i.e., without tearing) deformed to a point, then $\beta_0(M) = 1$ and $\beta_k(M) = 0$ for all $k \geq 1$; such a space is called *contractible*. The simplest noncontractible space is a circle $S^1 \subseteq \mathbb{R}^2$, which has a single connected component and a single one-dimensional hole, so $\beta_0(S^1) = 1 = \beta_1(S^1)$ and $\beta_k(S^1) = 0$ for all $k \geq 2$. Figure 2.3 has a few more examples.

Intuitively, the more holes a space has, the more complex its topology. In other words, the larger the numbers in $\beta(M)$, the more complicated the topology of M. As such, we define its *topological complexity* by

$$\omega(M) \coloneqq \beta_0(M) + \beta_1(M) + \cdots + \beta_d(M). \tag{2.1}$$

While not as well-known as the *Euler characteristic* (which is an alternating signed sum of the Betti numbers), the topological complexity is also a classical notion in topology, appearing most notably in Morse theory [59]; one of its best known result is that the topological complexity of M gives a lower bound for the number of stationary points of a function

$f : M \to \mathbb{R}$ with nondegenerate Hessians. It also appears in many other contexts [60, 5], including neural networks. We highlight in particular the work of [8] that we mentioned earlier, which studies the topological complexity of the decision boundary of neural networks with activations that are Pfaffian functions [87, 26]. These include sigmoidal activations but not the ReLU nor leaky ReLU activations studied in this thesis. For piecewise linear activations like ReLU and leaky ReLU, the most appropriate theoretical upper bounds for topological complexity of decision boundaries are likely given by the number of linear regions [62, 89].

The goal of our work is different, we are interested not in the shape of the decision boundary of an l-layer neural network $\nu_l : \mathbb{R}^d \to \mathbb{R}^p$ but in the shapes of the input $M \subseteq \mathbb{R}^d$, output $\nu_l(M) \subseteq \mathbb{R}^q$, and all its intermediate layers $\nu_k(M)$, $k = 1, \ldots, l-1$. By so doing, we may observe how the shape of M is transformed as it passes through the layers of a well-trained neural network, thereby elucidating its workings. In other words, we would like to track the Betti numbers

$$\beta(M) \to \beta(\nu_1(M)) \to \beta(\nu_2(M)) \to \cdots \to \beta(\nu_{l-1}(M)) \to \beta(\nu_l(M)).$$

To do this in reality, we will have to estimate $\beta(M)$ from a point cloud data set, i.e., a finite set of points $T \subseteq M$ sampled from M, possibly with noise. The next section will describe the procedure to do this via persistent homology, which is by now a standard tool in topological data analysis. Readers who do not want to be bothered with the details just need to know that one may reliably estimate $\beta(M)$ by sampling points from M; those who like to know the details may consult the next section. The main idea is that the Betti numbers of M may be estimated by constructing a simplicial complex from T in one of several ways that depend on a 'persistent parameter', and then using simplicial homology to compute the Betti numbers of this simplicial complex. Roughly speaking, the 'persistent parameter' allows one to pick the right scale at which the point cloud T should be sampled so as to give a faithful

estimation of $\beta(M)$. Henceforth whenever we speak of $\beta(M)$, we mean the Betti numbers estimated in this fashion.

For simplicity of the preceding discussion, we have used M as a placeholder for any manifold. Take say a handwritten digits classification problem (see Section 2.8), then $M = M_0 \cup M_1 \cup \cdots \cup M_9$ has ten components, with M_i the manifold of all possible handwritten digits $i \in \{0, 1, \ldots, 9\}$. Here we are not interested in $\beta(M)$ per se but in $\beta(\nu_k(M_i))$ for all $k = 0, 1, \ldots, l$ and $i = 0, 1, \ldots, 9$ — so that we may see how each component is transformed as M passes through the layers, i.e., we will need to sample points from each of $\nu_k(M_i)$ to estimate its Betti numbers, for each component i and at each layer k.

2.3 Algebraic topology and persistent homology background

This section may be skipped by readers who are already familiar with persistent homology or are willing to take on faith what we wrote in the last two paragraphs of the last section. Here we will introduce background knowledge in algebraic topology — simplicial complex, homology, simplicial homology — and provide a brief exposition on selected aspects of topological data analysis — Vietoris–Rips complex, persistent homology, practical homology computations — that we need for our purposes.

2.3.1 Simplicial complexes

A k-dimensional *simplex*, or k-simplex, σ in $\mathbb{R}^d$, is the convex hull of $k+1$ affinely independent points $v_0, \ldots, v_k \in \mathbb{R}^d$. A 0-simplex is a point, a 1-simplex is a line segment, a 2-simplex is a triangle, and a 3-simplex is a tetrahedron. A k-simplex is represented by listing the set of its $k + 1$ *vertices* and denoted $\sigma = [v_0, \ldots, v_k]$. The *faces* of a k-simplex are simplices of dimensions 0 to $k - 1$ formed by convex hulls of proper subsets of its *vertex set* $\{v_0, \ldots, v_k\}$. For example, the faces of a line segment/1-simplex are its end points, which are 0-simplices; the faces of a triangle/2-simplex are its three sides, which are 1-simplices, and its three

vertices, which are 0-simplices.

An m-dimensional *geometrical simplicial complex* K in $\mathbb{R}^d$ is a finite collection of simplices in $\mathbb{R}^d$ of dimensions at most m that are (i) glued together along faces, i.e., any intersection between two simplices in K is necessary a face of both of them; and (ii) include all faces of all its simplices, e.g., if the simplex $\sigma_1 = [v_0, v_1, v_2]$ is in K, then the simplices $[v_0, v_1], [v_1, v_2]$, $[v_0, v_2], [v_0], [v_1], [v_2]$ must all also belong to K. Behind each geometrical simplicial complex is an *abstract simplicial complex* — a list of simplices $K = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ with the property that if $\tau \subseteq \sigma \in K$, then $\tau \in K$. This combinatorial description of an abstract simplicial complex is exactly how we describe a graph, i.e., 1-dimensional simplicial complex, as an abstract collection of edges, i.e., 1-simplices, comprising pairs of vertices. Conversely, any abstract simplicial complex can be realized geometrically as a geometrical simplicial complex in $\mathbb{R}^d$ like in Figure 2.4, an example of a 3-dimensional simplicial complex in $\mathbb{R}^3$. The abstract description of a simplicial complex allows us to treat its simplices as elements in a vector space, a key to define simplicial homology, as we will see in Section 2.3.3.

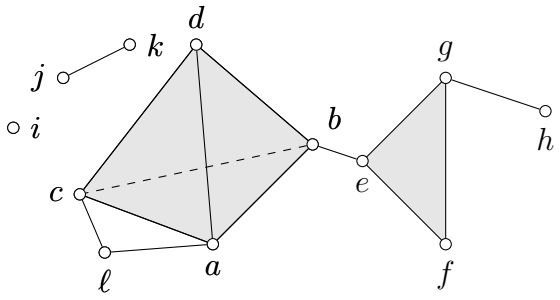


Figure 2.4: The picture shows a *geometrical simplicial complex* in $\mathbb{R}^3$. It is a geometrical realization of an *abstract simplicial complex* $K = \{[a, b, c, d], \ldots, [\ell]\}$ comprising 32 simplices: a single 3-simplex $[a, b, c, d]$, five 2-simplices such as $[a, c, d]$ and $[e, f, g]$, eighteen 1-simplices such as $[e, b]$ and $[g, h]$, fourteen 0-simplices $[a], \ldots, [\ell]$. Note that in the geometrical simplicial complex, the simplices intersect along faces.

2.3.2 Homology and Betti numbers

Homology is an abstract way to encode the topology of a space by means of a chain of vector spaces and linear maps. We refer readers to [51] for an elementary treatment requiring

nothing more than linear algebra and graph theory. Here we will give an even simpler treatment restricted to $\mathbb{F}_2 = \{0, 1\}$, the field of two elements with arithmetic performed modulo 2, which is enough for our analysis.

Let $C_0, C_1, \ldots, C_d$ be vector spaces over $\mathbb{F}_2$. Let $\partial_k : C_k \to C_{k-1}$ be linear maps called *boundary operators* that satisfy the condition that "a boundary of a boundary is trivial," i.e.,

$$\partial_k \circ \partial_{k+1} = 0 \tag{2.2}$$

for all $k = 0, \ldots, d$. A *chain complex* refers to the sequence

$$0 \xrightarrow{\partial_{d+1}} C_d \xrightarrow{\partial_d} C_{d-1} \xrightarrow{\partial_{d-1}} \cdots \xrightarrow{\partial_{k+1}} C_k \xrightarrow{\partial_k} C_{k-1} \xrightarrow{\partial_{k-1}} \cdots \xrightarrow{\partial_2} C_1 \xrightarrow{\partial_1} C_0 \xrightarrow{\partial_0} 0,$$

where we set $C_{d+1} = C_{-1} = 0$, the trivial subspace. The elements in the image of ∂_k are called *boundaries* and elements in the kernel of ∂_{k-1} are called *cycles*. Clearly $\ker(\partial_k)$ and $\operatorname{im}(\partial_{k+1})$ are both subspaces of C_k and by (2.2),

$$B_k \coloneqq \operatorname{im}(\partial_{k+1}) \subseteq \ker(\partial_k) =: Z_k.$$

We may form the quotient vector space

$$H_k \coloneqq Z_k / B_k = \ker(\partial_k) / \operatorname{im}(\partial_{k+1}), \qquad k = 0, 1, \ldots, d,$$

and we will call it the kth *homology group* — the 'group' here refers to the structure of H_k as an abelian group under addition. The elements of H_k are called *homology classes*; note that these are cosets or equivalence classes of the form

$$[z] = z + B_k = \{z + b \in Z_k : b \in B_k\}. \tag{2.3}$$

In particular $[z+b]=[z]$ for any $b \in B_k$. The dimension of H_k as a vector space is denoted

$$\beta_k \coloneqq \dim(H_k), \qquad k = 0, 1, \ldots, d.$$

This has special topological significance when H_k is the homology group of a topological space like a simplicial complex K and is called the kth *Betti number* of K. Intuitively β_k counts the number of k-dimensional holes in K. Note that by definition, H_k has a *basis* comprising homology classes $[z_1], \ldots, [z_{\beta_k}]$ for some $z_1, \ldots, z_{\beta_k} \in Z_k \subseteq C_k$.

2.3.3 Simplicial homology

We present a very simple exposition of simplicial homology tailored to our purposes. The simplification stems partly from our working over a field of two elements $\mathbb{F}_2 \coloneqq \{0, 1\}$. In particular $-1 = +1$ and we do not need to concern with signs.

Given an abstract simplicial complex K, we define an $\mathbb{F}_2$-vector space $C_k(K)$ in the following way: Let $K^{(k)} = \{\sigma_1, \ldots, \sigma_m\}$ be the set of all k-dimensional simplices in K. Then an element of $C_k(K)$ is a formal linear combination:

$$\sum_{j=1}^{m} n_j \sigma_j, \qquad n_j = 0 \text{ or } 1.$$

In other words, $C_k(K)$ is a vector space over $\mathbb{F}_2$ with $K^{(k)}$ as a basis.

The boundary operators $\partial_k : C_k(K) \to C_{k-1}(K)$ are defined on a k-simplex $\sigma = [v_0, \ldots, v_k]$ by

$$\partial_k \sigma \coloneqq \sum_{j=0}^{k} [v_0, \ldots, \hat{v}_j, \ldots, v_k], \tag{2.4}$$

where $\hat{v}_j$ indicates that v_j is omitted from σ, and extended linearly to all of $C_k(K)$, i.e.,

$$\partial_k \left(\sum_{j=1}^{m} n_j \sigma_j \right) \coloneqq \sum_{j=1}^{m} n_j \partial_k \sigma_j.$$

For example, $\partial_1[a,b] = a+b$, $\partial_2[a,b,c] = [a,b]+[b,c]+[c,a]$, $\partial_2([a,b,c]+[d,e,f]) = \partial_2[a,b,c]+\partial_2[d,e,f]$.

Working over $\mathbb{F}_2$ simplifies calculations enormously. In particular, it is easy to check that $\partial_k \circ \partial_{k+1} = 0$ for all $k = 0,\dots,d$, as each $(k-2)$-simplex appears twice in the resulting sum and $2 = 0$ in $\mathbb{F}_2$. Thus (2.2) holds and $\partial_k : C_k(K) \to C_{k-1}(K)$, $k = 0,\dots,d+1$ form a chain complex. The kth homology of the simplicial complex K is then $H_k(K) = \ker(\partial_k)/\operatorname{im}(\partial_{k+1})$ with ∂_k as defined in (2.4). Working over $\mathbb{F}_2$ also guarantees that $H_k(K)$ takes the simple form $\mathbb{F}_2^{\beta_k}$ where β_k is the kth Betti number, i.e.,

$$\beta_k(K) = \dim\big(H_k(K)\big) = \operatorname{nullity}(\partial_k) - \operatorname{rank}(\partial_{k+1}), \tag{2.5}$$

for $k = 0,1,\dots,d$. Let $m_k \coloneqq |K^{(k)}|$, the number of k-simplices in K. To compute $\beta_k(K)$, note that with the k-simplices in $K^{(k)}$ as basis, ∂_k is an $m_{k-1} \times m_k$ matrix with entries in $\mathbb{F}_2$ and the problem in (2.5) reduces to linear algebra over $\{0,1\}$ with modulo 2 arithmetic. While this seems innocuous, the cost of computing Betti numbers becomes prohibitive when the size of the simplicial complex $|K| = m_0+m_1+\cdots+m_d$ is large. The number of simplices in a d-dimensional simplicial complex K is bounded above by

$$|K| \leq \sum_{i=0}^{d} \binom{n}{i+1} \tag{2.6}$$

where n is the size of the vertex set, i.e., $n = m_0$, and the bound is obtained by summing over the maximal number of simplices of each dimension. The cost of computing $\beta(K)$ is $\approx O(|K|^{2.38})$ [84].

We conclude with a discussion of simplicial maps, which we will need in persistent homology. Let K_1 and K_2 be two abstract simplicial complexes. A *simplicial map* is a map defined on their vertex sets $f : K_1^{(0)} \to K_2^{(0)}$ so that for each simplex $\sigma = [v_0,\dots,v_k] \in K_1$, we have that $[f(v_0),\dots,f(v_k)]$ is a simplex in K_2. Such a map induces a map between chain

complexes that we will also denote by f, slightly abusing notation, defined by

$$f : C_k(K_1) \to C_k(K_2), \qquad \sum_{j=1}^{m} n_j \sigma_j \mapsto \sum_{j=1}^{m} n_j f(\sigma_j),$$

that in turn induces a map between homologies

$$H_k(f) : H_k(K_1) \to H_k(K_2), \qquad \left[\sum_{j=1}^{m} n_j \sigma_j\right] \mapsto \left[\sum_{j=1}^{m} n_j f(\sigma_j)\right] \tag{2.7}$$

for all $k = 0, 1, \ldots, d+1$. Recall that $[z] \in H_k$ is a shorthand for homology class (2.3).

The composition of two simplicial maps $f : K_1^{(0)} \to K_2^{(0)}$ and $g : K_2^{(0)} \to K_3^{(0)}$ is also a simplicial map $g \circ f : K_1^{(0)} \to K_3^{(0)}$ and thus induces a map between homologies $H_k(g \circ f) : H_k(K_1) \to H_k(K_3)$ for any $k = 0, 1, \ldots, d+1$. For the type of simplicial complex (Vietoris–Rips) and simplicial maps (inclusions) we consider in this chapter, we have that $H_k(g \circ f) = H_k(g) \circ H_k(f)$, a property known as *functoriality*.

2.3.4 Vietoris–Rips complex

There are several ways to obtain a simplicial complex from a point cloud data set but one stands out for its simplicity and widespread adoption in topological data analysis. Note that a point cloud data set is simply a finite set of n points $X \subseteq \mathbb{R}^d$. We will build an abstract simplicial complex K with vertex set $K^{(0)} = X$.

Let δ be a metric on $\mathbb{R}^d$. The *Vietoris–Rips complex* at scale $\varepsilon \geq 0$ on X is denoted by $\mathrm{VR}_\varepsilon(X)$ and defined to be the simplicial complex whose vertex set is X and whose k-simplices comprise all simplices $[x_0, \ldots, x_k]$ satisfying $\delta(x_i, x_j) \leq 2\varepsilon$ for all $i, j = 0, 1, \ldots, k$. In other words,

$$\mathrm{VR}_\varepsilon(X) \coloneqq \big\{[x_0, \ldots, x_k] : \delta(x_i, x_j) \leq 2\varepsilon,\ x_0, \ldots, x_k \in X,\ k = 0, 1, \ldots, n\big\}.$$

It follows immediately from definition that $\mathrm{VR}_\varepsilon(X)$ is an abstract simplicial complex. Note

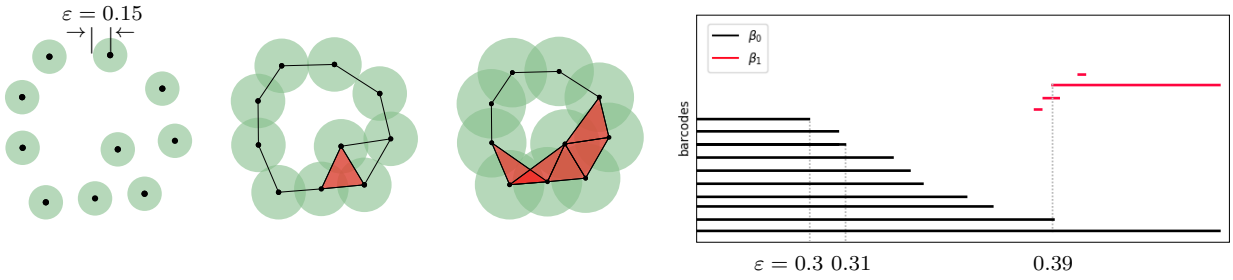


Figure 2.5: *Left*: Vietoris–Rips complex on ten points in $\mathbb{R}^2$ at scales $\varepsilon = 0.15$, 0.4, 0.6. *Right*: Persistence barcodes diagram obtained from filtration of the Vietoris–Rips complex with scale ε varying from 0 to 4. Barcodes show two most prominent topological features of the point cloud, the long black line at the bottom and the long red line near the top, revealing the topology of a circle, i.e., $\beta_0 = \beta_1 = 1$. A 0-homology class dies at times $\varepsilon = 0.3$, 0.31, and 0.39; a 1-homology class is simultaneously born at time $\varepsilon = 0.39$.

that it depends on two things — the scale ε and the choice of metric δ. Figure 2.5 shows an example of Vietoris–Rips complex constructed from a point cloud data set of ten points in $\mathbb{R}^2$ at three different scales $\varepsilon = 0.15$, 0.4, 0.6 and with δ given by the Euclidean norm.

For a point cloud $X \subseteq M \subseteq \mathbb{R}^d$ sampled from a manifold M embedded in $\mathbb{R}^d$, the most appropriate metric δ is the geodesic distance on M and not the Euclidean norm on $\mathbb{R}^d$. This is usually estimated from X using the graph geodesic distance as we will see in Section 2.6.3.

When X is sampled from a manifold $M \subseteq \mathbb{R}^d$, then for a dense enough sample, and at sufficiently small scale, the topology of $\mathrm{VR}_\varepsilon(X)$ recovers the true topology M in an appropriate sense, made precise in the following result in [70]:

Proposition 2.3.1 (Niyogi–Smale–Weinberger)**.** *Let $X = \{x_1, \ldots, x_n\} \subseteq \mathbb{R}^d$ be $(\varepsilon/2)$-dense in a compact Riemannian manifold $M \subseteq \mathbb{R}^d$, i.e., for every $p \in M$, there exists $x \in X$ such that $\|p - x\| < \varepsilon/2$. Let τ be the condition number of M. Then for any $\varepsilon < \sqrt{3\tau/5}$, the union of balls $V = \bigcup_{i=1}^n B_\varepsilon(x_i)$ deformation retracts to M. In particular, the homology of V equals the homology of M.*

Roughly speaking the condition number of a manifold embedded in $\mathbb{R}^d$ encodes its local and global curvature properties but the details are too technical for us to go into here.

2.3.5 Persistent homology

The Vietoris–Rips complex $\mathrm{VR}_\varepsilon(X)$ of a point cloud data set involves a parameter ε. Here we will discuss how this may be determined.

The homology classes, are very sensitive to small changes. For example, punching a small hole in a sphere has little effect on its geometry but has large consequence on its topology — even a very small hole would kill the H_2 homology class, turning a sphere into a topological disk. This also affects the estimation of Betti numbers of a manifold from a subset of sampled point cloud data: there are many scenarios where moving a single point can significantly change the homology estimates. *Persistent homology* [23] addresses this problem by blending geometry and topology. It allows one to reliably estimate the Betti numbers of a manifold from a point cloud data set, and to a large extent avoids the problem of the extreme sensitivity of topology to perturbations. In machine learning lingo, Betti numbers are features associated with the point cloud, and persistent homology enables one to identify the features that are robust to noise.

Informally, the idea of persistent homology is to introduce a geometric scale ε that varies from 0 to ∞ into homology calculations. At a scale of zero, $\mathrm{VR}_0(X) = \{[x] : x \in X\}$ is a collection of 0-dimensional simplices with $\beta_0 = |X|$ and all other Betti numbers zero. In machine learning lingo the simplicial $\mathrm{VR}_0(X)$ 'overfits' the data X, giving us a discrete topological space. As ε increases, more and more distant points come together to form higher and higher-dimensional simplices in $\mathrm{VR}_\varepsilon(X)$ and its topology becomes richer. But as $\varepsilon \to \infty$, eventually all points in X become vertices of a single $|X|$-dimensional simplex, giving us a contractible topological space. So at the extreme ends $\varepsilon = 0$ and $\varepsilon \to \infty$, we have trivial (discrete and indiscrete) topologies and the true answer we seek lies somewhere in between — to obtain a 'right' scale ε_*, we use the so-called *persistence barcodes*. Figure 2.5 shows an example of a persistence barcode diagram. This is the standard output of persistent homology calculations and it provides a summary of the evolution of topology across all scales. Generally speaking, a persistence barcode is an interval $[\varepsilon, \varepsilon']$ where its left-end point

ε is the scale at which the new feature appears or *born*; and its right-end point ε' is the scale at which that feature disappears or *die*. The length of the interval $\varepsilon' - \varepsilon$ is the *persistence* of that feature. Features that are non-robust to perturbations will produce short intervals; conversely, features that persist long enough, i.e., produce long intervals, are thought to be prominent features of the underlying manifold. For our purpose, the feature in question will always be a homology class in kth homology group. The collection of all persistence barcodes over $k = 0, 1, \ldots, d$ then gives us our persistence barcode diagram. If we sample a point cloud satisfying Proposition 2.3.1 from a sphere with a small punctured hole, we expect to see a single prominent interval corresponding to $\beta_2 = 1$, and a short interval corresponding to the small hole. The persistence barcode would allow us to identify a scale ε_* at which all prominent topological features of M are represented, assuming that such a scale exists. In the following we will assume that we are interested in selecting ε_* from a list of finitely many scales $\varepsilon_0 < \varepsilon_1 < \cdots < \varepsilon_m$ but that they could go as fine as we want. For our purpose, the simplicial complex below are taken to be $K_j = \mathrm{VR}(X, \varepsilon_j)$, $j = 0, 1, \ldots, m$, but the following discussion holds more generally.

We provide the details for computing persistence barcodes for homology groups, or *persistent homology* in short. This essentially tracks the evolution of homology in a *filtration* of simplicial complexes, which is chain of a nested simplicial complexes

$$K_0 \subseteq K_1 \subseteq K_2 \subseteq \cdots \subseteq K_m. \tag{2.8}$$

We let $f_j : K_j \hookrightarrow K_{j+1}$, $j = 1, \ldots, m-1$ denote the inclusion maps where each simplex of K_j is sent to the same simplex in $K_j \subseteq K_{j+1}$ and regarded as a simplex in K_{j+1}. As f_j is obviously a simplicial map and induces a linear map $H_k(f_j)$ between the homologies of $H_k(K_j)$ and $H_k(K_{j+1})$ as discussed in Section 2.3.3, composing inclusions $f_{j+p} \circ \cdots \circ f_j$ gives us a linear map between any two complexes in a filtration $H_k(K_j)$ and $H_k(K_{j+p})$, $j = 0, 1, \ldots, m$, $p = 1, 2, \ldots, m-j$. The index j is often referred to as 'time' in this context.

As such, for any $i < j$, one can tell whether two simplices belonging to two different homology classes in $H_k(K_i)$ are mapped to the same homology class in $H_k(K_j)$ — if this happens, one of the homology class is said to have *died* while the other has *persisted* from time i to j. If a homology class in $H_k(K_{j+1})$ is not in the image of $H_k(f_j)$, we say that its homology class is *born* at time $j+1$. The persistence barcodes simply keep track of the birth and death times of the homology classes.

To be completely formal, we have the two-dimensional complex called a *persistent complex* shown in Figure 2.6 with horizontal maps given by boundary maps $\partial_k : C_k(K_j) \to C_{k-1}(K_j)$ and vertical maps given by simplicial maps $f_j : C_k(K_j) \to C_k(K_{j+1})$. Thanks to a well-known structure theorem [90] which guarantees that a barcodes diagram completely describes the structure of a persistent complex in an appropriate sense, we may avoid persistent complexes like Figure 2.6 and work entirely with persistence barcodes diagram like the one on the right of Figure 2.5.

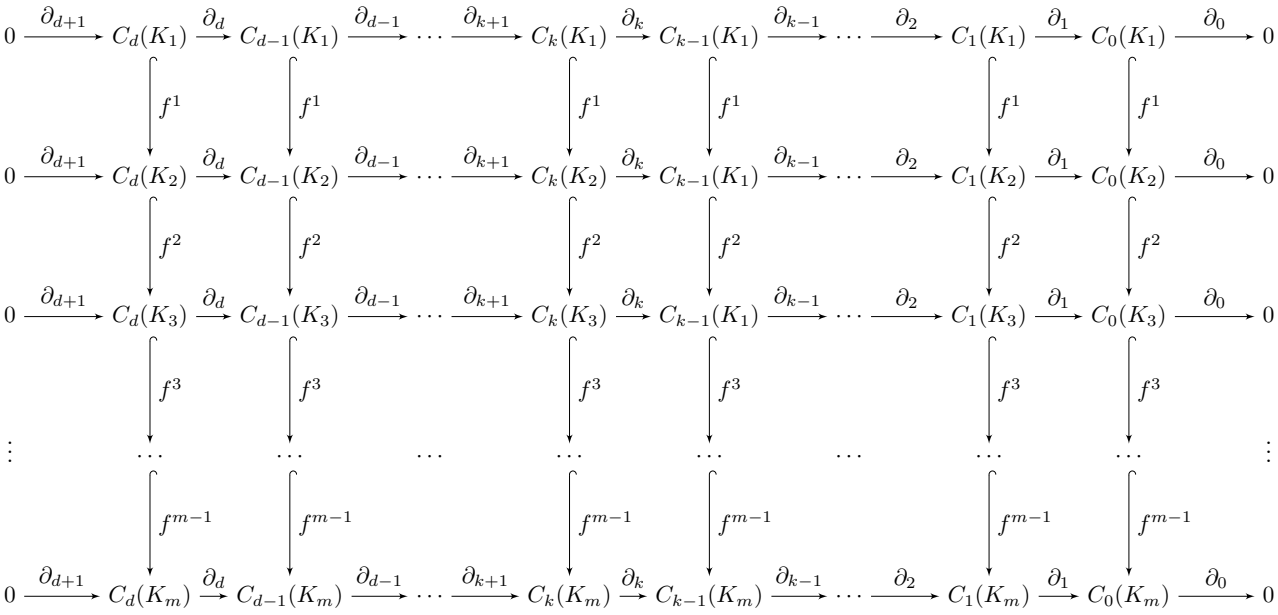


Figure 2.6: Persistence complex of the filtration $K_0 \subseteq K_1 \subseteq \cdots \subseteq K_m$.

Henceforth we let $K_j = \mathrm{VR}(X, \varepsilon_j)$, $j = 0, 1, \ldots, m$, be the Vietoris–Rips complex of our point cloud data at scales $\varepsilon_0 < \varepsilon_1 < \cdots < \varepsilon_m$. An important fact to note is that

persistence barcodes may be computed without having to compute homology at every scale ε_j, or, equivalently, at every time j. To identify the homology classes in $H_k(K_j)$ that persist from time j to time $j+p$, there is no need to compute $H_k(K_{j+1}), \ldots, H_k(K_{j+p})$ individually as one might think. Rather, one considers the *p-persistent kth homology group*

$$H_k^{j,p} = Z_k^j / (B_k^{j+p} \cap Z_k^j),$$

where $p = 1, 2 \ldots, m-j$. This captures the cycles in $C_k(K_j)$ that contribute to homology in $C_k(K_{j+p})$. One may consistently choose a basis for each $H_k^{j,p}$ so that the basis elements are compatible for homologies across $H_k(K_{j+1}), \ldots, H_k(K_{j+p})$ for all possible values of k and p. This allows one to track the persistence of each homology class throughout the filtration (2.8) and thus obtain the persistence barcodes: roughly speaking, with the right basis, we may simultaneously represent the boundary maps on $C_k(K_j)$ as matrices in an column-echelon form and read-off the dimension of $H_k^{j,p}$, known as the *p-persistent kth Betti number* $\beta_k^{j,p}$, from the pivot entries in these matrices. For details we refer readers to [23, 90].

2.3.6 Homology computations in practice

Actual computation of homology from a point cloud data set is more involved than what one might surmise from the description in the last few sections. We will briefly discuss some of the issues involved.

Before we even begin to compute the homology of the point cloud data $X \subseteq M \subseteq \mathbb{R}^d$, we will need to perform a few preprocessing steps, as depicted in Figure 2.7. These steps are standard practice in topological data analysis: (i) We smooth out X and discard outliers to reduce noise. (ii) We then select the scale ε and constructing the corresponding Vietoris–Rips complex $\mathrm{VR}_\varepsilon(X)$. (iii) We simplify $\mathrm{VR}_\varepsilon(X)$ in a way that reduces its size but leaving its topology unchanged. All processing operations that can have an effect on the homology are in steps (i) and (ii), and the homology of the simplicial complex $\mathrm{VR}_\varepsilon(X)$ is assumed to

closely approximate that of the underlying manifold M. The simplification in step (iii) is done to accelerate computations without altering homology.

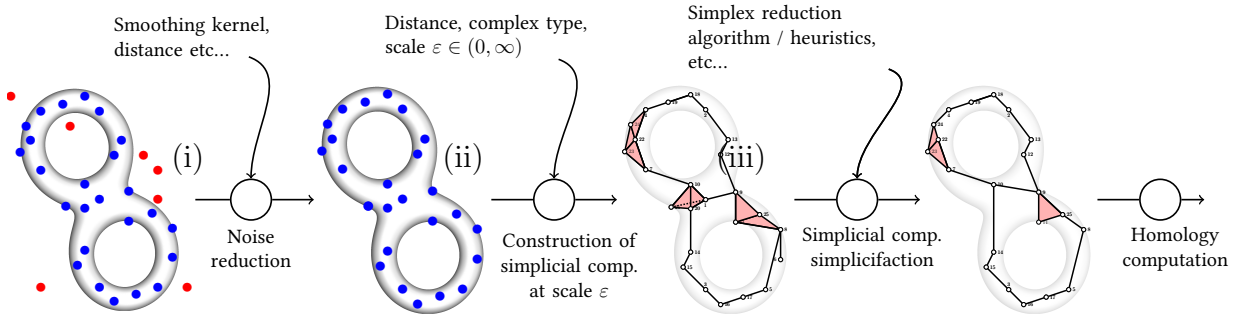


Figure 2.7: Pipeline for computation of homology from a point cloud data.

Note that the size of the final simplicial complex on which we perform homology calculations is the most important factor in computational cost. While increasing the number of points sampled from a manifold, i.e., the size of X, up to the point in Proposition 2.3.1 improves the accuracy of our homology estimates, it also results in a simplicial complex $\mathrm{VR}_\varepsilon(X)$ that is prohibitively large for carrying out computations, as we saw in (2.6). But since we are not concerned with the geometry of the underlaying manifold, only its topology, it is desirable to construct a small simplicial complex with minimal topology distortion. A well-known simplification is the *Witness complex* [19], which gives results of nearly the same quality with a simplicial complex of a smaller size constructed from so-called landmark points. Many other methods have been proposed for this [9, 22, 61], and while we will take advantage of these techniques in our calculations, we will not discuss them here.

The takeaway is that persistence barcodes are considerably more expensive to compute than homology at a single fixed scale ε. Therefore, running full persistent homology in the context of modern deep neural network poses some big challenges: modern deep neural networks operate on very high dimensional big data sets, a setting in which persistent homology cannot be used directly due to computation and memory complexity. This situation is exacerbated by the fact that neural networks are randomly trained (with potentially big variation in the learned decision boundaries), therefore one needs to run many computations

to obtain reliable results. Furthermore, an automated statistical analysis of persistent homology is still an active area of research and often requires additional large computational effort. It seems therefore largely beyond the reach of current technology to try to analyze topology of many of the standard deep learning data sets (such as `SVHN`, `CIFAR-10`, `ImageNet` [68, 45, 21]). We will return to this point later when we introduce our methodology for monitoring topology transformations in a neural network. In particular, we will see in Section 2.6.3 that our experiments are designed in such a way that although we will compute homology at every layer, we only need to compute persistence barcodes once, before the data set is passed through the layers.

2.4 Overview of problem and methodology

We will use *binary classification*, the most basic and fundamental problem in supervised learning, as our platform for studying how neural networks change topology. More precisely, we seek to classify two different probability distributions supported on two disjoint manifolds $M_a,\ M_b \subseteq \mathbb{R}^d$. The distance $\inf\{\|x-y\| : x \in M_a,\ y \in M_b\}$ can be arbitrarily small but not zero. So there exists an ideal classifier with zero prediction error. Here and henceforth, $\|\cdot\|$ will denote the Euclidean norm in $\mathbb{R}^d$.

We sample a large but finite set of points $T \subseteq M_a \cup M_b$ uniformly and densely, so that the Betti numbers of M_a and M_b can be faithfully obtained from the point cloud data sets $T \cap M_a$ and $T \cap M_b$ as described in Section 2.3. Our training set is a labeled point cloud data set, i.e., $x \in T$ is labeled to indicate whether $x \in M_a$ or M_b. We will use $T_a \coloneqq T \cap M_a$ and $T_b \coloneqq T \cap M_b$, or rather, their Vietoris–Rips complex as described in Section 2.3.4, as finite proxies for M_a and M_b.

Our feedforward neural network $\nu : \mathbb{R}^d \to [0,1]$ is given by the usual composition

$$\nu = s \circ f_l \circ f_{l-1} \circ \cdots \circ f_2 \circ f_1, \tag{2.9}$$

where each *layer* of the network $f_j : \mathbb{R}^{n_j} \to \mathbb{R}^{n_{j+1}}$, $j = 1, \dots, l$, is the composition of an affine map $\rho_j : \mathbb{R}^{n_j} \to \mathbb{R}^{n_{j+1}}$, $x \mapsto A_j x + b_j$, with an *activation* function $\sigma : \mathbb{R}^{n_{j+1}} \to \mathbb{R}^{n_{j+1}}$; and $s : \mathbb{R}^{n_l} \to [0, 1]$ is the *score* function. The *width* n_j is the number of nodes in the jth layer and we set $n_1 = d$ and $n_l = p$. For $j = 1, \dots, l$, the composition of the first through jth layers is denoted

$$\nu_j \coloneqq f_j \circ \cdots \circ f_2 \circ f_1 \quad \text{and} \quad \nu = s \circ \nu_l.$$

We assume that s is a *linear classifier* and thus the decision boundary of s is a hyperplane in $\mathbb{R}^p$. For notational convenience later, we define the '$(l+1)$th layer' $\nu_{l+1} \coloneqq s$ to be the score function and the '0th layer' ν_0 to be the identity function on $\mathbb{R}^d$.

We train an l-layer neural network $\nu : \mathbb{R}^d \to [0, 1]$ on a training set $T \subseteq M_a \cup M_b$ to classify samples into class a or b. As usual, the network's output for a sample $x \in T$ is interpreted to be the probability of $x \in M_a$. In all our experiments, we train ν until it correctly classifies all $x \in T$ — we will call such a network ν *well-trained*. In fact, we sampled T so densely that in reality ν also has near zero misclassification error on any test set $S \subseteq (M_a \cup M_b) \setminus T$; and we trained ν so thoroughly that its output is concentrated near 0 and 1. For all intents and purposes, we may treat ν as an ideal classifier.

We deliberately choose $M_a \cup M_b$ to have doubly complicated topologies in the following sense:

(i) For each $i = a, b$, the component M_i itself will have complicated topologies, with multiple components, i.e., large $\beta_0(M_i)$, as well as multiple k-dimensional holes, i.e., large $\beta_k(M_i)$.

(ii) In addition, M_a and M_b will be entangled in a topologically complicated manner. See Figures 2.8 and 2.9 for example. They not only cannot be separated by a hyperplane but any decision boundary $D \subseteq \mathbb{R}^d$ that separates them will necessarily have complicated topology itself.

In terms of the topological complexity in (2.1), $\omega(M_a)$, $\omega(M_b)$, $\omega(D)$ are all large.

Our experiments are intended to show the topologies of $\nu_j(M_a)$ and $\nu_j(M_b)$ evolve as j runs from 1 through l, for different manifolds M_a, M_b entangled in different ways, for different number of layers l and choices of widths $n_1, \ldots, n_d$, and different activations σ. Getting ahead of ourselves, the results will show that a well-trained neural network $\nu : \mathbb{R}^d \to [0, 1]$ reduces the topological complexity of M_a and M_b on a layer-by-layer basis until, at the output, we see a simple disentangled arrangement where the point cloud T gets mapped into two clusters of points $\nu(T_a)$ and $\nu(T_b)$ on opposite ends of $[0, 1]$. This indicates that an initial decision boundary $D \subseteq \mathbb{R}^d$ of complicated topology ultimately gets transformed into a hyperplane in $\mathbb{R}^p$ by the time it reaches the final layer. We measure and track the topologies of $\nu_j(M_a)$ and $\nu_j(M_b)$ directly, but our approach only permits us to indirectly observe the topology of the decision boundary separating them.

2.5 Real versus simulated data

We perform our experiments on a range of both real-world and simulated data sets to validate our premise that a neural network operates by simplifying topology. We explain why each is indispensable to our goal.

Unlike real-world data, simulated data may be generated in a controlled manner with well-defined topological features that are known in advance (crucial for finding a single scale for all homology computations). Moreover, with simulated data we have clean samples and may skip the denoising step mentioned in the previous section. We can generate samples that are uniformly distributed on the underlaying manifold, and ensure that the assumptions of Section 2.4 are satisfied. In addition, we may always simulate a data set with a perfect classifier, whereas such a classifier for a real-wold data set may not exist when the probability distributions of different categories overlap. For convincing results, we train our neural network to perfect accuracy on training set and near-zero generalization error — this may be impossible for real-world data. Evidently if there is no complete separation of one category M_a from the other M_b, i.e., $M_a \cap M_b \neq \varnothing$, the manifold $M = M_a \cup M_b$ will be impossible

to disentangle. Such is often the case with real-world data sets, which means that they may not fit our required setup in Section 2.4.

Nevertheless, the biggest issue with real-world data sets is that they have *vastly* more complicated topologies that are nearly impossible to determine in advance. Even something as basic as the Mumford data set [49], a mere collection of 3×3-pixels of high contrast patches of natural images, took many years to have its topology determined [15] and whether the conclusion (that it has the topology type of a Klein bottle) is correct is still a matter of debate. Figuring out, say, the topology of the manifold of cat images within the space of all possible images is well-beyond our capabilities for the foreseeable future.

Since our experiments on simulated data allow us to pick the right scale to compute homology, we only need to compute homology at one single scale. On the other hand, for real data we will need to find the persistence barcodes, i.e., determine homology over a full range of scales. Consequently, our experiments on simulated data are extensive — we repeat our experiments for each simulated data set over a large number of neural networks of different architectures to examine their effects on topology changes. In all we ran more than 10,000 homology computations on our simulated data sets since we can do them fast and accurate. In comparison, our experiments on real-world data are more limited in scope as it is significantly more expensive to compute persistence barcodes then to compute homology at a single scale.

As such, we use simulated data to fully explore and investigate the effects of depth, width, shapes, activation functions, and various combinations of these factors on the topology-changing power of neural networks. Thereafter we use real-world data to validate the findings we draw from the simulated data sets.

2.6 Methodology

In this section, we will describe the full details of our methodology for (i) simulating topologically nontrivial data sets in a binary classification problem; (ii) training a variety of

neural networks to near-perfect accuracy for such a problem; (iii) determining the homology of the data set as it passes through the layers of such a neural network. For real data sets, step (i) is of course irrelevant, but steps (ii) and (iii) will apply with minor modifications; these discussions will be deferred to Section 2.8.

The key underlying reason for designing our experiments in the way we did is relative computational costs:

- multidimensional persistent homology is much more costly than persistent homology;
- persistent homology is much more costly than homology;
- homology is much more costly than training neural networks.

As such, we train 1,286 neural networks to near zero generalization errors; for each neural network, we compute homology at every layer but we compute persistent homology only once; and we avoid multidimensional persistent homology altogether.

2.6.1 Generating data sets

We generate three point cloud data sets D-I, D-II, D-III in a controlled manner to have complicated but manageable topologies that we *know in advance*.

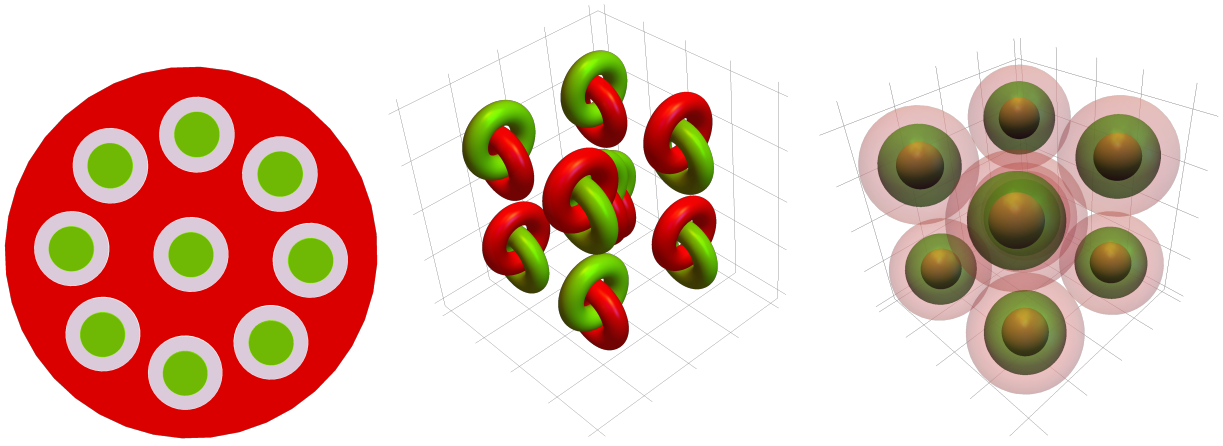


Figure 2.8: The manifolds underlying data sets D-I, D-II, D-III (left to right). The green M_a represents category a, the red M_b represents category b.

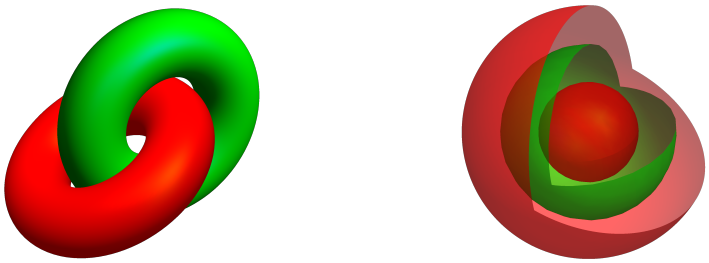


Figure 2.9: *Left*: D-II comprises nine pairs of such interlocking rings. *Right*: D-III comprises nine units of such doubly concentric spheres.

D-I is sampled from a two-dimensional manifold consisting of M_a, nine green disks, positioned in M_b, a larger disk with nine holes as on the left in Figure 2.8. We clearly have $\beta(M_a) = (9, 0)$ and $\beta(M_b) = (1, 9)$ (one connected component, nine holes). **D-II** is sampled from a three-dimensional manifold comprising nine disjoint pairs of red solid torus interlocked with a green solid torus (a single pair is shown in Figure 2.9). M_a (resp. M_b) is the union of all nine green (resp. red) tori. So $\beta(M_a) = \beta(M_b) = (9, 9, 0)$. **D-III** is sampled from a three-dimensional manifold comprising nine disjoint units of the following — a large red sphere enclosing a smaller green sphere enclosing a red ball; the green sphere is trapped between the red sphere and the red ball. M_a is the union of all nine green spheres and M_b is the union of the nine spheres and nine balls. So we have $\beta(M_a) = (9, 0, 9)$ and $= \beta(M_b) = (18, 0, 9)$ (see Figures 2.8 and 2.9 for more details, but note on Figure 2.9 the spheres are shown with portions omitted). In all cases, the two categories a and b are entangled in such a way that any decision boundary separating the two categories necessarily has highly complex topology.

The point cloud data sets D-I and D-III are sampled on a grid whereas D-II is sampled uniformly from the solid tori. The difference in sampling schemes is inconsequential for all intents and purposes in this part of the thesis, as the samples are sufficiently dense that there is no difference in training and testing behaviors.

2.6.2 Training neural networks

Our goal is to examine the topology changing effects of (i) different *activations*: hyperbolic tangent, leaky ReLU set to be $\max(x, 0.2x)$, and ReLU; (ii) different *depths* of four to ten layers; (iii) different *widths* of six to fifty neurons. So for any given data set (D-I, D-II, D-III) and any given architecture (depth, width, activation), we tracked the Betti numbers through all layers for at least 30 well-trained neural networks. The repetition is necessary — given that neural network training involves a fair amount of randomization in initialization, batching, optimization, etc — to ensure that what we observe is not a fluke.

To train these neural networks to our requirements — recall that this means zero training error and a near-zero ($\approx 0.01\%$) generalization error — we relied on TensorFlow (version 1.12.0 on Ubuntu 16.04.1). Training is done on cross-entropy categorical loss with standard Adam optimizer [44] for up to 18,000 training epochs. Learning rate is set to 0.02–0.04 with an exponential decay, i.e., $\eta^{t/d}$ where t is the training epoch normalized by $d = 2500$. For the 'bottleneck architectures' where the widths narrow down in the middle layers (see Table 2.1), the decay is set to 4000 and $\eta = 0.5$. We use the softmax function as the score function in all of our networks, i.e., $s : \mathbb{R}^p \to \mathbb{R}^p$ whose ith coordinate is

$$s_i(x) = e^{x_i}/(e^{x_1} + \cdots + e^{x_p}), \qquad i = 1, \ldots, p,$$

where p is the number of categories. In our case, $p = 2$ and $i = a, b$.

Table 2.1 summarizes our results: the data set used, the activation type, the widths of each layer, and the number of successfully trained neural networks of that architecture obtained. The first number in the sequence of the third column gives the dimension of the input, which is two for the two-dimensional D-I and three for the three-dimensional D-II and D-III. The last number in that sequence is always two since they are all binary classification problems. To give readers an idea, training any one of these neural networks to near zero generalization error takes at most 10 minutes, often much less.

data set	activation	neurons in each layer	#
D-I	tanh	2-15-15-15-15-15-15-15-15-15-2	30
D-I	leaky ReLU	2-15-15-15-15-15-15-15-15-15-2	30
D-I	leaky ReLU	2-05-05-05-05-03-05-05-05-05-2	30
D-I	leaky ReLU	2-15-15-15-15-03-15-15-15-15-2	30
D-I	leaky ReLU	2-50-50-50-50-50-50-50-50-50-2	30
D-I	ReLU	2-15-15-15-15-15-15-15-15-15-2	30
D-II	tanh	3-15-15-15-15-15-15-15-15-15-2	32
D-II	leaky ReLU	3-15-15-15-15-15-15-15-15-15-2	36
D-II	ReLU	3-15-15-15-15-15-15-15-15-15-2	31
D-II	tanh	3-25-25-25-25-25-25-25-25-25-2	30
D-II	leaky ReLU	3-25-25-25-25-25-25-25-25-25-2	30
D-II	ReLU	3-25-25-25-25-25-25-25-25-25-2	30
D-III	tanh	3-15-15-15-15-15-15-15-15-15-2	30
D-III	leaky ReLU	3-15-15-15-15-15-15-15-15-15-2	46
D-III	ReLU	3-15-15-15-15-15-15-15-15-15-2	30
D-III	tanh	3-50-50-50-50-50-50-50-50-50-2	30
D-III	leaky ReLU	3-50-50-50-50-50-50-50-50-50-2	30
D-III	ReLU	3-50-50-50-50-50-50-50-50-50-2	34
D-I	tanh	2-15-15-15-15-2	30
D-I	tanh	2-15-15-15-15-15-15-15-15-2	30
D-I	leaky ReLU	2-15-15-15-15-2	30
D-I	leaky ReLU	2-15-15-15-15-15-15-15-15-2	30
D-I	ReLU	2-15-15-15-15-2	30
D-I	ReLU	2-15-15-15-15-15-15-15-15-2	30
D-II	tanh	3-15-15-15-15-2	31
D-II	tanh	3-15-15-15-15-15-2	31
D-II	tanh	3-15-15-15-15-15-15-15-2	30
D-II	leaky ReLU	3-15-15-15-15-2	31
D-II	leaky ReLU	3-15-15-15-15-15-2	30
D-II	leaky ReLU	3-15-15-15-15-15-15-2	30
D-II	leaky ReLU	3-15-15-15-15-15-15-15-2	31
D-II	leaky ReLU	3-15-15-15-15-15-15-15-15-2	42
D-II	ReLU	3-15-15-15-15-2	32
D-II	ReLU	3-15-15-15-15-15-2	32
D-II	ReLU	3-15-15-15-15-15-15-15-15-2	31
D-III	tanh	3-15-15-15-15-15-15-2	30
D-III	tanh	3-15-15-15-15-15-15-15-15-2	31
D-III	leaky ReLU	3-15-15-15-15-15-15-2	30
D-III	leaky ReLU	3-15-15-15-15-15-15-15-15-2	30
D-III	ReLU	3-15-15-15-15-15-15-2	33
D-III	ReLU	3-15-15-15-15-15-15-15-15-2	32

Table 2.1: First column specifies the data set on which we train the networks. Next two columns give the activation used and a sequence giving the number of neurons in each layer. Last column gives the number of well-trained networks obtained.

2.6.3 Computing homology

For each of the neural networks obtained in Section 2.6.2, we track how the topology of the respective point cloud data set changes as it passed through the layers. This represents the bulk of the computational effort, way beyond that required for training neural networks in Section 2.6.2. With simulated data, we are essentially assured of a perfectly clean data set and the preprocessing step in Figure 2.7 of Section 2.3.6 may be omitted. We describe the rest of the work involved below.

The metric δ used to form our Vietoris–Rips complex is given by the graph geodesic distance on the k-nearest neighbors graph determined by the point cloud $X \subseteq \mathbb{R}^d$. As this depends on k, a positive integer specifying the number of neighbors used in the graph construction, we denote the metric by δ_k. In other words, the Euclidean distance on $\mathbb{R}^d$ is used only to form the k-nearest neighbors graph and do not play a role thereafter. For any $x_i, x_j \in X$, the distance $\delta_k(x_i, x_j)$ is given by the minimal number of edges between them in the k-nearest neighbors graph. Each edge, regardless of its Euclidean length, has the same length of one when measured in δ_k.

The metric δ_k has the effect of normalizing distances across layers of a neural network while preserving connectivity of nearest neighbors. This is important for us as the local densities of a point cloud can vary enormously as it passes through a layer of a well-trained neural network — each layer stretches and shrinks different regions, dramatically altering geometry as one can see in the bottom halves of Figures 2.11, 2.12, and 2.13. Using an intrinsic metric like δ_k ameliorates this variation in densities; it is robust to geometric changes and yet reveals topological ones. Furthermore, our choice of δ_k allows for comparison across layers with different numbers of neurons. Note that if $d \neq p$, the Euclidean norms on $\mathbb{R}^d$ and $\mathbb{R}^p$ are two different metrics on two different spaces with no basis for comparison. Had we used Euclidean norms, two Vietoris–Rips complexes of the same scale ε in two different layers cannot be directly compared — the scale needs to be calibrated somehow to reflect that they live in different spaces. Using δ_k avoids this problem.

This leaves us with two parameters to set: k, the number of neighbors in the nearest neighbors graph and ε, the scale at which to build our Vietoris–Rips complex. This is where persistent homology, described at length in Section 2.3.5, comes into play. Informed readers may think that we should be using *multidimensional persistence* since there are two parameters but this is prohibitively expensive as the problem is EXPSPACE-complete [16] and its results are not quite what we need, for one, there is no multidimensional analogue of persistence barcodes [17]. To choose an appropriate (k_*, ε_*) for a point cloud $X \subseteq M \subseteq \mathbb{R}^d$,

we construct a filtered complex over the two parameters: Let $\mathrm{VR}_{k,\varepsilon}(X)$ be the Vietoris–Rips complex of X with respect to the metric δ_k at scale ε. In our case, we know the topology of the underlying manifold M completely as we generated it in Section 2.6.1 as part of our data sets. Thus we may ascertain whether our chosen value (k_*, ε_*) gives a Vietoris–Rips complex $\mathrm{VR}_{k_*,\varepsilon_*}(X)$ with the same homology as M.

Set $\varepsilon = 1$, we determine a value of k_* with persistent homology on the k-filtered complex in the metric δ_k with correct zeroth homology, i.e., k_* is chosen so that

$$\beta_0\big(\mathrm{VR}_{k_*,1}(X)\big) = \beta_0(M).$$

Set $k = k_*$, we determine a value of ε_* with persistent homology on the ε-filtered complex in the metric δ_{k_*} with correct first and second homologies, i.e., ε_* is chosen so that

$$\beta_1\big(\mathrm{VR}_{k_*,\varepsilon_*}(X)\big) = \beta_1(M) \qquad \text{and} \qquad \beta_2\big(\mathrm{VR}_{k_*,\varepsilon_*}(X)\big) = \beta_2(M).$$

If there is a range of parameters that all recover the correct homology we pick our (k_*, ε_*) closest to the middle of the range. Once these parameters are set, we keep them fixed for our homology computations across all layers of the network.

The parameters chosen via the aforementioned procedure for our data sets are as follows. For D-I, we have $k_* = 14$ neighbors, and scale is set at $\varepsilon_* = 2.5$; recall that this means that $x_0, x_1, \ldots, x_n \in X$ form an n-simplex in $\mathrm{VR}_{14,2.5}(X)$ whenever $\delta_{14}(x_i, x_j) \leq 2.5$ for all i, j. For both D-II and D-III, we have $k_* = 35$ and $\varepsilon_* = 2.5$. Figure 2.10 shows the Betti numbers for these three data sets over a range of values of (k, ε): green (resp. red) dots indicate integral points on the (k, ε)-plane with correct (resp. incorrect) Betti numbers and the blue dot marks the (k_*, ε_*) selected in each case.

Our homology and persistent homology computations are all performed using the Eirene package in Julia 0.6.4 [37]. To give readers an idea, the time required to compute a single Betti number from a point cloud X ranges from a few tens of seconds, if $X \subseteq \mathbb{R}^5$ is the

output of a five-neuron-wide layer, to at most 30 minutes, if $X \subseteq \mathbb{R}^{50}$ is the output of a 50-neuron wide layer. On the other hand, the time taken for the persistent homology computations to obtain k_* and ε_* is in excess of 80 minutes. These computations are run in parallel over 12 cores of an Intel i7-8750H-2.20GHz processor with 9,216KB cache and 32GB DDR4-2666MHz RAM. The jobs are fed in a queue, with a single core limited to 9GB of memory.

2.6.4 Overview of our experiments

All our experiments on simulated data (those on real data omits the first step) may be described at a high level as follows: (i) generate a manifold $M = M_a \cup M_b \subseteq \mathbb{R}^d$ with $M_a \cap M_b = \varnothing$; (ii) densely sample point cloud $X \subseteq M$ and let $X_i := X \cap M_i$, $i = a, b$; (iii) train l-layer neural network $\nu : \mathbb{R}^d \to [0, 1]$ on the labeled training set $X_a \cup X_b$ to classify points on M; (iv) compute homology of the output at the jth layer $\nu_j(X_i)$, $j = 0, 1, \ldots, l, l+1$ and $i = a, b$. This allows us to track how the topology of M_i, by way of (persistent) homology of X_i, as it passes through the layers. Steps (i) and (ii) are described in Section 2.6.1, step (iii) in Section 2.6.2, and step (iv) in Section 2.6.3. The neural network notations are as in Section 2.4. Results will be described in Section 2.7. In reality, the point cloud X_i used in

data set	training neural networks	homology computations
D-I	7,800	2,600
D-II	45,000	11,250
D-III	37,800	9,450

Table 2.2: Comparison of sample sizes for computations in Sections 2.6.2 and 2.6.3.

step (iv) is not the same as the training set X_i used in step (iii) as it is considerably more expensive to compute homology (see Section 2.6.3) than to train a neural network to near zero generalization error (see Section 2.6.2). As such the size of point clouds used for our homology computations are a fraction (about 1/4) that used to train our neural networks.

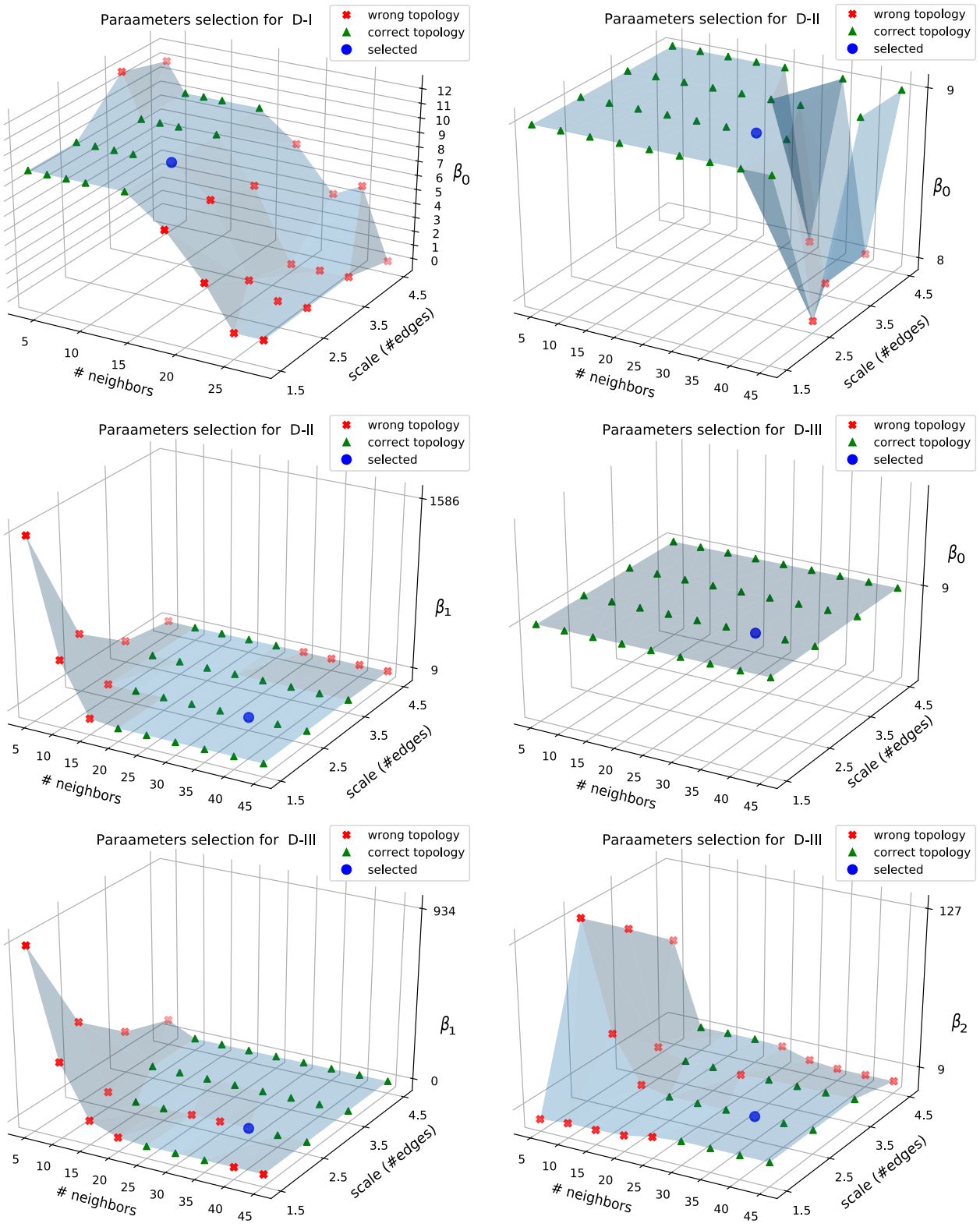


Figure 2.10: For each combination of parameters k and ε, we determine whether the homology of $\mathrm{VR}_{k,\varepsilon}(X)$ matches the homology of the manifold M from which X is sampled. We marked those values of (k, ε) with correct homology in green and those with incorrect homology in red. Our choice of (k, ε), naturally chosen among the correct ones, is marked in blue.

2.7 Results and discussions

We present the results from our experiments to analyze how a well-trained neural network simplifies the topology of a data set. More importantly, we discuss what one may surmise from these results. We start with the three simulated data sets D-I, D-II, D-III in Section 2.6.1 since the results are the most striking in this case. To validate that these observations indeed extend to real data, we repeat our experiments on four real-world data sets in Section 2.8.

Topological simplification evident across training instances. Figure 2.11 records our simplest data set D-I, where M_a comprises nine contractible components and so higher Betti numbers are irrelevant (all zero). Here we present every curve corresponding to every neural network trained on D-I, recall that we do at least 30 runs for each experiment to account for the inherent randomness, and they all show consistent profiles — a clear decay in β_0 across the layers although hyperbolic tangent activation (blue graph) shows larger variance in this decay than leaky ReLU (red graph) and ReLU (green graph). The profiles shown in Figure 2.11 are representative of other experiments on higher Betti numbers and on other data sets. To avoid clutter, in the corresponding figures for D-II and D-III (Figures 2.12 and 2.13), we omit curves corresponding to the individual runs and show only the curve of their means (dark curve in middle) and the region of half standard deviation (shaded region). The bottom diagrams in Figure 2.11 show how M_a changes from layer-to-layer by projecting onto its first two principal components (note that the intervening layers are in $\mathbb{R}^{15}$).

Nonhomeomorphic activations induce rapid topology changes. As the faint blue lines in Figure 2.11 reveal, hyperbolic tangent activation is less effective at reducing Betti numbers, occasionally even increasing them over layers. In all data sets, across all our experiments, the nonhomeomorphic activation ReLU exhibits the most rapid reductions in all Betti numbers. The top halves of Figures 2.11, 2.12 and 2.13 show results for a 10-layer network (see caption for specifics). The different rates at which topological changes occur are

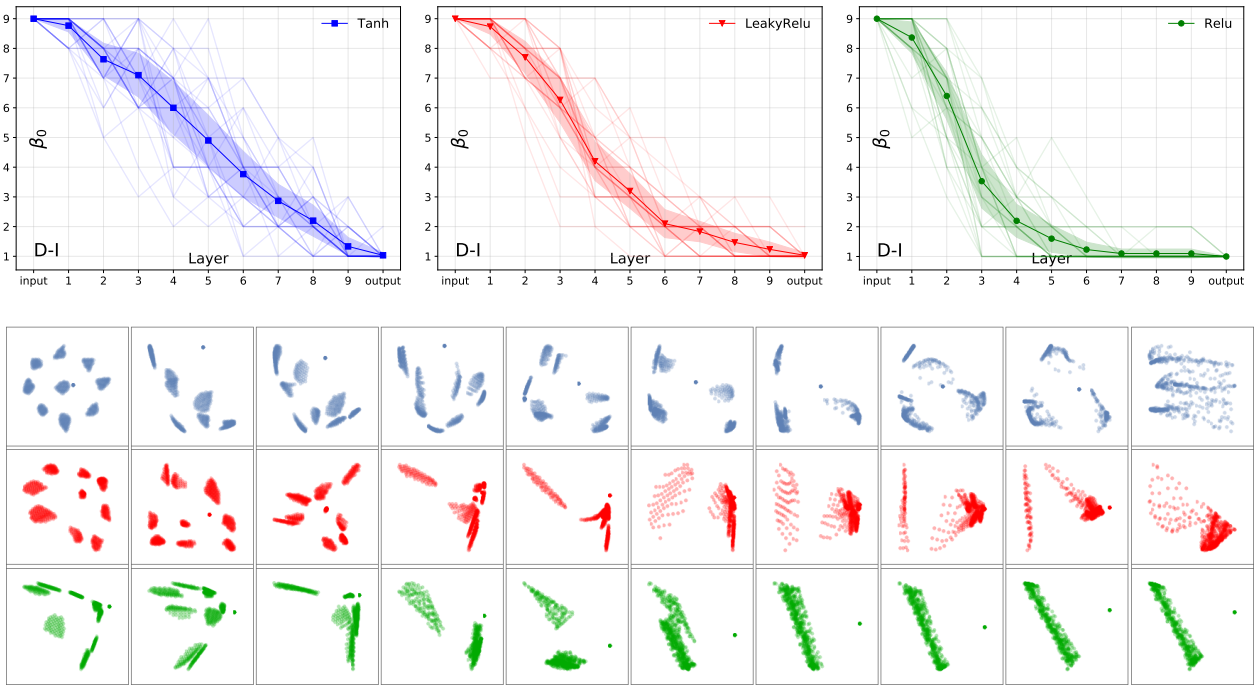


Figure 2.11: *Top*: Faint curves show individual profiles, dark curves show averaged profiles of $\beta_0\big(\nu_k(M_a)\big)$, $k = 1, \ldots, 10$, in data set D-I. Shaded region is the region of $\pm$ half standard deviation about average curve. Networks have different activations — blue for tanh, red for leaky ReLU, green for ReLU; but same architecture — ten layers, two neurons in the first and the last layers, fifteen in the intervening layers. *Bottom*: Projections of $\nu_k(M_a)$, $k = 1, \ldots, 10$, on the first two principal components, color-coded according to activations.

also evident from the principal components projections in the bottom half of these figures.

Efforts depend on topological features. Some topological features evidently require more layers to simplify than others. The hardest one is the interlocking tori in the data set D-II. The profile of $\beta_1\big(\nu_l(M_a)\big)$ in the graph on the right of Figure 2.12 shows that some loops survive across many layers, especially so when activated with hyperbolic tangent (blue): both the (blue) principal components projections and the (blue) profile show that the loops persist considerably longer than any other features in any of the three data sets.

Effects of depth on topology change. Reducing the depth of a constant-width network beyond a certain threshold makes it increasingly difficult to train the network to high accuracy — the percentage of successfully trained networks drops noticeably. Moreover,

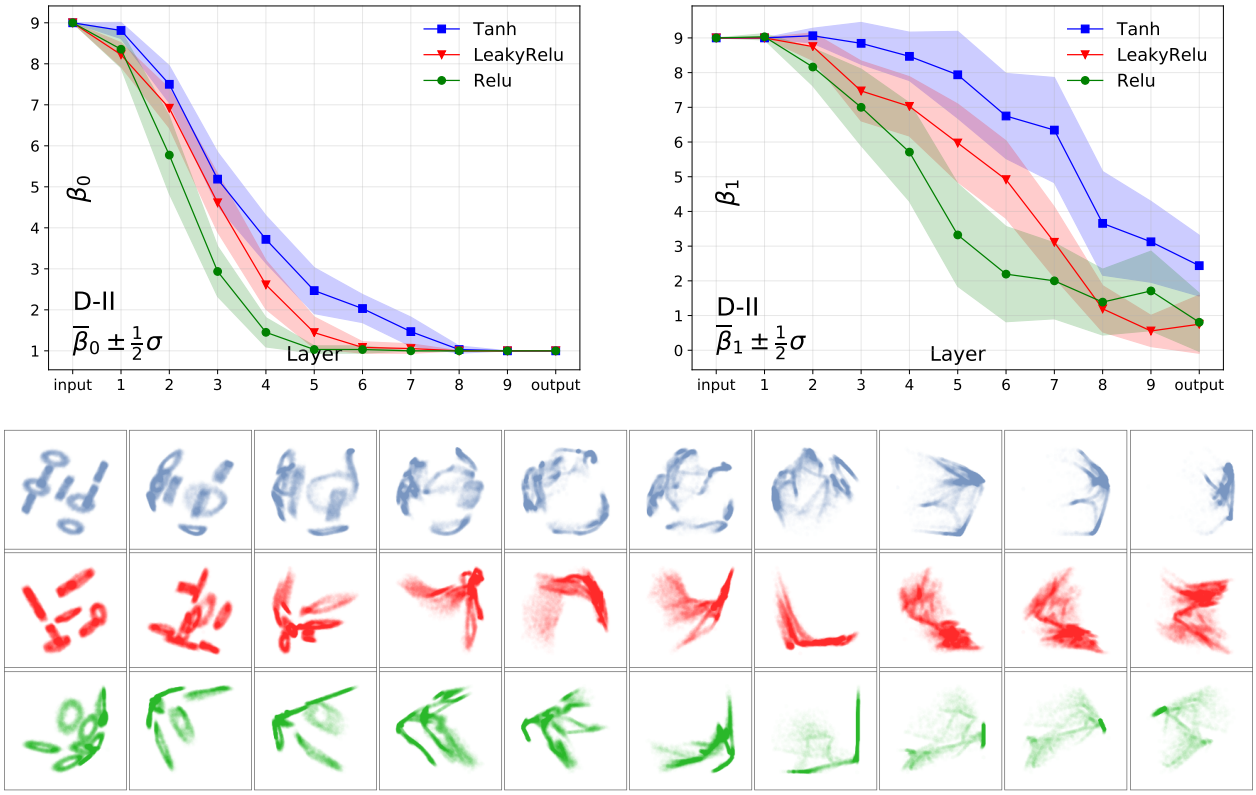


Figure 2.12: *Top:* Profiles of $\beta_0\big(\nu_k(M_a)\big)$ and $\beta_1\big(\nu_k(M_a)\big)$ for data set D-II, $k = 1, \ldots, 10$. Network's architecture: three-dimensional input, two-dimensional output, and fifteen neurons in intervening layers one to nine, with different activations. *Bottom*: Projections of $\nu_k(M_a)$, $k = 1, \ldots, 10$, on the first two principal components.

as the depth is reduced, the burden of changing topology does not spread evenly across all layers but becomes concentrated in the final layers. The initial layers do not appear to play a big role in changing topology, reducing depth simply makes the final layers 'work harder' to produce larger reductions in Betti numbers. Figure 2.14 shows this effect.

Effects of width on topology change. For the data set D-I, we compare three sets of ten-layer networks: (i) narrow networks with six neurons in each layer; (ii) 'bottleneck' networks with 15, 15, 15, 15, 3, 15, 15, 15, 15 neurons respectively in layers one through nine — notice the three neuron bottleneck layer; (iii) wide networks with fifty neurons in each layer. The left graph in Figure 2.15 suggests that a bottleneck layer forces large topological changes, and a narrow network changes topology faster than a wider one. The other two

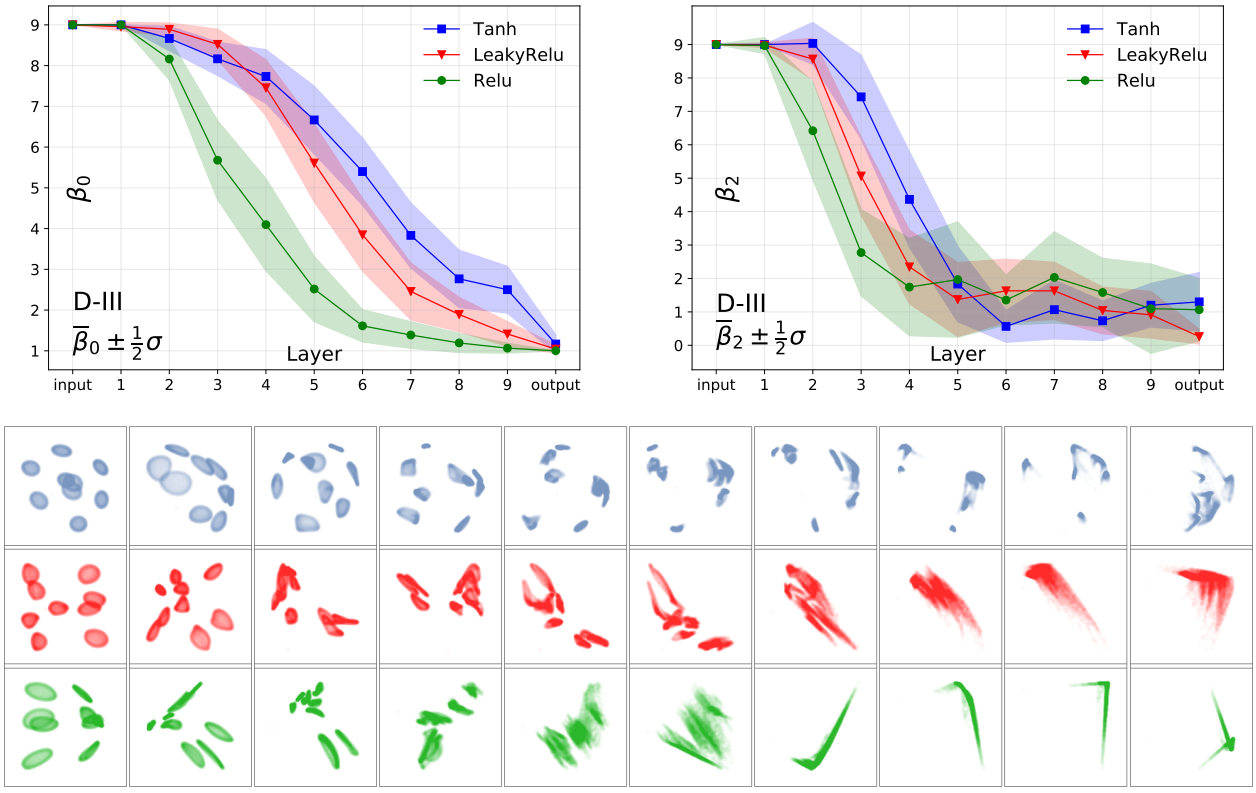


Figure 2.13: *Top:* Profiles of $\beta_0\big(\nu_k(M_a)\big)$ and $\beta_2\big(\nu_k(M_a)\big)$ for data set D-III, $k = 1, \ldots, 10$. Network's architecture: three-dimensional input, two-dimensional output, and fifteen neurons in intervening layers one to nine, with different activations. *Bottom*: Projections of $\nu_k(M_a)$, $k = 1, \ldots, 10$, on first two principal components.

graphs compare a 15-neuron wide network with a 50-neuron wide one, both with ten layers, on data sets D-II and D-III respectively. However, for the same choice of activation, the difference between them is negligible. Also, reducing the width below fifteen neurons makes training to high accuracy increasing more difficult, i.e., the percentage of successfully trained networks starts to drop.

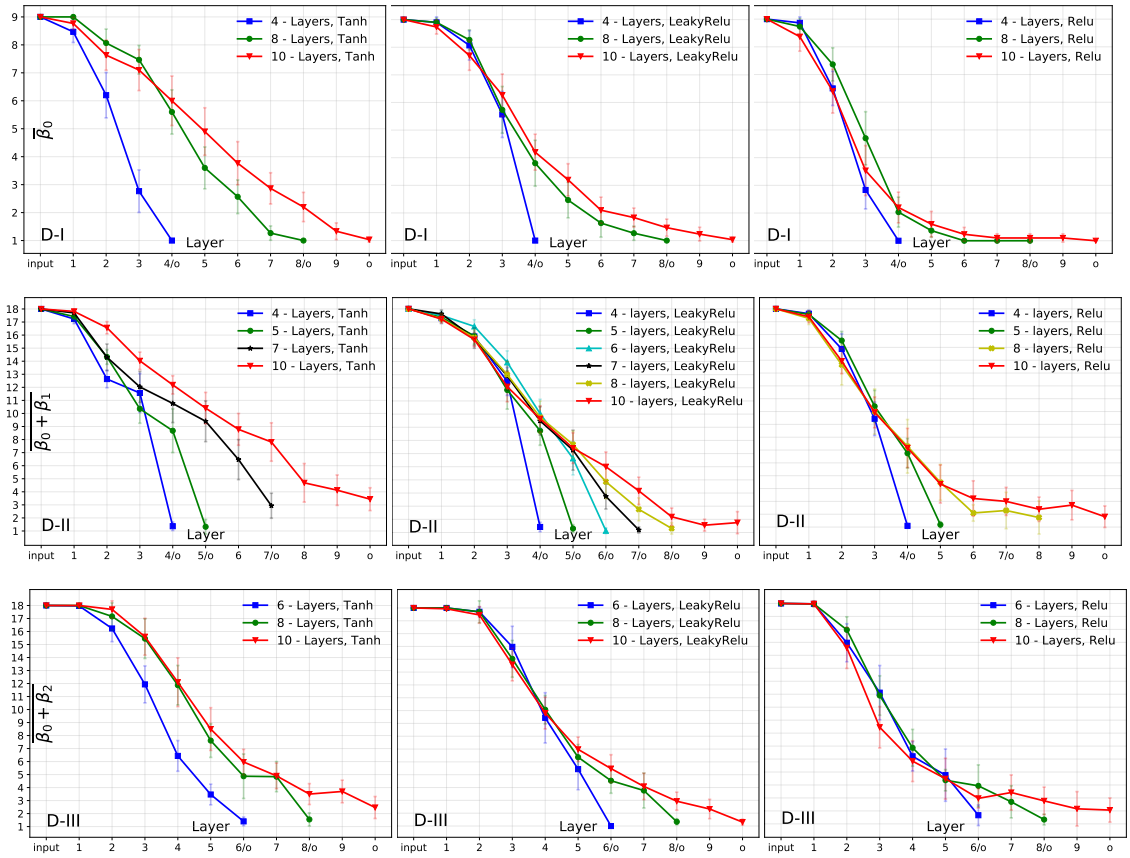


Figure 2.14: Mean values of topological complexity $\omega\big(\nu_k(M_a)\big)$, $k = 1, \ldots, l$, for fifteen-neuron-wide networks of varying depths. Error bars indicate $\pm$ half standard deviation about the mean.

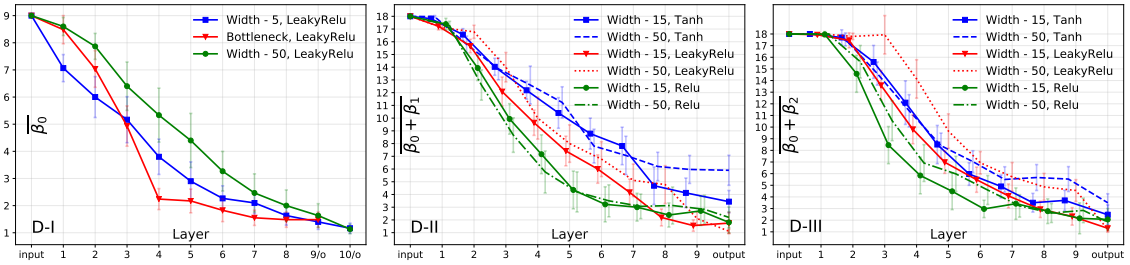


Figure 2.15: Mean values of topological complexity $\omega\big(\nu_k(M_a)\big)$, $k = 1, \ldots, l$, for ten-layer deep networks of varying widths. Error bars indicate $\pm$ half standard deviation about the mean.

2.8 Consistency with real-world data

The results in Section 2.7 are deduced from experiments on the simulated data sets D-I, D-II, D-III generated in Section 2.6.1. It is naturally to ask if these results remain valid on real data. In this section, we will see that they are, with some mild caveats. The key difference between real and simulated data is only in the amount of computational effort required to carry out our experiments — they are much more expensive for real data sets.

We will validate our results on four real-world data sets from (i) MNIST Handwritten Digits [47], (ii) HTRU2 High Time-Resolution Universe Survey [54], (iii) UCI Banknotes Authentication [52], (iv) UCI Sensorless Drive Diagnostic [6]. These data sets were chosen on the basis that they are real-valued and may be trained to high accuracy. The goal, as usual, is to observe how their topology changes as they pass through the layers of well-trained neural networks. To this end, our methodology in Section 2.6 applies to these data sets with some modifications:

- For real data, it is no longer possible to obtain the kind of near-perfectly trained neural networks in Section 2.6.2 that we could readily obtain with simulated data. As such, we adjust our expectations accordingly. By a well-trained neural network on a real data set, we mean one whose test accuracy ranges between 95 to 98 percent (recall that for simulated data, we required 99.99% or better).

- Unlike the simulated data sets in Section 2.6.1, we do not already know the topology of our real data sets and this has to be determined with persistent homology. More importantly, for real data, it is no longer possible to set a single scale for observing topological changes across different layers, as described in Section 2.6.3 — we have to compute persistent homology in every layer to track topological changes.

The complexity of real-world data and the need to calculate full persistent homology at every layer limits the number of experiments that we could run. As it will be prohibitively expensive to carry out extensive exploratory tests across a range of different architectures

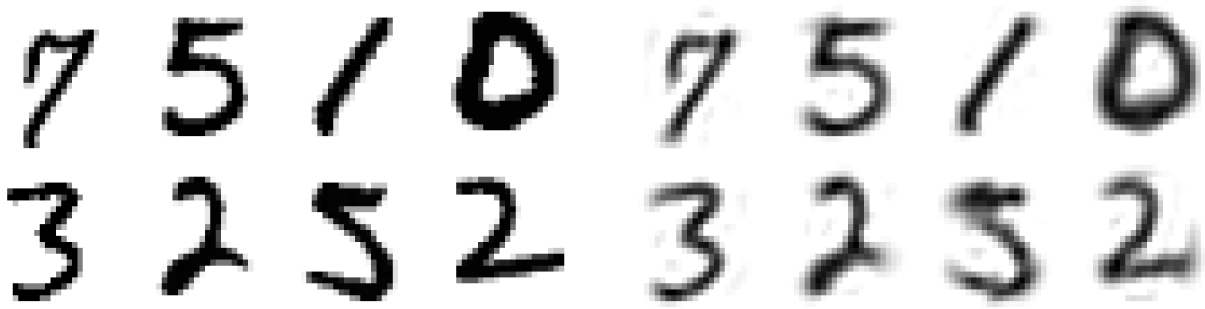


Figure 2.16: *Left*: Original MNIST handwritten digits. *Right*: MNIST handwritten digits projected onto first fifty principal components.

like what we did on simulated data (see Table 2.1), we will keep both width (10 neurons) and depth (10 layers) fixed in this section. In any case, our experiments on real data sets are not intended to be exploratory but to corroborate the findings in Section 2.7 that we deduced from simulated data. We seek confirmation on two findings in particular: the reduction in topological complexity through the layers and the relative effectiveness of ReLU over tanh activations in achieving this. Note that while we will only present the persistence barcodes at the output of first, the middle, and the final layer, we computed persistence homology in every layer; interested readers may easily get those for other layers from our publicly available codes.

MNIST Handwritten Digits [47]. Each of the 70,000 images in the MNIST handwritten digits data set is an image of size 28×28-pixels and collectively forms a point cloud on some manifold $M \subseteq \mathbb{R}^{784}$. Computing persistent homology for a 784-dimensional point cloud is way beyond what our computing resource could handle and we first reduce dimension by projecting onto its leading 50 principal components. Nevertheless, the dimension-reduced images remain to be of reasonably high-quality; we show a comparison of a few original digits alongside their principal component projections onto $\mathbb{R}^{50}$ in Figure 2.16. We will take the dimension-reduced point cloud X as the starting point for our experiments and will loosely refer to $X \subseteq M \subseteq \mathbb{R}^{50}$ as the MNIST data set.

Since we would like to have a binary classification problem for consistency with all our

other experiments, instead of regarding the MNIST data as a classification problem with ten classes, we reduce it to a problem of classifying a chosen (any) digit a versus all non-a digits. So our manifold is $M = M_a \cup M_b$, where M_a is the 'manifold all handwritten a' and M_b is the union of the 'manifolds of all other non-a handwritten digits.' Following our methodology in Section 2.6, we train a neural network with ten layers and ten neuron in each layer on a labelled point cloud $X_a = X \cap M_a$ and $X_b = X \cap M_b$ to classify points in M_a and M_b; then with persistent homology we analyze how M_a is transformed after each layer in the network. While we fix the depth and widths, we vary the activation among tanh, leaky ReLU, and ReLU. We used 60,000 samples to train and the remaining 10,000 samples to test our neural networks; we use one third of the test set for our persistent homology calculations.

activation	scale ε	$k=0$	1	2	3	4	5	6	7	8	9	10
tanh	1.5	525	408	356	266	233	145	156	88	30	20	9
	2.5	6	5	2	1	3	14	12	8	1	4	4
	3.5	1	1	1	1	1	1	1	1	1	1	3
leaky ReLU	1.5	525	340	182	108	38	16	10	8	1	1	1
	2.5	6	6	6	5	1	1	2	1	1	1	1
	3.5	1	1	1	1	1	1	1	1	1	1	1
ReLU	1.5	525	199	106	27	13	6	1	1	1	1	1
	2.5	6	2	6	6	2	1	1	1	1	1	1
	3.5	1	1	1	1	1	1	1	1	1	1	1

Table 2.3: Topological complexity $\omega(M_a) = \beta_0\big(\nu_k(M_a)\big) + \beta_1\big(\nu_k(M_a)\big) + \beta_2\big(\nu_k(M_a)\big)$ at layers $k = 0, 1, \ldots, 10$ with M_a the 'manifold of handwritten a'. Network has 50-dimensional input, 2-dimensional output, and is 10-dimensional in intermediate layers. For each of three activation types, we show homology at three scales $\varepsilon = 1.5$, 2.5, 3.5.

The results of our experiments for the digit $a = 0$ are shown in Table 2.3. What we see in the table corroborates our earlier findings on simulated data sets — an unmistakable reduction in topological complexity through the layers, with ReLU activation reducing topological complexity most rapidly when compared to the other two activations. With tanh activation, reduction in topological complexity is not only much slower but the network fails to reduce M_a to a topological disk, despite having ten layers. The persistence barcodes

diagram for the MNIST data set is much larger than those for the next three data sets but it is not much more informative than our summary statistics in Table 2.3. As such we do not show it here although it can just as readily be generated from our program.

HTRU2 High Time Resolution Universe Survey [54]. This data set consists of statistics of radio source signals from 17,898 stars, measured during the High Time Resolution Universe Survey (HTRU2) experiment to identify pulsars. For our purpose, it suffices to know that pulsars are stars that produce radio emission measurable on earth. In the HTRU2 data set, each recorded radio emission is described by eight continuous variable: four are statistics of the radio signal called 'integrated profile' and the other four are statistics of the 'DM-SNR curve' that tracks frequency components of the signal versus its arrival time. The radio sources are labeled by a or b according to whether the source is a pulsar or not. We show a small portion of this data set in Table 2.4.

Star #	1	2	3	4	5
Mean (integral profile)	140.5625	102.5078	103.0156	136.7500	99.3672
Standard Deviation (integral profile)	55.6838	45.5499	39.3416	57.1784	41.5722
Excess Kurtosis (integral profile)	−0.2346	0.2829	0.3233	−0.0684	0.4653
Skewness (integral profile)	−0.6996	0.4199	1.0511	−0.6362	4.1541
Mean (DM-SNR)	3.1998	1.3587	3.1212	3.6423	1.6773
Standard Deviation (DM-SNR)	19.1104	13.0790	21.7447	20.9593	61.7190
Excess Kurtosis (DM-SNR)	7.9755	13.3121	7.7358	6.8965	2.2088
Skewness (DM-SNR)	74.2422	212.5970	63.1720	53.5937	127.3930
Pulsar 'a' or not 'b'	b	a	b	b	b

Table 2.4: Five entries from HTRU2. The first eight rows are statistics of the radio signal from that star. The last row indicates whether the respective star is a pulsar 'a' or not 'b'.

We take a 3,278 subsample of the HTRU2 data set so that we have an equal number of pulsar and non-pulsars. This is a point cloud $X \subseteq M \subseteq \mathbb{R}^8$ with $M = M_a \cup M_b$ a union of the 'pulsar manifold' M_a and 'non-pulsar manifold' M_b; the point clouds $X_a = X \cap M_a$ and $X_b = X \cap M_b$ each has 1,639 points. We use 80% of this balanced data X for training the neural networks and the remaining 20% for testing. For persistent homology computations, we use the test set but we first passed it through a local outlier removal algorithm in [11] for

denoising. Again, our neural networks have ten layers with ten neurons in each layer and are activated with either ReLU or tanh.

In Figure 2.17, we show the persistence barcode diagrams for $\nu_k(X_a)$ in the first, middle, and last layer, i.e., $k = 1, 5, 10$. The scatter plots below the barcode diagrams show, for $k = 1, 5, 10$, the projections of $\nu_k(X_a)$ (red) and $\nu_k(X_b)$ (blue) onto the three leading principal components. These persistent barcodes tell the same story for the HTRU2 data as Table 2.3 does for the MNIST data and Figures 2.11, 2.12, 2.13 do for the simulated data D-I, D-II, D-III: Topology is simplified as the data passes through the layers; and ReLU does a better job than tanh activation at simplifying topological complexity.

UCI Banknotes Authentication [52]. This data set is derived from 400×400-pixels gray scale images of 1,372 genuine and forged banknotes; small patches ranging in sizes from 96×96 to 128×128-pixels are extracted from the images and wavelet-transformed. Figure 2.18 shows three of these small extracted patches.

The UCI Banknotes data set does not contain any images but is simply a list of four statistics computed from the wavelet coefficients of the image patches, together with a label indicating whether the patch is from a genuine 'a' or forged 'b' banknote. We show five of these entries in Table 2.5; the full data set comprises 1,372 entries like these.

Banknote #	1	2	3	4	5	6
Variance (wavelet coef.)	−1.3971	4.5459	3.8660	3.4566	0.3292	0.3901
Skewness (wavelet coef.)	3.3191	8.1674	−2.6383	9.5228	−4.4552	−0.1428
Kurtosis (wavelet coef.)	−1.3927	−2.4586	1.9242	−4.0112	4.5718	−0.0319
Entropy (wavelet coef.)	−1.9948	−1.4621	0.1065	−3.5944	−0.9888	0.3508
Genuine 'a' or forged 'b'	a	b	b	b	b	a

Table 2.5: Five entries from the UCI Banknotes data set. The first four rows are statistics of the wavelet coefficients of banknotes image patches. The last row indicates whether the respective banknote is genuine 'a' or forged 'b'.

As with the HTRU2 data set, we subsample 1,200 entries from the UCI Banknotes data set so that we have an equal number of genuine and forged samples; we use 80% of this data set for training and 20% for testing; and for persistent homology computations, we

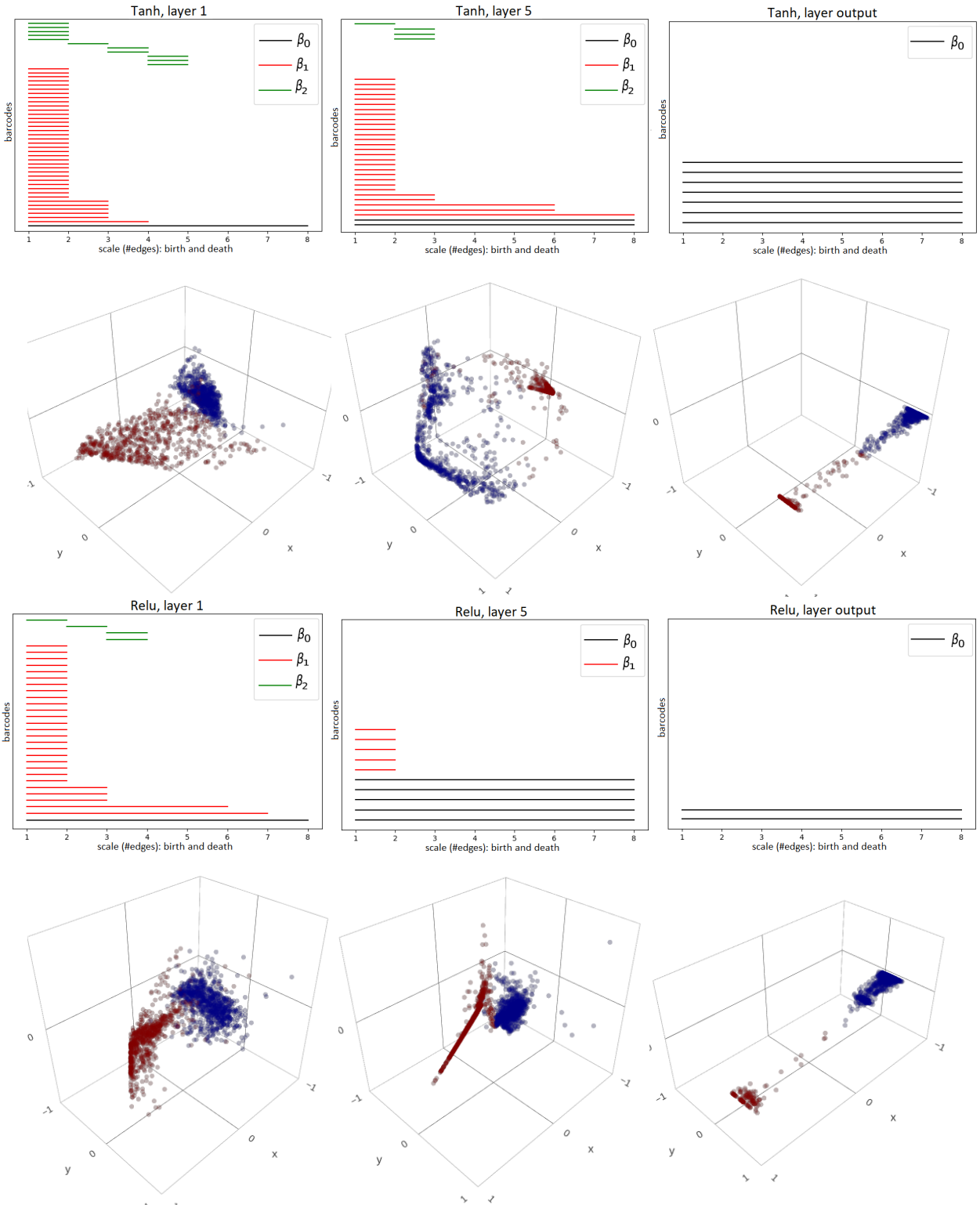


Figure 2.17: Persistence barcode diagrams for neural networks trained on HTRU2 show topology changes in the 'pulsar manifold' M_a as it passes through layers activated with tanh (top) and ReLU (bottom). Scatter plots show principal component projections of M_a (red) and the 'non-pulsar manifold' M_b (blue) at the corresponding layers.

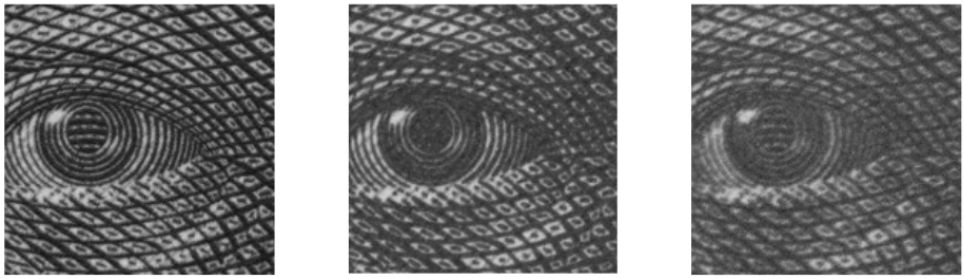


Figure 2.18: Texture sample for genuine banknote (*left*), high-quality forgery (*middle*) and low-quality forgery (*right*). The figures are taken from [52].

preprocess the data with the outliers removal algorithm in [11]. For our purpose, the UCI Banknotes data set is a point cloud $X \subseteq M \subseteq \mathbb{R}^4$ with $M = M_a \cup M_b$ a union of the 'manifold of genuine banknotes' M_a and 'manifold of forged banknotes' M_b; the point clouds $X_a = X \cap M_a$ and $X_b = X \cap M_b$ each has 600 points.

When X_a and X_b are passed through well-trained neural networks (ten layers, ten neurons in each layer, ReLU or tanh-activated), we obtained results consistent with all earlier experiments. The persistence barcodes diagrams in Figure 2.19 show that Betti numbers β_1 and β_2 are reduced to zero for both activations, β_0 successfully reduces to one when ReLU-activated but is stuck at two when tanh-activated. Also, reduction of Betti numbers happens more rapidly with ReLU-activation. These observations are also reflected in the respective principal components scatter plots below each persistence barcodes diagram.

UCI Sensorless Drive Diagnostic [6]. This data set concerns a printed circuit board that operates a specific type of drive motor. The goal is to classify twelve types of common defects in the drive motor based on 49 measurements of electric currents at various locations on the printed circuit board. Table 2.6 shows a sample of five such entries in this data set, which has a total of 58,509 such entries.

As in the case of the MNIST data set, instead of regarding the UCI Drive data as a classification problem with twelve classes, we reduce it to a binary classification problem of classifying a type a defect versus all other eleven types of defects. So our manifold is $M = M_a \cup M_b$ where M_a is the 'manifold of type a defects' and M_b is the union of the

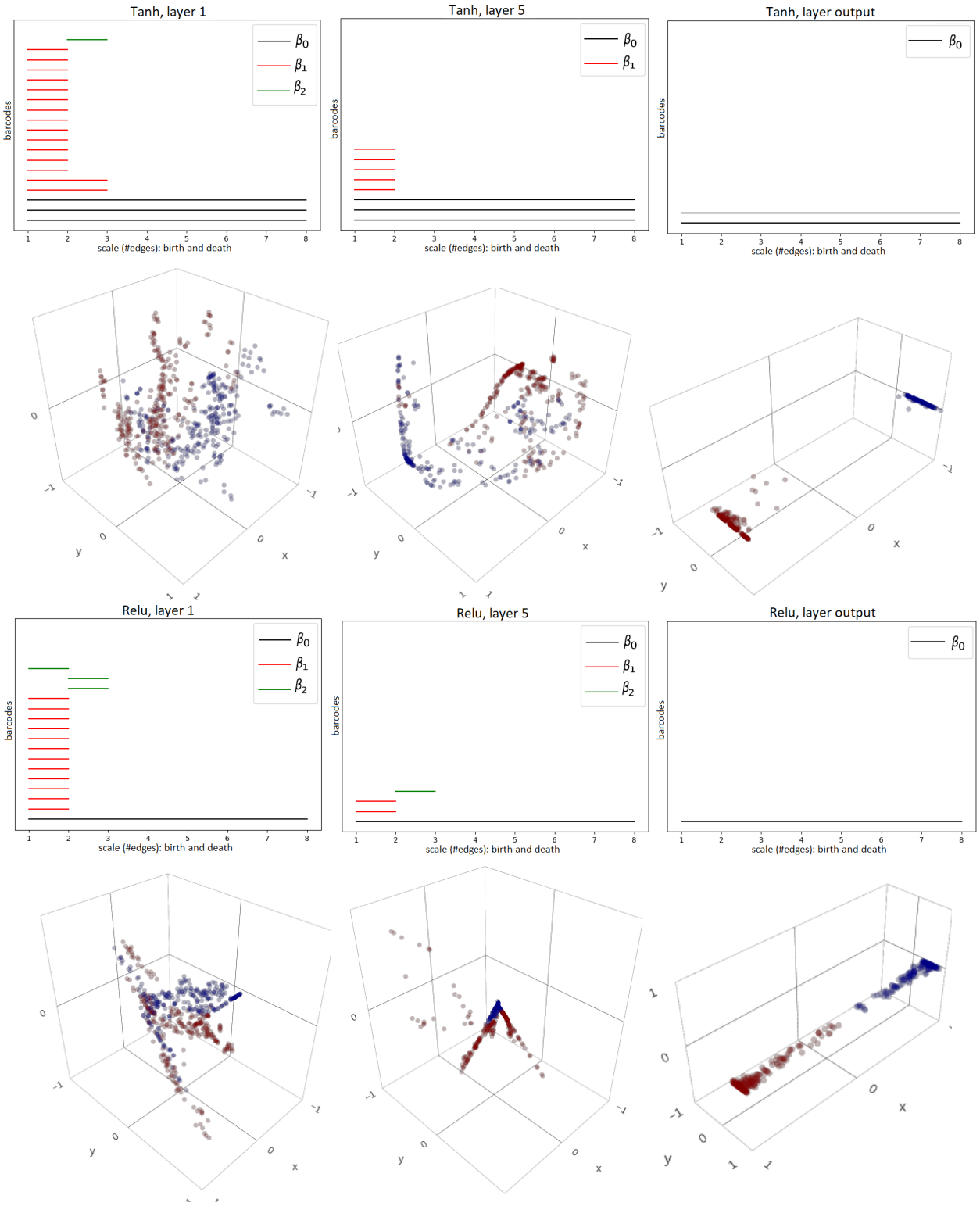
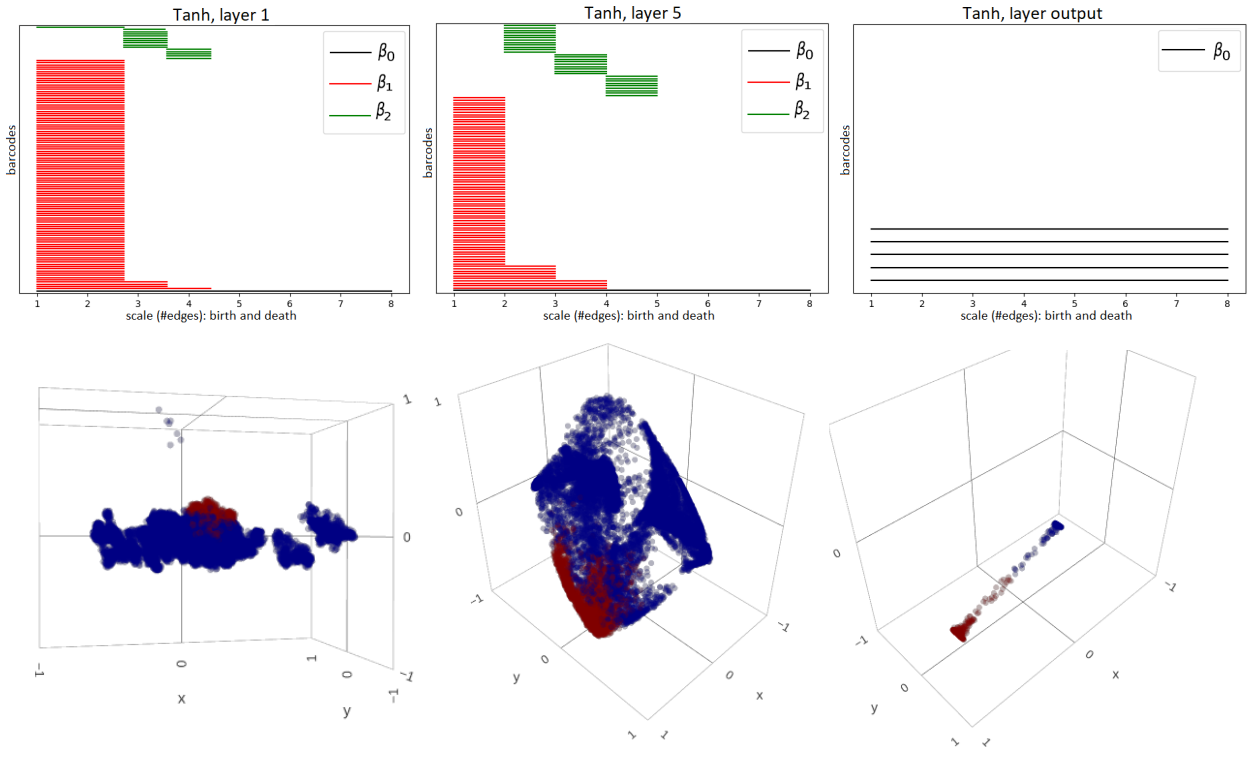


Figure 2.19: Persistence barcode diagrams for neural networks trained on UCI Banknotes data show topology changes in the 'manifold of genuine banknotes' M_a as it passes through layers activated with tanh (top) and ReLU (bottom). Scatter plots show principal component projections of M_a (red) and the 'manifold of forged banknotes' M_b (blue) at the corresponding layers.

sample #	1	2	3	4	5
Elect. curr. 1	-3.015×10^{-7}	-2.952×10^{-6}	-2.952×10^{-6}	-4.961×10^{-6}	-6.501×10^{-6}
Elect. curr. 2	8.260×10^{-6}	-5.248×10^{-6}	-3.184×10^{-6}	-2.088×10^{-6}	-6.208×10^{-6}
⋮ ⋮	⋮ ⋮	⋮ ⋮	⋮ ⋮	⋮ ⋮	⋮ ⋮
Elect. curr. 49	-1.500×10^{0}	-1.501×10^{0}	-1.496×10^{0}	-1.497×10^{0}	-1.500×10^{0}
Failure types	a	a	a	b	b

Table 2.6: Five entries from the UCI Drive data set. Last row indicates whether the failure is of type 'a' or one of the other eleven types, all of which are indicated as 'b'.

'manifolds of all other types of defects.' Of the 58,509 entries in the UCI Drive data, we choose a random subset of 10,600 as our point cloud $X \subseteq M \subseteq \mathbb{R}^{49}$. The rest of the experiments is exactly as in the previous two cases (HTRU2 and UCI Banknotes data). The results are shown in Figure 2.20 and they are fully consistent with the results in Figures 2.17 and 2.19, supporting the same conclusions we drew from all previously examined data sets.



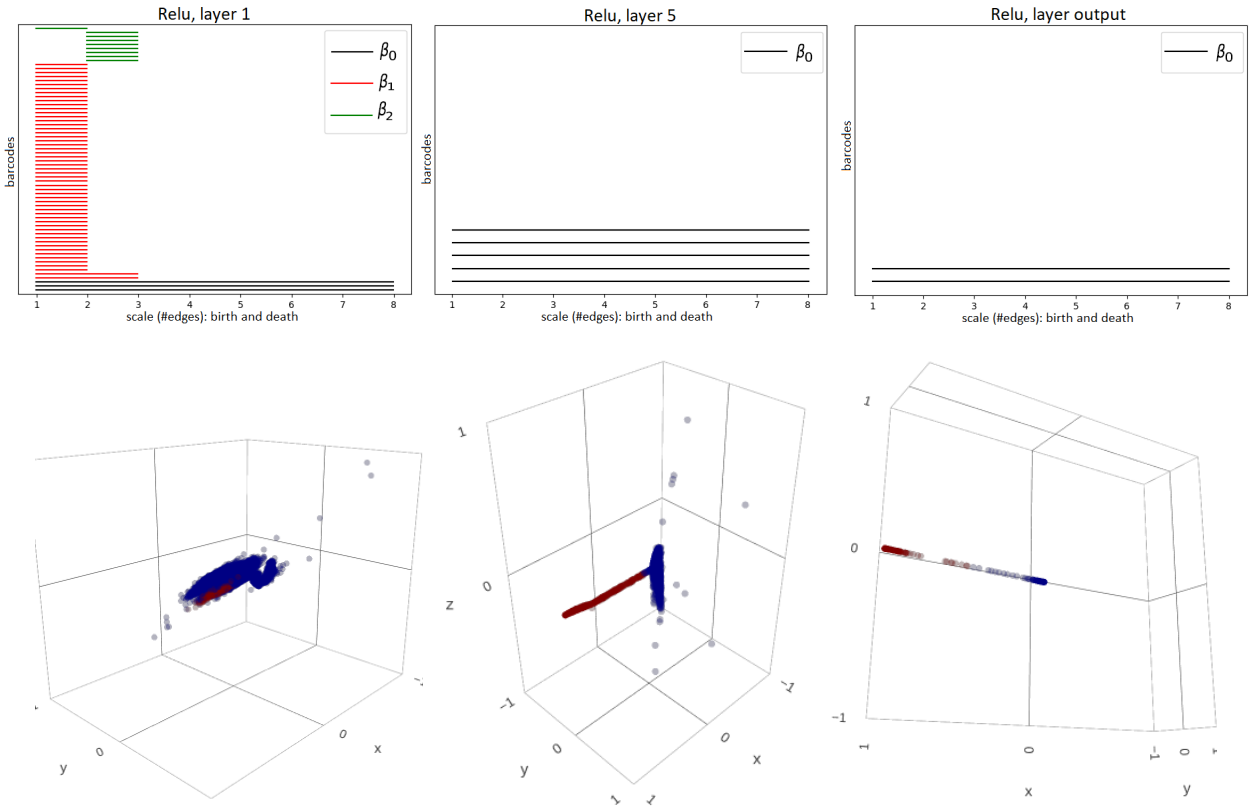
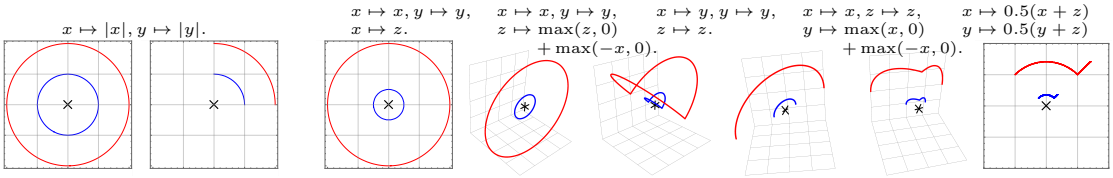


Figure 2.20: Persistence barcode diagrams for neural networks trained on UCI Drive data show topology changes in the 'manifold of type a defects' M_a as it passes through layers activated with tanh (previous page) and ReLU (above). Scatter plots show principal component projections of M_a (red) and the 'manifolds of all other types of defects' M_b (blue) at the corresponding layers.

2.9 Concluding discussions

Our findings support the view that deep neural networks operate by transforming topology, gradually simplifying topologically complicated data shapes and arrangements in the input space until it becomes linearly separable in the output space. We proffered some new insights on the roles of the deep layers and of rectified activations, namely, that they are mechanisms that aid topological changes. As this article is an empirical study intended to provide evidence for the above point-of-view, we did not investigate the actual mechanics of how a ReLU-activated neural network carries out topological changes. We conclude our article with a few speculative words about the 'topology changing mechanism' of neural networks, mainly to serve as pointers for future work.

Consider the concentric red and blue circles on the left of Figure 2.21a, two one-dimensional manifolds embedded in $\mathbb{R}^2$. By the Jordan Curve Theorem, there is no homeomorphism $\mathbb{R}^2 \to \mathbb{R}^2$ that will transform the two circles into two sets that can be separated by a hyperplane in $\mathbb{R}^2$. Nevertheless it is easy to achieve this with a *many-to-one map* like $(x, y) \mapsto (|x|, |y|)$ as shown in Figure 2.21a that allows one to 'fold' a set. An alternative way to achieve this is with an *embedding into higher dimensional space* like in Figure 2.21b where we did $\mathbb{R}^2 \to \mathbb{R}^3 \to \mathbb{R}^3 \to \mathbb{R}^3 \to \mathbb{R}^3 \to \mathbb{R}^2$ with operations within $\mathbb{R}^3$ that disentangles the red and blue circles. Our speculation is that in a neural network, (i) the ReLU activation is a many-to-one map that can 'fold' a space; (ii) the excess width[4] of the intermediate layers affords a higher dimensional *space* in which to transform the data; (iii) the depth on the other hand plays the role of *time*, every additional layer affords additional time to transform the data. To elaborate on the last point (iii), note that since we are limited to affine transformations and ReLU activation, a substantial change to the topology of a space may require a longer sequence of these operations, and by 'time', we simply mean the length of this sequence.



(a) many-to-one maps (b) embedding into higher dimensions

Figure 2.21: Examples of topology change: (a) Two neurons activated with the absolute value function can disentangle two concentric circles in a single step, transforming them into linearly separable sets. (b) The same can be achieved by first mapping to a higher dimension space and performing the disentangling operations therein.

Adding to the first point (i), since an affine map takes the form $x \mapsto U\Sigma Vx+b$, it provides the capability to translate (by the vector b), rotate/reflect (by the orthogonal matrices U and V), and stretch/shrink (by the nonnegative diagonal matrix Σ); ReLU-activation adds folding to the arsenal — an important capability. For example, to transform the surface

4. Width in excess of input dimension.

of a donut (torus) into the surface of a croissant (sphere) as in Figure 2.22, the first two operations may be achieved with appropriate affine maps but the last one requires that we fold the doubly-pinched torus into a croissant surface.

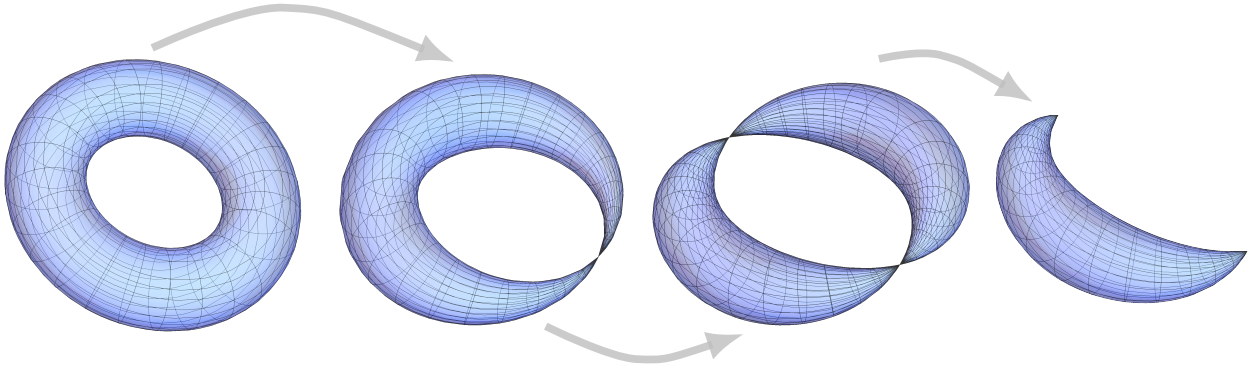


Figure 2.22: Donut to croissant: torus $\rightarrow$ pinched torus $\rightarrow$ doubly-pinched torus $\rightarrow$ sphere. Betti numbers: $(1, 2, 1) \rightarrow (1, 1, 1) \rightarrow (1, 1, 2) \rightarrow (1, 0, 1)$.

CHAPTER 3

TROPICAL ALGEBRA OF DEEP NEURAL NETWORKS

3.1 Introduction

The goal of our work in this chapter is to establish connections between neural network and tropical geometry in the hope that they will shed light on the workings of deep neural networks. Tropical geometry is a new area in algebraic geometry that has seen an explosive growth in the recent decade but remains relatively obscure outside pure mathematics. We will focus on feedforward neural networks with rectified linear units (ReLU) and show that they are analogues of *rational functions*, i.e., ratios of two multivariate polynomials f, g in variables $x_1, \dots, x_d$,

$$\frac{f(x_1, \dots, x_d)}{g(x_1, \dots, x_d)},$$

in *tropical algebra*. For standard and trigonometric polynomials, it is known that *rational approximation* — approximating a target function by a ratio of two polynomials instead of a single polynomial — vastly improves the quality of approximation without increasing the degree. This gives our analogue: An ReLU neural network is the tropical ratio of two tropical polynomials, i.e., a tropical rational function. More precisely, if we view a neural network as a function $\nu : \mathbb{R}^d \to \mathbb{R}^p$, $x = (x_1, \dots, x_d) \mapsto (\nu_1(x), \dots, \nu_p(x))$, then ν is a tropical rational map, i.e., each ν_i is a tropical rational function. In fact, we will show that:

> *the family of functions represented by feedforward neural networks with rectified linear units and integer weights is exactly the family of tropical rational maps.*

It immediately follows that there is a *semifield* structure on this family of functions. More importantly, this establishes a bridge between neural networks[1] and tropical geometry that allows us to view neural networks as well-studied tropical geometric objects. This insight allows us to closely relate boundaries between linear regions of a neural network to tropical

1. Henceforth a "neural network" will always mean a feedforward neural network with ReLU activation.

hypersurfaces and thereby facilitate studies of decision boundaries of neural networks in classification problems as tropical hypersurfaces. Furthermore, the number of linear regions, which captures the complexity of a neural network [63, 77, 2], can be bounded by the number of vertices of the polytopes associated with the neural network's tropical rational representation. Lastly, a neural network with one hidden layer can be completely characterized by zonotopes, which serve as building blocks for deeper networks.

In Sections 3.2 and 3.3 we introduce basic tropical algebra and tropical algebraic geometry of relevance to us. We state our assumptions precisely in Section 3.4 and establish the connection between tropical geometry and multilayer neural networks in Section 3.5. We analyze neural networks with tropical tools in Section 3.6, proving that a deeper neural network is exponentially more expressive than a shallow network — though our objective is not so much to perform state-of-the-art analysis but to demonstrate that tropical algebraic geometry can provide useful insights. All proofs are deferred to Section 3.8.

3.2 Tropical Algebra

Roughly speaking, tropical algebraic geometry is an analogue of classical algebraic geometry over $\mathbb{C}$, the field of complex numbers, but where one replaces $\mathbb{C}$ by a semifield[2] called the tropical semiring, to be defined below. We give a brief review of tropical algebra and introduce some relevant notations. See [41, 56] for an in-depth treatment.

The most fundamental component of tropical algebraic geometry is the *tropical semiring* $\mathbb{T} \coloneqq \big(\mathbb{R} \cup \{-\infty\}, \oplus, \odot\big)$, also known as the max-plus algebra. The two operations $\oplus$ and $\odot$, called *tropical addition* and *tropical multiplication* respectively, are defined as follows.

Definition 3.2.1. *For $x, y \in \mathbb{R}$, their* tropical sum *is $x \oplus y \coloneqq \max\{x, y\}$; their* tropical product *is $x \odot y \coloneqq x + y$; the* tropical quotient *of x over y is $x \oslash y \coloneqq x - y$.*

For any $x \in \mathbb{R}$, we have $-\infty \oplus x = 0 \odot x = x$ and $-\infty \odot x = -\infty$. Thus $-\infty$ is the tropical

2. A semifield is a field sans the existence of additive inverses.

additive identity and 0 is the tropical multiplicative identity. Furthermore, these operations satisfy the usual laws of arithmetic: associativity, commutativity, and distributivity. The set $\mathbb{R} \cup \{-\infty\}$ is therefore a semiring under the operations $\oplus$ and $\odot$. While it is not a ring (lacks additive inverse), one may nonetheless generalize many algebraic objects (e.g., matrices, polynomials, tensors, etc) and notions (e.g., rank, determinant, degree, etc) over the tropical semiring — the study of these, in a nutshell, constitutes the subject of tropical algebra.

Let $\mathbb{N} = \{n \in \mathbb{Z} : n \geq 0\}$. For an integer $a \in \mathbb{N}$, raising $x \in \mathbb{R}$ to the ath power is the same as multiplying x to itself a times. When standard multiplication is replaced by tropical multiplication, this gives us *tropical power*:

$$x^{\odot a} := x \odot \cdots \odot x = a \cdot x,$$

where the last $\cdot$ denotes standard product of real numbers; it is extended to $\mathbb{R} \cup \{-\infty\}$ by defining, for any $a \in \mathbb{N}$,

$$-\infty^{\odot a} := \begin{cases} -\infty & \text{if } a > 0, \\ 0 & \text{if } a = 0. \end{cases}$$

A tropical semiring, while not a field, possesses one quality of a field: Every $x \in \mathbb{R}$ has a tropical multiplicative inverse given by its standard additive inverse, i.e., $x^{\odot(-1)} := -x$. Though not reflected in its name, $\mathbb{T}$ is in fact a *semifield*.

One may therefore also raise $x \in \mathbb{R}$ to a negative power $a \in \mathbb{Z}$ by raising its tropical multiplicative inverse $-x$ to the positive power $-a$, i.e., $x^{\odot a} = (-x)^{\odot(-a)}$. As is the case in standard real arithmetic, the tropical additive inverse $-\infty$ does not have a tropical multiplicative inverse and $-\infty^{\odot a}$ is undefined for $a < 0$. For notational simplicity, we will henceforth write x^a instead of $x^{\odot a}$ for tropical power when there is no cause for confusion. Other algebraic rules of tropical power may be derived from definition; see Section 3.7.2.

We are now in a position to define tropical polynomials and tropical rational functions.

In the following, x and x_i will denote variables (i.e., indeterminates).

Definition 3.2.2. *A* tropical monomial *in d variables $x_1, \ldots, x_d$ is an expression of the form*

$$c \odot x_1^{a_1} \odot x_2^{a_2} \odot \cdots \odot x_d^{a_d}$$

where $c \in \mathbb{R} \cup \{-\infty\}$ and $a_1, \ldots, a_d \in \mathbb{N}$. As a convenient shorthand, we will also write a tropical monomial in multiindex notation as cx^α where $\alpha = (a_1, \ldots, a_d) \in \mathbb{N}^d$ and $x = (x_1, \ldots, x_d)$. Note that $x^\alpha = 0 \odot x^\alpha$ as 0 is the tropical multiplicative identity.

Definition 3.2.3. *Following notations above, a* tropical polynomial *$f(x) = f(x_1, \ldots, x_d)$ is a finite tropical sum of tropical monomials*

$$f(x) = c_1 x^{\alpha_1} \oplus \cdots \oplus c_r x^{\alpha_r},$$

where $\alpha_i = (a_{i1}, \ldots, a_{id}) \in \mathbb{N}^d$ and $c_i \in \mathbb{R} \cup \{-\infty\}$, $i = 1, \ldots, r$. We will assume that a monomial of a given multiindex appears at most once in the sum, i.e., $\alpha_i \neq \alpha_j$ for any $i \neq j$.

Definition 3.2.4. *Following notations above, a* tropical rational function *is a standard difference, or, equivalently, a tropical quotient of two tropical polynomials $f(x)$ and $g(x)$:*

$$f(x) - g(x) = f(x) \oslash g(x).$$

We will denote a tropical rational function by $f \oslash g$, where f and g are understood to be tropical polynomial functions.

It is routine to verify that the set of tropical polynomials $\mathbb{T}[x_1, \ldots, x_d]$ forms a semiring under the standard extension of $\oplus$ and $\odot$ to tropical polynomials, and likewise the set of tropical rational functions $\mathbb{T}(x_1, \ldots, x_d)$ forms a semifield. We regard a tropical polynomial $f = f \oslash 0$ as a special case of a tropical rational function and thus $\mathbb{T}[x_1, \ldots, x_d] \subseteq$

$\mathbb{T}(x_1, \ldots, x_d)$. Henceforth any result stated for a tropical rational function would implicitly also hold for a tropical polynomial.

A d-variate tropical polynomial $f(x)$ defines a function $f : \mathbb{R}^d \to \mathbb{R}$ that is a *convex function* in the usual sense as taking max and sum of convex functions preserve convexity [10]. As such, a tropical rational function $f \oslash g : \mathbb{R}^d \to \mathbb{R}$ is a *DC function* or *difference-convex function* [36, 85].

We will need a notion of vector-valued tropical polynomials and tropical rational functions.

Definition 3.2.5. *$F : \mathbb{R}^d \to \mathbb{R}^p$, $x = (x_1, \ldots, x_d) \mapsto (f_1(x), \ldots, f_p(x))$, is called a* tropical polynomial map *if each $f_i : \mathbb{R}^d \to \mathbb{R}$ is a tropical polynomial, $i = 1, \ldots, p$, and a* tropical rational map *if $f_1, \ldots, f_p$ are tropical rational functions. We will denote the set of tropical polynomial maps by $\mathrm{Pol}(d, p)$ and the set of tropical rational maps by $\mathrm{Rat}(d, p)$. So $\mathrm{Pol}(d, 1) = \mathbb{T}[x_1, \ldots, x_d]$ and $\mathrm{Rat}(d, 1) = \mathbb{T}(x_1, \ldots, x_d)$.*

3.3 Tropical Hypersurfaces

There are tropical analogues of many notions in classical algebraic geometry [41, 56], among which are *tropical hypersurfaces*, tropical analogues of algebraic curves in classical algebraic geometry. Tropical hypersurfaces are a principal object of interest in tropical geometry and will prove very useful in our approach towards neural networks. Intuitively, the tropical hypersurface of a tropical polynomial f is the set of points x where f is not linear at x.

Definition 3.3.1. *The* tropical hypersurface *of a tropical polynomial $f(x) = c_1 x^{\alpha_1} \oplus \cdots \oplus c_r x^{\alpha_r}$ is*

$$\mathcal{T}(f) \coloneqq \left\{ x \in \mathbb{R}^d : c_i x^{\alpha_i} = c_j x^{\alpha_j} = f(x) \text{ for some } \alpha_i \neq \alpha_j \right\},$$

i.e., the set of points x at which the value of f at x is attained by two or more monomials in f.

A tropical hypersurface divides the domain of f into convex cells on each of which f is linear. These cells are convex polyhedrons, i.e., defined by linear inequalities with integer

coefficients: $\{x \in \mathbb{R}^d : Ax \leq b\}$ for $A \in \mathbb{Z}^{m\times d}$ and $b \in \mathbb{R}^m$. For example, the cell where a tropical monomial $c_j x^{\alpha_j}$ attains its maximum is $\{x \in \mathbb{R}^d : c_j + \alpha_j^\mathsf{T} x \geq c_i + \alpha_i^\mathsf{T} x \text{ for all } i \neq j\}$. Tropical hypersurfaces of polynomials in two variables (i.e., in $\mathbb{R}^2$) are called *tropical curves*.

Just like standard multivariate polynomials, every tropical polynomial comes with an associated *Newton polygon*.

Definition 3.3.2. *The* Newton polygon *of a tropical polynomial* $f(x) = c_1 x^{\alpha_1} \oplus \cdots \oplus c_r x^{\alpha_r}$ *is the convex hull of* $\alpha_1, \ldots, \alpha_r \in \mathbb{N}^d$*, regarded as points in* $\mathbb{R}^d$*,*

$$\Delta(f) \coloneqq \mathrm{Conv}\big\{\alpha_i \in \mathbb{R}^d : c_i \neq -\infty,\ i = 1, \ldots, r\big\}.$$

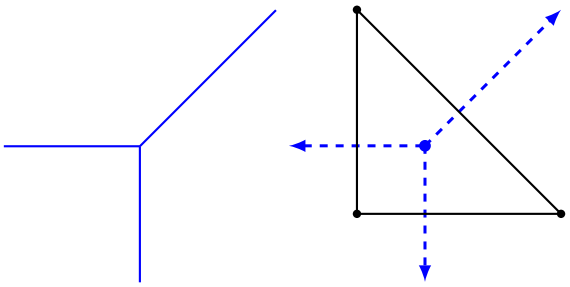


Figure 3.1: $x \oplus y \oplus 0$. *Left*: Tropical curve. *Right*: (Dual subdivision of) the Newton polygon and tropical curve.

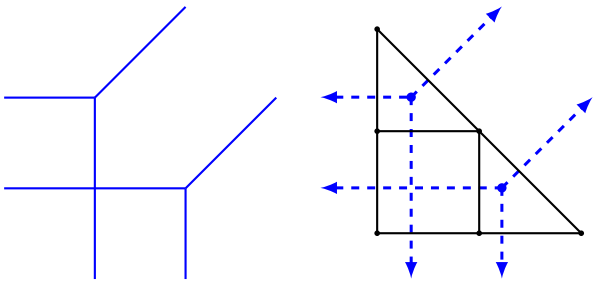


Figure 3.2: $1 \odot x_1^2 \oplus 1 \odot x_2^2 \oplus 2 \odot x_1 x_2 \oplus 2 \odot x_1 \oplus 2 \odot x_2 \oplus 2$. *Left*: Tropical curve. *Right*: Dual subdivision of the Newton polygon and tropical curve.

A tropical polynomial f determines a dual subdivision of $\Delta(f)$, constructed as follows. First, lift each α_i from $\mathbb{R}^d$ into $\mathbb{R}^{d+1}$ by appending c_i as the last coordinate. Denote the

convex hull of the lifted $\alpha_1, \ldots, \alpha_r$ as

$$\mathcal{P}(f) \coloneqq \text{Conv}\{(\alpha_i, c_i) \in \mathbb{R}^d \times \mathbb{R} : i = 1, \ldots, r\}. \tag{3.1}$$

Next let $\text{UF}\big(\mathcal{P}(f)\big)$ denote the collection of upper faces in $\mathcal{P}(f)$ and $\pi : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ be the projection that drops the last coordinate. The dual subdivision determined by f is then

$$\delta(f) \coloneqq \big\{\pi(p) \subseteq \mathbb{R}^d : p \in \text{UF}\big(\mathcal{P}(f)\big)\big\}.$$

$\delta(f)$ forms a polyhedral complex with support $\Delta(f)$. By [56, Proposition 3.1.6], the tropical hypersurface $\mathcal{T}(f)$ is the $(d-1)$-skeleton of the polyhedral complex dual to $\delta(f)$. This means that each vertex in $\delta(f)$ corresponds to one "cell" in $\mathbb{R}^d$ where the function f is linear. Thus, the number of vertices in $\mathcal{P}(f)$ provides an upper bound on the number of linear regions of f.

Figure 3.2 shows the Newton polygon and dual subdivision for the tropical polynomial $f(x_1, x_2) = 1 \odot x_1^2 \oplus 1 \odot x_2^2 \oplus 2 \odot x_1 x_2 \oplus 2 \odot x_1 \oplus 2 \odot x_2 \oplus 2$. Figure 3.3 shows how we may find the dual subdivision for this tropical polynomial by following the aforementioned procedures; with step-by-step details given in Section 3.7.3.

Tropical polynomials and tropical rational functions are clearly piecewise linear functions. As such a tropical rational map is a piecewise linear map and the notion of *linear region* applies.

Definition 3.3.3. *A* linear region *of $F \in \text{Rat}(d, m)$ is a maximal connected subset of the domain on which F is linear. The number of linear regions of F is denoted $\mathcal{N}(F)$.*

Note that a tropical *polynomial* map $F \in \text{Pol}(d, m)$ has convex linear regions but a tropical *rational* map $F \in \text{Rat}(d, n)$ generally has nonconvex linear regions. In Section 3.6.3, we will use $\mathcal{N}(F)$ as a measure of complexity for an $F \in \text{Rat}(d, n)$ given by a neural network.

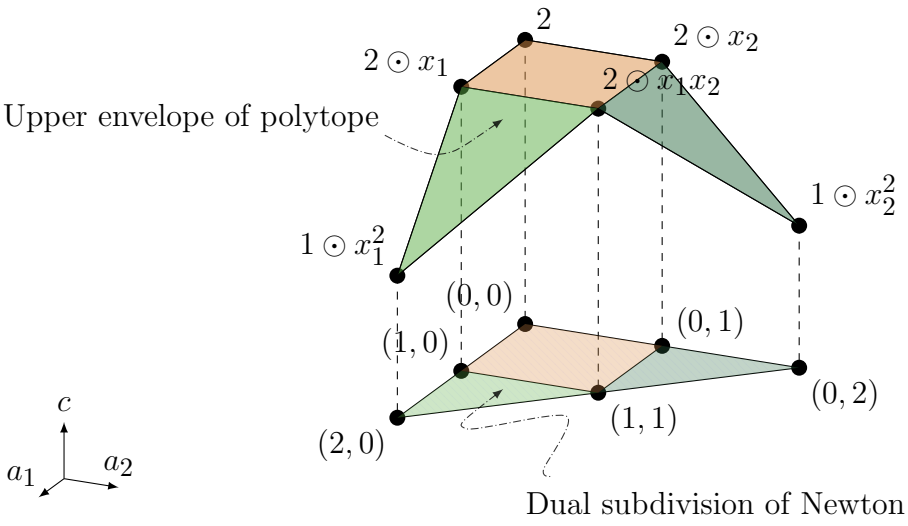


Figure 3.3: $1 \odot x_1^2 \oplus 1 \odot x_2^2 \oplus 2 \odot x_1 x_2 \oplus 2 \odot x_1 \oplus 2 \odot x_2 \oplus 2$. The dual subdivision can be obtained by projecting the edges on the upper faces of the polytope.

3.3.1 Transformations of Tropical Polynomials

Our analysis of neural networks will require figuring out how the polytope $\mathcal{P}(f)$ transforms under tropical power, sum, and product. The first is straightforward.

Proposition 3.3.4. *Let f be a tropical polynomial and let $a \in \mathbb{N}$. Then*

$$\mathcal{P}(f^a) = a\mathcal{P}(f).$$

$a\mathcal{P}(f) = \{ax : x \in \mathcal{P}(f)\} \subseteq \mathbb{R}^{d+1}$ is a scaled version of $\mathcal{P}(f)$ with the same shape but different volume.

To describe the effect of tropical sum and product, we need a few notions from convex geometry. The *Minkowski sum* of two sets P_1 and P_2 in $\mathbb{R}^d$ is the set

$$P_1 + P_2 \coloneqq \left\{x_1 + x_2 \in \mathbb{R}^d : x_1 \in P_1, x_2 \in P_2\right\};$$

and for $\lambda_1, \lambda_2 \geq 0$, their *weighted Minkowski sum* is

$$\lambda_1 P_1 + \lambda_2 P_2 \coloneqq \left\{\lambda_1 x_1 + \lambda_2 x_2 \in \mathbb{R}^d : x_1 \in P_1, x_2 \in P_2\right\}.$$

Weighted Minkowski sum is clearly commutative and associative and generalizes to more than two sets. In particular, the Minkowski sum of line segments is called a *zonotope*.

Let $\mathcal{V}(P)$ denote the set of vertices of a polytope P. Clearly, the Minkowski sum of two polytopes is given by the convex hull of the Minkowski sum of their vertex sets, i.e., $P_1 + P_2 = \mathrm{Conv}\big(\mathcal{V}(P_1) + \mathcal{V}(P_2)\big)$. With this observation, the following is immediate.

Proposition 3.3.5. *Let $f, g \in \mathrm{Pol}(d,1) = \mathbb{T}[x_1, \ldots, x_d]$ be tropical polynomials. Then*

$$\mathcal{P}(f \odot g) = \mathcal{P}(f) + \mathcal{P}(g),$$
$$\mathcal{P}(f \oplus g) = \mathrm{Conv}\big(\mathcal{V}(\mathcal{P}(f)) \cup \mathcal{V}(\mathcal{P}(g))\big).$$

We reproduce below part of [32, Theorem 2.1.10] and derive a corollary for bounding the number of verticies on the upper faces of a zonotope.

Theorem 3.3.6 (Gritzmann–Sturmfels)**.** *Let $P_1, \ldots, P_k$ be polytopes in $\mathbb{R}^d$ and let m denote the total number of nonparallel edges of $P_1, \ldots, P_k$. Then the number of vertices of $P_1 + \cdots + P_k$ does not exceed*

$$2\sum_{j=0}^{d-1} \binom{m-1}{j}.$$

The upper bound is attained if all P_i's are zonotopes and all their generating line segments are in general positions.

Corollary 3.3.7. *Let $P \subseteq \mathbb{R}^{d+1}$ be a zonotope generated by m line segments $P_1, \ldots, P_m$. Let $\pi : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ be the projection. Suppose P satisfies:*

(i) *the generating line segments are in general positions;*

(ii) *the set of projected vertices $\{\pi(v) : v \in \mathcal{V}(P)\} \subseteq \mathbb{R}^d$ are in general positions.*

Then P has

$$\sum_{j=0}^{d} \binom{m}{j}$$

vertices on its upper faces. If either (i) *or* (ii) *is violated, then this becomes an upper bound.*

As we mentioned, linear regions of a tropical polynomial f correspond to vertices on $\mathrm{UF}\big(\mathcal{P}(f)\big)$ and the corollary will be useful for bounding the number of linear regions.

3.4 Neural Networks

While we expect our readership to be familiar with feedforward neural networks, we will nevertheless use this short section to define them, primarily for the purpose of fixing notations and specifying the assumptions that we retain throughout this chapter. We restrict our attention to fully connected feedforward neural networks.

Viewed abstractly, an L-layer feedforward neural network is a map $\nu : \mathbb{R}^d \to \mathbb{R}^p$ given by a composition of functions

$$\nu = \sigma^{(L)} \circ \rho^{(L)} \circ \sigma^{(L-1)} \circ \rho^{(L-1)} \dots \circ \sigma^{(1)} \circ \rho^{(1)}.$$

The *preactivation* functions $\rho^{(1)}, \dots, \rho^{(L)}$ are affine transformations to be determined and the *activation* functions $\sigma^{(1)}, \dots, \sigma^{(L)}$ are chosen and fixed in advanced.

We denote the *width*, i.e., the number of nodes, of the lth layer by n_l, $l = 1, \cdots, L-1$. We set $n_0 \coloneqq d$ and $n_L \coloneqq p$, respectively the dimensions of the input and output of the network. The output from the lth layer will be denoted by

$$\nu^{(l)} \coloneqq \sigma^{(l)} \circ \rho^{(l)} \circ \sigma^{(l-1)} \circ \rho^{(l-1)} \dots \circ \sigma^{(1)} \circ \rho^{(1)},$$

i.e., it is a map $\nu^{(l)} : \mathbb{R}^d \to \mathbb{R}^{n_l}$. For convenience, we assume $\nu^{(0)}(x) \coloneqq x$.

The affine function $\rho^{(l)} : \mathbb{R}^{n_{l-1}} \to \mathbb{R}^{n_l}$ is given by a *weight* matrix $A^{(l)} \in \mathbb{Z}^{n_l \times n_{l-1}}$ and a *bias* vector $b^{(l)} \in \mathbb{R}^{n_l}$:

$$\rho^{(l)}(\nu^{(l-1)}) \coloneqq A^{(l)}\nu^{(l-1)} + b^{(l)}.$$

The (i,j)th coordinate of $A^{(l)}$ will be denoted $a_{ij}^{(l)}$ and the ith coordinate of $b^{(l)}$ by $b_i^{(l)}$.

Collectively they form the *parameters* of the lth layer.

For a vector input $x \in \mathbb{R}^{n_l}$, $\sigma^{(l)}(x)$ is understood to be in the coordinatewise sense; so $\sigma : \mathbb{R}^{n_l} \to \mathbb{R}^{n_l}$. We assume the final output of a neural network $\nu(x)$ is fed into a *score function* $s : \mathbb{R}^p \to \mathbb{R}^m$ that is application specific. When used as an m-category classifier, s may be chosen, for example, to be a soft-max or sigmoidal function. The score function is quite often regarded as the last layer of a neural network but this is purely a matter of convenience and we will not assume this. We will make the following mild assumptions on the architecture of our feedforward neural networks and explain next why they are indeed mild:

(a) the weight matrices $A^{(1)}, \dots, A^{(L)}$ are integer-valued;

(b) the bias vectors $b^{(1)}, \dots, b^{(L)}$ are real-valued;

(c) the activation functions $\sigma^{(1)}, \dots, \sigma^{(L)}$ take the form

$$\sigma^{(l)}(x) := \max\{x, t^{(l)}\},$$

where $t^{(l)} \in (\mathbb{R} \cup \{-\infty\})^{n_l}$ is called a *threshold* vector.

Henceforth all neural networks in our subsequent discussions will be assumed to satisfy (a)–(c).

(b) is completely general but there is also no loss of generality in (a), i.e., in restricting the weights $A^{(1)}, \dots, A^{(L)}$ from real matrices to integer matrices, as:

- real weights can be approximated arbitrarily closely by rational weights;
- one may then 'clear denominators' in these rational weights by multiplying them by the least common multiple of their denominators to obtain integer weights;
- keeping in mind that scaling all weights and biases by the same positive constant has no bearing on the workings of a neural network.

The activation function in (c) includes both ReLU activation ($t^{(l)} = 0$) and identity map ($t^{(l)} = -\infty$) as special cases. Aside from ReLU, our tropical framework will apply to

piecewise linear activations such as leaky ReLU and absolute value, and with some extra effort, may be extended to max pooling, maxout nets, etc. But it does not, for example, apply to activations such as hyperbolic tangent and sigmoid.

In this work, we view an ReLU network as the simplest and most canonical model of a neural network, from which other variants that are more effective at specific tasks may be derived. Given that we seek general theoretical insights and not specific practical efficacy, it makes sense to limit ourselves to this simplest case. Moreover, ReLU networks already embody some of the most important elements (and mysteries) common to a wider range of neural networks (e.g., universal approximation, exponential expressiveness); they work well in practice and are often the go-to choice for feedforward networks. We are also not alone in limiting our discussions to ReLU networks [63, 2].

3.5 Tropical Algebra of Neural Networks

We now describe our tropical formulation of a multilayer feedforward neural network satisfying (a)–(c).

A multilayer feedforward neural network is generally nonconvex, whereas a tropical polynomial is always convex. Since most nonconvex functions are a difference of two convex functions [36], a reasonable guess is that a feedforward neural network is the difference of two tropical polynomials, i.e., a tropical rational function. This is indeed the case, as we will see from the following.

Consider the output from the first layer in neural network

$$\nu(x) = \max\{Ax + b,\, t\},$$

where $A \in \mathbb{Z}^{p\times d}, b \in \mathbb{R}^p$, and $t \in (\mathbb{R} \cup \{-\infty\})^p$. We will decompose A as a difference of two nonnegative integer-valued matrices, $A = A_+ - A_-$ with $A_+, A_- \in \mathbb{N}^{p\times d}$; e.g., in the

standard way with entries

$$a_{ij}^{+} \coloneqq \max\{a_{ij}, 0\}, \qquad a_{ij}^{-} \coloneqq \max\{-a_{ij}, 0\}$$

respectively. Since

$$\max\{Ax + b, t\} = \max\{A_{+}x + b,\, A_{-}x + t\} - A_{-}x,$$

we see that every coordinate of one-layer neural network is a difference of two tropical polynomials. For networks with more layers, we apply this decomposition recursively to obtain the following result.

Proposition 3.5.1. *Let $A \in \mathbb{Z}^{m\times n}$, $b \in \mathbb{R}^m$ be the parameters of the $(l+1)$th layer, and let $t \in (\mathbb{R} \cup \{-\infty\})^m$ be the threshold vector in the $(l+1)$th layer. If the nodes of the lth layer are given by tropical rational functions,*

$$\nu^{(l)}(x) = F^{(l)}(x) \oslash G^{(l)}(x) = F^{(l)}(x) - G^{(l)}(x),$$

i.e., each coordinate of $F^{(l)}$ and $G^{(l)}$ is a tropical polynomial in x, then the outputs of the preactivation and of the $(l+1)$th layer are given by tropical rational functions

$$\begin{aligned} \rho^{(l+1)} \circ \nu^{(l)}(x) &= H^{(l+1)}(x) - G^{(l+1)}(x), \\ \nu^{(l+1)}(x) = \sigma \circ \rho^{(l+1)} \circ \nu^{(l)}(x) &= F^{(l+1)}(x) - G^{(l+1)}(x) \end{aligned}$$

respectively, where

$$\begin{aligned} F^{(l+1)}(x) &= \max\big\{H^{(l+1)}(x),\, G^{(l+1)}(x) + t\big\}, \\ G^{(l+1)}(x) &= A_{+}G^{(l)}(x) + A_{-}F^{(l)}(x), \\ H^{(l+1)}(x) &= A_{+}F^{(l)}(x) + A_{-}G^{(l)}(x) + b. \end{aligned}$$

We will write $f_i^{(l)}$, $g_i^{(l)}$ *and* $h_i^{(l)}$ *for the* i*th coordinate of* $F^{(l)}$, $G^{(l)}$ *and* $H^{(l)}$ *respectively. In tropical arithmetic, the recurrence above takes the form*

$$\begin{aligned} f_i^{(l+1)} &= h_i^{(l+1)} \oplus (g_i^{(l+1)} \odot t_i), \\ g_i^{(l+1)} &= \left[\bigodot_{j=1}^{n} (f_j^{(l)})^{a_{ij}^-}\right] \odot \left[\bigodot_{j=1}^{n} (g_j^{(l)})^{a_{ij}^+}\right], \\ h_i^{(l+1)} &= \left[\bigodot_{j=1}^{n} (f_j^{(l)})^{a_{ij}^+}\right] \odot \left[\bigodot_{j=1}^{n} (g_j^{(l)})^{a_{ij}^-}\right] \odot b_i. \end{aligned} \tag{3.2}$$

Repeated applications of Proposition 3.5.1 yield the following.

Theorem 3.5.2 (Tropical characterization of neural networks)**.** *A feedforward neural network under assumptions* (a)–(c) *is a function* $\nu : \mathbb{R}^d \to \mathbb{R}^p$ *whose coordinates are tropical rational functions of the input, i.e.,*

$$\nu(x) = F(x) \oslash G(x) = F(x) - G(x)$$

where F *and* G *are tropical polynomial maps. Thus* ν *is a tropical rational map.*

Note that the tropical rational functions above have real coefficients, not integer coefficients. The integer weights $A^{(l)} \in \mathbb{Z}^{n_l \times n_{l-1}}$ have gone into the powers of tropical monomials in f and g, which is why we require our weights to be integer-valued, although as we have explained, this requirement imposes little loss of generality.

By setting $t^{(1)} = \cdots = t^{(L-1)} = 0$ and $t^{(L)} = -\infty$, we obtain the following corollary.

Corollary 3.5.3. *Let* $\nu : \mathbb{R}^d \to \mathbb{R}$ *be an ReLU activated feedforward neural network with integer weights and linear output. Then* ν *is a tropical rational function.*

A more remarkable fact is the converse of Corollary 3.5.3.

Theorem 3.5.4 (Equivalence of neural networks and tropical rational functions)**.**

(i) *Let* $\nu : \mathbb{R}^d \to \mathbb{R}$*. Then* ν *is a tropical rational function if and only if* ν *is a feedforward neural network satisfying assumptions* (a)–(c)*.*

(ii) *A tropical rational function $f \oslash g$ can be represented as an L-layer neural network, with*

$$L \leq \max\{\lceil \log_2 r_f \rceil, \lceil \log_2 r_g \rceil\} + 2,$$

where r_f and r_g are the number of monomials in the tropical polynomials f and g respectively.

We would like to acknowledge the precedence of [2, Theorem 2.1], which demonstrates the equivalence between ReLU-activated L-layer neural networks with *real* weights and d-variate continuous piecewise functions with *real* coefficients, where $L \leq \lceil \log_2(d+1) \rceil + 1$.

By construction, a tropical rational function is a continuous piecewise linear function. The continuity of a piecewise linear function automatically implies that each of the pieces on which it is linear is a polyhedral region. As we saw in Section 3.3, a tropical polynomial $f : \mathbb{R}^d \to \mathbb{R}$ gives a tropical hypersurface that divides $\mathbb{R}^d$ into *convex* polyhedral regions defined by linear inequalities with integer coefficients: $\{x \in \mathbb{R}^d : Ax \leq b\}$ with $A \in \mathbb{Z}^{m \times d}$ and $b \in \mathbb{R}^m$. A tropical rational function $f \oslash g : \mathbb{R}^d \to \mathbb{R}$ must also be a continuous piecewise linear function and divide $\mathbb{R}^d$ into polyhedral regions on each of which $f \oslash g$ is linear, although these regions are *nonconvex* in general. We will show the converse — any continuous piecewise linear function with integer coefficients is a tropical rational function.

Proposition 3.5.5. *Let $\nu : \mathbb{R}^d \to \mathbb{R}$. Then ν is a continuous piecewise linear function with integer coefficients if and only if ν is a tropical rational function.*

Corollary 3.5.3, Theorem 3.5.4, and Proposition 3.5.5 collectively imply the equivalence of

(i) tropical rational functions,

(ii) continuous piecewise linear functions with integer coefficients,

(iii) neural networks satisfying assumptions (a)–(c).

An immediate advantage of this characterization is that the set of tropical rational functions $\mathbb{T}(x_1, \ldots, x_d)$ has a semifield structure as we pointed out in Section 3.2, a fact that we have

implicitly used in the proof of Proposition 3.5.5. However, what is more important is not the algebra but the *algebraic geometry* that arises from our tropical characterization. We will use tropical algebraic geometry to illuminate our understanding of neural networks in the next section.

The need to stay within tropical algebraic geometry is the reason we did not go for a simpler and more general characterization (that does not require the integer coefficients assumption). A *tropical signomial* takes the form

$$\varphi(x) = \bigoplus_{i=1}^{m} b_i \bigodot_{j=1}^{n} x_j^{a_{ij}},$$

where $a_{ij} \in \mathbb{R}$ and $b_i \in \mathbb{R} \cup \{-\infty\}$. Note that a_{ij} is not required to be integer-valued nor nonnegative. A *tropical rational signomial* is a tropical quotient $\varphi \oslash \psi$ of two tropical signomials φ, ψ. A *tropical rational signomial map* is a function $\nu = (\nu_1, \ldots, \nu_p) : \mathbb{R}^d \to \mathbb{R}^p$ where each $\nu_i : \mathbb{R}^d \to \mathbb{R}$ is a tropical rational signomial $\nu_i = \varphi_i \oslash \psi_i$. The same argument we used to establish Theorem 3.5.2 gives us the following.

Proposition 3.5.6. *Every feedforward neural network with ReLU activation is a tropical rational signomial map.*

Nevertheless tropical signomials fall outside the realm of tropical algebraic geometry and we do not use Proposition 3.5.6 in the rest of this chapter.

3.6 Tropical Geometry of Neural Networks

Section 3.5 defines neural networks via tropical algebra, a perspective that allows us to study them via tropical algebraic geometry. We will show that the decision boundary of a neural network is a subset of a tropical hypersurface of a corresponding tropical polynomial (Section 3.6.1). We will see that, in an appropriate sense, zonotopes form the geometric building blocks for neural networks (Section 3.6.2). We then prove that the geometry of the

function represented by a neural network grows vastly more complex as its number of layers increases (Section 3.6.3).

3.6.1 Decision Boundaries of a Neural Network

We will use tropical geometry and insights from Section 3.5 to study decision boundaries of neural networks, focusing on the case of two-category classification for clarity. As explained in Section 3.4, a neural network $\nu : \mathbb{R}^d \to \mathbb{R}^p$ together with a choice of score function $s : \mathbb{R}^p \to \mathbb{R}$ give us a classifier. If the output value $s(\nu(x))$ exceeds some decision threshold c, then the neural network predicts x is from one class (e.g., x is a CAT image), and otherwise x is from the other category (e.g., a DOG image). The input space is thereby partitioned into two disjoint subsets by the *decision boundary* $\mathcal{B} \coloneqq \{x \in \mathbb{R}^d : \nu(x) = s^{-1}(c)\}$. Connected regions with value above the threshold and connected regions with value below the threshold will be called the *positive regions* and *negative regions* respectively.

We provide bounds on the number of positive and negative regions and show that there is a tropical polynomial whose tropical hypersurface contains the decision boundary.

Proposition 3.6.1 (Tropical geometry of decision boundary)**.** *Let $\nu : \mathbb{R}^d \to \mathbb{R}$ be an L-layer neural network satisfying assumptions* (a)–(c) *with $t^{(L)} = -\infty$. Let the score function $s : \mathbb{R} \to \mathbb{R}$ be injective with decision threshold c in its range. If $\nu = f \oslash g$ where f and g are tropical polynomials, then*

(i) *its decision boundary $\mathcal{B} = \{x \in \mathbb{R}^d : \nu(x) = s^{-1}(c)\}$ divides $\mathbb{R}^d$ into at most $\mathcal{N}(f)$ connected positive regions and at most $\mathcal{N}(g)$ connected negative regions;*

(ii) *its decision boundary is contained in the tropical hypersurface of the tropical polynomial $s^{-1}(c) \odot g(x) \oplus f(x) = \max\{f(x),\, g(x) + s^{-1}(c)\}$, i.e.,*

$$\mathcal{B} \subseteq \mathcal{T}(s^{-1}(c) \odot g \oplus f). \tag{3.3}$$

The function $s^{-1}(c) \odot g \oplus f$ is not necessarily linear on every positive or negative region

and so its tropical hypersurface $\mathcal{T}(s^{-1}(c) \odot g \oplus f)$ may further divide a positive or negative region derived from $\mathcal{B}$ into multiple linear regions. Hence the "$\subseteq$" in (3.3) cannot in general be replaced by "=".

3.6.2 Zonotopes as Geometric Building Blocks of Neural Networks

From Section 3.3, we know that the number of regions a tropical hypersurface $\mathcal{T}(f)$ divides the space into equals the number of vertices in the dual subdivision of the Newton polygon associated with the tropical polynomial f. This allows us to bound the number of linear regions of a neural network by bounding the number of vertices in the dual subdivision of the Newton polygon.

We start by examining how geometry changes from one layer to the next in a neural network, more precisely:

Question. *How are the tropical hypersurfaces of the tropical polynomials in the* $(l+1)$*th layer of a neural network related to those in the* l*th layer?*

The recurrent relation (3.2) describes how the tropical polynomials occurring in the $(l+1)$th layer are obtained from those in the lth layer, namely, via three operations: tropical sum, tropical product, and tropical powers. Recall that a tropical hypersurface of a tropical polynomial is dual to the dual subdivision of the Newton polytope of the tropical polynomial, which is given by the projection of the upper faces on the polytopes defined by (3.1). Hence the question boils down to how these three operations transform the polytopes, which is addressed in Propositions 3.3.4 and 3.3.5. We follow notations in Proposition 3.5.1 for the next result.

Lemma 3.6.2. *Let* $f_i^{(l)}$, $g_i^{(l)}$, $h_i^{(l)}$ *be the tropical polynomials produced by the* i*th node in the* l*th layer of a neural network, i.e., they are defined by* (3.2)*. Then* $\mathcal{P}\big(f_i^{(l)}\big)$, $\mathcal{P}\big(g_i^{(l)}\big)$, $\mathcal{P}\big(h_i^{(l)}\big)$ *are subsets of* $\mathbb{R}^{d+1}$ *given as follows:*

(i) $\mathcal{P}\big(g_i^{(1)}\big)$ *and* $\mathcal{P}\big(h_i^{(1)}\big)$ *are points.*

(ii) $\mathcal{P}\big(f_i^{(1)}\big)$ *is a line segment.*

(iii) $\mathcal{P}\big(g_i^{(2)}\big)$ *and* $\mathcal{P}\big(h_i^{(2)}\big)$ *are zonotopes.*

(iv) *For* $l \geq 1$,

$$\mathcal{P}\big(f_i^{(l)}\big) = \mathrm{Conv}\big[\mathcal{P}\big(g_i^{(l)} \odot t_i^{(l)}\big) \cup \mathcal{P}\big(h_i^{(l)}\big)\big]$$

if $t_i^{(l)} \in \mathbb{R}$, *and* $\mathcal{P}\big(f_i^{(l)}\big) = \mathcal{P}\big(h_i^{(l)}\big)$ *if* $t_i^{(l)} = -\infty$.

(v) *For* $l \geq 1$, $\mathcal{P}\big(g_i^{(l+1)}\big)$ *and* $\mathcal{P}\big(h_i^{(l+1)}\big)$ *are weighted Minkowski sums,*

$$\mathcal{P}\big(g_i^{(l+1)}\big) = \sum_{j=1}^{n_l} a_{ij}^{-}\mathcal{P}\big(f_j^{(l)}\big) + \sum_{j=1}^{n_l} a_{ij}^{+}\mathcal{P}\big(g_j^{(l)}\big),$$
$$\mathcal{P}\big(h_i^{(l+1)}\big) = \sum_{j=1}^{n_l} a_{ij}^{+}\mathcal{P}\big(f_j^{(l)}\big) + \sum_{j=1}^{n_l} a_{ij}^{-}\mathcal{P}\big(g_j^{(l)}\big) + \{b_i e\},$$

where a_{ij}, b_i *are entries of the weight matrix* $A^{(l+1)} \in \mathbb{Z}^{n_{l+1} \times n_l}$ *and bias vector* $b^{(l+1)} \in \mathbb{R}^{n_{l+1}}$, *and* $e \coloneqq (0, \ldots, 0, 1) \in \mathbb{R}^{d+1}$.

A conclusion of Lemma 3.6.2 is that zonotopes are the building blocks in the tropical geometry of neural networks. Zonotopes are studied extensively in convex geometry and, among other things, are intimately related to hyperplane arrangements [31, 33, 58, 39]. Lemma 3.6.2 connects neural networks to this extensive body of work but its full implication remains to be explored. In Section 3.7.4, we show how one may build these polytopes for a two-layer neural network.

3.6.3 Geometric Complexity of Deep Neural Networks

We apply the tools in Section 3.3 to study the complexity of a neural network, showing that a deep network is much more expressive than a shallow one. Our measure of complexity is geometric: we will follow [63, 77] and use the number of linear regions of a piecewise linear function $\nu : \mathbb{R}^d \to \mathbb{R}^p$ to measure the complexity of ν.

We would like to emphasize that our upper bound below does not improve on that obtained in [77] — in fact, our version is more restrictive given that it applies only to neural networks satisfying (a)–(c). Nevertheless our goal here is to demonstrate how tropical geometry may be used to derive the same bound.

Theorem 3.6.3. *Let $\nu : \mathbb{R}^d \to \mathbb{R}$ be an L-layer real-valued feedforward neural network satisfying* (a)–(c)*. Let $t^{(L)} = -\infty$ and $n_l \geq d$ for all $l = 1, \ldots, L-1$. Then $\nu = \nu^{(L)}$ has at most*

$$\prod_{l=1}^{L-1} \sum_{i=0}^{d} \binom{n_l}{i}$$

linear regions. In particular, if $d \leq n_1, \ldots, n_{L-1} \leq n$, the number of linear regions of ν is bounded by $\mathcal{O}\big(n^{d(L-1)}\big)$.

Proof. If $L = 2$, this follows directly from Lemma 3.6.2 and Corollary 3.3.7. The case of $L \geq 3$ is in Section 3.8.7. □

As was pointed out in [77], this upper bound closely matches the lower bound $\Omega\big((n/d)^{(L-1)d} n^d\big)$ in [63, Corollary 5] when $n_1 = \cdots = n_{L-1} = n \geq d$. Hence we surmise that the number of linear regions of the neural network grows polynomially with the width n and exponentially with the number of layers L.

3.7 Details and Proofs

3.7.1 Illustration of Our Neural Network

Figure 3.4 summarizes the architecture and notations of the feedforward neural network discussed in this thesis.

3.7.2 Tropical Power

As in Section 3.2, we write $x^a = x^{\odot a}$; aside from this slight abuse of notation, $\oplus$ and $\odot$ denote tropical sum and product, $+$ and $\cdot$ denote standard sum and product in all other

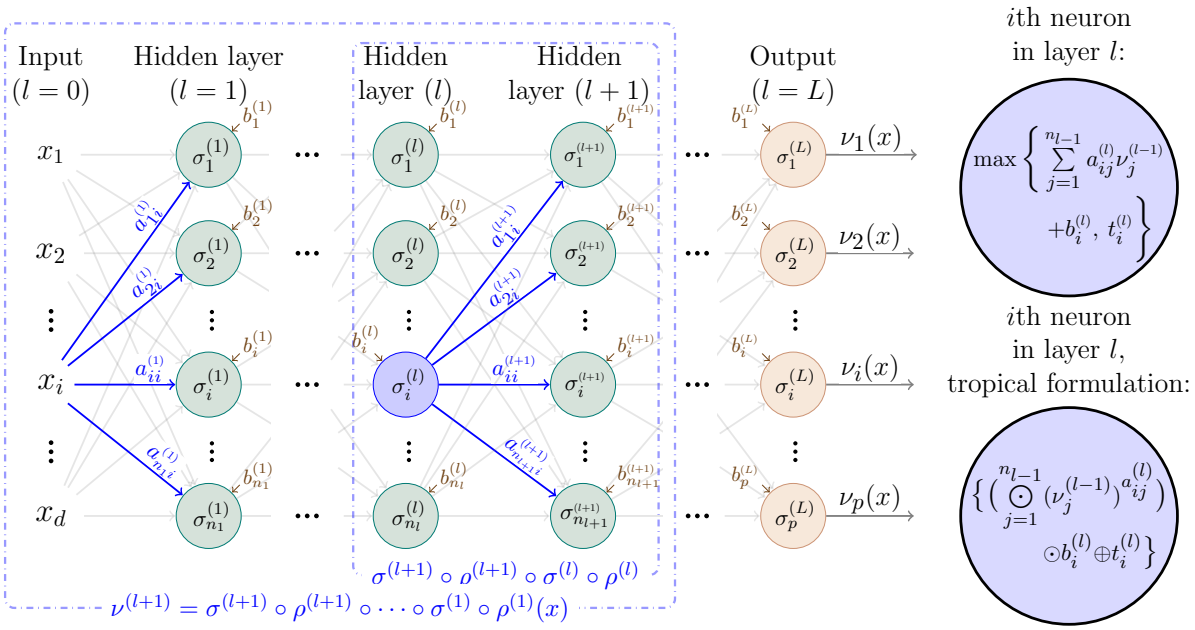


Figure 3.4: General form of an ReLU feedforward neural network $\nu : \mathbb{R}^d \to \mathbb{R}^p$ with L layers.

contexts. Tropical power evidently has the following properties:

- For $x, y \in \mathbb{R}$ and $a \in \mathbb{R}$, $a \geq 0$,

$$(x \oplus y)^a = x^a \oplus y^a \quad \text{and} \quad (x \odot y)^a = x^a \odot y^a.$$

  If a is allowed negative values, then we lose the first property. In general $(x \oplus y)^a \neq x^a \oplus y^a$ for $a < 0$.

- For $x \in \mathbb{R}$,

$$x^0 = 0.$$

- For $x \in \mathbb{R}$ and $a, b \in \mathbb{N}$,

$$(x^a)^b = x^{a \cdot b}.$$

- For $x \in \mathbb{R}$ and $a, b \in \mathbb{Z}$,

$$x^a \odot x^b = x^{a+b}.$$

- For $x \in \mathbb{R}$ and $a, b \in \mathbb{Z}$,

$$x^a \oplus x^b = x^a \odot (x^{a-b} \oplus 0) = x^a \odot (0 \oplus x^{a-b}).$$

3.7.3 Examples of Tropical Curves and Dual Subdivision of Newton Polygon

Let $f \in \text{Pol}(2,1) = \mathbb{T}[x_1, x_2]$, i.e., a bivariate tropical polynomial. It follows from our discussions in Section 3.3 that the tropical hypersurface $\mathcal{T}(f)$ is a planar graph dual to the dual subdivision $\delta(f)$ in the following sense:

(i) Each two-dimensional face in $\delta(f)$ corresponds to a vertex in $\mathcal{T}(f)$.

(ii) Each one-dimensional edge of a face in $\delta(f)$ corresponds to an edge in $\mathcal{T}(f)$. In particular, an edge from the Newton polygon $\Delta(f)$ corresponds to an unbounded edge in $\mathcal{T}(f)$ while other edges correspond to bounded edges.

Figure 3.3 illustrates how we may find the dual subdivision for the tropical polynomial $f(x_1, x_2) = 1 \odot x_1^2 \oplus 1 \odot x_2^2 \oplus 2 \odot x_1 x_2 \oplus 2 \odot x_1 \oplus 2 \odot x_2 \oplus 2$. First, find the convex hull

$$\mathcal{P}(f) = \text{Conv}\{(2,0,1), (0,2,1), (1,1,2), (1,0,2), (0,1,2), (0,0,2)\}.$$

Then, by projecting the upper envelope of $\mathcal{P}(f)$ to $\mathbb{R}^2$, we obtain $\delta(f)$, the dual subdivision of the Newton polygon.

3.7.4 Polytopes of a Two-Layer Neural Network

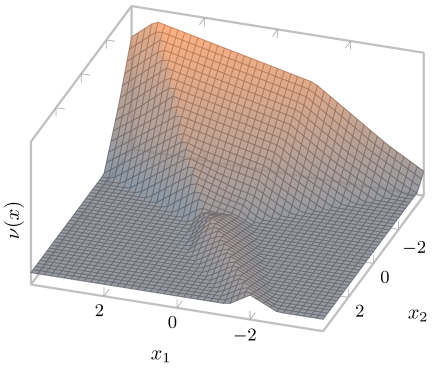
We illustrate our discussions in Section 3.6.2 with a two-layer example. Let $\nu : \mathbb{R}^2 \to \mathbb{R}$ be with $n_0 = 2$ input nodes, $n_1 = 5$ nodes in the first layer, and $n_2 = 1$ nodes in the output:

$$\omega = \nu^{(1)}(x) = \max\left\{ \begin{bmatrix} -1 & 1 \\ 1 & -3 \\ 1 & 2 \\ -4 & 1 \\ 3 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 \\ -1 \\ 2 \\ 0 \\ -2 \end{bmatrix}, 0 \right\},$$

$$\nu^{(2)}(\omega) = \max\{\omega_1 + 2\omega_2 + \omega_3 - \omega_4 - 3\omega_5, 0\}.$$

We first express $\nu^{(1)}$ and $\nu^{(2)}$ as tropical rational maps,

$$\nu^{(1)} = F^{(1)} \oslash G^{(1)}, \quad \nu^{(2)} = f^{(2)} \oslash g^{(2)},$$



where

$$y \coloneqq F^{(1)}(x) = H^{(1)}(x) \oplus G^{(1)}(x),$$

$$z \coloneqq G^{(1)}(x) = \begin{bmatrix} x_1 \\ x_2^3 \\ 0 \\ x_1^4 \\ 0 \end{bmatrix}, \qquad H^{(1)}(x) = \begin{bmatrix} 1 \odot x_2 \\ (-1) \odot x_1 \\ 2 \odot x_1 x_2^2 \\ x_2 \\ (-2) \odot x_1^3 x_2^2 \end{bmatrix},$$

and

$$
\begin{aligned}
f^{(2)}(x) &= g^{(2)}(x) \oplus h^{(2)}(x), \\
g^{(2)}(x) &= y_4 \odot y_5^3 \odot z_1 \odot z_2^2 \odot z_3 \\
&= (x_2 \oplus x_1^4) \odot ((-2) \odot x_1^3 x_2^2 \oplus 0)^3 \odot x_1 \odot (x_2^3)^2, \\
h^{(2)}(x) &= y_1 \odot y_2^2 \odot y_3 \odot z_4 \odot z_5^3 \\
&= (1 \odot x_2 \oplus x_1) \odot ((-1) \odot x_1 \oplus x_2^3)^2 \odot (2 \odot x_1 x_2^2 \oplus 0) \odot x_1^4.
\end{aligned}
$$

We will write $F^{(1)} = (f_1^{(1)}, \ldots, f_5^{(1)})$ and likewise for $G^{(1)}$ and $H^{(1)}$. The monomials occurring in $g_j^{(1)}(x)$ and $h_j^{(1)}(x)$ are all of the form $cx_1^{a_1}x_2^{a_2}$. Therefore $\mathcal{P}(g_j^{(1)})$ and $\mathcal{P}(h_j^{(1)})$, $j = 1, \ldots, 5$, are points in $\mathbb{R}^3$.

Since $F^{(1)} = G^{(1)} \oplus H^{(1)}$, $\mathcal{P}(f_j^{(1)})$ is a convex hull of two points, and thus a line segment in $\mathbb{R}^3$. The Newton polygons associated with $f_j^{(1)}$, equal to their dual subdivisions in this case, are obtained by projecting these line segments back to the plane spanned by a_1, a_2, as shown on the left in Figure 3.5.

The line segments $\mathcal{P}(f_j^{(1)})$, $j = 1, \ldots, 5$, and points $\mathcal{P}(g_j^{(1)})$, $j = 1, \ldots, 5$, serve as building blocks for $\mathcal{P}(h^{(2)})$ and $\mathcal{P}(g^{(2)})$, which are constructed as weighted Minkowski sums:

$$
\begin{aligned}
\mathcal{P}(h^{(2)}) &= \mathcal{P}(f_4^{(1)}) + 3\mathcal{P}(f_5^{(1)}) + \mathcal{P}(g_1^{(1)}) + 2\mathcal{P}(g_2^{(1)}) + \mathcal{P}(g_3^{(1)}), \\
\mathcal{P}(g^{(2)}) &= \mathcal{P}(f_1^{(1)}) + 2\mathcal{P}(f_2^{(1)}) + \mathcal{P}(f_3^{(1)}) + \mathcal{P}(g_4^{(1)}) + 3\mathcal{P}(g_5^{(1)}).
\end{aligned}
$$

$\mathcal{P}(g^{(2)})$ and the dual subdivision of its Newton polygon are shown on the right in Figure 3.5. $\mathcal{P}(h^{(2)})$ and the dual subdivision of its Newton polygon are shown on the left in Figure 3.6. $\mathcal{P}(f^{(2)})$ is the convex hull of the union of $\mathcal{P}(g^{(2)})$ and $\mathcal{P}(h^{(2)})$. The dual subdivision of its Newton polygon is obtained by projecting the upper faces of $\mathcal{P}(f^{(2)})$ to the plane spanned by a_1, a_2. These are shown on the right in Figure 3.6.

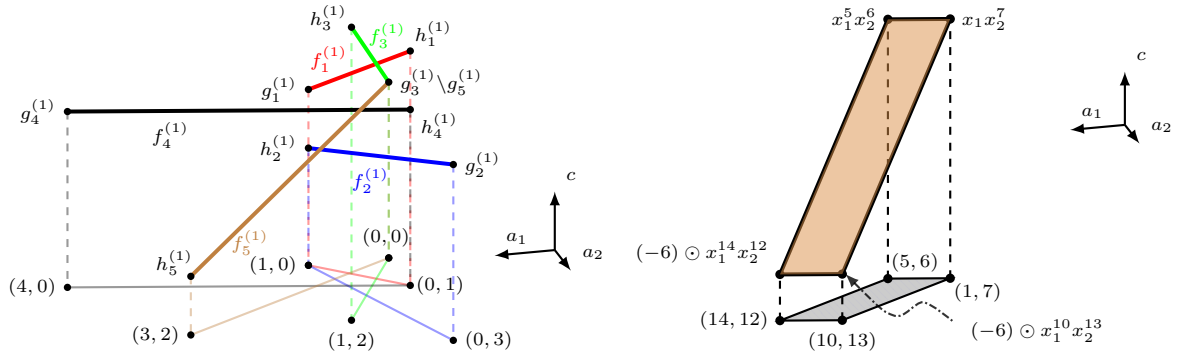


Figure 3.5: *Left*: $\mathcal{P}(f_j^{(1)})$ and dual subdivision of $\Delta(f_j^{(1)})$, $j = 1, \ldots, 5$. *Right*: $\mathcal{P}(g^{(2)})$ and dual subdivision of $\Delta(g^{(2)})$. In both figures, dual subdivisions have been translated along the $-c$ direction (downwards) and separated from the polytopes for visibility.

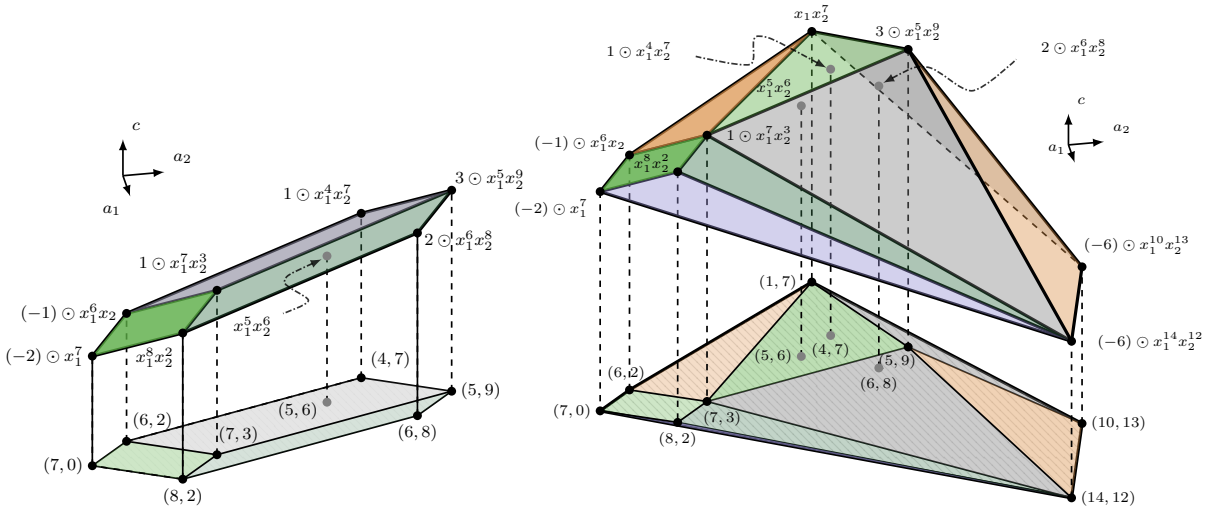


Figure 3.6: *Left*: The polytope associated with $h^{(2)}$ and its dual subdivision. *Right*: $\mathcal{P}(f^{(2)})$ and dual subdivision of $\Delta(f^{(2)})$. In both figures, dual subdivisions have been translated along the $-c$ direction (downwards) and separated from the polytopes for visibility.

3.8 Proofs

3.8.1 Proof of Corollary 3.3.7

Proof. Let V_1 and V_2 be the sets of vertices on the upper and lower envelopes of P respectively. By Theorem 3.3.6, P has

$$n_1 \coloneqq 2\sum_{j=0}^{d}\binom{m-1}{j}$$

vertices in total. By construction, we have $|V_1 \cup V_2| = n_1$. It is well-known that zonotopes are centrally symmetric and so there are equal number of vertices on the upper and lower envelopes, i.e., $|V_1| = |V_2|$. Let $P' \coloneqq \pi(P)$ be the projection of P into $\mathbb{R}^d$. Since the projected vertices are assumed to be in general positions, P' must be a d-dimensional zonotope generated by m nonparallel line segments. Hence, by Theorem 3.3.6 again, P' has

$$n_2 \coloneqq 2\sum_{j=0}^{d-1}\binom{m-1}{j}$$

vertices. For any vertex $v \in P$, $\pi(v)$ is a vertex of P' if and only if v belongs to both the upper and lower envelopes, i.e., $v \in V_1 \cap V_2$. Therefore the number of vertices on P' equals $|V_1 \cap V_2|$. By construction, we have $|V_1 \cap V_2| = n_2$. Consequently the number of vertices on the upper envelope is

$$|V_1| = \frac{1}{2}(|V_1 \cup V_2| - |V_1 \cap V_2|) + |V_1 \cap V_2| = \frac{1}{2}(n_1 - n_2) + n_2 = \sum_{j=0}^{d}\binom{m}{j}. \qquad \square$$

3.8.2 Proof of Proposition 3.5.1

Proof. Writing $A = A_+ - A_-$, we have

$$\begin{aligned}
\rho^{(l+1)}(x) &= \big(A_+ - A_-\big)\big(F^{(l)}(x) - G^{(l)}(x)\big) + b \\
&= \big(A_+F^{(l)}(x) + A_-G^{(l)}(x) + b\big) - \big(A_+G^{(l)}(x) + A_-F^{(l)}(x)\big) \\
&= H^{(l+1)}(x) - G^{(l+1)}(x), \\
\nu^{(l+1)}(x) &= \max\big\{\rho^{(l+1)}(y),\, t\big\} \\
&= \max\big\{H^{(l+1)}(x) - G^{(l+1)}(x),\, t\big\} \\
&= \max\big\{H^{(l+1)}(x),\, G^{(l+1)}(x) + t\big\} - G^{(l+1)}(x) \\
&= F^{(l+1)}(x) - G^{(l+1)}(x).
\end{aligned}$$

□

3.8.3 Proof of Theorem 3.5.4

Proof. It remains to establish the "only if" part. We will write $\sigma_t(x) \coloneqq \max\{x, t\}$. Any tropical monomial $b_i x^{\alpha_i}$ is clearly such a neural network as

$$b_i x^{\alpha_i} = (\sigma_{-\infty} \circ \rho_i)(x) = \max\{\alpha_i^{\mathsf{T}} x + b_i, -\infty\}.$$

If two tropical polynomials p and q are represented as neural networks with l_p and l_q layers respectively,

$$\begin{aligned}
p(x) &= \big(\sigma_{-\infty} \circ \rho_p^{(l_p)} \circ \sigma_0 \circ \ldots \sigma_0 \circ \rho_p^{(1)}\big)(x), \\
q(x) &= \big(\sigma_{-\infty} \circ \rho_q^{(l_q)} \circ \sigma_0 \circ \ldots \sigma_0 \circ \rho_q^{(1)}\big)(x),
\end{aligned}$$

then $(p \oplus q)(x) = \max\{p(x), q(x)\}$ can also be written as a neural network with $\max\{l_p, l_q\}+1$ layers:

$$(p \oplus q)(x) = \sigma_{-\infty}\big([\sigma_0 \circ \rho_1](y(x)) + [\sigma_0 \circ \rho_2](y(x)) - [\sigma_0 \circ \rho_3](y(x))\big),$$

where $y : \mathbb{R}^d \to \mathbb{R}^2$ is given by $y(x) = (p(x), q(x))$ and $\rho_i : \mathbb{R}^2 \to \mathbb{R}$, $i = 1, 2, 3$, are linear functions defined by

$$\rho_1(y) = y_1 - y_2, \quad \rho_2(y) = y_2, \quad \rho_3(y) = -y_2.$$

Thus, by induction, any tropical polynomial can be written as a neural network with ReLU activation. Observe also that if a tropical polynomial is the tropical sum of r monomials, then it can be written as a neural network with no more than $\lceil \log_2 r \rceil + 1$ layers.

Next we consider a tropical rational function $(p \oslash q)(x) = p(x) - q(x)$ where p and q are tropical polynomials. Under the same assumptions, we can represent $p \oslash q$ as

$$(p \oslash q)(x) = \sigma_{-\infty}\big([\sigma_0 \circ \rho_4](y(x)) - [\sigma_0 \circ \rho_5](y(x)) + [\sigma_0 \circ \rho_6](y(x)) - [\sigma_0 \circ \rho_7](y(x))\big)$$

where $\rho_i : \mathbb{R}^2 \to \mathbb{R}^2$, $i = 4, 5, 6, 7$, are linear functions defined by

$$\rho_4(y) = y_1, \quad \rho_5(y) = -y_1, \quad \rho_6(y) = -y_2, \quad \rho_7(y) = y_2.$$

Therefore $p \oslash q$ is also a neural network with at most $\max\{l_p, l_q\} + 1$ layers.

Finally, if f and g are tropical polynomials that are respectively tropical sums of r_f and r_g monomials, then the discussions above show that $(f \oslash g)(x) = f(x) - g(x)$ is a neural network with at most $\max\{\lceil \log_2 r_f \rceil, \lceil \log_2 r_g \rceil\} + 2$ layers. □

3.8.4 Proof of Proposition 3.5.5

Proof. It remains to establish the "if" part. Let $\mathbb{R}^d$ be divided into N polyhedral region on each of which ν restricts to a linear function

$$\ell_i(x) = a_i^{\mathsf{T}} x + b_i, \quad a_i \in \mathbb{Z}^d, \quad b_i \in \mathbb{R}, \quad i = 1, \ldots, L,$$

i.e., for any $x \in \mathbb{R}^d$, $\nu(x) = \ell_i(x)$ for some $i \in \{1, \ldots, L\}$. It follows from [86] that we can find N subsets of $\{1, \ldots, L\}$, denoted by S_j, $j = 1, \ldots, N$, so that ν has a representation

$$\nu(x) = \max_{j=1,\ldots,N} \min_{i \in S_j} \ell_i.$$

It is clear that each ℓ_i is a tropical rational function. Now for any tropical rational functions p and q,

$$\min\{p, q\} = -\max\{-p, -q\} = 0 \oslash [(0 \oslash p) \oplus (0 \oslash q)] = [p \odot q] \oslash [p \oplus q].$$

Since $p \odot q$ and $p \oplus q$ are both tropical rational functions, so is their tropical quotient. By induction, $\min_{i \in S_j} \ell_i$ is a tropical rational function for any $j = 1, \ldots, N$, and therefore so is their tropical sum ν. □

3.8.5 Proof of Proposition 3.5.6

Proof. For a one-layer neural network $\nu(x) = \max\{Ax + b, t\} = (\nu_1(x), \ldots, \nu_p(x))$ with $A \in \mathbb{R}^{p \times d}$, $b \in \mathbb{R}^p$, $x \in \mathbb{R}^d$, $t \in (\mathbb{R} \cup \{-\infty\})^p$, we have

$$\nu_k(x) = \left(b_k \odot \bigodot_{j=1}^{d} x_j^{a_{kj}} \right) \oplus t_k = \left(b_k \odot \bigodot_{j=1}^{d} x_j^{a_{kj}} \right) \oplus \left(t_k \odot \bigodot_{j=1}^{d} x_j^{0} \right), \qquad k = 1, \ldots, p.$$

So for any $k = 1, \ldots, p$, if we write $\bar{b}_1 = b_k$, $\bar{b}_2 = t_k$, $\bar{a}_{1j} = a_{kj}$, $\bar{a}_{2j} = 0$, $j = 1, \ldots, d$, then

$$\nu_k(x) = \bigoplus_{i=1}^{2} \bar{b}_i \bigodot_{j=1}^{d} x_j^{\bar{a}_{ij}}$$

is clearly a tropical signomial function. Therefore ν is a tropical signomial map. The result for arbitrary number of layers then follows from using the same recurrence as in the proof in Section 3.8.2, except that now the entries in the weight matrix are allowed to take real values, and the maps $H^{(l)}(x)$, $G^{(l)}(x)$, $F^{(l)}(x)$ are tropical signomial maps. Hence every

layer can be written as a tropical rational signomial map $\nu^{(l)} = F^{(l)} \oslash G^{(l)}$. $\square$

3.8.6 Proof of Proposition 3.6.1

We prove a slightly more general result.

Proposition 3.8.1 (Level sets)**.** *Let* $f \oslash g \in \mathrm{Rat}(d, 1) = \mathbb{T}(x_1, \ldots, x_d)$.

(i) *Given a constant* $c > 0$*, the* level set

$$\mathcal{B} \coloneqq \{x \in \mathbb{R}^d : f(x) \oslash g(x) = c\}$$

divides $\mathbb{R}^d$ *into at most* $\mathcal{N}(f)$ *connected polyhedral regions where* $f(x) \oslash g(x) > c$*, and at most* $\mathcal{N}(g)$ *such regions where* $f(x) \oslash g(x) < c$.

(ii) *If* $c \in \mathbb{R}$ *is such that there is no tropical monomial in* $f(x)$ *that differs from any tropical monomial in* $g(x)$ *by* c*, then the level set* $\mathcal{B}$ *is contained in a tropical hypersurface,*

$$\mathcal{B} \subseteq \mathcal{T}(\max\{f(x),\, g(x) + c\}) = \mathcal{T}(c \odot g \oplus f).$$

Proof. We show that the bounds on the numbers of connected positive (i.e., above c) and negative (i.e., below c) regions are as we claimed in (i). The tropical hypersurface of f divides $\mathbb{R}^d$ into $\mathcal{N}(f)$ convex regions $C_1, \ldots, C_{\mathcal{N}(f)}$ such that f is linear on each C_i. As g is piecewise linear and convex over $\mathbb{R}^d$, $f \oslash g = f - g$ is piecewise linear and concave on each C_i. Since the level set $\{x : f(x) - g(x) = c\}$ and the superlevel set $\{x : f(x) - g(x) \geq c\}$ must be convex by the concavity of $f - g$, there is at most one positive region in each C_i. Therefore the total number of connected positive regions cannot exceed $\mathcal{N}(f)$. Likewise, the tropical hypersurface of g divides $\mathbb{R}^d$ into $\mathcal{N}(g)$ convex regions on each of which $f \oslash g$ is convex. The same argument shows that the number of connected negative regions does not exceed $\mathcal{N}(g)$.

We next address (ii). Upon rearranging terms, the level set becomes

$$\mathcal{B} = \big\{x \in \mathbb{R}^d : f(x) = g(x) + c\big\}.$$

Since $f(x)$ and $g(x) + c$ are both tropical polynomial, we have

$$f(x) = b_1 x^{\alpha_1} \oplus \cdots \oplus b_r x^{\alpha_r},$$
$$g(x) + c = c_1 x^{\beta_1} \oplus \cdots \oplus c_s x^{\beta_s},$$

with appropriate multiindices $\alpha_1, \ldots, \alpha_r, \beta_1, \ldots, \beta_s$, and real coefficients $b_1, \ldots, b_r, c_1, \ldots, c_s$. By the assumption on the monomials, we have that $x_0 \in \mathcal{B}$ only if there exist i, j so that $\alpha_i \neq \beta_j$ and $b_i x_0^{\alpha_i} = c_j x_0^{\beta_j}$. This completes the proof since if we combine the monomials of $f(x)$ and $g(x)+c$ by (tropical) summing them into a single tropical polynomial, $\max\{f(x),\, g(x)+c\}$, the above implies that on the level set, the value of the combined tropical polynomial is attained by at least two monomials and therefore $x_0 \in \mathcal{T}(\max\{f(x),\, g(x) + c\})$. □

Proposition 3.6.1 follows immediately from Proposition 3.8.1 since the decision boundary $\{x \in \mathbb{R}^d : \nu(x) = s^{-1}(c)\}$ is a level set of the tropical rational function ν.

3.8.7 Proof of Theorem 3.6.3

The linear regions of a tropical polynomial map $F \in \mathrm{Pol}(d, m)$ are all convex but this is not necessarily the case for a tropical rational map $F \in \mathrm{Rat}(d, n)$. Take for example a bivariate real-valued function $f(x, y)$ whose graph in $\mathbb{R}^3$ is a pyramid with base $\{(x, y) \in \mathbb{R}^2 : x, y \in [-1, 1]\}$ and zero everywhere else, then the linear region where f vanishes is $\mathbb{R}^2 \setminus \{(x, y) \in \mathbb{R}^2 : x, y \in [-1, 1]\}$, which is nonconvex. The nonconvexity invalidates certain geometric arguments that only apply in the convex setting. Nevertheless there is a way to subdivide each of the nonconvex linear regions into convex ones to get ourselves back into the

convex setting. We will start with the number of *convex* linear regions for tropical rational maps although later we will deduce the required results for the number of linear regions (without imposing convexity).

We first extend the notion of tropical hypersurface to tropical rational maps: Given a tropical rational map $F \in \mathrm{Rat}(d, m)$, we define $\mathcal{T}(F)$ to be the boundaries between adjacent linear regions. When $F = (f_1, \ldots, f_m) \in \mathrm{Pol}(d, m)$, i.e., a tropical polynomial map, this set is exactly the union of tropical hypersurfaces $\mathcal{T}(f_i)$, $i = 1, \ldots, m$. Therefore this definition of $\mathcal{T}(F)$ extends Definition 3.3.1.

For a tropical rational map F, we will examine the smallest number of convex regions that form a refinement of $\mathcal{T}(F)$. For brevity, we will call this the *convex degree* of F; for consistency, the number of linear regions of F we will call its *linear degree*. We define convex degree formally below. We will write $F|_C$ to mean the restriction of map F to $C \subseteq \mathbb{R}^d$.

Definition 3.8.2. *The* convex degree *of a tropical rational map $F \in \mathrm{Rat}(d, n)$ is the minimum division of $\mathbb{R}^d$ into convex regions over which F is linear, i.e.*

$$\mathcal{N}_c(F) \coloneqq \min\big\{n : C_1 \cup \cdots \cup C_n = \mathbb{R}^d,\ C_i \text{ convex, } F|_{C_i} \text{ linear}\big\}.$$

Note that $C_1, \ldots, C_{\mathcal{N}_c(F)}$ either divide $\mathbb{R}^d$ into the same regions as $\mathcal{T}(F)$ or form a refinement.

For $m \leq d$, we will denote by $\mathcal{N}_c(F \mid m)$ the maximum convex degree obtained by restricting F to an m-dimensional affine subspace in $\mathbb{R}^d$, i.e.,

$$\mathcal{N}_c(F \mid m) \coloneqq \max\big\{\mathcal{N}_c(F|_\Omega) : \Omega \subseteq \mathbb{R}^d \text{ is an } m\text{-dimensional affine space}\big\}.$$

For any $F \in \mathrm{Rat}(d, n)$, there is at least one tropical polynomial map that subdivides $\mathcal{T}(F)$, and so convex degree is well-defined (e.g., if $F = (p_1 \oslash q_1, \ldots, p_n \oslash q_n) \in \mathrm{Rat}(d, n)$, then we may choose $P = (p_1, \ldots, p_n, q_1, \ldots, q_n) \in \mathrm{Pol}(d, 2n)$). Since the linear regions of a tropical polynomial map are always convex, we have $\mathcal{N}(F) = \mathcal{N}_c(F)$ for any $F \in \mathrm{Pol}(d, n)$.

Let $F = (f_1, \ldots, f_n) \in \mathrm{Rat}(d, n)$ and $\alpha = (a_1, \ldots, a_n) \in \mathbb{Z}^n$. Consider the tropical

rational function[3]

$$F^\alpha \coloneqq \alpha^\mathsf{T} F = a_1 f_1 + \cdots + a_n f_n = \bigodot_{j=1}^{n} f_j^{a_j} \in \mathrm{Rat}(d, 1).$$

For some α, F^α may have fewer linear regions than F, e.g, $\alpha = (0, \ldots, 0)$. As such, we need the following notion.

Definition 3.8.3. *$\alpha = (a_1, \ldots, a_n) \in \mathbb{Z}^n$ is said to be a* general exponent *of $F \in \mathrm{Rat}(d, n)$ if the linear regions of F^α and the linear regions of F are identical.*

We show that general exponent always exists for any $F \in \mathrm{Rat}(d, n)$ and may be chosen to have all entries nonnegative.

Lemma 3.8.4. *Let $F \in \mathrm{Rat}(d, n)$. Then*

(i) *$\mathcal{N}(F^\alpha) = \mathcal{N}(F)$ if and only if α is a general exponent;*

(ii) *F has a general exponent $\alpha \in \mathbb{N}^n$.*

Proof. It follows from the definition of tropical hypersuface that $\mathcal{T}(F^\alpha)$ and $\mathcal{T}(F)$ comprise respectively the points $x \in \mathbb{R}^d$ at which F^α and F are not differentiable. Hence $\mathcal{T}(F^\alpha) \subseteq \mathcal{T}(F)$, which implies that $\mathcal{N}(F^\alpha) < \mathcal{N}(F)$ unless $\mathcal{T}(F^\alpha) = \mathcal{T}(F)$. This concludes (i).

For (ii), we need to show that there always exists an $\alpha \in \mathbb{N}^n$ such that F^α divides its domain $\mathbb{R}^d$ into the same set of linear regions as F. In other words, for every pair of adjacent linear regions of F, the $(d-1)$-dimensional face in $\mathcal{T}(F)$ that separates them is also present in $\mathcal{T}(F^\alpha)$ and so $\mathcal{T}(F^\alpha) \supseteq \mathcal{T}(F)$.

Let L and M be adjacent linear regions of F. The differentials of $F|_L$ and $F|_M$ must have integer coordinates, i.e., $dF|_L, dF|_M \in \mathbb{Z}^{n \times d}$. Since L and M are distinct linear regions, we must have $dF|_L \neq dF|_M$ (or otherwise L and M can be merged into a single linear region). Note that the differentials of $F^\alpha|_L$ and $F^\alpha|_M$ are given by $\alpha^\mathsf{T} dF|_L$ and $\alpha^\mathsf{T} dF|_M$.

3. This is in the sense of a tropical power but we stay consistent to our slight abuse of notation and write F^α instead of $F^{\odot\alpha}$.

To ensure the $(d-1)$-dimensional face separating L and M still exists in $\mathcal{T}(F^\alpha)$, we need to choose α so that $\alpha^\mathsf{T} dF|_L \neq \alpha^\mathsf{T} dF|_M$. Observe that the solution to $(dF|_L - dF|_M)^\mathsf{T}\alpha = 0$ is contained in a one-dimensional subspace of $\mathbb{R}^n$.

Let $\mathcal{A}(F)$ be the collection of all pairs of adjacent linear regions of F. Since the set of α that degenerates two adjacent linear regions into a single one, i.e.,

$$\mathcal{S} \coloneqq \bigcup_{(L,M)\in\mathcal{A}(F)} \{\alpha \in \mathbb{N}^n : (dF|_L - dF|_M)^\mathsf{T}\alpha = 0)\},$$

is contained in a union of a finite number of hyperplanes in $\mathbb{R}^n$, $\mathcal{S}$ cannot cover the entire lattice of nonnegative integers $\mathbb{N}^n$. Therefore the set $\mathbb{N}^n \cap (\mathbb{R}^n \setminus \mathcal{S})$ is nonempty and any of its element is a general exponent for F. □

Lemma 3.8.4 shows that we may study the linear degree of a tropical rational map by studying that of a tropical rational function, for which the results in Section 3.3.1 apply.

We are now ready to prove a key result on the convex degree of composition of tropical rational maps.

Theorem 3.8.5. *Let $F = (f_1, \ldots, f_m) \in \mathrm{Rat}(n, m)$ and $G \in \mathrm{Rat}(d, n)$. Define $H = (h_1, \ldots, h_m) \in \mathrm{Rat}(d, m)$ by*

$$h_i \coloneqq f_i \circ G, \qquad i = 1, \ldots, m.$$

Then

$$\mathcal{N}(H) \leq \mathcal{N}_c(H) \leq \mathcal{N}_c(F \mid d) \cdot \mathcal{N}_c(G).$$

Proof. Only the upper bound requires a proof. Let $k = \mathcal{N}_c(G)$. By the definition of $\mathcal{N}_c(G)$, there exist convex sets $C_1, \ldots, C_k \subseteq \mathbb{R}^d$ whose union is $\mathbb{R}^d$ and on each of which G is linear. So $G|_{C_i}$ is some affine function ρ_i. For any i,

$$\mathcal{N}_c(F \circ \rho_i) \leq \mathcal{N}_c(F \mid d),$$

by the definition of $\mathcal{N}_c(F \mid d)$. Since $F \circ G = F \circ \rho_i$ on C_i, we have

$$\mathcal{N}_c(F \circ G) \leq \sum_{i=1}^{k} \mathcal{N}_c(F \circ \rho_i).$$

Hence

$$\mathcal{N}_c(F \circ G) \leq \sum_{i=1}^{k} \mathcal{N}_c(F \circ \rho_i) \leq \sum_{i=1}^{k} \mathcal{N}_c(F \mid d) = \mathcal{N}_c(F \mid d) \cdot \mathcal{N}_c(G). \qquad \square$$

We now apply our observations on tropical rational functions to neural networks. The next lemma follows directly from Corollary 3.3.7.

Lemma 3.8.6. *Let $\sigma^{(l)} \circ \rho^{(l)} : \mathbb{R}^{n_{l-1}} \to \mathbb{R}^{n_l}$ where $\sigma^{(l)}$ and $\rho^{(l)}$ are the affine transformation and activation of the lth layer of a neural network. If $d \leq n_l$, then*

$$\mathcal{N}_c(\sigma^{(l)} \circ \rho^{(l)} \mid d) \leq \sum_{i=0}^{d} \binom{n_l}{i}.$$

Proof. $\mathcal{N}_c(\sigma^{(l)} \circ \rho^{(l)} \mid d)$ is the maximum convex degree of a tropical rational map $F = (f_1, \ldots, f_{n_l}) : \mathbb{R}^d \to \mathbb{R}^{n_l}$ of the form

$$f_i(x) \coloneqq \sigma_i^{(l)} \circ \rho^{(l)} \circ (b_1 \odot x^{\alpha_1}, \ldots, b_{n_{l-1}} \odot x^{\alpha_{n_{l-1}}}), \qquad i = 1, \ldots, n_l.$$

For a general affine transformation $\rho^{(l)}$,

$$\rho^{(l)}(b_1 \odot x^{\alpha_1}, \ldots, b_{n_{l-1}} \odot x^{\alpha_{n_{l-1}}}) = \left(b'_1 \odot x^{\alpha'_1}, \ldots, b'_{n_l} \odot x^{\alpha'_{n_l}}\right) \eqqcolon G(x)$$

for some $\alpha'_1, \ldots, \alpha'_{n_l}$ and $b'_1, \ldots, b'_{n_l}$, and we denote this map by $G : \mathbb{R}^d \to \mathbb{R}^{n_l}$. So $f_i = \sigma_i^{(l)} \circ G$. By Theorem 3.8.5, we have $\mathcal{N}_c(\sigma^{(l)} \circ \rho^{(l)} \mid d) = \mathcal{N}_c(\sigma^{(l)} \mid d) \cdot \mathcal{N}_c(G) = \mathcal{N}_c(\sigma^{(l)} \mid d)$; note that $\mathcal{N}_c(G) = 1$ as G is a linear function.

We have thus reduced the problem to determining a bound on the convex degree of a single layer neural network with n_l nodes $\nu = (\nu_1, \ldots, \nu_{n_l}) : \mathbb{R}^d \to \mathbb{R}^{n_l}$. Let $\gamma = (c_1, \ldots, c_{n_l}) \in \mathbb{N}^{n_l}$

be a nonnegative general exponent for ν. Note that

$$\bigodot_{j=1}^{n_l} \nu_j^{c_j} = \bigodot_{j=1}^{n_l} \left[\left(\bigodot_{i=1}^{d} b_i \odot x^{a_{ji}^+} \right) \oplus \left(\bigodot_{i=1}^{d} x^{a_{ji}^-} \right) \odot t_j \right]^{c_j} - \bigodot_{j=1}^{n_l} \left(\bigodot_{i=1}^{d} x^{a_{ji}^-} \right)^{c_j}.$$

Since the last term is linear in x, we may drop it without affecting the convex degree of the entire expression. It remains to determine an upper bound for the number of linear regions of the tropical polynomial

$$h(x) = \bigodot_{j=1}^{n_l} \left[\left(\bigodot_{i=1}^{d} b_i \odot x^{a_{ji}^+} \right) \oplus \left(\bigodot_{i=1}^{d} x^{a_{ji}^-} \right) \odot t_j \right]^{c_j},$$

which we will obtain by counting vertices of the polytope $\mathcal{P}(h)$. By Propositions 3.3.4 and 3.3.5 the polytope $\mathcal{P}(h)$ is given by a weighted Minkowski sum

$$\sum_{j=1}^{n_l} c_j \mathcal{P}\left[\left(\bigodot_{i=1}^{d} b_i \odot x^{a_{ji}^+} \right) \oplus \left(\bigodot_{i=1}^{d} x^{a_{ji}^-} \right) \odot t_j \right].$$

By Proposition 3.3.5 again,

$$\mathcal{P}\left[\left(\bigodot_{i=1}^{d} b_i \odot x^{a_{ji}^+} \right) \oplus \left(\bigodot_{i=1}^{d} x^{a_{ji}^-} \right) \odot t_j \right] = \operatorname{Conv}\big(\mathcal{V}(\mathcal{P}(f)) \cup \mathcal{V}(\mathcal{P}(g))\big)$$

where

$$f(x) = \bigodot_{i=1}^{d} b_i \odot x^{a_{ji}^+} \qquad \text{and} \qquad g(x) = \left(\bigodot_{i=1}^{d} x^{a_{ji}^-} \right) \odot t_j$$

are tropical monomials. Therefore $\mathcal{P}(f)$, $\mathcal{P}(g)$ are just points in $\mathbb{R}^{d+1}$ and $\operatorname{Conv}\big(\mathcal{V}(\mathcal{P}(f)) \cup \mathcal{V}(\mathcal{P}(g))\big)$ is a line in $\mathbb{R}^{d+1}$. Hence $\mathcal{P}(h)$ is a Minkowski sum of n_l line segments in $\mathbb{R}^{d+1}$, i.e., a zonotope, and Corollary 3.3.7 completes the proof. □

Using Lemma 3.8.6, we obtain a bound on the number of linear regions created by one layer of a neural network.

Theorem 3.8.7. *Let $\nu : \mathbb{R}^d \to \mathbb{R}^{n_L}$ be an L-layer neural network satisfying assumptions* (a)–(c) *with $F^{(l)}$, $G^{(l)}$,$H^{(l)}$, and $\nu^{(l)}$ as defined in Proposition 3.5.1. Let $n_l \geq d$ for all*

$l = 1, \ldots, L$. *Then*

$$\mathcal{N}_c(\nu^{(1)}) = \mathcal{N}(G^{(1)}) = \mathcal{N}(H^{(1)}) = 1, \qquad \mathcal{N}_c(\nu^{(l+1)}) \leq \mathcal{N}_c(\nu^{(l)}) \cdot \sum_{i=0}^{d} \binom{n_{l+1}}{i}.$$

Proof. The $l = 1$ case follows from the fact that $G^{(1)}(x) = A_-^{(1)}x$ and $H^{(1)}(x) = A_+^{(1)}x + b^{(1)}$ are both linear, which in turn forces $\mathcal{N}_c(\nu^{(1)}) = 1$ as in the proof of Lemma 3.8.6. Since $\nu^{(l)} = (\sigma^{(l)} \circ \rho^{(l)}) \circ \nu^{(l-1)}$, the recursive bound follows from Theorem 3.8.5 and Lemma 3.8.6. □

Theorem 3.6.3 follows from applying Theorem 3.8.7 recursively.

3.9 Conclusion

We argue that feedforward neural networks with rectified linear units are, modulo trivialities, nothing more than tropical rational maps. To understand them we often just need to understand the relevant tropical geometry.

In this part of the thesis, we took a first step to provide a proof-of-concept: questions regarding decision boundaries, linear regions, how depth affect expressiveness, etc, can be translated into questions involving tropical hypersurfaces, dual subdivision of Newton polygon, polytopes constructed from zonotopes, etc.

As a new branch of algebraic geometry, the novelty of tropical geometry stems from both the algebra and geometry as well as the interplay between them. It has connections to many other areas of mathematics. Among other things, there is a tropical analogue of linear algebra [12] and a tropical analogue of convex geometry [27]. We cannot emphasize enough that we have only touched on a small part of this rich subject.

REFERENCES

[1] Robert J. Adler, Omer Bobrowski, Matthew S. Borman, Eliran Subag, and Shmuel Weinberger. Persistent homology for random fields and complexes. In *Borrowing strength: theory powering applications—a Festschrift for Lawrence D. Brown*, volume 6 of *Inst. Math. Stat. (IMS) Collect.*, pages 124–143. Inst. Math. Statist., Beachwood, OH, 2010.

[2] Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. Understanding deep neural networks with rectified linear units. *International Conference on Learning Representations*, 2018.

[3] M. F. Atiyah. Geometry, topology and physics. *Quart. Journ. Royal Astrophysics Soc.*, 29(3):287–299, September 1988. Delivered as the 11th Arthur Milne Lecture at Oxford University. Republished in Atiyah's *Collected works*, vol. 6.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv:1409.0473*, 2014.

[5] Saugata Basu and Anthony Rizzie. Multi-degree bounds on the Betti numbers of real varieties and semi-algebraic sets and applications. *Discrete Comput. Geom.*, 59(3):553–620, 2018.

[6] Christian Bayer, Olaf Enge-Rosenblatt, Martyna Bator, and Uwe Mönks. Sensorless drive diagnosis using automated feature extraction, significance ranking and reduction. In *Proceedings of 2013 IEEE 18th Conference on Emerging Technologies & Factory Automation, ETFA 2013, Cagliari, Italy, September 10-13, 2013*, pages 1–4, 2013.

[7] Yoshua Bengio and Olivier Delalleau. On the expressive power of deep architectures. In *International Conference on Algorithmic Learning Theory*, pages 18–36. Springer, 2011.

[8] Monica Bianchini and Franco Scarselli. On the complexity of neural network classifiers: A comparison between shallow and deep architectures. *IEEE transactions on neural networks and learning systems*, 25(8):1553–1565, 2014.

[9] Jean-Daniel Boissonnat, Siddharth Pritam, and Divyansh Pareek. Strong Collapse for Persistence. In *ESA 2018 - 26th Annual European Symposium on Algorithms*, pages 67:1–67:13, Helsinki, Finland, August 2018.

[10] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[11] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. Lof: Identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD '00, page 93–104, New York, NY, USA, 2000. Association for Computing Machinery.

[12] Peter Butkovič. *Max-linear systems: theory and algorithms*. Springer Science & Business Media, 2010.

[13] Gunnar Carlsson. The shape of data. In *Foundations of computational mathematics, Budapest 2011*, volume 403 of *London Math. Soc. Lecture Note Ser.*, pages 16–44. Cambridge Univ. Press, Cambridge, 2013.

[14] Gunnar Carlsson. Topological pattern recognition for point cloud data. *Acta Numer.*, 23:289–368, 2014.

[15] Gunnar Carlsson, Tigran Ishkhanov, Vin De Silva, and Afra Zomorodian. On the local behavior of spaces of natural images. *Int. J. Comput. Vis.*, 76(1):1–12, 2008.

[16] Gunnar Carlsson, Gurjeet Singh, and Afra Zomorodian. Computing multidimensional persistence. *J. Comput. Geom.*, 1(1):72–100, 2010.

[17] Gunnar Carlsson and Afra Zomorodian. The theory of multidimensional persistence. *Discrete Comput. Geom.*, 42(1):71–93, 2009.

[18] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Training deep neural networks with low precision multiplications. *arXiv preprint arXiv:1412.7024*, 2014.

[19] Vin De Silva and Gunnar Carlsson. Topological estimation using witness complexes. In Markus Gross, Hanspeter Pfister, Marc Alexa, and Szymon Rusinkiewicz, editors, *SPBG'04 Symposium on Point - Based Graphics 2004*. The Eurographics Association, 2004.

[20] Olivier Delalleau and Yoshua Bengio. Shallow vs. deep sum-product networks. In *Advances in Neural Information Processing Systems*, pages 666–674, 2011.

[21] Jia Deng, Wei Dong, Richard Socher, Li-Jia. Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[22] Pawel Dlotko and Hubert Wagner. Simplification of complexes for persistent homology computations. *Homology Homotopy Appl.*, 16(1):49–63, 2014.

[23] Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. In *41st Annual Symposium on Foundations of Computer Science (Redondo Beach, CA, 2000)*, pages 454–463. IEEE Comput. Soc. Press, Los Alamitos, CA, 2000.

[24] David Eigen, Jason Rolfe, Robert Fergus, and Yann Lecun. Understanding deep architectures using a recursive convolutional network. In *International Conference on Learning Representations (ICLR2014), CBLS, April 2014*, 2014.

[25] Ronen Eldan and Ohad Shamir. The power of depth for feedforward neural networks. In *Conference on Learning Theory*, pages 907–940, 2016.

[26] Andrei Gabrielov, Nicolai Vorobjov, and Thierry Zell. Betti numbers of semialgebraic and sub-Pfaffian sets. *J. London Math. Soc. (2)*, 69(1):27–43, 2004.

[27] Stéphane Gaubert and Ricardo Katz. Max-plus convex geometry. In *International Conference on Relational Methods in Computer Science*. Springer, 2006.

[28] Robert Ghrist. Barcodes: the persistent topology of data. *Bull. Amer. Math. Soc. (N.S.)*, 45(1):61–75, 2008.

[29] Chad Giusti, Eva Pastalkova, Carina Curto, and Vladimir Itskov. Clique topology reveals intrinsic geometric structure in neural correlations. *Proc. Natl. Acad. Sci. USA*, 112(44):13455–13460, 2015.

[30] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, pages 315–323, 2011.

[31] Curtis Greene and Thomas Zaslavsky. On the interpretation of whitney numbers through arrangements of hyperplanes, zonotopes, non-radon partitions, and orientations of graphs. *Transactions of the American Mathematical Society*, 280(1):97–126, 1983.

[32] Peter Gritzmann and Bernd Sturmfels. Minkowski addition of polytopes: computational complexity and applications to gröbner bases. *SIAM Journal on Discrete Mathematics*, 6(2):246–269, 1993.

[33] Leonidas J Guibas, An Nguyen, and Li Zhang. Zonotopes as bounding volumes. In *Proceedings of 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 803–812. SIAM, 2003.

[34] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *International Conference on Machine Learning*, pages 1737–1746, 2015.

[35] William H. Guss and Ruslan Salakhutdinov. On characterizing the capacity of neural networks using algebraic topology. *CoRR*, abs/1802.04443, 2018.

[36] Philip Hartman. On functions representable as a difference of convex functions. *Pacific Journal of Mathematics*, 9(3):707–713, 1959.

[37] Gregory Henselman and Robert Ghrist. Matroid Filtrations and Computational Persistent Homology. *ArXiv e-prints*, June 2016.

[38] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[39] Olga Holtz and Amos Ron. Zonotopal algebra. *Advances in Mathematics*, 227(2):847–894, 2011.

[40] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: training neural networks with low precision weights and activations. *J. Mach. Learn. Res.*, 18:Paper No. 187, 30, 2017.

[41] Ilia Itenberg, Grigory Mikhalkin, and Eugenii I Shustin. *Tropical algebraic geometry*, volume 35. Springer Science & Business Media, 2009.

[42] Nal Kalchbrenner and Phil Blunsom. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, 2013.

[43] Firas A. Khasawneh, Elizabeth Munch, and Jose A. Perea. Chatter classification in turning using machine learning and topological data analysis. *CoRR*, abs/1804.02261, 2018.

[44] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[45] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research), 2009.

[46] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, May 2017.

[47] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Intelligent Signal Processing*, pages 306–351. IEEE Press, 2001.

[48] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[49] Ann B. Lee, Kim Steenstrup Pedersen, and David Mumford. The nonlinear statistics of high-contrast patches in natural images. *International Journal of Computer Vision*, 54(1-3):83–103, 2003.

[50] Li Li, Wei-Yi Cheng, Benjamin S. Glicksberg, Omri Gottesman, Ronald Tamler, Rong Chen, Erwin P. Bottinger, and Joel T. Dudley. Identification of type 2 diabetes subgroups through topological analysis of patient similarity. *Science translational medicine*, 7(311):311ra174–311ra174, 2015.

[51] Lek-Heng Lim. Hodge laplacians on graphs. *SIAM Rev.*, 62(4):to appear, 2020.

[52] Volker Lohweg, Jan Leif Hoffmann, Helene Dörksen, Roland Hildebrand, Eugen Gillich, Jürg Hofmann, and Johannes Schaede. Banknote authentication with mobile devices. In *Media Watermarking, Security, and Forensics 2013, Burlingame, CA, USA, February 5-7, 2013, Proceedings*, page 866507, 2013.

[53] Namita Lokare, Jorge Silva, Ilknur Kaynar-Kabul, and Gregory Naisat. Analytic system for graphical interpretability of and improvement of machine learning models, U.S. Patent 20190080253, 2019.

[54] Rob J. Lyon, Benjamin. W. Stappers, S. Cooper, J. M. Brooke, and J. D. Knowles. Fifty years of pulsar candidate selection: from simple filters to a new principled real-time classification approach. *Monthly Notices of the Royal Astronomical Society*, 459(1):1104–1123, 04 2016.

[55] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of the International Conference on Machine Learning*, Atlanta, Georgia, 2013.

[56] Diane Maclagan and Bernd Sturmfels. *Introduction to tropical geometry*, volume 161. American Mathematical Society, 2015.

[57] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, Dec 1943.

[58] Peter McMullen. On zonotopes. *Transactions of the American Mathematical Society*, 159:91–109, 1971.

[59] J. Milnor. *Morse theory*. Based on lecture notes by M. Spivak and R. Wells. Annals of Mathematics Studies, No. 51. Princeton University Press, Princeton, N.J., 1963.

[60] John W. Milnor. On the Betti numbers of real varieties. *Proc. Amer. Math. Soc.*, 15:275–280, 1964.

[61] Konstantin Mischaikow and Vidit Nanda. Morse theory for filtrations and efficient computation of persistent homology. *Discrete Comput. Geom.*, 50(2):330–353, 2013.

[62] Guido Montúfar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 2924–2932, Cambridge, MA, USA, 2014. MIT Press.

[63] Guido F. Montúfar, Razvan Pascanu, KyungHyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2924–2932, 2014.

[64] Eduardo Nahmad-Achar. *Differential topology and geometry with applications to physics*. IOP Expanding Physics. IOP Publishing, Bristol, 2018.

[65] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 807–814, 2010.

[66] Gregory Naitzat, Namita Lokare, Jorge Silva, and Ilknur Kaynar-Kabul. M-boost: Profiling and refining deep neural networks with topological data analysis. In *KDD 2018 Workshop on Interactive Data Exploration and Analytics, London, UK, August 20, 2018*, 2018.

[67] Mikio Nakahara. *Geometry, topology and physics*. Graduate student series in physics. Hilger, Bristol, 1990.

[68] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

[69] Jessica L. Nielson, Jesse Paquette, Aiwen W. Liu, Cristian F. Gu, C. Amy Tovar, Tomoo Inoue, A Irvine, John C. Gensel, Jennifer Kloke, Tanya C. Petrossian, et al. Topological data analysis for discovery in preclinical spinal cord injury and traumatic brain injury. *Nature communications*, 6:8581, 2015.

[70] Partha Niyogi, Stephen Smale, and Shmuel Weinberger. Finding the homology of submanifolds with high confidence from random samples. *Discrete Comput. Geom.*, 39(1-3):419–441, 2008.

[71] Christopher Olah. Neural networks, manifolds, and topology. `http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/`, 2014.

[72] Michael L. Overton. *Numerical computing with IEEE floating point arithmetic*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2001. Including one theorem, one rule of thumb, and one hundred and one exercises.

[73] Paul T. Pearson. Visualizing clusters in artificial neural networks using morse theory. *Adv. Artificial Neural Systems*, 2013:486363:1–486363:8, 2013.

[74] Jose A. Perea, Anastasia Deckard, Steven B. Haase, and John Harer. Sw1pers: Sliding windows and 1-persistence scoring; discovering periodicity in gene expression time series data. *BMC Bioinformatics*, 16:257:1–257:12, 2015.

[75] Jose A. Perea and John Harer. Sliding windows and persistence: an application of topological methods to signal analysis. *Found. Comput. Math.*, 15(3):799–838, 2015.

[76] Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. In *Advances in Neural Information Processing Systems*, pages 3360–3368, 2016.

[77] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. In *International Conference on Machine Learning*, pages 2847–2854, 2017.

[78] Karthikeyan Natesan Ramamurthy, Kush Varshney, and Krishnan Mody. Topological data analysis of decision boundaries with application to model selection. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5351–5360, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[79] Bastian Rieck, Matteo Togninalli, Christian Bock, Michael Moor, Max Horn, Thomas Gumbsch, and Karsten M. Borgwardt. Neural persistence: A complexity measure for deep neural networks using algebraic topology. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[80] Michael Robinson. *Topological signal processing*. Mathematical Engineering. Springer, Heidelberg, 2014.

[81] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: support vector machines, regularization, optimization, and beyond*. Adaptive computation and machine learning series. MIT Press, 2002.

[82] Frank Seide, Gang Li, and Dong Yu. Conversational speech transcription using context-dependent deep neural networks. In *Twelfth annual conference of the international speech communication association*, 2011.

[83] Gurjeet Singh, Facundo Mémoli, and Gunnar E. Carlsson. Topological methods for the analysis of high dimensional data sets and 3d object recognition. In *Symposium on Point Based Graphics, Prague, Czech Republic, 2007. Proceedings*, pages 91–100, 2007.

[84] Arne Storjohann. Near optimal algorithms for computing smith normal forms of integer matrices. In *Proceedings of the 1996 International Symposium on Symbolic and Algebraic Computation, ISSAC '96, Zurich, Switzerland, July 24-26, 1996*, pages 267–274, 1996.

[85] Pham Dinh Tao and Le Thi Hoai An. The dc (difference of convex functions) programming and dca revisited with dc models of real world nonconvex optimization problems. *Annals of Operations Research*, 133(1–4):23–46, 2005.

[86] JM Tarela and MV Martinez. Region configurations for realizability of lattice piecewise-linear models. *Mathematical and Computer Modelling*, 30(11-12):17–27, 1999.

[87] Thierry P. Zell. *Quantitative study of semi-Pfaffian sets*. ProQuest LLC, Ann Arbor, MI, 2003. Thesis (Ph.D.)–Purdue University.

[88] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv:1611.03530*, 2016.

[89] Liwen Zhang, Gregory Naitzat, and Lek-Heng Lim. Tropical geometry of deep neural networks. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5824–5832, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[90] Afra Zomorodian and Gunnar Carlsson. Computing persistent homology. *Discrete Comput. Geom.*, 33(2):249–274, 2005.