ACCESSING THE TOPOLOGICAL PROPERTIES OF NEURAL NETWORK FUNCTIONS

by

MARISSA MASDEN

A DISSERTATION

Presented to the Department of Mathematics
and the Divison of Graduate Studies of the University of Oregon
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

June 2023

DISSERTATION APPROVAL PAGE

Student: Marissa Masden

Title: Accessing the Topological Properties of Neural Network Functions

This dissertation has been accepted and approved in partial fulfillment of the requirements for the Doctor of Philosophy degree in the Department of Mathematics by:

Dev Sinha	Chair
Benjamin Young	Core Member
Peter Ralph	Core Member
Luca Mazzucato	Core Member
Thanh Nguyen	Institutional Representative

and

Krista Chronister	Vice Provost for Graduate Studies

Original approval signatures are on file with the University of Oregon Division of Graduate Studies.

Degree awarded June 2023

DISSERTATION ABSTRACT

Marissa Masden

Doctor of Philosophy

Department of Mathematics

June 2023

Title: Accessing the Topological Properties of Neural Network Functions

We provide a framework for analyzing the geometry and topology of the canonical polyhedral complex of ReLU neural networks, which naturally divides the input space into linear regions. Beginning with a category appropriate for analyzing neural network layer maps, we give a categorical definition. We then use our foundational results to produce a duality isomorphism between cellular poset of the canonical polyhedral complex and a cubical set. This duality uses sign sequences, an algebraic tool from hyperplane arrangements and oriented matroid theory.

Our theoretical results lead to algorithms for computing not only the canonical polyhedral complex itself but topological invariants of its substructures such as the decision boundary, as well as for evaluating the presence of PL critical points. Using these algorithms, we produce some of the first empirical measurements of the topology of the decision boundary of neural networks, both at initialization and during training. We observing that increasing the width of neural networks decreases the variability observed in their topological expression, but increasing depth increases variability.

A code repository containing Python and Sage code implementing some of the algorithms described herein is available in the included supplementary material.

CURRICULUM VITAE

NAME OF AUTHOR: Marissa Masden

GRADUATE AND UNDERGRADUATE SCHOOLS ATTENDED:

University of Oregon, Eugene, OR
Walla Walla University, College Place, WA

DEGREES AWARDED:

Doctor of Philosophy, Mathematics, 2023, University of Oregon
Bachelor of Science, Mathematics, 2015, Walla Walla University
Bachelor of Science, Chemistry, 2015, Walla Walla University

AREAS OF SPECIAL INTEREST:

Applied Topology, especially in the context of machine learning and the computational and mathematical sciences.

PROFESSIONAL EXPERIENCE:

Graduate Employee, University of Oregon, 2017-2023

GRANTS, AWARDS AND HONORS:

NSF Research Training Grant, University of Oregon Department of Mathematics, Summer 2022.

Johnson Fellowship, University of Oregon Department of Mathematics, Summer 2019.

Dean's First Year Merit Award, University of Oregon Department of Mathematics, AY 2017-2018

ACKNOWLEDGEMENTS

First, I would like to thank my advisor, Dev, not only for his invaluable support, guidance, and inspiration in the process of research and writing, but also for his enthusiasm for and belief in my work.

I am also grateful towards all of those who have served as a mentor to me in other capacities. This includes the members of my dissertation committee, as well as the graduate students preceding me, who have all offered advice and provided a sense of belonging. Special thanks towards Eli Grigsby and Kathryn Lindsey for additionally welcoming me into their sphere of collaboration.

Finally, (uncountably) infinite love and thanks to my husband, Kyle, for his unwavering encouragement, patience, and sacrifice. No words can fully express my appreciation.

TABLE OF CONTENTS

Chapter Page

LIST OF FIGURES

LIST OF TABLES

Table Page

CHAPTER I

INTRODUCTION

Artificial neural networks have rapidly impacted our society, but they are still far from mathematically understood. Lack of understanding of machine learning behavior leaves space for unintended consequences of its use. Traditionally, the mathematical study of machine learning relies on the domains of statistics and functional analysis. However, there are geometric interpretations of the behavior of the basic objects of study. For example, in a *classification task*, data sampled from Euclidean space must be classified into two categories, and the *decision boundary* is a geometric object delineating the boundary between two regions of Euclidean space that a machine learning model has learned to represent the classification.

In this thesis, we develop a theoretical framework and implementable algorithms for fully analyzing neural network functions from the perspective of geometry and topology. The techniques developed apply theoretically to almost all ReLU neural networks, relying on a remarkable duality between the polyhedral complex describing the neural network function and a subcomplex of a cube. The resulting theoretical framework leads to algorithms which are implementable in practice for input dimensions up to 10 or several intermediate layers. These algorithms allow for exact computation of the topological behavior of real neural networks of arbitrary size, limited only by computational power. Previously, such computations could only be performed for either input dimension two or one hidden layer. We thus open the black box of the geometry of a neural network, visualizing small neural networks in their entirety and obtaining exact topological

measurements of their level sets, as pictured in Figure 1 for the training of a neural network on a torus-shaped dataset.

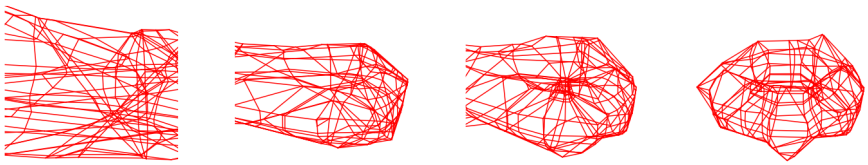


FIGURE 1. The typical evolution of the decision boundary of a neural network successfully trained on a torus-shaped dataset.

In a bit more detail, we establish a combinatorial description of ReLU neural network functions' **canonical polyhedral complex**, using algebraic tools from hyperplane arrangements and oriented matroid theory in Chapter III, expanding geometric dualities from these theories to this more complicated setting. The organization through cubical duality gives an approach to obtain explicit chain complexes describing the cellular topology of the level sets of these functions. These descriptions are implementable in code, so in Chapter IV we produce algorithms computing the described structure when given a specific neural network. Using this code, we obtain empirical measurements of the topology of neural networks at initialization and during training, shared in Chapter V.

1.1. Algebraic Topology and Machine Learning

This thesis is motivated by the question of whether machine learning algorithms can be expected to *topologically generalize to unseen examples*. More precisely,

Problem 1. In a given hypothesis class, does there exist a function whose sublevel or level sets can represent the characteristic function of a given manifold? What is

the probability that the sublevel sets of a function sampled from that hypothesis class have given topological invariants?

This question is broad, and mostly out of reach with current tools. Instead, we begin to approach the following subproblem:

Problem 2. In the hypothesis class of fully-connected, feedforward ReLU neural networks with a given parametrization, classify the achievable and/or probable topology and geometry of the functions' level and sublevel sets.

To understand these questions, here we discuss the utility of algebraic topology for understanding the performance of a neural network model.

Classification as a Topological Task

To understand topology's role in generalization, consider the *manifold hypothesis* [9], which suggests that, in many applications, real-valued data can be modeled as being sampled from an embedded manifold (or a manifold with singularities) in its underlying feature space $\mathbb{R}^n$, with some n-dimensional Gaussian noise. (See Figure 2)

Under this hypothesis, the ideal classifier — the function $p(y = 1|x)$— is smooth, and generically has manifold level sets. The 0.5 level set, consisting of points which are equally likely to belong to each class, is called the *ideal decision boundary*. For a network F to correctly perform a simple classification task, its decision boundary $\{x \mid F(x) = 0.5\}$ must have the same topological invariants as the ideal decision boundary, otherwise we say it fails to topologically generalize. As seen in Figure 3, a neural network might perform with perfect accuracy on data sampled from a manifold, but fail to topologically generalize. In other words,

FIGURE 2. (Left) An ideal data manifold. (Middle) Addition of noise in $\mathbb{R}^2$. (Right) Data sampled from a manifold with noise.

FIGURE 3. (Left) A decision boundary which exhibits topological generalization. (Right) A decision boundary which fails to topologically generalize.

topological generalization cannot be deduced *a priori* by measuring the network's performance on a sample of the data. In Section 5.3 we train small neural networks and explicitly measure their topological generalization.

Furthermore, if the neural networks are selected from a hypothesis class (such as a specific architecture) which does not contain functions which can express the appropriate topological invariants, there will necessarily be failures of topological generalization by all neural networks trained on the given data [14].

Known Topological Failures of Neural Network Behavior

Contemporary neural network functions experience topological and geometric failures in their learned behaviors. For example, in the context of binary classification, toy datasets explored in [14] demonstrate empirically that the architecture of a neural network is predictive of whether it will successfully learn to classify data given generated topological properties. Furthermore, neural networks' architecture were shown to restrict learnable topology. However, theoretical bounds on the exact topological restrictions on the decision boundary were unexplored.

In a more contemporary setting, is common to view generative models as learning a density function on a manifold in their output space. Essentially, random points are generated in $\mathbb{R}^n$ and a neural network function is optimized towards a function whose outputs on these random points are approximately sampled from a density function on an n-dimensional output manifold. It is not surprising that interpolation problems may occur, as direct interpolation between points in $\mathbb{R}^n$ will generally fail to stay on a given submanifold of $\mathbb{R}^n$, but a generative model is specifically designed so that any point in its input *latent space* should output a plausible object of the appropriate manifold, and the goal is that interpolation between those points should correspond to a meaningful path in the output space.

To be more explicit, consider the context of autoencoders, a simple form of generative model (Figure 4). In training an autoencoder, we suppose that data is sampled from an k-dimensional submanifold of $\mathbb{R}^n$. A neural network is trained to compress the data from $\mathbb{R}^n$ into the latent space $\mathbb{R}^k$, and then decompress back into $\mathbb{R}^n$, via a function

$$\mathbb{R}^n \xrightarrow{F_1} \mathbb{R}^k \xrightarrow{F_2} \mathbb{R}^n$$

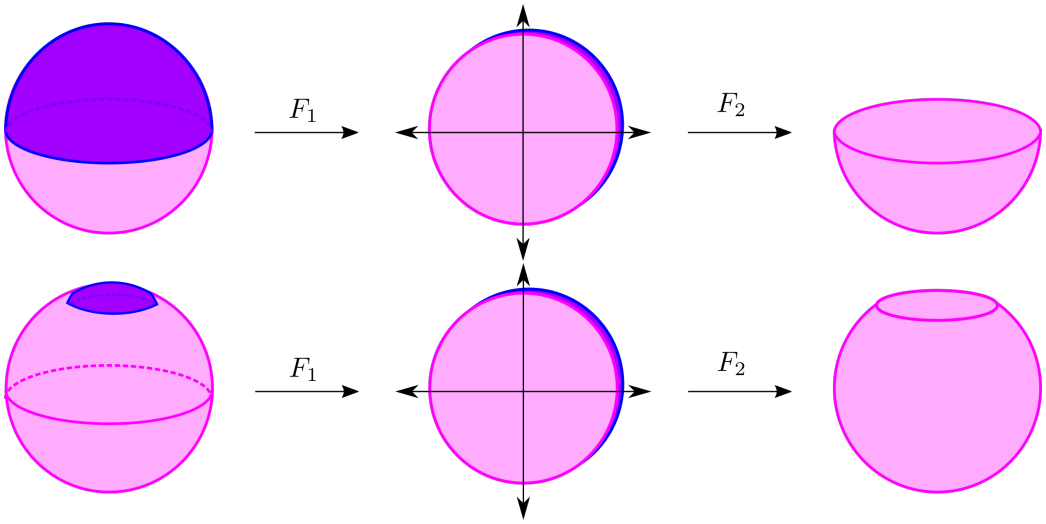


FIGURE 4. In both figures, F_1 is two-to-one. Top: Autoencoder mode collapse due to the failure of F_2 to be surjective. Bottom: The more points which satisfy $F_2 \circ F_1(x) = id(x)$, the less "continuous" the intermediate composites become.

This function is trained to mimize the distance $x - F_2(F_1(x))$, under the heuristic that that points sampled from $\mathbb{R}^k$ should be sent to points in M by approximately the same distribution as the data. If $M \subseteq \mathbb{R}^n$ is a closed k-manifold, the topological problems are obvious; no set of continuous functions F_1 and F_2 can satisfy $F_2 \circ F_1 = id_M$ if M has any nontrivial homology. As the composite is optimized to be as close to the identity as possible, the result is frequently that nearby points in $\mathbb{R}^k$ are sent to vastly different points on M, and interpolation between these points in $\mathbb{R}^k$ passes through out-of-distribution points in $\mathbb{R}^n$. The alternative is significant portions of M fail to be in the image of $F_2 \circ F_1$, a state known as *mode collapse* (Figure 5).

This is still a problem in modern generative models. Even large generative models can exhibit demonstrable topological failure in their behavior. For example, in [20] a large generative model, trained to generate faces from random noise, is demonstrated to have discontinuities of this form through the existence of a

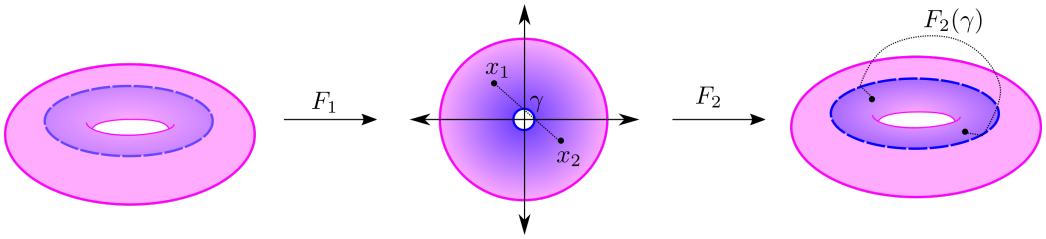


FIGURE 5. Topological errors in interpolation through the latent space of an autoencoder $\mathbb{R}^3 \to \mathbb{R}^2 \to \mathbb{R}^3$.

two-dimensional subset of output images generated from a continuous patch in input space. The patch in input space has the topology of a disc, but the apparent discontinuity in output space demonstrates a topological failure in the latent space embedding.

1.2. Topology and the Linear Regions of ReLU Neural Networks

Models for machine learning based on the *rectified linear unit*, or *ReLU* [23] (popularized in 2010), are widely acknowledged to have improved convergence properties for and decreased computational expense of training neural networks, among other benefits. The mathematical objects under study in this thesis are the class of *fully-connected, feedforward ReLU neural networks,* which may be thought of mathematically as a class of real-parametrized functions.

For fully-connected ReLU networks [23], the canonical polyhedral complex of the network, as defined by [10], encodes its decomposition of input space into piecewise linear regions and determines key structures such as the decision boundary for a binary classification task. Investigation of properties and characterizations of this decomposition of input space are ongoing, in particular with respect to counting the top-dimensional linear regions [26, 17, 22, 25, 28], since these bounds give one measure of the expressivity of the associated network

architecture. It is common to describe linear regions of the input space, $\mathbb{R}^{n_0}$, using "activation patterns" or "neural codes" recorded as vectors in $\{0,1\}^N$ [18]. Unfortunately, having a list of which activation patterns are present in the interiors of the linear regions does not determine their pairwise intersection properties (Theorem 3.3.5), and computing the intersections of these regions directly is not numerically stable. Furthermore, the polyhedra comprising the linear regions appear, at first glance, arbitrarily complicated.

Until now, a theoretical understanding of adjacency between regions and more generally the connectivity of lower-dimensional faces has been undocumented. In this thesis, we give the first full account of face relations in all dimensions, which are encoded by activation patterns we call sign sequences in $\{-1,0,1\}^N$. Such face relations are important because understanding the face relations in the canonical polyhedral complex is necessary to relate combinatorial properties of the polyhedral complex of a network to the topology of regions into which the decision boundary partitions input space, geometric measurements such as the presence of critical points [11], or other notions of topological expressivity, as explored by [14, 5].

1.3. Comparison to Prior Work

The seminal paper [10] establishes a high-level view of the cellular structure of the canonical polyhedral complex, but does not establish explicit low-dimensional information as we do here. Under weak assumptions, they show the canonical polyhedral complex's $(n_0 - 1)$-skeleton may be described as the preimages of hyperplanes from each layer map, but arbitrary k-skeletons are unexamined for $k < (n_0 - 1)$, as are general face relations. The subsequent work [11] establishes local

models for the polyhedral structure at the intersection of hyperplanes in shallow neural networks, but does not address deeper network structures as we do here.

Hyperplane arrangements and sign sequences are used in [16, 17] to study linear regions, but primarily for computing volumes and counting top-dimensional regions, and not obtaining adjacency relations. In particular, in these works properties of hyperplane arrangements are used to establish *statistical* properties of the canonical polyhedral complex. While our work does rely on properties of hyperplane arrangements in a similar way, we focus on encoding the full face poset to obtain *topological* information.

Others who approach explicit computation of linear regions as in [32] do so using halfspace intersections, and we use vertices. While in theory one could intersect top-dimensional regions pairwise to obtain their shared faces, this is not numerically stable, especially when the linear equations involved arise from matrix multiplication. Our algorithm tracks known equalities and uses discrete signs to track face relations, avoiding issues potentially arising from numerical error in polyhedral intersection. Furthermore, in contrast to approaches by [6] and other neuroscientific and biological applications where boundary structure is not biologically meaningful, in the context of artificial networks the boundary intersections are in fact computable.

Other characterizations of the combinatorics of ReLU networks' polyhedral complexes exist, but lack explicit implementation or applicability to deeper networks. The paper [4] describes the regions of the canonical polyhedral complex according to the roots of a polynomial, but no algorithm is presented on how to obtain these roots, nor how to explicitly determine whether two polyhedra are connected by a shared face. The authors of [31] provide a tropical characterization

of the polyhedral complex including its face relations, but rely on the translation of network functions to tropical rational functions with integer coefficients, which is a discontinuous operation. In contrast, we conjecture that the signs of vertices present in the sign sequence cubical complex are stable in open sets of parameter space and could therefore be recorded to track changes in a network's topology during training. The subsequent application of tropical geometry by [2] appears limited to networks with single hidden layers. Additionally, [18] establish a characterization of the regions of single-layer hyperplane networks which relies on similar sign labelings, but the methods do not apply to deeper networks.

Finally, some reachability analysis algorithms using polyhedral methods such as those by [29, 30] use signs to determine how polyhedra have been "split" by a layer map, tracking the face lattices of the output polyhedra of each layer. In theory, this approach could plausibly record which polytopes in the input space intersect in shared regions, but these papers consider polytopes independently from another in the next layer for purposes of parallelization. It is unclear whether enough information is retained to store which "splits" line up between polytopes, as this is not needed in their applications. Additionally, this approach requires storing all faces for the face lattice in each polytope, whereas we show this information is fully contained in the sign sequences of the vertices.

1.4. Dissertation Summary

This dissertation establishes a combinatorial description of ReLU neural network functions' **canonical polyhedral complex** using algebraic tools from hyperplane arrangements and oriented matroid theory. This structure gives an approach to obtain explicit topological descriptions of the level sets of neural

network functions, which are furthermore implementable in code, computing the described structure when given a specific neural network. In addition, tools from *piecewise linear Morse theory* are applicable to the broader (not generally PL-Morse) class of ReLU neural network functions, giving general tools for understanding the topology of sublevel and level sets of ReLU functions [11]. Here we additionally provide local characterizations of vertices of the canonical polyhedral complex before approaching an extension of this understanding to deep neural networks.

In Chapter II we introduce a number of preliminaries (Sections 2.1-2.2), followed by the introduction of a category which preserves the combinatorics of polyhedral complexes under piecewise linear maps in Section 2.3, finally presenting an alternative development of the canonical polyhedral complex within this category in Section 2.4.

In Chapter III, we then seek to understand the cellular topology of the canonical polyhedral complex through its face poset and local characterizations of its vertices. These characterizations rely on piecewise linear analogs of transversality, reviewed in Section 3.1. Then, in 3.2 we introduce appropriate conditions for a ReLU neural network to satisfy the results of the rest of this work, and show that these conditions are (fiberwise) generic. In the following Section 3.3, we see that the bent hyperplane arrangement inherits a number of properties from co-oriented hyperplane arrangements in $\mathbb{R}^{n_i}$ under these transversality assumptions about the layers of a neural network. For example, a co-oriented hyperplane arrangement of m hyperplanes in $\mathbb{R}^n$ induces an assignment of a *sign sequence* in $\{-1, 0, 1\}^m$ to each point in $\mathbb{R}^n$ [1]. This sign sequence pulls back to the canonical polyhedral complex, and uniquely identifies each cell of the complex, providing a

combinatorial description of each cell [32], which we reprove here in an alternative approach.

In Section 3.4 we then show that the *composition operation* between sign sequences, derived from the same operation in oriented matroids [3], is additionally well-defined as an operation between cells of the canonical polyhedral complex, and fully encodes the cellular poset of the canonical polyhedral complex. As a result, we see in Section 3.5 that the geometric dual of the canonical polyhedral complex is a cubical complex consisting of n_0-dimensional cubes in an N-dimensional hypercube $[-1,1]^N$, where $N = \sum_{i=1}^{m} n_i$ is the sum of the intermediate dimensions of the ReLU neural network.

An immediate corollary is that for almost all ReLU neural networks the full cellular poset can be generated by the sign sequences of the vertices of the canonical polyhedral complex together with the coboundary operation, which is dual to the boundary operation in the cube. Methods to use this sign sequence information to obtain the topological invariants of the decision boundary and the presence of PL critical points are then described in Sections 3.6 and 3.7, respectively.

In Chapter IV we then explore how the vertices of the canonical polyhedral complex, together with their sign sequences, can be computed in a relatively numerically stable way, even though sign evaluation in $\{-1,0,1\}$ is unstable. This allows for the explicit computation of a structure encoding the canonical polyhedral complex without needing to store information separately about all cells, which is combinatorially explosive.

We provide algorithms in Sections 4.1 and 4.3, which we have implemented in Python, which compute the canonical polyhedral complex of a given neural

network together with Sage scripts which evaluate topological invariants of a decision boundary with an added point at infinity. Discussion of the numerical stability of these algorithms is provided in Section 4.2. Once the vertices of the canonical polyhedral complex are computed, tracking face relations of cells using sign sequences is computationally straightforward, and each bent hyperplane is readily encoded as the cellular subcomplex of the canonical polyhedral complex whose cells' sign sequences contain a zero in a distinguished coordinate. We discuss both a naïve and a more efficient method for obtaining this same information.

We finish in Chapter V with empirical experiments about neural networks. For random neural networks at initialization, we obtain distributional information about the number and minimum pairwise distance between vertices (Section 5.1), as well as statistics about the mod-two homology of the decision boundary with the distinguished point at infinity at initialization (Section 5.2). We additionally track topological invariants of the decision boundary for neural networks which have been trained on synthetic data for specific topological tasks (Section 5.3). These empirical measurements demonstrate that increasing depth increases the variability of neural networks' topological behavior both at initialization and during training, but increasing width appears to decrease the observed variability, even though increased width is known to increase networks' theoretical topological expressivity.

1.5. Further Directions

The tools developed in this work can be useful by providing a local characterization of vertices of $\mathcal{C}(F)$, which can be used to obtain geometric properties of ReLU functions. For example, access to solid angles would allow for extensions to work such as [4] on distributional properties of local curvature of

decision boundaries. This local characterization additionally makes piecewise linear Morse theory as in [11, 12] applicable to $\mathcal{C}(F)$, as discussed briefly in section 3.7. Morse theory describes local extrema and saddle points by diagonalizing a second derivative matrix. Replacing a 0 with ± 1 in a vertex's sign sequence identifies opposite edges incident to a vertex, giving axes for a "piecewise linear second derivative."

Another possible future application of this work is to analyze topological generalization of real networks. A key indicator of a network's generalizability is whether its sublevel sets have the appropriate topological properties [5]. In addition, the architecture of a classification network influences the topology of the expressible decision boundaries of that network [14]. Empirically, topological data analysis provides practical approximation for low-dimensional features of high dimensional data. While approximations exist to obtain the topological properties of a network's decision boundaries using topological data analysis [21], these properties are dependent on the geometry of the network function. In places where the network's decision boundary has high curvature, the approximation methods may lead to inaccuracy between the true topology of a network's decision boundary and the topology which is approximated by persistent homology methods, but it is precisely those locations where a network is vulnerable to adversarial examples [8]. A measure of the true topology of the decision boundary could provide a metric for comparison.

Lastly, it may be possible to obtain an explicit theoretical understanding of the evolution of a network's decision boundary through a training path, and not merely an empirical understanding as is described here. We believe that changes in the face structure of $\mathcal{C}(F)$ should be "discrete" in that they only change at

finitely many locations in a general training path, which would thus open an avenue for tracking the vertices of $\mathcal{C}(F)$ as the network trains in parameter space, and establishing theoretical limits on the possible discrete changes that may occur to the set of vertices of $\mathcal{C}(F)$ during training.

CHAPTER II

CATEGORICALLY DEVELOPING THE CANONICAL POLYHEDRAL COMPLEX

We proceed by defining what is meant by a polyhedral complex and a neural network before defining the primary object of study, the **canonical polyhedral complex** $\mathcal{C}(F)$ of a neural network F, as introduced by Grigsby and Lindsey [10]. We provide an alternative characterization of $\mathcal{C}(F)$ as a natural construction arising from the composition operation in the appropriate category.

2.1. Preliminaries: ReLU Neural Networks

The ReLU function became popularized in the early 2010s as a function which enabled the improved training of neural networks [23]. However, because ReLU neural networks are constructed from functions which are not smooth, many classical results about smooth functions cannot be applied to their analysis or their sublevel and level set topology.

Definition 2.1.1 (ReLU, σ_n)**.** The following terms refer to activation functions of a neural network:

- ReLU: $\mathbb{R} \to \mathbb{R}$ denotes the function $\text{ReLU}(x) := \max\{0, x\}$.
- $\sigma_n : \mathbb{R}^n \to \mathbb{R}^n$ is the function which applies ReLU coordinatewise. By abuse, we often drop n and write σ when dimensionality is clear.

We introduce the definition of a neural network here to motivate the notation used in the rest of this document. Aligning with the framework in [10], we investigate the following class of neural network functions.

Definition 2.1.2 ([10], Definition 2.1)**.** Let $n_0, ..., n_m \in \mathbb{N}$. A **fully-connected ReLU neural network** with **architecture** $(n_0, ..., n_m, 1)$ is a collection $\mathcal{N} = \{A_i\}$ of affine maps $A_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_{i+1}}$ for $i = 0, ..., m$. Such a collection determines a function $F_{\mathcal{N}} : \mathbb{R}^{n_0} \to \mathbb{R}$, the ***associated neural network map***, given by the composite

$$\mathbb{R}^{n_0} \xrightarrow{F_1=\text{ReLU}\circ A_1} \mathbb{R}^{n_1} \xrightarrow{F_2=\text{ReLU}\circ A_2} \mathbb{R}^{n_2} \xrightarrow{F_3=\text{ReLU}\circ A_3} ... \xrightarrow{F_m=\text{ReLU}\circ A_m} \mathbb{R}^{n_m} \xrightarrow{G=A_{m+1}} \mathbb{R}^1$$

We say that this network has **depth** $m+1$ and **width** $\max\{n_1, ..., n_m, 1\}$. The maps F_k are called the kth **layer maps**.

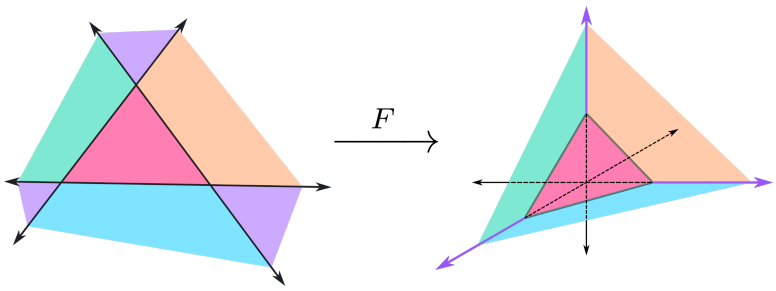


FIGURE 6. A layer map, with its associated hyperplane arrangement.

Each neural network function can be decomposed into its intermediate composites, and this perspective will become useful later. We introduce the following notation here to capture this process:

Definition 2.1.3. If $F = G \circ F_m \circ ... \circ F_1$ is a ReLU neural network with $F : \mathbb{R}^{n_0} \to \mathbb{R}$, then we denote:

$$F_{(k)} = F_k \circ ... \circ F_1$$

and write that this is F **ending at the** k**th layer**.

Likewise, we denote

$$F^{(k)} = G \circ F_m \circ ... \circ F_k$$

and call $F^{(k)} : \mathbb{R}^{n_{k-1}} \to \mathbb{R}$ by F **starting at the kth layer**.

Thus $F = F^{(k)} \circ F_{(k-1)}$ for any k.

In addition to decomposing neural networks by their intermediate composites, each layer map is associated to a hyperplane arrangement in a canonical way. In [10] it is established that a co-oriented hyperplane arrangement,

$$\mathbf{A}_k = \{\mathbf{H}_1^{(k)}, ..., \mathbf{H}_{n_k}^{(k)}\}$$

in $\mathbb{R}^{n_{k-1}}$ may be associated to each layer map F_k, defined by the affine solution set arrangement of A_k as follows:

Definition 2.1.4 ([10] §2)**.** If the affine map A_k is given by $(W^{(k)}|b^{(k)})$, with rows $W_i^{(k)}$ not identically zero, the solution set of $(W_i^{(k)}|b_i^{(k)}) \cdot (\vec{x}|1) = 0$ gives hyperplane $\mathbf{H}_i^{(k)}$, with co-orientation defined by a unit normal vector in the direction of the gradient of $(W_i|b_i) \cdot (\vec{x}|1)$. Furthermore, the **positive half-space** $H_i^{(k)+}$

$$\mathbf{H}_i^{(k)+} := \{\vec{x} \in \mathbb{R}^n : (W_i^{(k)}|b_i^{(k)}) \cdot (\vec{x}|1) > 0\}$$

and the region $\mathbf{H}_i^{(k)-}$ is defined analogously.

Note: It is traditional to call $W^{(k)}$ the **weights** of A_k and $b^{(k)}$ the **biases** of A_k.

Properties of these hyperplane arrangements will be useful for deriving properties of neural networks. The first, most basic property is genericity,

which captures whether each layer map in isolation is associated with a generic hyperplane arrangement.

Definition 2.1.5 ([10] 2.9)**.** A layer map F_k is said to be **generic** if the corresponding affine solution set arrangement $\mathbf{H}_i^{(k)}$ is generic as a hyperplane arrangement. A neural network whose layer maps are all generic is called a **generic neural network**.

In [10] it is shown that (Lebesgue) almost all neural networks of a given architecture are generic. Generic hyperplane arrangements have well-documented properties; see [1].

2.2. Preliminaries: Polyhedral Geometry

We will associate a polyhedral complex to each ReLU neural network. Polyhedral complexes are constructed from polyhedral sets. Here we use the term *polyhedral set* to refer to an intersection of halfspaces which may be unbounded, in contrast to polytopes, which are generally defined to be compact. A summary of relevant results and definitions about polyhedral complexes and complexes arising from affine hyperplane arrangements found in [10], Sections 2-3, and [12], Sections 1-2, with the most relevant information repeated below.

Definition 2.2.1 (Polyhedra, Polyhedral Complex, cf. [10])**.**

- A **polyhedron** is an intersection of the form $\bigcap_{1\leq i\leq m} H_i^+$ for some set of (codimension 1) hyperplanes $H_1, ..., H_m \subset \mathbb{R}^n$. Here H_i^+ is the half-space of $\mathbb{R}^n$ consisting of the union of the hyperplane H_i and one of the connected components of $\mathbb{R}^n \setminus H_i$.

– A point is on the **interior** of a polyhedron P if it is on the interior of P with respect to the subspace topology of the affine span of P, except when P is a point, in which case by convention its interior is nonempty. We use the notation P° to denote the interior of P.

– A **face** of a polyhedron P embedded in $\mathbb{R}^n$ is a set of the form $H \cap P$, where H is a codimension 1 hyperplane in $\mathbb{R}^n$ and $H \cap P$ does not contain any interior point of P. (The empty set is a face of any polyhedron.) A hyperplane which intersects P in a nonempty face is called a **supporting hyperplane** of P. All other hyperplanes which intersect P are called **cutting hyperplanes** of P. We denote the relation "C is a face of D" with $C \leq D$.

Under this definition, polyhedra may not be bounded, but they are always closed. As a result, an affine hyperplane arrangement induces a natural polyhedral decomposition on its ambient space.

For convenience, we occasionally make use of the following:

Definition 2.2.2. Each **proper face** F of P is itself a polyhedral set in $\mathbb{R}^n$ of dimension less than or equal to $n-1$. We call the inclusion $\iota_F^P : F \to P$ the **characteristic inclusion of F in P.**

If D and F are faces of P satisfying $D \subseteq F$, then $\iota_D^P = \iota_F^P \circ \iota_D^F$. Finally, polyhedral complexes are defined in [10, 12] as follows:

Definition 2.2.3 ([10], 3.11)**.** A **polyhedral complex** $\mathcal{C}$ of dimension d is a finite set of polyhedral sets of dimension k embedded in $\mathbb{R}^n$, for $0 \leq k \leq d$, called the **cells** of $\mathcal{C}$, satisfying the properties:

(i) if $P \in \mathcal{C}$, then every face of P is in $\mathcal{C}$.

(ii) If $P, Q \in \mathcal{C}$, then $P \cap Q$ is the single mutual face of P and Q.

Note that it is not necessarily the case that $n = d$, that is, polyhedral complexes of dimension $d < n$ may be embedded in $\mathbb{R}^n$.

Next, we provide notation for the underlying set of a polyhedral complex.

Definition 2.2.4 ([10]). If $\mathcal{C}$ is a polyhedral complex, we denote by $|\mathcal{C}|$ the **underlying set** of $\mathcal{C}$, given by the union of all cells in $\mathcal{C}$.

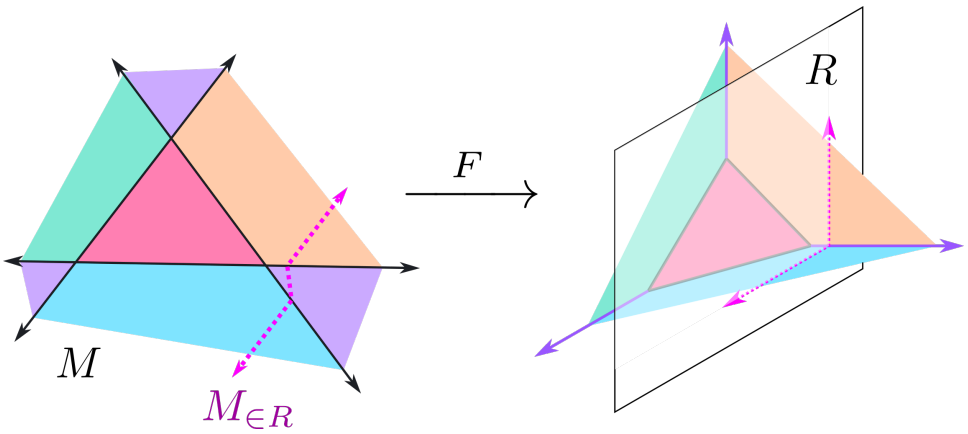


FIGURE 7. If $F : \mathbb{R}^2 \to \mathbb{R}^3$ is a layer map, R and $M_{\in R}$ pictured.

This notion of each polyhedral complex having an underlying set is used when constructing new polyhedral complexes through piecewise linear maps (see Figure 7):

Definition 2.2.5 ([10], [12]). Let M and R be polyhedral complexes with a map $f : |M| \to \mathbb{R}^r$ linear on cells of M, where R is embedded in $\mathbb{R}^r$. The **level set complex of** f is the set $M_{\in R}$ defined as:

$$M_{\in R} := \{S \cap f^{-1}(Y) | S \in M, Y \in R\}$$

It is straightforward and established in [10, 12] that this does indeed result in a polyhedral complex and the resulting map is cellular on the subdivision $M_{\in R}$, which we will use to construct a category for polyhedral complexes in which combinatorial information is preserved under isomorphisms.

Later, we will additionally make use of the following lemma regarding boundary relations, rewritten so as to not require the $M_{\in R}$ notation.

Lemma 2.2.6 (cf. [12], Lemma 2.4)**.** *Let $M \subseteq \mathbb{R}^m$ and $N \subseteq \mathbb{R}^n$ be polyhedral complexes and $f : |M| \to \mathbb{R}^n$ be continuous and affine on cells of M. Let $\leq$ denote face relations in the respective polyhedral complexes. If $C \leq C'$ are polyhedra in M, and $D \leq D'$ are polyhedra in N, then*

$$C \cap f^{-1}(D) \leq C' \cap f^{-1}(D')$$

is a face relation in the polyhedral complex consisting of the cells

$$\{C \cap F^{-1}(D) : D \in N, C \in M\}.$$

Finally, we use the following notation for consistency with previous work, providing a notation for affine hyperplane arrangement $R^{(i)}$ associated to an affine map A_i.

Definition 2.2.7 ($R^{(i)}$, π_j, cf. [10], Definition 6.7)**.** Let $A_i : \mathbb{R}^{n_{i-1}} \to \mathbb{R}^{n_i}$ be an affine function for $1 \leq i \leq n$. Denote by $R^{(i)}$ the polyhedral complex associated to the hyperplane arrangement in $\mathbb{R}^{n_{i-1}}$, induced by the hyperplanes given by the solution set to $H_{ij} = \{x \in \mathbb{R}^n : \pi_j \circ A_i(x) = 0\}$, where π_j is the linear projection onto the jth coordinate in $\mathbb{R}^{n_i}$.

2.3. A Category for ReLU Neural Networks' Layer Maps

There is no one generally accepted category for polyhedral complexes, so we define a suitable category consisting, intuitively, of finite, geometrically realizable polyhedral complexes, beginning with the objects:

Definition 2.3.1 (Objects in Euclidean Polyhedral Complexes)**.** An object in Euclidean Polyhedral Complexes (EPoCx) is a finite **Euclidean polyhedral complex**, that is, a polyhedral complex $\mathcal{C} = \{P_1 ... P_n\}$ embedded in $\mathbb{R}^m$.

Classically, a piecewise affine linear function is a continuous map $f : |\mathcal{C}| \to |\mathcal{D}|$ such that there exists a polyhedral subdivision $\mathcal{C}'$ of $\mathcal{C}$ on which f is affine linear. If we take these to be morphisms in this category with no further structure, then the combinatorial structure (face poset) of polyhedral complexes is not defined up to isomorphism.

To enforce the definition of combinatorial structure in this category up to isomorphism between polyhedral complexes, we define morphisms to be generated through functions on their underlying sets of the following two types.

Definition 2.3.2 (Morphisms for Euclidean Polyhedral Complexes)**.** We define morphisms in EPoCx in the following way:

- A **polyhedral subdivision** of a polyhedral complex $\mathcal{C}$ is a polyhedral complex $\mathcal{C}'$ such that $|\mathcal{C}'| = |\mathcal{C}|$ and every cell $C' \in \mathcal{C}'$ is a subset of a cell $C \in \mathcal{C}$. If $\mathcal{C}'$ is a subdivision of $\mathcal{C}$ then there is a formal **subdivision morphism** from $\mathcal{C}$ to $\mathcal{C}'$.

- A **strictly affine morphism** from a polyhedral complex $\mathcal{C}$ to $\mathcal{D}$ is a function $|\mathcal{C}| \to |\mathcal{D}|$ which is continuous, affine on cells, and the image of the interior of

each cell $C^\circ \in \mathcal{C}$ is contained in the interior of a single cell $D^\circ \in \mathcal{D}$. (This cell D is therefore determined by C).

– A **morphism in EPoCx** is a subdivision morphism followed by a strictly affine morphism.

Strictly affine morphisms are so named due to being a stronger condition than simply requiring the restriction of the function to each cell to be affine (which is the traditional definition of piecewise linear functions). However, strictly affine is a weaker condition than *cellular* morphisms (which are ones in which the image of each cell $C \in \mathcal{C}$ would need to be *equal* to a cell in $\mathcal{D}$.) Two morphisms in EPoCx may be considered as equal if and only if their subdivision morphisms and their strictly affine morphisms are equal. We choose to permit only formal subdivisions, and not the inverse, in order to ensure, as shall be seen, that the combinatorics of a polyhedral complex are preserved under isomorphism.

To equip EPoCx with structure of a category we must define composition between these morphisms. We note briefly that the identity morphism $\iota : \mathcal{C} \to \mathcal{C}$ is given by the trivial subdivision which keeps the same set of polyhedra, followed by the identity morphism on $|\mathcal{C}|$. However, composition is more complicated.

Lemma 2.3.3. *Let $F : \mathcal{C} \to \mathcal{D}$ and $G : \mathcal{D} \to \mathcal{E}$ be morphisms in EPoCx, equipped with associated subdivision morphisms $F_S : \mathcal{C} \to \mathcal{C}'$ and $G_S : \mathcal{D} \to \mathcal{D}'$, and equipped with strictly affine morphisms given by associated functions $f : |\mathcal{C}'| \to |\mathcal{D}|$ and $g : |\mathcal{D}'| \to |\mathcal{E}|$. Then we define $G \circ F : \mathcal{C} \to \mathcal{E}$ as follows.*

First, the associated function on underlying sets is given by $g \circ f$, and second, the associated subdivision is $\mathcal{C}'' = \mathcal{C}'_{\in \mathcal{D}'}$ (Definition 2.2.5).

This construction assures the associativity of $\circ$ in EPoCx.

Proof. First, we must justify that $g \circ f$ is a strictly affine morphism from $|\mathcal{C}''|$ to $\mathcal{D}$ to finish ensuring that morphisms in EPoCx are closed under composition.

We must see that $g \circ f$ is affine on cells in $\mathcal{C}''$ and that for each cell C in $\mathcal{C}''$, $g \circ f(C)$ is contained in a single cell of $\mathcal{E}$. Let $C \in \mathcal{C}''$. Then $C \subset S$ for some $S \in \mathcal{C}'$. Because f is strictly affine, $f|_S$ is affine and therefore $f|_C$ is affine. Next, by definition, $C = S \cap f^{-1}(Y)$ for some $S \in \mathcal{C}'$ and some $Y \in \mathcal{D}'$. This implies that $f(C) \subset Y$ for some polyhedron $Y \in \mathcal{D}'$. Because g is strictly affine, $g|_Y$ is affine, so $g|_{f(C)}$ is affine. Furthermore, because g is strictly affine, $g(Y)$ is contained in a single cell $E \in \mathcal{E}$, so $g(f(C))$ is also contained in that same cell.

Next, we show associativity. Function composition on sets is already associative. Thus, we only need to ensure that the subdivision morphism associated with $H \circ (G \circ F)$ is equal to the subdivision associated with $(H \circ G) \circ F$.

Let $F : \mathcal{C} \to \mathcal{D}$, $G : \mathcal{D} \to \mathcal{E}$ and $H : \mathcal{E} \to \mathcal{C}(F)$ be morphisms in EPoCx, with associated subdivisions $\mathcal{C}_f$, $\mathcal{D}_g$ and $\mathcal{E}_h$ respectively. The cells of the subdivision associated to $H \circ (G \circ F)$ are given by the following:

$$\mathcal{C}_{h(gf)} = \{T \cap (g \circ f)^{-1}(Z) \mid T \in \mathcal{C}_{gf}, Z \in \mathcal{E}_h\}$$

where $\mathcal{C}_{gf} = \{S \cap f^{-1}(Y) \mid S \in \mathcal{C}_f, Y \in \mathcal{D}_g\}$.

In comparison, the cells of the subdivision associated to $(H \circ G) \circ F$ are given by the following:

$$\mathcal{C}_{(hg)f} = \{S \cap f^{-1}(X) \mid S \in \mathcal{C}_f, X \in \mathcal{D}_{hg}\}$$

where $\mathcal{D}_{hg} = \{Y \cap g^{-1}(Z) \mid Y \in \mathcal{D}_g, Z \in \mathcal{E}_h\}$

Compressing these set definitions, we have:

$$\mathcal{C}_{h(gf)} = \{(S \cap f^{-1}(Y)) \cap (g \circ f)^{-1}(Z) \mid S \in \mathcal{C}_f, Y \in \mathcal{D}_g, Z \in \mathcal{E}_h\}$$

and

$$\mathcal{C}_{(hg)f} = \{S \cap f^{-1}(Y \cap g^{-1}(Z)) \mid S \in \mathcal{C}_f, Y \in \mathcal{D}_g, Z \in \mathcal{E}_h\}$$

Written in this way, we observe that since preimages distribute over intersection, $S \cap f^{-1}(Y \cap g^{-1}(Z)) = (S \cap (f^{-1}(Y)) \cap (g \circ f)^{-1}(Z))$ and the two sets of polyhedra (and therefore the two subdivisions) are equal.

□

This structure is sufficient to ensure that relevant combinatorial structure is defined up to isomorphism between polyhedral complexes, since a composition of morphisms in this category can only induce further subdivisions of a polyhedral complex.

Lemma 2.3.4. *In EPoCx, a strictly affine morphism $F : \mathcal{C} \to \mathcal{D}$ induces an order-preserving map on the face poset of $\mathcal{C}$ to the face poset of $\mathcal{D}$. Resultingly, if $\mathcal{C}$ and $\mathcal{D}$ are isomorphic in EPoCx, they have order- and grade-equivalent face posets, where grading is given by face dimension.*

Proof. First, let $C \leq C'$ in $\mathcal{C}$ and let $F : \mathcal{C} \to \mathcal{D}$ be a strictly affine morphism. By definition of strictly affine, there is a unique cell $D' \in \mathcal{D}$ such that $F(C'^{\circ}) \subset D'^{\circ}$. Since C is contained in the closure of C', we have that $f(C)$ is contained in the closure of D'. So, $f(C)$ must be contained in either D'° or the interior of one of the faces of D'. The map between sets of polyhedra given by sending each cell $C \in \mathcal{C}$ to the unique $D \in \mathcal{D}$ with $f(C^{\circ}) \subset D^{\circ}$ is thus order-preserving.

If $\mathcal{C}$ and $\mathcal{D}$ are isomorphic in EPoCx by $F : \mathcal{C} \to \mathcal{D}$ and $G : \mathcal{D} \to \mathcal{C}$ (and f, g the corresponding induced map on underlying sets) we note that if F or G

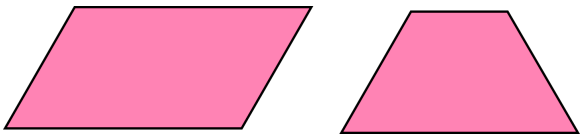


FIGURE 8. Two combinatorially equivalent but nonisomorphic polyhedral complexes in EPoCx. No affine function will send the parallelogram directly to the trapezoid without subdivision.

includes any subdivision except the trivial subdivision morphism on their respective polyhedral complexes then the subdivision associated with $G \circ F$ or $F \circ G$, which is a further subdivision, will not be the trivial subdivision, making either $G \circ F$ or $F \circ G$ not equal to the identity in EPoCx on its respective polyhedral complex. Thus, both F and G may be taken to be strictly affine maps and by the previous paragraph must induce order-preserving maps on the face posets of $\mathcal{C}$ and $\mathcal{D}$. Furthermore the composites $g \circ f$ and $f \circ g$ induce order isomorphisms from the face poset of $\mathcal{C}$ to itself and the face poset of $\mathcal{D}$ to itself, so the face posets of $\mathcal{C}$ and $\mathcal{D}$ are isomorphic with isomorphisms given by the induced maps from f and g.

We observe that if $f(C^\circ) \subset D^\circ$ then $g(D^\circ) \subset C^\circ$ by the order isomorphism. However, $(g \circ f)|_C = id_C$ so $g(D) = C$. By symmetry, $f(C) = D$. Thus, the two cells must be topologically homeomorphic and therefore the same dimension. □

Two polyhedral complexes which are combinatorially equivalent might not be isomorphic in this category. Indeed, no affine map will send a parallelogram in $\mathbb{R}^2$ to a general quadrilateral in $\mathbb{R}^2$ even though the two polyhedra are combinatorially equivalent (Fig. 8). We highlight this as a *geometric* obstruction to isomorphism, in contrast with the more obvious combinatorial obstructions to isomorphism. So, while the face poset is an isomorphism invariant in this category, it does not uniquely determine each polyhedral complex even in the most simple cases.

We additionally wish to establish that pullbacks exist in this category and furthermore the general construction $M_{\in R}$ is a pullback in this category.

Lemma 2.3.5. *If $F : \mathcal{C} \to \mathcal{Z}$ and $G : \mathcal{D} \to \mathcal{Z}$ are morphisms in EPoCx with associated subdivisions $\mathcal{C}_f$ and $\mathcal{D}_g$ then the pullback $\mathcal{P}$ of f and g exists and can be characterized as the polyhedral complex whose cells can are given (possibly not uniquely) by*

$$\{(x, y) \in C \times D : f(x) = g(y)\}$$

for cells $C \in \mathcal{C}_f$ and $D \in \mathcal{D}_g$.

Proof. We observe that $\mathcal{P}$ is a polyhedral complex when equipped with the given cell structure.

Define the coordinatewise projections p_1 and p_2 of each cell $P = \{(x, y) \in C \times D : f(x) = g(y)\}$ to C and D. We note that p_1 and p_2 affine on P and their images are contained entirely on C and D. Thus taking p_1 and p_2 on each cell of $\mathcal{P}$ extends to strictly affine morphisms on $\mathcal{P}$, thus these are morphisms in EPoCx and the following diagram commutes:

$$\begin{array}{ccc} \mathcal{P} & \xrightarrow{P_2} & \mathcal{D} \\ \downarrow{\scriptstyle P_1} & & \downarrow{\scriptstyle G} \\ \mathcal{C} & \xrightarrow{F} & \mathcal{Z} \end{array}$$

Observe that subdivision on $\mathcal{P}$ given by $P_1 \circ F$ is equal to the subdivision on $\mathcal{P}$ given by p_1, which is the trivial subdivision, by construction.

Next, suppose $\mathcal{Q}$ is another polyhedral complex such that the following diagram commutes:

$$\begin{array}{ccc} \mathcal{Q} & \xrightarrow{Q_2} & \mathcal{D} \\ \downarrow{\scriptstyle Q_1} & & \downarrow{\scriptstyle G} \\ \mathcal{C} & \xrightarrow{F} & \mathcal{Z} \end{array}$$

Then $u = (q_1, q_2) : |\mathcal{Q}| \to |\mathcal{C}| \times |\mathcal{D}|$ is a piecewise linear map with image in $|\mathcal{P}|$. We also note the underlying set of $|\mathcal{P}|$ is equal to the pullback in **Sets**, so the map $u : |\mathcal{Q}| \to |\mathcal{P}|$ is the unique map in **Set** respecting the commutative diagram by the uniqueness of pullbacks in **Set**.

We also need to show that there is a unique choice of corresponding subdivision of the map $|\mathcal{Q}| \to |\mathcal{P}|$ such that this diagram commutes. As $F \circ Q_1 = G \circ Q_2$, in particular the subdivisions corresponding to both composites must be equal. Let $\mathcal{Q}_{fg}$ be this shared subdivision. We claim that setting the subdivision corresponding to U to the same subdivision as $\mathcal{Q}_{fg}$ is the unique choice of subdivision such that the diagram commutes in EPoCx.

Selecting this subdivision does ensure that $U : |\mathcal{Q}| \to |\mathcal{P}|$ is a morphism in EPoCx. Let Q be a cell in the subdivision $\mathcal{Q}_{fg}$. By construction we know that $q_1|_Q : Q \to |\mathcal{C}|$ and $q_2|_Q : Q \to |\mathcal{D}|$ are affine on Q and have image in C and D respectively, where C and D are cells of $\mathcal{C}_f$ and $\mathcal{D}_g$ respectively. Thus the product $(q_1, q_2) : Q \to C \times D$ is affine. As $g \circ q_1 = f \circ q_2$ the image of (q_1, q_2) is contained in a single cell P of $\mathcal{P}$. Therefore, (q_1, q_2) is a strictly affine morphism.

Additionally, since the subdivisions corresponding to $F \circ P_1$ and $G \circ P_2$ are the trivial subdivision on $\mathcal{P}$, the composite subdivision corresponding to $F \circ P_1 \circ U$ must equal the subdivision corresponding to U. As the composite $F \circ P_1 \circ U$ must have the subdivision Q_{fg} this must be the same subdivision corresponding to U.

Thus, not only is the choice of setwise function for U unique, but the choice of subdivision is as well, and $\mathcal{P}$ is a pullback in EPoCx. □

In particular, the polyhedral complex $M_{\in R}$ is isomorphic to the pullback in this diagram:

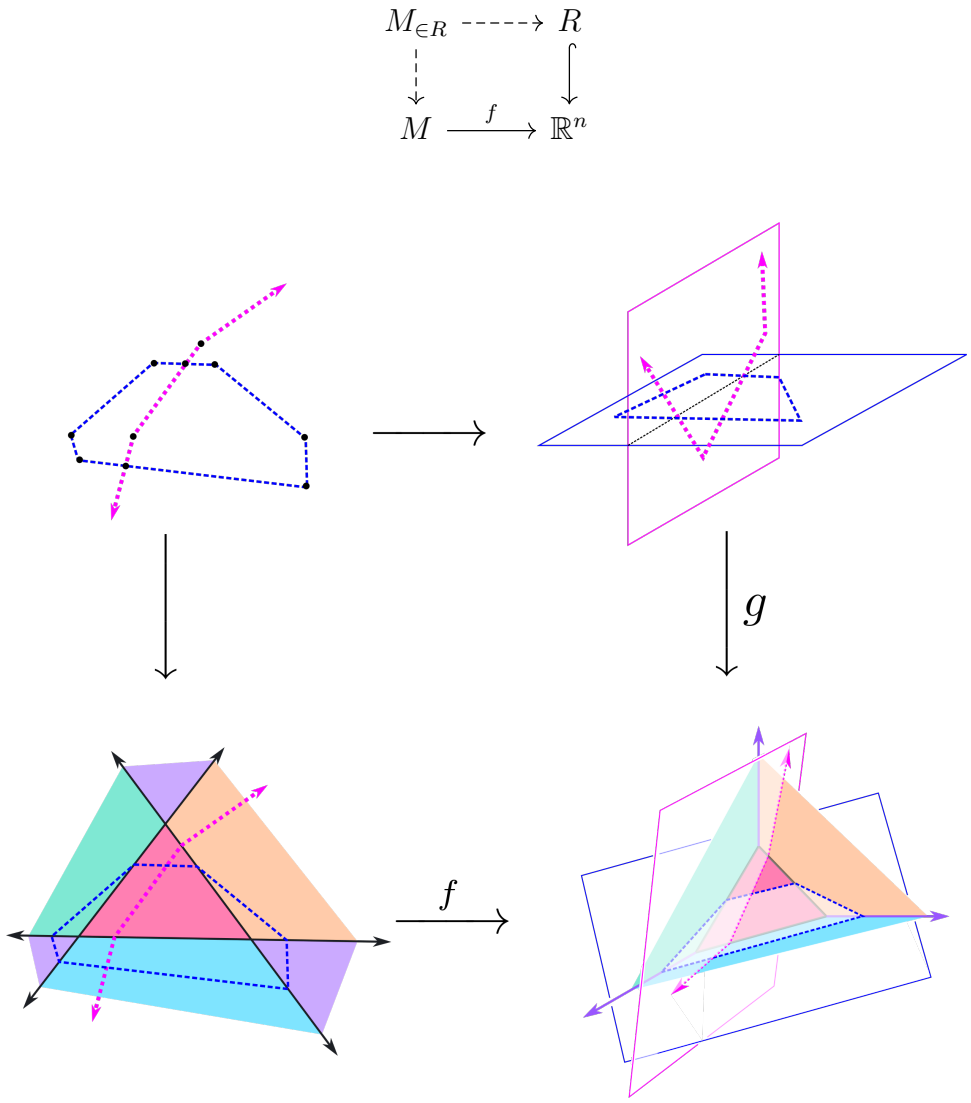


FIGURE 9. The pullback diagram of $M_{\in R}$.

2.4. The Canonical Polyhedral Complex Constructed Categorically

For fully-connected ReLU networks [23], the canonical polyhedral complex of the network, as defined by Grigsby and Lindsey [10], encodes its decomposition of input space into linear regions and determines key structures such as the decision boundary for a binary classification task. Investigation of properties and characterizations of this decomposition of input space are ongoing, in particular

with respect to counting the top-dimensional linear regions [26, 17, 22, 25, 28], since these bounds give one measure of the expressivity of the associated network architecture.

Below we provide several equivalent definitions for the canonical polyhedral complex, and show that they are equivalent to the definition by Grigsby and Lindsey [10].

The Canonical Polyhedral Complex

As a piecewise-affine linear function, a neural network function F_N, which we simplify to F, defines an obvious polyhedral decomposition of input space, namely into the (largest) polyhedra on which it is affine-linear. However [10] shows the utility of considering not only the decomposition which F itself defines, but the common refinement of decompositions by intermediate composites.

We may define the canonical polyhedral complex $\mathcal{C}(F)$ as the subdivision in EPoCx of $\mathbb{R}^{n_0}$ induced by the composite $F_m \circ ... \circ F_1$ cf. Definition 2.3.2, where each of the $F_i : \mathbb{R}^{n_{i-1}} \to \mathbb{R}^{n_i}$ consists of the subdivision given by $R^{(i)}$ followed by the piecewise affine function, as described.

For implementation, we prefer a definition through explicit identification of cells, using further language from [10]:

Definition 2.4.2 ([10], Definition 8.1)**.** If F is a ReLU neural network, the **node map** $F_{i,j}$ is defined by:

$$\pi_j \circ A_i \circ F_{i-1} \circ ... \circ F_1 : \mathbb{R}^{n_0} \to \mathbb{R}$$

Remark 2.4.3. Note here that π_j is projection on to the jth coordinate, i indexes layers, and j indexes neurons. For fixed i, the solutions to $\pi_j \circ A_i = 0$ are hyperplanes in $\mathbb{R}^{n_{i-1}}$, which together form a hyperplane arrangement described earlier. These hyperplane arrangements are equipped naturally with the structure of a polyhedral complex. Recall that we denote the polyhedral complex associated to A_i by $R^{(i)}$ (Definition 2.2.7).

In particular, the locus in input space where $F_{ij} = 0$ is of particular interest, and to draw analogies to hyperplane arrangements, we use the phrase "bent hyperplane."

Definition 2.4.4 ([10], Definition 6.1)**.** A **bent hyperplane** of $\mathcal{C}(F)$ is the preimage of 0 under a node map, that is, $F_{ij}^{-1}(0)$ for fixed i, j.

A bent hyperplane can contain polyhedral regions with codimension less than one, but this occurs with zero probability (see Figure 10). The conditions under which the bent hyperplanes' maximal cells are always codimension 1 are listed by [10].

The canonical polyhedral complex $\mathcal{C}(F)$ consists precisely of the polyhedra in $\mathbb{R}^{n_0}$ which may be described by selecting one polyhedron $R_i \subset \mathbb{R}^{n_i}$ from each hyperplane arrangement $R^{(i)}$, considering the preimages $(F_{i-1} \circ .. \circ F_0)^{-1}(R_i)$ in $\mathbb{R}^{n_0}$, and then taking the intersection of the resulting polyhedra, which we formalize in Definition 2.4.5.

The original definition of the canonical polyhedral complex $\mathcal{C}(F)$ uses the notion of a "level set complex," defined by [12]. We streamline the definition, working more directly in two ways.

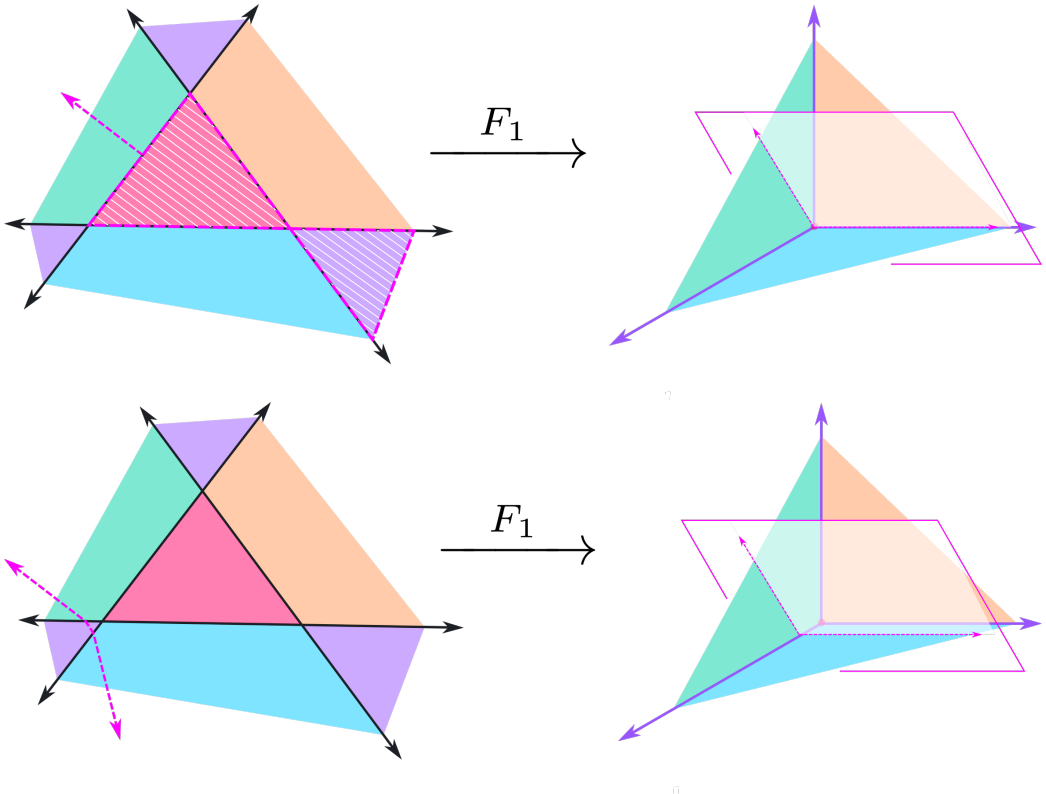


FIGURE 10. Top: A bent hyperplane with nongeneric behavior (some cells are codimension 0). Bottom: A more general codimension-1 bent hyperplane.

Definition 2.4.5 (Canonical Polyhedral Complex $\mathcal{C}(F)$, cf. [10], Definition 6.7)**.** Let $F : \mathbb{R}^{n_0} \to \mathbb{R}$ be a ReLU neural network with m layers and let $F_{(k)}$ and $F^{(k)}$ be as in 2.1.3. Define $\mathcal{C}(F)$ as follows:

1. (Forward Construction) Define $\mathcal{C}(F_{(1)})$ by $R^{(1)}$ (Definition 2.2.7). Then let $\mathcal{C}(F_{(k)})$ be defined in terms of $\mathcal{C}(F_{(k-1)})$ as the polyhedral complex consisting of the following cells:

$$\mathcal{C}(F_{(k)}) = \left\{ C \cap F_{(k-1)}^{-1}(R) : C \in \mathcal{C}(F_{(k-1)}), R \in R^{(k)} \right\}$$

   Then $\mathcal{C}(F)$ is given by $\mathcal{C}(F_{(m)})$.

2. (Backwards Construction) Define $\mathcal{C}(F^{(m)})$ by $R^{(m)}$. Then $\mathcal{C}(F^{(k-1)})$ can be defined from $\mathcal{C}(F^{(k)})$ as the polyhedral complex consisting of the following cells:

$$\mathcal{C}(F^{(k-1)}) = \left\{ R \cap F_{k-1}^{-1}(C) : R \in R^{(k-1)}, C \in \mathcal{C}(F^{(k)}) \right\}$$

   Then $\mathcal{C}(F)$ is given by $\mathcal{C}(F^{(1)})$.

Remark 2.4.6. While this defines $\mathcal{C}(F)$ mathematically, this does not describe a tractable algorithm for its computation, which is forthcoming in Section 4.1.

It is established by [10] and [12] that each intermediate complex $\mathcal{C}(F_k \circ ... \circ F_1)$ is a polyhedral complex which subdivides the previous one, and resultingly F is affine linear on each cell of $\mathcal{C}(F)$. However, this is also immediately apparent by considering $\mathcal{C}(F)$ as representing a subdivision morphism corresponding to a composite in EPoCx. So, the work necessary to define EPoCx pays off here:

Lemma 2.4.7. *The forward and backwards definitions of $\mathcal{C}(F)$ are equivalent.*

Proof. Each of the F_i be identified with the morphism in EPoCx consisting of subdividing $\mathbb{R}^{n_i}$ into $R^{(i)}$ followed by the layer map F_i as a function. The first definition inductively expresses the subdivision of $\mathbb{R}^{n_0}$ given by the the morphism $F_k \circ (F_{k-1} \circ ... \circ F_1)$:

$$\mathcal{C}(F_{(k)}) = \left\{ C \cap F_{(k-1)}^{-1}(R) : C \in \mathcal{C}(F_{(k-1)}), R \in R^{(k)} \right\}$$

The second definition inductively expresses the subdivision of $\mathbb{R}^{n_{k-1}}$ given by $(F_m \circ ... \circ F_{k+1}) \circ F_k$:

$$\mathcal{C}(F^{(k)}) = \left\{ R \cap F_k^{-1}(C) : R \in R^{(k)}, C \in \mathcal{C}(F^{(k+1)}) \right\}$$

Composition of morphisms in EPoCx is associative, so the resulting expressions are equal. □

CHAPTER III

ACCESSING THE TOPOLOGY OF NEURAL NETWORK FUNCTIONS

3.1. Piecewise Linear Transversality

The notion of transversality on cells will be critical for the next section. Function transversality is discussed in textbooks such as [13].

Definition 3.1.1 ([10], Definition 4.5)**.** Let X be a polyhedral complex of dimension d in $\mathbb{R}^n$, let $f : |X| \to \mathbb{R}^r$ be a map which is smooth on all cells of X and let Z be a smoothly embedded submanifold (without boundary) of $\mathbb{R}^r$. We say f is **transverse on cells of** X to Z and write $f \pitchfork_X Z$ if the restriction of f to the interior C° of every k-cell C of X is transverse to Z when $0 \leq k \leq d$.

Remark 3.1.2. By convention the interior of each 0-cell is nonempty. Otherwise, we would need to add the condition that the restriction of f to every 0-cell of X is transverse to Z.

Lastly, we will need the following notions of *generic* regarding hyperplane arrangements and neural networks, respectively.

Definition 3.1.3 ([10] Definitions 2.7, 2.9)**.** A hyperplane arrangement in $\mathbb{R}^n$ is called **generic** if each all sets of k hyperplanes intersect in an affine space of dimension $n-k$. A neural network is called **generic** if all of its affine maps A_i have generic corresponding hyperplane arrangements, $R^{(i)}$.

In [10] it is also established that the union of bent hyperplanes of $\mathcal{C}(F)$ form the $(n_0 - 1)$-faces of $\mathcal{C}(F)$ with probability 1. In the next section we expand on this characterization for lower-dimensional subcomplexes.

3.2. Supertransversal Neural Networks

Our results apply to certain subsets of ReLU neural networks, called *generic* (Definition 3.1.3) and *supertransversal* (Definition 3.2.1), additional technical conditions which are nonrestrictive in light of the lemma that almost all networks are supertransversal. That almost all networks are, additionally, *generic*, was established by [10]. As a result, in all but a measure-zero subset of neural networks, the theory developed below will hold.

Definition 3.2.1. Let F be a ReLU neural network of depth m. Let $\mathbf{F}^{(i)} : \mathbb{R}^{n_i} \to \mathbb{R}$ be the neural network defined by the last $m - i$ layers of F as in definition 2.1.3. Suppose, for all $1 \leq i \leq n$, F_i is transverse on cells of $R^{(i-1)}$ to the interior of all cells of $\mathcal{C}(F^{(i)})$. Then we call F a **supertransversal** neural network.

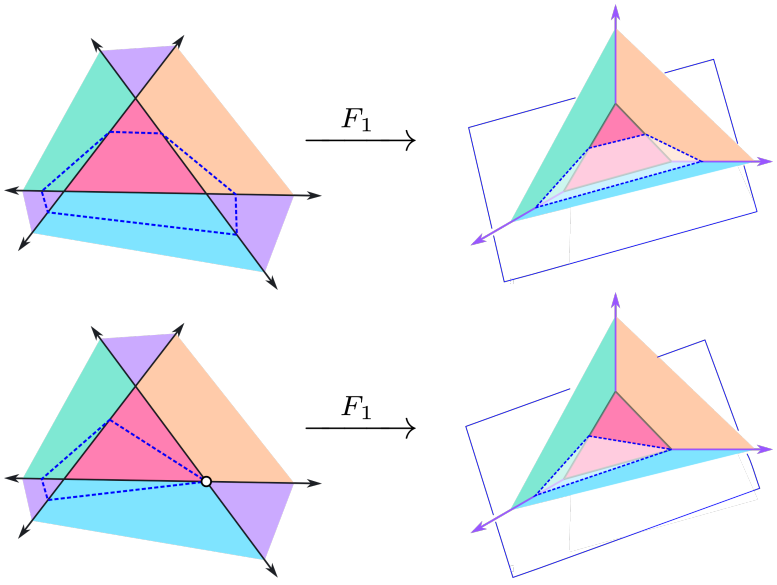


FIGURE 11. (Top) A supertransversal neural network. (Bottom) A non-supertransversal neural network.

Supertransversality ensures a analog of hyperplane arrangement genericity in the bent hyperplane arrangement associated with a neural network. We will show

that supertransversality guarantees that codimensionality is preserved in each cell under preimages. As seen in Figure 11, in a nonsupertransversal neural network, three bent hyperplanes in $\mathbb{R}^2$ may intersect in a 0-cell (as highlighted), whereas generically we would hope that they do not intersect. However, it is clear from examining the image of F_1 in the second layer that by a slight perturbation of the hyperplane associated with F_2 will lead to another supertransversal neural network. Indeed, even though the condition of network supertransversality is stronger than the notion of network transversality in [10] (Definition 8.2), it still holds on a full-measure subset of neural network parameter space, as we show here.

Lemma 3.2.2. *Supertransversality is full measure in $\mathbb{R}^P$, where P is the set of network parameters.*

Proof. First, a single-layer neural network $F^{(m)} : \mathbb{R}^{n_m} \to \mathbb{R}$ is trivially supertransversal; $\mathbb{R}$ has one cell which is already full dimension.

Next suppose that $F^{(k)}$ is supertransversal, and let $F_{k-1} : \mathbb{R}^{n_{k-1}} \to \mathbb{R}^{n_k}$ be a network layer. Suppose it is the case that F_{k-1} is nontransverse on some cell R of $R^{(i-1)}$ to some cell C of $\mathcal{C}(\mathbf{F}^{(k)})$. If so, it must be the case that $F_{k-1}(R) \cap C$ is nonempty and $T(F_{k-1}(R)) \oplus T(C) \neq \mathbb{R}^{n_k}$. Call $T(F_{k-1}(R)) \oplus T(C)$ by $T_{R,C}$. If $T_{R,C} \neq \mathbb{R}^{n_{k-1}}$, then it is instead a vector subspace of less than full rank, and an affine translation of $\mathcal{C}(F^{(k)})$ by any vector in $\mathbb{R}^{n_k} - T_{R,C}$ will ensure $F_{k-1}(R) \cap C$ is subsequently empty, since F is affine on R and C contained in an affine subspace of $\mathbb{R}^{n_k}$.

Let $\delta_{R,C}$ be the minimum distance between pairs of points in $F_{k-1}(R)$ and C. Since these cells are closed (though not necessarily compact), this is well defined, and furthermore if $R \cap C = \emptyset$, then $\delta_{R,C} > 0$. Let $\delta = \{\min \delta_{R,C} : F_{k-1}(R) \cap C = \emptyset\}$. Then $\delta > 0$.

Since there are finitely many cells R and C, the set

$$\mathbb{R}^{n_{k-1}} - \bigcup_{R,C} T_{R,C}$$

is generic in $\mathbb{R}^{n_{k-1}}$, where the union is taken over only those cells where $T_{R,C}$ is not full rank. An affine translation of $\mathcal{C}(F^{(k)})$ by any vector in this set with magnitude greater than 0 but less than δ yields a supertransversal network.

Since this can be performed regardless of the weights and biases of F_k, and an affine translation of the input space of $F^{(k)}$ does not change its supertransversality properties, the network $F_{k-1} \circ F^{(k)}$ is supertransversal on a full-measure subset of parameter space, which completes our inductive step. □

In order to establish additional properties regarding the existence of cells satisfying certain relations in supertransversal networks, we will rely on the following lemma.

Lemma 3.2.3. *Let $f : M \to \mathbb{R}^n$ be a PL map affine on cells of an embedded polyhedral complex $M \subset \mathbb{R}^m$. Let N be a polyhedral complex embedded in $\mathbb{R}^n$. Suppose f is transverse on cells of M to the interior of all cells of N.*

If $C \leq C'$ is a face relation in M, $D \leq D'$ is a face relation in N, and $f(C^\circ) \cap D$ is nonempty, then $f(C'^\circ) \cap D'^\circ$ is nonempty.

Proof. First we show that $f(C^\circ) \cap D'^\circ$ is nonempty. If $D = D'$ then we are done. Otherwise, consider $f(C)$, which is the image of the polyhedron C under an affine map. If the affine span A of $f(C)$ does not intersect the interior of D', then $A \cap D'$ is contained in a proper face E' of D'. In this case, $T(f(C)) \oplus T(E') \neq \mathbb{R}^n$, and f is not transverse on C to E'. Therefore, $A \cap D'^\circ$ is nonempty.

Within A we have $\partial(A \cap D') \subseteq A \cap \partial(D')$. As a result letting $x \in C^\circ$ and $f(x) \in D'$, every open neighborhood of $f(x)$ in A must have nontrivial intersection with the interior of D'. Take an open neighborhood N of x in C°. We note that $f : C \to A$ is a submersion (locally a surjective linear map). Thus $f(N)$ is open in A, and $N \cap D'^\circ$ must be nonempty, so $C^\circ \cap f^{-1}(D'^\circ)$ is nonempty.

To see that $D'^\circ \cap f(C'^\circ)$ is nonempty, take $x \in C^\circ$ with $f(x) \in D'^\circ$. If N is a neighborhood of $f(x)$ in D'° then $f^{-1}(N)$ must be an open neighborhood of x in M, containing $x \in C$. As $C \subset \partial C'$, $f^{-1}(N) \cap C'$ is nonempty, so $f(C'^\circ) \cap D'$ is nonempty. □

3.3. Combinatorially Characterizing $\mathcal{C}(F)$ with Sign Sequences

For networks with N neurons, binary strings of length N (which we will denote using -1 and 1) may serve as a labeling scheme to describe which neurons are active at a point in a ReLU network's input space [18] on the interior of cells of $\mathcal{C}(F)$. To encode all face relationships, we use *sign sequences*, which include 0 as a possible sign.

An algorithm for the computation of the sign sequence complex appears in section 4.1.

Sign Sequences

The combinatorial characterization of $\mathcal{C}(F)$ is through the following combinatorial construction called *sign sequences*. The primary use of these sign sequences is to track face relations, but first we show that sign sequences are sufficent to list the cells of $\mathcal{C}(F)$. Though this is proven in various forms elsewhere [19], we provide a different proof using differential topological methods.

Definition 3.3.2. Define $s : \mathcal{C}(F) \to \{-1, 0, 1\}^N$ by $s_{ij}(C) = \text{sgn}(F_{ij}(C))$. We call $s(C)$ the **sign sequence** of the cell C.

This construction is used in the theory of oriented matroids and hyperplane arrangements, cf. [1], and in particular the construction may be used to identify polyhedra in an affine hyperplane arrangement by denoting which halfspaces and hyperplanes were intersected to form that region. However, many of the properties fail to hold for arbitrary PL manifold arrangements. We must show that the construction still provides a combinatorial description of the polyhedra of the network:

Theorem 3.3.3. *The function s is well-defined and injective on cells of $\mathcal{C}(F)$.*

Proof. To see that s is well-defined, suppose $x_1, x_2 \in \mathbb{R}^{n_0}$ are such that $\text{sgn}(F_{ij}(x_1)) \neq \text{sgn}(F_{ij}(x_2))$ for some i, j. We wish to show that x_1 and x_2 are not in the same cell of $\mathcal{C}(F)$. However, we see that the images $F_{i-1} \circ ... \circ F_1(x_1)$ and $F_{i-1} \circ ... \circ F_1(x_2)$ cannot be in the same cell of of $R^{(i)}$ (the induced polyhedral decomposition of $\mathbb{R}^{n_{i-1}}$ by $\mathbf{A}_i$), because they differ in their location relative to the jth hyperplane. Thus x_1 and x_2 are in different cells of $\mathcal{C}(F_{(i)})$. As $\mathcal{C}(F)$ is a further polyhedral subdivision of $\mathcal{C}(F_{(i)})$, x_1 and x_2 are in different cells of $\mathcal{C}(F)$. So, s is well defined.

Next, suppose C_0 and C_1 are cells such that $s(C_0) = s(C_1)$. Let $x_0 \in C_0$ and $x_1 \in C_1$. We wish to show $C_0 = C_1$. We proceed by induction on layers in the forward direction.

We show first that, as a base case for induction, x_0 and x_1 are in the same cell in $\mathcal{C}(F_1)$.

Indeed since $s(C_0) = s(C_1)$, x_0 and x_1 are contained in the same cell of $R^{(1)}$, following the corresponding fact for hyperplane arrangements, this immediately means that x_0 and x_1 are in the same cell of $\mathcal{C}(F_1)$.

Now suppose as an inductive hypothesis that x_0 and x_1 are in the same cell of $\mathcal{C}(F_{(k)})$. Call $y_0 = F_{(k)}(x_0)$ and $y_1 = F_{(k)}(x_1)$. Because $\text{sgn}(F_{(k+1)j}(x_1)) = \text{sgn}(F_{(k+1)j}(x_2))$ for all $0 \leq j \leq n_k$, this implies that y_0 and y_1 are in the same intersection of halfspaces and hyperplanes in the co-oriented hyperplane arrangement $\mathbf{A}_{k+1}$, that is, the same cell of $R^{(k+1)}$. Therefore, as x_1 and x_2 are in the same cell C of $\mathcal{C}(F_{(k)})$ and their image is in the same cell C' of $R^{(k+1)}$ we conclude x_0 and x_1 are in the same cell in $\mathcal{C}(F_{(k+1)})$ given by

$$C \cap (F_{(k+1)})^{-1}(C')$$

That this is a unique polyhedral cell in $\mathcal{C}(F_{(k+1)})$, follows the work in [12], Lemma 2.5. By induction, as F is composed of finitely many layers, x_0 and x_1 are in the same cell of $\mathcal{C}(F)$. □

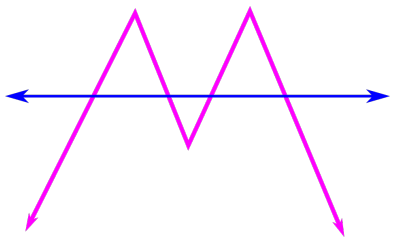
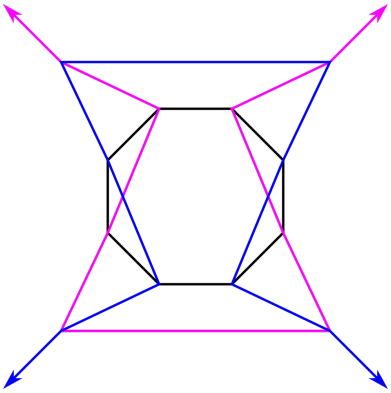


FIGURE 12. A simple manifold arrangement which is not sign codable because there are two PL submanifolds, but more than four regions.

The injectivity of s is special to constructions arising from hyperplane arrangements, and not general manifold arrangements. Indeed, a set of codimension-1 PL submanifolds may easily subdivide $\mathbb{R}^n$ in a way which this injectivity fails; see Figure 12.

FIGURE 13. An example of a manifold arrangement satisfying strong hypotheses which is still not sign codable.



Definition 3.3.4. Let $M_1, ..., M_m \subseteq \mathbb{R}^n$ be codimension-1 co-oriented manifolds embedded in $\mathbb{R}^n$, with M_i^+ and M_i^-in $\mathbb{R}^n$ corresponding to the two connected components in $\mathbb{R}^n \backslash M$ and M_i^0 corresponding to M itself. We then call the manifold arrangement **sign codable** if the set $M_1^{s_1} \cap ... \cap M_m^{s_m}$ is connected for all sign sequences $\{s_i\}_{i=1}^m \in \{-, 0, +\}^m$

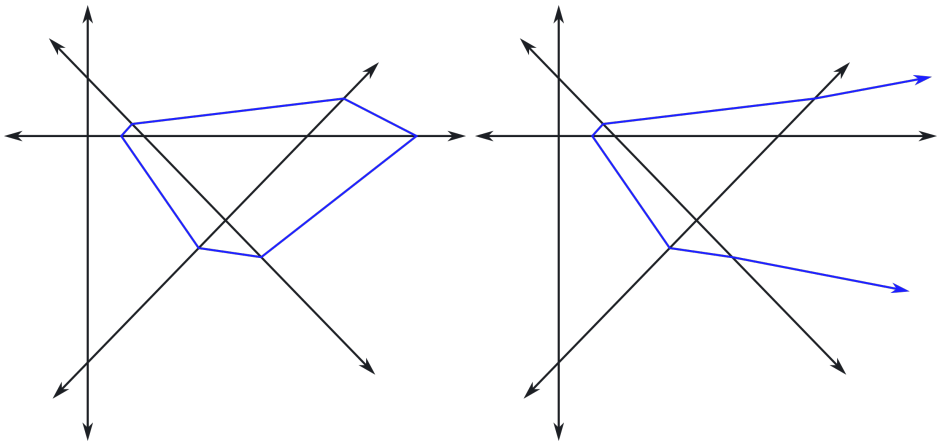
Even if M_i are codimension-1 connected co-orientable PL manifolds embedded in $\mathbb{R}^n$ whose intersection subdivides $\mathbb{R}^n$ into polyhedral regions, and the resulting polyhedral subdivision is dual to a cubical complex, it is possible that the manifold arrangmeent is not sign-codable. As seen in Figure 13, labeling each region by its location relative to the co-orientation of those manifolds fails to be injective. The pictured example depicts 3 PL submanifolds in $\mathbb{R}^2$ whose embedding has the aforementioned properties, but there are too many cells (14 vertices, 30 edges, and 17 2-gons) to be labeled by the 27 possible labelings in $\{-1, 0, 1\}^3$.

Later, we will show that not only is $\mathcal{C}(F)$ sign codable, but its full intersection poset can be reconstructed from vertices and sign sequences. It is more traditional to work "top down," considering the top dimensional polyhedra and their faces. But unlike theories such as hyperplane arrangements and oriented matroids, there

is no guarantee that knowing the sign sequences of the top-dimensional regions allows one to deduce the sign sequences of the zero-dimensional regions (circuit-cocircuit duality does not hold). The following example is an explicit illustration of this fact.

Theorem 3.3.5. *There exists a pair of networks F_1 and F_2 such that the set of strings encoding the activation patterns in the interiors of the cells of $\mathcal{C}(F_1)$ and $\mathcal{C}(F_2)$ are equal, but the polyhedral complexes $\mathcal{C}(F_1)$ and $\mathcal{C}(F_2)$ are not combinatorially equivalent.*

FIGURE 14. Two neural networks with the same sign sequences of the top dimensional regions but different combinatorics.



Proof. We provide an example in Figure 14. Two neural networks $F_1 : \mathbb{R}^2 \to \mathbb{R}^4 \to \mathbb{R}$ and $F_2 : \mathbb{R}^2 \to \mathbb{R}^4 \to \mathbb{R}$ have the pictured canonical polyhedral complexes (not to scale). One has a bounded decision boundary, whereas the other is unbounded. However, both have identical sign sequences of their top-dimensional regions, given by:

(-1, 1, -1, 1, 1), (-1, 1, 1, -1, 1), (1, -1, -1, -1, 1),
(-1, 1, -1, 1, -1), (-1, -1, 1, -1, 1) (-1, 1, 1, 1, -1),
(1, -1, -1, 1, -1), (1, -1, 1, -1, 1), (-1, -1, 1, 1, -1),
(-1, 1, 1, 1, 1), (-1, -1, -1, 1, -1), (1, -1, -1, 1, 1)
(-1, -1, 1, 1, 1), (1, -1, 1, 1, -1), (1, -1, 1, 1, 1),
(1, 1, -1, 1, -1), (1, 1, -1, 1, 1)

That these canonical polyhedral complexes have the same set of sign sequences of their top-dimensional regions is more easily seen by looking at the differences between the two pictures, which only depend on the blue "decision boundary." Each region which is subdivided into two regions by the blue bent hyperplane in the left image is also subdivided by the blue bent hyperplane in the right image.

The two canonical polyhedral complexes pictured have differing combinatorics and differing topology of their decision boundaries, but the set of sign sequences of the top dimensional regions is equal (see Theorem 3.3.5). Explicit weights and biases for this construction are available in the supplementary materials, and additionally recorded explicitly in Appendix A. □

Dimension of Cells from Sign Sequences

As shown in [10], it is not always the case that the preimages $F_{ij}^{-1}(0)$ are $(n_0 - 1)$-dimensional. In order to establish dimension of cells in general, supertransversality and genericity are key. In fact, the sign sequence of a cell is determinative under these conditions, as it encodes the dimension of the cell.

Lemma 3.3.7. *Let F be generic and supertransversal. Let C be a k-cell of $\mathcal{C}(F)$. Then $s(C)$ has exactly $n_0 - k$ entries which are zero. That is, C is contained in the intersection of $n_0 - k$ bent hyperplanes.*

Proof. This is certainly true for any neural network of the form $G \circ F_m$ satisfying the condition that F_m is generic as a layer map, as the bent hyperplane arrangement is equal to the hyperplane arrangement, which is a generic hyperplane arrangement (Definition 3.1.3).

We proceed via induction, using the backwards construction of $\mathcal{C}(F)$ (Definition 2.4.5).

Suppose by way of induction $F^{(i)} = G \circ F_m \circ ... \circ F_i$, and that $\mathcal{C}(F^{(i)})$ satisfies the condition that $C \in \mathcal{C}(F^{(i)})$ is a k-cell if and only if C is contained in exactly $n_i - k$ bent hyperplanes.

Now, suppose F_{i-1} is transverse on the cells of $R^{(k-1)}$ to C for all C in $\mathcal{C}(F^{(i)})$. Consider $F^{(i-1)} = G \circ F_m \circ ... \circ F_i \circ F_{i-1}$.

Let C' be a cell in $\mathcal{C}(F^{(i-1)})$. By definition, C' is given by $F_1^{-1}(C) \cap D$ for some $C \in \mathcal{C}(F^{(i)})$ and some minimal (by inclusion) cell D of $R^{(i-1)}$. In particular, we may assume C' is not contained in any proper face of D. If D has codimension ℓ, then D is in the intersection of exactly ℓ hyperplanes in $R^{(i-1)}$ by the genericity of F. Because F_1 is transverse on cells of $R^{(i-1)}$ to C, $\text{codim}(C')$ in the interior of D is equal to $\text{codim}(C) = k$, with total codimension $k + \ell$.

As C is contained in the intersection of exactly k bent hyperplanes in $\mathcal{C}(F^{(i)})$, C' is contained in the intersection of the preimage of precisely those same k bent hyperplanes in $\mathcal{C}(\mathbf{F}^{(i-1)})$. Furthermore C' is contained in the intersection of the ℓ hyperplanes in $R^{(i-1)}$ which intersect to form D, and no additional hyperplanes as C' is not contained in any proper face of D. Thus C' is contained in the

intersection of precisely $k + \ell$ bent hyperplanes in $\mathcal{C}(\mathbf{F}^{(i-1)})$ and has codimension $k + \ell$. The number of zeros in $s(C')$ must be equal to the number of bent hyperplanes it is contained in, by definition. This completes the inductive step. $\square$

3.4. Algebra of Sign Sequences

Now we can define a key algebraic structure which will lead to the ability to deduce the structure of $\mathcal{C}(F)$ in general, via a particular algebraic structure which allows us to generate all sign sequences of cells from the sign sequences of the vertices. The following holds for all supertransversal networks (and does not rely on the layer maps being generic). It is a result of the existence of a multiplicative "composition" derived from oriented matroid theory [3].

Lemma 3.4.1. *Let F be a supertransversal neural network.*

If C and D are two cells of $\mathcal{C}(F)$, the product $S(C) \cdot S(D)$ given by:

$$(S(C) \cdot S(D))_{ij} = \begin{cases} S(C)_{ij} & \text{if } S(C)_{ij} \neq 0 \\ S(D)_{ij} & \text{otherwise} \end{cases}$$

is well-defined as a product between sign sequences. That is, there exists a cell E in $\mathcal{C}(F)$ such that $S(C) \cdot S(D) = S(E)$ for all such cells C and D.

Furthermore, $C \leq E$, that is, C is a face of E or equal to E.

Thus, sign sequences of a supertransversal network form a semigroup.

Proof. First, this is true for any single-layer network $F : \mathbb{R}^{n_m} \to \mathbb{R}$, since it is true for hyperplane arrangements; see [1], Section 1.4 for a treatment.

Now, suppose these properties hold for any supertransversal k-layer neural network and inductively, using the backwards construction of $\mathcal{C}(F)$, let F be a $k+1$-layer supertransversal neural network. Then $F^{(2)} = G \circ F_{k+1} \circ ... \circ F_2$ is a k-layer

supertransversal network and our inductive hypothesis holds for $\mathcal{C}(F^{(2)})$. We will denote the sign sequences of cells with respect to $\mathcal{C}(F^{(2)})$ by $S^{(2)}(C)$.

Let C and D be cells of $\mathcal{C}(F) = \mathcal{C}(F^{(2)} \circ F_1)$. Then $C = R_1 \cap F_1^{-1}(C')$ and $D = R_2 \cap F_1^{-1}(D')$, for cells R_1, R_2 in $R^{(1)}$ and cells $C', D' \in \mathcal{C}(F^{(2)})$, by the definition of $\mathcal{C}(F)$. Now, by inductive hypothesis $S^{(2)}(C') \cdot S^{(2)}(D') = S^{(2)}(E')$ for some cell E' in $\mathcal{C}(F^{(2)})$, and C' is a face of E'.

Denote the sign sequences with respect to $R^{(1)}$ as S_1. Since $R^{(1)}$ is a polyhedral complex induced by an affine hyperplane arrangement, $S_1(R_1) \cdot S_1(R_2) = S_1(R_3)$ for R_3 a region in $R^{(1)}$, and R_1 is a face of R_3 or equal to R_3.

Let $E = R_3 \cap F_1^{-1}(E')$. We wish to show that $S(C) \cdot S(D) = S(E)$, and that C is a face of E or equal to it. Now, $S(C)$ is obtained by $S_1(R_1)$ concatenated with $S^{(2)}(C')$, and likewise for the other cells. Since $S_1(R_1) \cdot S_1(R_2) = S_1(R_3)$ by the corresponding hyperplane arrangement and $S^{(2)}(C') \cdot S^{(2)}(D') = S(E')$ by inductive hypothesis, by concatenation this gives $S(C) \cdot S(D) = S(E)$.

To see that E is nonempty we must note that since F is supertransversal, F_1 is transverse on R_1 to C', and apply Lemma 3.2.3 to obtain that $R_3^\circ \cap F_1^{-1}(E')^\circ$ is nonempty.

Lastly, we recall from Lemma 2.2.6 that $C' \leq E'$ and $R_1 \leq R_3$ implies that $(F_1^{-1}(C') \cap R_1) \leq (F_1^{-1}(E') \cap R_3)$, that is, $C \leq E$. □

The following properties of the product defined above continue to follow from similar constructions in hyperplane arrangements [1].

Lemma 3.4.2. *For all supertransversal networks, the following relations hold for all C and D in $\mathcal{C}(F)$, where the relation $\leq$ denotes "is a face of":*

1. *$C \leq D$ if and only if $S(C) \cdot S(D) = S(D)$*

2. *$S(C) \cdot S(D) = S(D) \cdot S(C)$ if and only if there is a cell E with $D \leq E$ and $C \leq E$.*

3. *$S(C) \cdot S(D) = S(C)$ if and only if all bent hyperplanes which contain D also contain C.*

Proof.

1. We have already shown if $S(C) \cdot S(D) = S(D)$ then $C \leq D$. If $S(C) \cdot S(D) \neq S(D)$ then there is some index where $s_{ij}(C) = \pm 1$ and $s_{ij}(D) = -s_{ij}(C)$. If so, then C and D are sent to opposite sides of some hyperplane in some layer; this cannot occur if $C \leq D$.

2. Immediate from the previous statement and Lemma 3.4.1.

3. $S(C) \cdot S(D) = S(C)$ if and only if for all node maps for which $F_{ij}(C) = 0$, we also have $F_{ij}(D) = 0$. But this is true if and only if all bent hyperplanes which contain C also contain D.

□

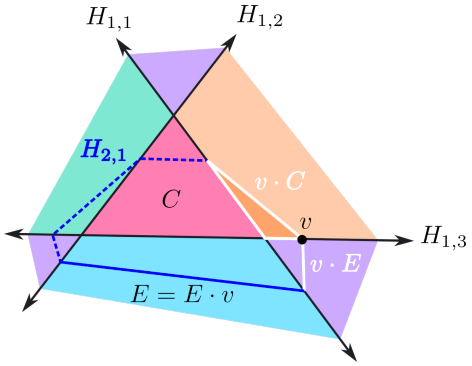


FIGURE 15. "Multiplication" of polyhedra on the sign sequence cubical complex, pictured geometrically.

TABLE 1. The sign sequence of the cells in Figure 15, together with some computed products.

Cell	Sign Sequence
v	(1,1, 0, 0)
E	(1,1,-1, 0)
C	(1,1, 1,-1)
$v \cdot C$	(1,1, 1,-1)
$v \cdot E$	(1,1,-1, 0)

These characterizations are primarily useful for using code to track face relations via a discrete structure. As implemented in Section 3.6, we can use these to compute the topological properties of the decision boundary of a network using sign sequences. These properties can be seen illustrated in Figure 15. If the three hyperplanes $H_{1,1}, H_{1,2}$ and $H_{1,3}$ are co-oriented towards the cell C, and $H_{2,1}$ is co-oriented away from the cell C, the pictured cells have the sign sequences indicted in Table 1. The product is computed and pictured for certain pairs of cells.

3.5. The Duality Between $\mathcal{C}(F)$ and $\mathcal{S}(F)$

We now assemble the ideas from the previous sections to present a duality between the canonical polyhedral complex of a generic, supertransversal neural network and a pure cubical complex, providing new structure to the combinatorics of the canonical polyhedral complex. This duality sends k-cells in $\mathcal{C}(F)$ to $(n_0 - k)$-cells of the cube. We will call this subcomplex $\mathcal{S}(F)$.

This shows that for a given neural network the properties of the sign sequence complex make it straightforward to compute the combinatorial structure of the polyhedral complex of a network across all dimensions upon obtaining the sign sequences of the vertices of $\mathcal{C}(F)$.

Definition 3.5.1. Here, we use the term *pure* to refer to a polyhedral complex where every face is contained in some other polyhedron of uniform top dimension. In this case every maximal cell of $\mathcal{S}(F)$ is an (n_0)-cell.

In particular, vertices of $\mathcal{C}(F)$ correspond to n_0-cells of $\mathcal{S}(F)$. Since the complex $\mathcal{S}(F)$ is pure and n_0-dimensional, knowing which n_0-cells are present is sufficient to determine all face relations in the subcomplex of the hypercube. An important corollary is:

Corollary. For generic, supertransversal neural networks, the sign sequences of the vertices of $\mathcal{C}(F)$ determine the face poset of the polyhedral complex $\mathcal{C}(F)$.

For a visualization, see Figure 16. On the left we depict $\mathcal{C}(F)$ for a specific neural network function $F : \mathbb{R}^2 \to \mathbb{R}^3 \to \mathbb{R}$. The three straight lines and the solid colored regions together form $R^{(1)}$, which is also $\mathcal{C}(F_1)$. In the middle we see that $F_1 : \mathbb{R}^2 \to \mathbb{R}^3$ is piecewise linear on cells of $\mathcal{C}(F_1)$. The hyperplane in $\mathbb{R}^3$ is the hyperplane associated with $A_2 : \mathbb{R}^3 \to \mathbb{R}$, and this hyperplane together with the two halfspaces on either side of it form $R^{(2)}$. The cells of $\mathcal{C}(F)$ on the left are mathematically determined by taking one region R of $R^{(2)}$, considering its preimage $F_1^{-1}(R)$, and taking the intersection of this preimage with a cell of $\mathcal{C}(F_1)$. On the right, the geometric dual sign sequence complex $\mathcal{S}(F)$ is superimposed in white over $\mathcal{C}(F)$, with one vertex for each region of $\mathcal{C}(F)$. As we prove in general, $\mathcal{S}(F)$ is cubical, with each two-cube (quadrilateral) containing a unique vertex of $\mathcal{C}(F)$.

Theorem 3.5.2. *For each generic, supertransversal neural network $F : \mathbb{R}^{n_0} \to \mathbb{R}$ with at least n_0 hidden units in the first layer, the image of the map $S : \mathcal{C}(F) \to \{-1, 0, 1\}^n$ uniquely defines a pure n_0-dimensional subcomplex of the hypercube $[-1, 1]^N$ endowed with the product CW structure. We call this subcomplex $\mathcal{S}(F)$.*

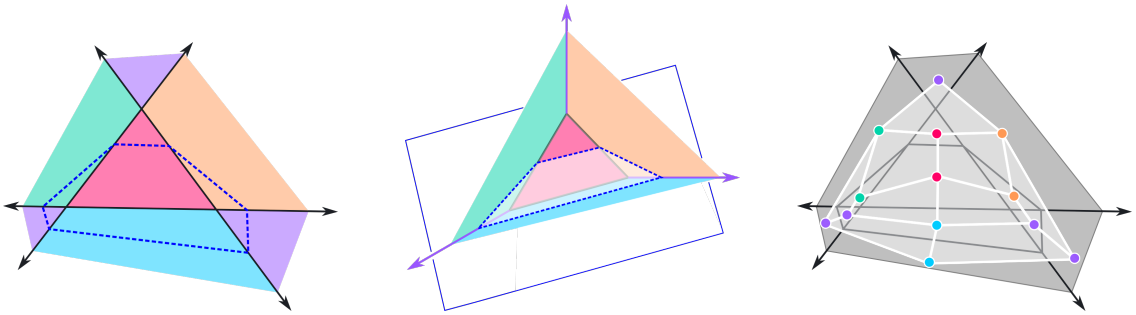


FIGURE 16. (Left) A canonical polyhedral complex and (Right) its geometric dual.

In the image, the vertices in $\mathcal{S}(F)$ correspond to n_0-cells in $\mathcal{C}(F)$, and in general the k-cells of $\mathcal{S}(F)$ correspond to codimension-k cells of $\mathcal{C}(F)$.

Proof. Recall that cubical faces of $[-1,1]^N$ (with its product CW structure) can be identified by sequences of $\{-1,0,1\}^N$.

First if F has at least n_0 hidden units in the first layer and it is generic, then $\mathcal{C}(F)$ contains vertices as some of its cells, since the intersection of n_0 hyperplanes in general position in $\mathbb{R}^{n_0}$ is a point. Since $\mathcal{C}(F)$ is a connected polyhedral complex, if any of its polyhedra have vertices, then all of them do (see [11], Corollary 5.29).

For any $C \in \mathcal{C}(F)$ there is a vertex $v \leq C$. There are n_0 coordinates where $S(v) = 0$ by Lemma 3.3.7. Furthermore $S(v) \cdot S(C) = S(C)$ by Lemma 3.4.1. Thus $S(C)$ is equal to $S(v)$ except those places where $S(v) = 0$. But this is equivalent to the condition that the n_0-cell $S(v) \in \mathcal{S}(F)$ has $S(C)$ on its boundary. So, every in the image of $S(F)$ is contained in an n_0-cube which is also in the image of $S(F)$. (There are no $n_0 + 1$-cubes in the image of $S(F)$ by Lemma 3.3.7.) Thus the image of $S(F)$ is "pure n_0-dimensional" in the sense that every cube in $S(F)$ is a face of an n_0-cube in $S(F)$.

Next we show that for a given n_0-cube in the image of $S(F)$, all its faces are in the image of $S(F)$. Our strategy is to show that there exists an edge corresponding to each possible sign sequence incident to the corresponding vertex.

Then we may apply the sign sequence multiplication in Lemma 3.4.1 to obtain all remaining faces. This is equivalent to establishing that a vertex v of $\mathcal{C}(F)$ has $2n_0$ neighboring edges, each of which are obtained by replacing a single 0 from $S(v)$ with 1 or -1. Of course, any vertex must be incident to at least n_0 edges since it belongs to a polyhedral complex with domain $\mathbb{R}^{n_0}$, so we show that if there exists an edge incident to v with $S_{ij}(E) = 1$ while $S_{ij}(v) = 0$, then there also exists an edge with $S_{ij}(E) = -1$ (and, by symmetry, vice versa).

Suppose that this is not the case for some v. Then without loss of generality there exists an earliest (i, j) node map satisfying that $F_{ij}(v) = 0$ but for all edges E neighboring v in $\mathcal{C}(F)$, $F_{ij}(E) \geq 0$, since for each edge E, $S(E)$ differs from $S(v)$ only in one location. Since the edge set of v is nonempty, this implies that F_{ij} cannot be affine on any affine subspace of $\mathbb{R}^{n_0}$ containing v unless $F_{ij} = 0$ on that subspace.

As v cannot be a vertex of $\mathcal{C}(F_{(i-1)})$, since it is contained in the intersection of fewer than n_0 bent hyperplanes before F_{ij}, it is contained in the interior of a larger cell C in $\mathcal{C}(F_{(i-1)})$. As a result, F_{ij} is affine on the interior of C. But by the previous paragraph, this means that $F_{ij}(C) = 0$, and thus $F_{(i)}$ is not transverse on C to a cell contained in $R^{(i)}$, and cannot be transverse on C to any polyhedral subdivision (including $\mathcal{C}(F^{(i)})$). This implies that there is a layer of $F_{(i-1)}$ which fails to be transverse on cells, which is a contradiction.

So, if v is a vertex of $\mathcal{C}(F)$, then for each node map F_{ij} such that $F_{ij}(v) = 0$, v has an incident edge with $F_{ij}(E) = 1$ and an incident edge with $F_{ij}(E) = -1$ by the same argument. We note by supertransversality that $S(E)$ must have $n_0 - 1$ entries which are zero. Also, since v is incident to E, by Lemma 3.4.2, $S(E)$ must have the same entries as $S(V)$ except possibly where $S(v) = 0$. This means $S(E) =$

$S(v)$ except for at the (i, j) coordinate, as required. Since this occurs at all node maps for which $F_{ij}(v) = 0$, we are done. □

Once the existing cells in $\mathcal{S}(F)$ have been located, we only need to establish an explicit duality. The majority of the work has already been done.

Lemma 3.5.3. *The face poset of $\mathcal{C}(F)$ is the opposite poset of the face poset of $\mathcal{S}(F)$, and the (mod-two) cellular boundary map of $\mathcal{S}(F)$ is dual to the (mod-two) cellular boundary map of $\mathcal{C}(F)$.*

Proof. In $[-1, 1]^N$, the cells consist of cubes which are uniquely defined by their center, at points given by sequences in $\{-1, 0, 1\}^N$. The dimension of each cube is given by the number of 0 entries in this sequence. The cellular boundary of this cube consists of cells one dimension lower, with a 1 or -1 replacing a 0 in the sign sequence, providing the (mod two) boundary in $\mathcal{S}(C)$. In $\mathcal{C}(F)$, if $s(C)$ is related to $s(D)$ by replacing one zero entry of C with a 1 or -1, by Lemmas 3.4.2 and 3.3.7 that this is equivalent to $C \leq D$ and $dim(C) + 1 = dim(D)$, which is equivalent to C being a face of D, and additionally being a term of the (mod two) cellular boundary of D. □

We next describe some possible applications of the sign sequences of $\mathcal{C}(F)$ before discussing how exactly to obtain the vertices in $\mathcal{C}(F)$ for a given neural network function.

3.6. Computing Decision Boundary Topology from $\mathcal{S}(F)$

The characterization of $\mathcal{C}(F)$ as dual to a cubical complex permits us to define a straightforward mod-two cellular boundary, which is the transpose of

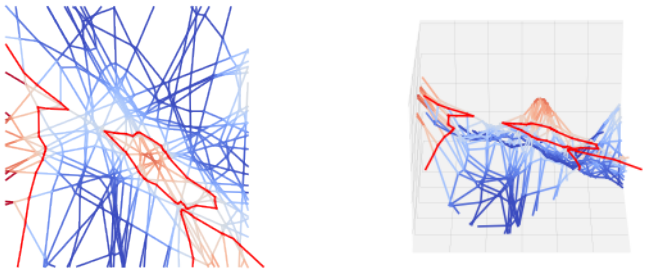


FIGURE 17. Left: The decision boundary of a randomly-initialized neural network $F : \mathbb{R}^2 \to \mathbb{R}$, in red, and the canonical polyhedral complex. Right: A graph of the function F.

the coboundary operation on the cubical complex, as seen in Lemma 3.5.3. This enables the computation of the Betti numbers of the decision boundary.

In practice, if C is a polyhedron in $\mathcal{C}(F)$, then replacing one location for which $S(C)_{ij} = 0$ with ± 1 gives the sign sequence of a polyhedron containing C, and doing this operation for all locations computes all terms in the cellular coboundary of C (ignoring orientation). We recover the decision boundary of a network as the subcomplex of cells C in $\mathcal{C}(F)$ whose faces D, including vertices, all satisfy $F(D) = 0$. By locating the vertices in $\mathcal{S}(F)$ which have a 0 as the last entry in their sign sequence, this coboundary operation, with image restricted to those cells whose last sign sequence entry is zero, gives a chain complex of cells of the decision boundary, for example as pictured in Figure 17.

In general, the presence of unbounded cells makes this map not quite correspond to a cellular chain complex. In particular, not every edge has two vertices. By adding a single 'vertex at infinity' to unbounded edges, the corresponding chain complex has a straightforward interpretation as the chain

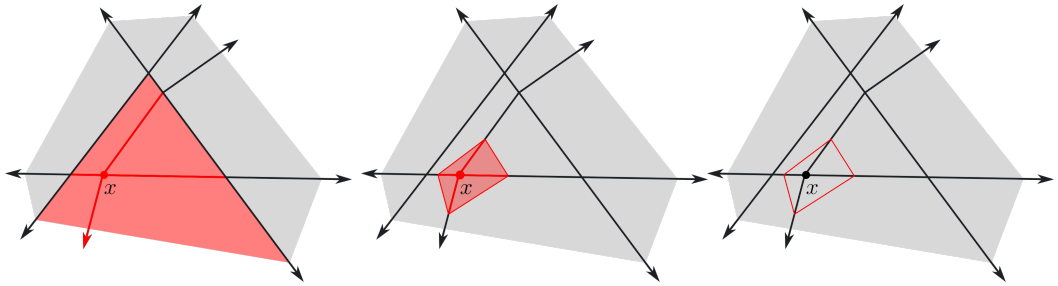


FIGURE 18. Left: The star of x in M. Middle: A cone neighborhood for x in M. Right: A link complex for x in M.

complex of the compactification of the decision boundary within the one-point compactification of $\mathbb{R}^n$.

Eplicit implementation of this algorithm is provided in the code repository provided in the supplementary material.

3.7. Local Combinatorics of Vertices for PL Morse Theory

The topology of sublevel and level sets for neural networks may also be approached through Piecewise Linear (PL) Morse theory. There is no one generally accepted approach to PL Morse theory. The thesis [12] provides a general framework which depends on the following local constructions at vertices; see Figure 18.

Definition 3.7.1 ([12], pgs. 15-16, 29)**.** Let M be a polyhedral complex, and let $x \in |M|^\circ$. The **star** of x in M is the union of all polyhedra which have a face in x. A **cone neighborhood** for x in M is a compact neighborhood of x in M which is equal to the cone on x with its boundary, L, called the **link** of P. A **link complex** of x in M is a polyhedral complex which may serve as the link of x in M.

It is always possible to find a link complex for x in M which is contained in its star.

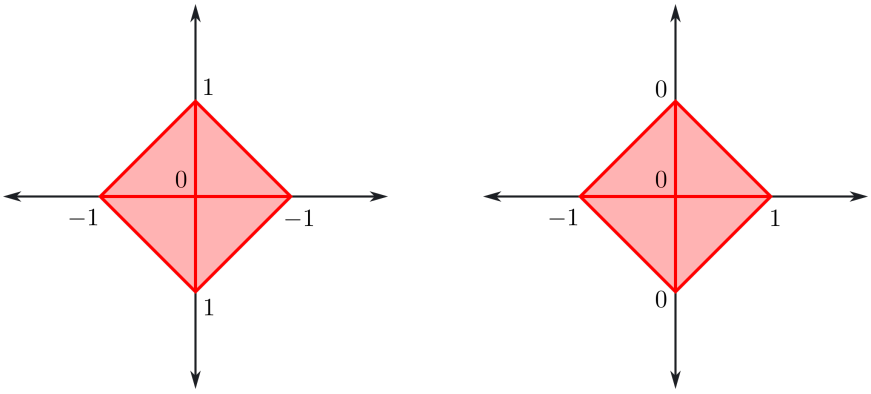


FIGURE 19. Left: An exemplar PL critical point of index 1 on the standard cross-polytope in $\mathbb{R}^2$. Right: The exemplar PL regular point.

Combining [12], Definition 3.1 with Thm. 3.11, we see that the following may be used as a definition for PL Morse critical points with index i (See Figure 19).

Definition 3.7.2. Let M be a combinatorial d-manifold and let $f : |M| \to \mathbb{R}$ be piecewise affine on cells. Let $x \in |M|$. Let $St(d)$ be the standard cross-polytope in $\mathbb{R}^d$ centered at the origin o and define $f_i : St(d) \to \mathbb{R}$ by

$$f_i(x_1, .., x_d) = \sum_{k=1}^{i} -|x_k| + \sum_{k=i+1}^{d} |x_k|$$

If there are combinatorially equivalent link complexes for x and o contained in the stars of x and o such that $f - f(x)$ and f_k have the same signs at corresponding vertices, then x is a critical point of f with index i.

The definition of PL Morse regular points is analogous, but with $f : St(d) \to \mathbb{R}$ defined as $f_i(x_1, ..., x_d) = x_1$. It is possible for individual points to not satisfy either definition, at which point they are degenerate critical points.

Under Def. 3.1, [12] establishes that sublevel sets of a PL Morse function behave as the sublevel sets of a smooth Morse function, in that the the homotopy type of $M_{\leq a}$ equals that of $M_{\leq b}$ so long as $[a, b]$ contains no critical points, and

there is an F-level preserving isotopy between them. Additionally, critical points may only occur at vertices of M or on cells for which F is constant. However, the theory in [12] is limited to compact polytopal complexes. In [11] we extend this theory to the noncompact setting of the canonical polyhedral complex, and show that flat cells (those cells on which F is constant) are the analog of critical points for ReLU neural networks. It is likely that the theory more generally applies in the category of Euclidean polyhedral complexes.

In [11], Proposition 6.6 we additionally jointly establish that for neural networks F with architecture $(n, m, 1)$ (that is, shallow neural networks with a single hiddden layer), the criticality of individual vertices of $\mathcal{C}(F)$ with respect to F can be determined by computing the gradient on the edges. However, no such statement was established for deeper neural networks because there was no guarantee regarding the combinatorial properties of neighborhoods of vertices. With the work of establishing the combinatorial properties of $\mathcal{C}(F)$ complete, we briefly extend these results here.

Theorem 3.7.3. *If F is a supertransversal, generic neural network with no flat cells except vertices, then a vertex v of $\mathcal{C}(F)$ is critical if and only if the grad-F orientation on edges incident to v corresponding to opposite signs in each coordinate where $s(v) = 0$ are either both pointing towards or pointing away from v for all pairs.*

Furthermore, the index of the critical vertex is given by the number of pairs of edges oriented towards v.

Proof. The sign sequence of a cell C incident to v can be identified with a region of the hyperplane arrangement induced by the coordinate axes in $\mathbb{R}^{n_0}$ by evaluating

$s_{ij}(C)$ for the (i,j) entries for which $s_{ij}(v) = 0$. This gives a combinatorial equivalence between the link complex of v and $St(n_0)$.

If the grad-F orientations are paired in opposite directions, the axes may be permuted so that the edges with relative grad-F orientations towards v correspond to the first i axes, giving a combinatorial equivalence with $St(n_0)$ with vertices having the same signs as f_i. Therefore v is critical, and i gives the index of v.

The other direction is more subtle. We exploit the fact that the described link complex $lk(v)$ has known connectivity and symmetry, given by the connectivity of $St(n_0)$. Select a link complex of v such that its vertices are in general position. Let E and $\tilde{E}$ be a pair of edges with grad-F orientations toward and away from v, respectively, and let w be the point on $lk(v) \cap E$, with $\tilde{w}$ giving the opposite point on $lk(v) \cap \tilde{E}$.

This allows us to reduce this question to the problem of checking that for any PL function on $St(n_0)$ with $g(1, 0...0) = 1$, $g(-1, 0, ...0) = -1$ and vertices in general position, there is a combinatorial equivalence of a subdivision of $St(n_0)$ with a different subdivision of $St(n_0)$ such that the standard PL regular f applied at the subdivision has the same signs.

We observe that because of general position $g^{-1}(0)$ intersects each *pair* of edges between w and $\tilde{w}$ in a unique point, and the reflection along the $x_1 = 0$ plane gives a combinatorially equivalent subdivision, for which each region between $g^{-1}(0)$ and the equator is combinatorially equivalent with itself via a second reflection. This ensures that the equatorial shift map $s : St(n_0) \to St(n_0)$ illustrated in Figure 20 actually gives a combinatorial equivalence between the subdivision induced by $g(0)$ and the standard cross-polytope, with equal signs to the standard regular $f : St(n_0) \to \mathbb{R}$, and v is a PL-regular point.

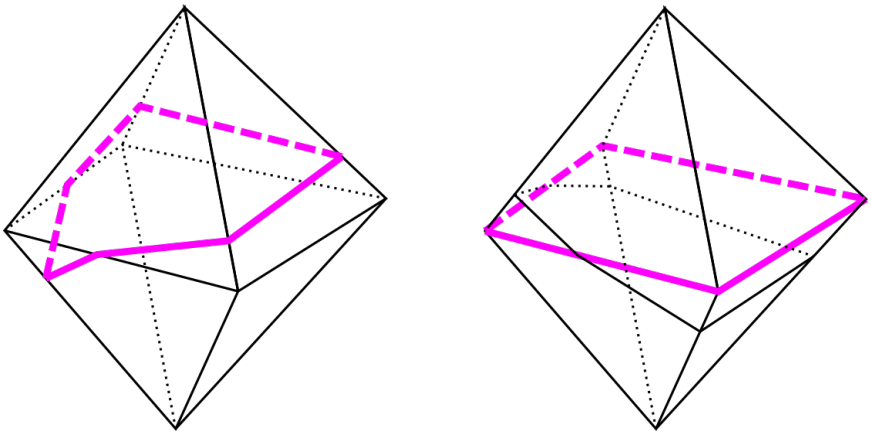


FIGURE 20. The equatorial shift map. Left: $g^{-1}(0)$ induces a subdivision on the boundary of $St(3)$ separating opposite points. Right: There is a combinatorial equivalence sending each cell in the subdivision on the left to the corresponding cell in the subdivision of the boundary of $St(3)$ given by the reflection of the first subdivision, shifting $g^{-1}(0)$ to the equator of $St(3)$.

□

In practice, this means if the vertices of $\mathcal{C}(F)$ have been computed, the critical vertices of $\mathcal{C}(F)$ which are not incident to unbounded edges can be identified explicitly by evaluating F on each vertex. The grad-F orientation of each edge is therefore away from v on the edge (v, w) if $F(w) > F(v)$, and towards v on that edge if $F(w) < F(v)$.

To evaluate whether a vertex incident to an unbounded edge is critical requires more work. Unbounded edges may be identified as those edges which are incident to only one vertex after applying the coboundary operation. Generically, each unbounded edge in $\mathcal{C}(F)$ has a sign sequence with exactly $n_0 - 1$ zero entries, and the vertex with exactly n_0 zero entries which it is incident to can be used to identify its direction as follows.

1. E is the intersection of $n_0 - 1$ bent hyperplanes, the last of which occurs in layer ℓ. This implies that E is next to a n_0-cell in layer $\ell - 1$ which F_ℓ does

not collapse, that is, on which $F_\ell(C)$ is nonzero. Identify the sign sequence of this cell in $C(F_{(\ell-1)})$. Call this sign sequence s.

2. Find the linear equations of H_i where i are the indices for which $s(E)$ is zero, relative to the sign sequence s. This determines a vector $\vec{v}$ in $\mathbb{R}^{n_0}$.

3. Let x be the unique vertex of $\mathcal{C}(F)$ incident to E. Its sign sequence differs from $s(E)$ in one (ij)-coordinate. Let A_{ij} be affine function which is equal to the node map F_{ij} when restricted to C. Consider $A_{ij}(x + \vec{v})$.

   (a) If $A_{ij}(x + \vec{v})$ is equal in sign to $s_{ij}(E)$ let $y = x + \vec{v}$.

   (b) Otherwise, let $y = x - \vec{v}$.

4. Evaluate $F(x)$ and $F(y)$. If $F(y) > F(x)$ then the grad-F orientation on E is away from x, otherwise the grad-F orientation on E is towards x.

Once grad-F orientations have been obtained, to determine whether v is a critical vertex, we apply Theorem 3.7.3. If all pairs of opposite edges to v point in opposite directions, then v is critical, with index given by the number of "downward-facing pairs," which are pairs of edges both with grad-F orientation pointing towards v.

CHAPTER IV

ALGORITHMS FOR COMPUTING $\mathcal{C}(F)$

4.1. Naïve Computation of $\mathcal{C}(F)$ by Looping through Regions

In order to make use of the algorithms outlined in Sections 3.6 and 3.7, it is necessary to obtain the canonical polyhedral complex of a neural network.

For a given neural network, sign sequences follow from locating potential vertices, thus knowing locations of the 0 entries in its sign sequence, and then evaluating the network to obtain its remaining signs.

We prove that the process of computing $\mathcal{C}(F)$ can be done iteratively through layers, beginning with the first layer. Letting $R^{(k)}$ be the polyhedral complex associated with the hyperplane arrangement in layer k, the complex $\mathcal{C}(F_k \circ ... \circ F_1)$ is given precisely by the intersection complex of $\mathcal{C}(F_{k-1} \circ ... \circ F_1)$ and $(F_{k-1} \circ ... \circ F_1)^{-1}(R^{(k)})$. (See Definition 2.4.5). To obtain the vertices of a particular network's canonical polyhedral complex, we may therefore begin by obtaining the vertices corresponding to $\mathcal{C}(F_1)$, its first layer's canonical polyhedral complex.

Lemma 4.1.1. *Let F be a supertransversal, generic neural network.*

The 0-cells of $\mathcal{C}(F_{(1)})$ are given by the solutions to

$$\{W_\alpha x = b_\alpha : \alpha \subset [n_1] \ \& \ |\alpha| = n_0\}$$

where W is the weight matrix of the network and α denotes a subset of the n_1 vertices.

A vertex v obtained by solving $W_\alpha x = b_\alpha$ satisfies $s_i(v) = 0$ iff $i \in \alpha$.

Proof. These are the vertices of a generic, affine hyperplane arrangement. □

The sign sequences of the top-dimensional regions which are present in $\mathcal{C}(F_1)$ can be determined by the sign sequences of the vertices of $\mathcal{C}(F_1)$, ignoring signs corresponding to neurons in later layers. The exact coboundary operation defining how to obtain these sign sequences is described by Lemma 3.5.3.

Following the computation of the first layer, subsequent layers' vertices may be found by analyzing the preimage of each bent hyperplane for additional intersections of k bent hyperplanes from the new layer together with $n_0 - k$ bent hyperplanes from the previous layers. Since $F_{k-1} \circ ... \circ F_1$ is affine on each region of $\mathcal{C}(F_{k-1} \circ ... \circ F_1)$, restricted to each region, $F_{ij}(x)$ is affine.

In order to compute the vertices of $\mathcal{C}(F)$ corresponding to bent hyperplanes from further layers, we loop through regions C of $\mathcal{C}(F_{(k-1)})$ and solve systems of linear equations arising from n_0 bent hyperplanes on that region, at least one of which corresponds to a new bent hyperplane F_{kj}. The following lemma guarantees that if we select these combinations of F_{ij} corresponding to earlier layers from only those on the boundary of $\mathcal{C}$, we are guaranteed to obtain all new vertices in $\mathcal{C}(F_{(k)})$. Furthermore, once such an intersection x is found with new bent hyperplanes we may determine whether the intersection belongs to the polyhedral complex by evaluating $F_{ij}(x)$ at only the bent hyperplanes which were not intersected. Thus, we do not have to determine whether $\text{sgn}(F_{ij}(x)) = 0$, removing a source of floating point error.

Lemma 4.1.2. *Let F be a generic, supertransversal neural network with at least n_0 hidden units in its first layer.*

If C is a cell of $\mathcal{C}(F_{k-1} \circ ... \circ F_1)$, then $F_{ij}(C)$ is affine for all $i \leq k$. Call the corresponding affine map $A_{ij} : \mathbb{R}^{n_0} \to \mathbb{R}$. Then,

1. *All* 0*-cells of* $\mathcal{C}(F_k \circ ... \circ F_1)$ *which are contained in the closure of* C *and which are not already in* $\mathcal{C}(F_{k-1} \circ ... \circ F_1)$ *are the solution to a system of* n_0 *affine equations, of which* $1 \leq \ell \leq n_0$ *are of the form:*

$$A_{km}(x) = 0$$

and $0 \leq n_0 - \ell \leq n_0 - 1$ *equations are of the form:*

$$A_{ij}(x) = 0; i < k$$

Here, the A_{ij} *of the* $n_0 - \ell$ *equations from earlier layers are selected such that there exists a vertex of* C *in the intersection of the corresponding bent hyperplanes. In other words, the remaining* $n_0 - \ell$ *equations describe the affine span of a face of* C*.*

2. *A solution to the system of equations described in (1) corresponds to a* 0*-cell of* $\mathcal{C}(F_k \circ ... \circ F_1)$ *contained in the closure of* C *if and only if, for all remaining* (i, j) *pairs with* $i \leq k - 1$*, we have that* $s_{ij}(v) = s_{ij}(C)$*.*

Proof. For statement (1), suppose that v is in the closure of C, where C is a cell of $\mathcal{C}(F_{k-1} \circ ... \circ F_1)$, and v is a vertex of $\mathcal{C}(F_k \circ ... \circ F_1)$. By Theorem 3.5.2, v is the solution to $F_{ij}(x) = 0$ for exactly n_0 node maps. Since $F_{ij}|_C = A_{ij}$, then $A_{ij}(v) = 0$ for those n_0 node maps. If $i < k$ for all of these node maps F_{ij}, then in fact v is a 0-cell of $\mathcal{C}(F_{k-1} \circ ... \circ F_1)$. So if v is a vertex of $\mathcal{C}(F_k \circ ... \circ F_1)$ and not a vertex of $\mathcal{C}(F_{k-1} \circ ... \circ F_1)$, at least one of the F_{ij} must be a node map with $i = k$. Thus, any vertex of $\mathcal{C}(F_k \circ ... \circ F_1)$ which is contained in the closure of C must be a solution to a system of equations of this form. For any solution of this form to be nonempty

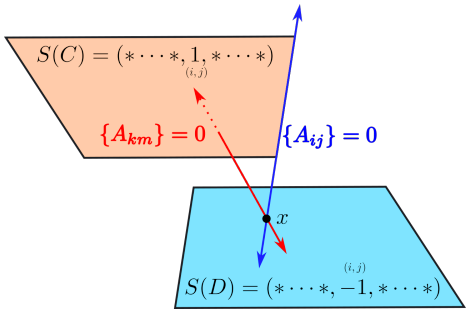


FIGURE 21. Illustration of the key step in Lemma 4.1.2

when intersecting with the closure of C, the the A_{ij} corresponding to this system of equations must satisfy the condition that $C \cap \bigcap\{x : A_{ij}(x) = 0\}$ is nonempty. Since none of the A_{ij} from earlier layers intersect the interior of C, the intersection of the A_{ij} are describing the linear span of a face of C, which must contain a vertex of C.

For statement (2), of course if v is a solution to the system of equations described in (1) and also is in the closure of C, then by Lemma 3.4.2, $s_{ij}(v) = s_{ij}(C)$ when $i \leq k-1$, except for where $s_{ij}(v) = 0$, which by Theorem 3.5.2 occurs for precisely the bent hyperplanes which were intersected to obtain s_{ij}.

In the other direction, if x is a solution to the above system of equations but is not a 0-cell of $\mathcal{C}(F)$, it must not be contained in the closure of C. Then x is contained in the interior of some other cell of $\mathcal{C}(F_{k-1} \circ ... \circ F_1)$, call it D, such that D is not a face of C (Figure 21). If there is some bent hyperplane corresponding to one of the remaining (i,j) pairs such that $s_{ij}(D) \neq s_{ij}(C)$ then we are done. Otherwise we will see a contradiction. If $S(D) = S(C)$ except at (i,j) pairs corresponding to some of the A_{ij}, then by our selection of equations earlier, there is a face E of C which has the sign sequence equal to zero at these coordinates (contained in the intersection of the solution of $A_{ij}x = 0$). If $E = D$, then D is a face of C and we have a contradiction. The only other option is that E is a proper face of D by Lemma 3.4.2. The intersection of the solutions to $A_{ij} = 0$ contains the

affine span of E, so the intersection of these with the closure of D is contained in a proper face of D, and so x, an element of this intersection, cannot be in the interior of D. This contradicts our assumption that x is in the interior of D.

This shows that if x is a solution to the above system of equations but is not a 0-cell of $\mathcal{C}(F)$, then there exists some (i, j) pair with $i \leq k-1$ such that $s_{ij}(x) \neq s_{ij}(C)$ and which does not correspond to the hyperplanes which were intersected.

□

Remark 4.1.3. When determining if a solution x to the system of equations in Lemma 4.1.2 is a vertex of $\mathcal{C}(F_{(k)})$, we look at its sign sequence. However, its sign relative to $\{A_{ij}\}$ is numerically unstable. We would like to guarantee it belongs to the closure of C by evaluating the node maps which do not include A_{ij}. A concern is that it is contained in a different cell D with identical signs to C in $\mathcal{C}(F_{(k-1)})$ except possibly in the locations of the A_{ij} which we intersected, which would make this task impossible. The argument in part (2) shows this does not occur, and the situation pictured in Figure 21 is impossible.

In summary, the following sequence of steps can be used to compute the sign sequences of $\mathcal{C}(F)$, as depicted in Figure 22.

Computing Sign Sequences. To obtain the vertices of $\mathcal{C}(F)$, and thus the n_0-cells of $\mathcal{S}(F)$:

1. Compute the intersections of the hyperplanes from the first layer, as in Lemma 4.1.1. Obtain their sign sequences by evaluating F_{ij} on each intersection. Restricting these sign sequences to the signs of F_{1j} obtains $\mathcal{C}(F_1)$.

2. To compute $\mathcal{C}(F_i)$, loop through regions C in $\mathcal{C}(F_{i-1})$. On each region C,

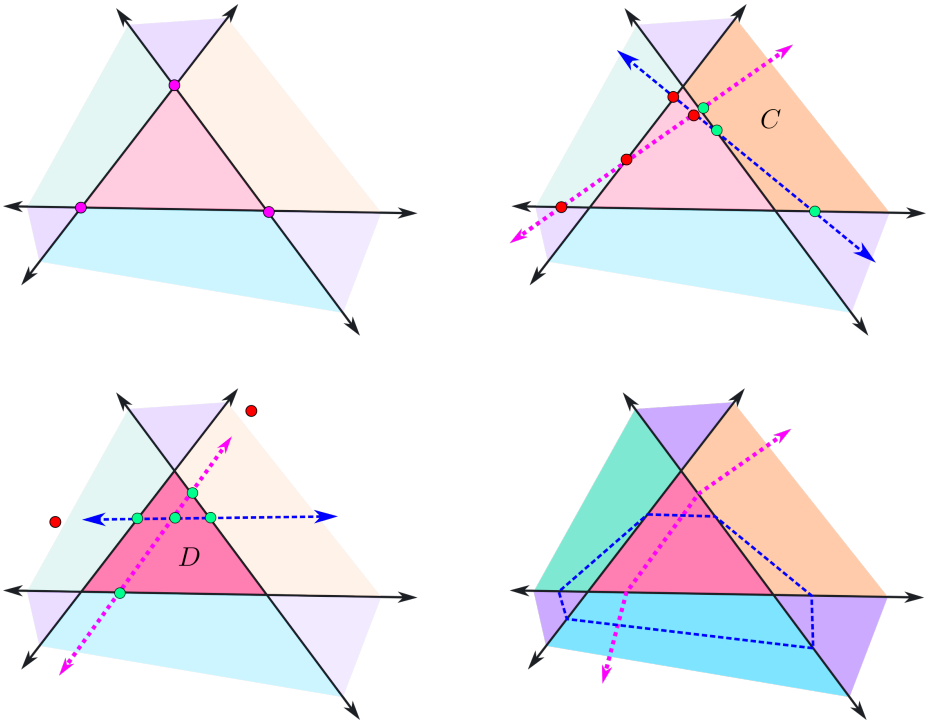


FIGURE 22. An illustration of the first algorithm for computing $\mathcal{C}(F)$. Upper left: Step 1. Upper right and bottom left: Step 2, keeping the green vertices and discarding the red ones for regions C and D respectively. Bottom right: The complete $\mathcal{C}(F)$.

(a) For $1 \leq k \leq n_0$, compute the intersections of k bent hyperplanes from the new layer with $n_0 - k$ bent hyperplanes from previous layers, the latter of which are selected so that their intersection forms an $n_0 - k$-face of C.

(b) Evaluate $F_{ij}(x)$ for each computed intersection x. Then keep x as a vertex of $\mathcal{C}(F)$ if and only if $F_{ij}(x) = F_{ij}(C)$ for $i \leq k-1$, following Lemma 4.1.2.

Explicit implementation of this algorithm is included in the code repository provided in the supplementary material.

We finally discuss the question of algorithmic complexity. As deep ReLU networks only have polynomially many regions in the number of hidden units, at least on average at initialization [17], and the number of possible combinations of k neurons from n_i neurons together with $n_0 - k$ neurons from $n_0 + ... + n_{k-1}$ neurons is also polynomial in the total number of hidden units, it is possible to obtain the canonical polyhedral complex $\mathcal{C}(F)$ in polynomial expected time in the number of hidden units.

Lemma 4.1.4. *Let $F : \mathbb{R}^{n_0} \to \mathbb{R}$ be a randomly-initialized ReLU neural network satisfying the conditions in [17].*

If $N = \sum_{i=1}^{m} n_i$ is the number of hidden units of F, then the average number of linear equations and sign sequence evaluations required for the computation of $\mathcal{S}(F)$ via the algorithm above is $O(N^{2n_0+1})$.

Proof. The first step in Lemma 4.1.1 involves solving $\binom{n_1}{n_0}$ equations, which is $O(n_1^{n_0})$. Then in the the recursive step in Lemma 4.1.2, in each subsequent layer $i \geq 2$, there are an average of $O((n_1 + \cdots + n_{i-1})^{n_0})$ regions to iterate through [17]. In each region, there are fewer than $\binom{n_1+\cdots+n_i}{n_0}$ new equations to solve. This yields the following big-O upper bound for the average computational complexity of this algorithm:

$$n_1^{n_0} + \sum_{i=1}^{m} (n_1 + \cdots + n_{i-1})^{n_0} n_1 + \cdots + n_i)^{n_0} \leq N^{n_0} + \sum_{i=1}^{m} N^{n_0}$$

Since $N = n_1 + \cdots + n_m$, and each of the $n_i \geq 1$, it must be the case that the depth $m \leq N$, so this expression is

$$O(N^{2n_0+1})$$

This is polynomial in N and exponential in n_0. □

We compare with some other possible naïve approaches to computing the face relations of $\mathcal{C}(F)$. Existing methods of tracking which vertices are present in a way which could hypothetically allow for tracking pairwise intersection between cells currently track the full face poset [29, 30], and are thus storage-intensive. Next, consider the approach of computing which regions are present and then computing their intersections. First, naïve search for all regions' activation patterns in $(-1, 1)^N$ would require determining whether there are solutions to 2^N linear inequalities in N variables, which would then define highly redundant descriptions of polyhedral regions. This can be done more optimally than checking each set of inequalities [26, 32]. However, without having tracked sign information, to determine if two polyhedra share a face and find the dimension of that face, requires finding whether the union of two sets of N linear inequalities is consistent with the expectation that there will be cancellation of redundancies by the *exact equality* of some linear combinations of these linear inequalities in order to observe the existence of lower-dimensional faces. Numerical error in the expression of these linear inequalities can thus lead to catastrophic failure in identifying shared lower-dimensional faces, especially the dimension of those faces, as we see in Section 4.2.

4.2. Avoiding Numerical Error and Singular Matrices

The algorithm in the previous section works perfectly under the assumption that systems of linear equations can be solved with complete accuracy. We next account for the practical limitations which arise in floating point computation.

Numerical Stability of Vertex Sign Sequences and Polyhedral Intersection

We first discuss the problem of computing the sign sequence of a vertex of $\mathcal{C}(F)$. Naively, if we compute a solution x to $F_{ij}(x) = 0$, and then numerically evaluate the node map $F_{ij}(x)$, the result may not be exactly zero due to floating point error. However, machine epsilon-level errors obtained when solving for the location of a vertex will not generally lead to errors in computing the sign sequence of a vertex, for the following reasons. When determining the sign sequence of a vertex, which of its signs are zero is determined by which hyperplanes were intersected, and the remaining signs are stable to small perturbations, since the sets $F_{ij} > 0$ and $F_{ij} < 0$ are open sets. As long as the error in computing solutions to linear equations is small compared to the size of the cells in the polyhedral complex, the proposed algorithm will find the correct sign sequence of each vertex, and as a result the correct combinatorics of the polyhedral complex.

More explicitly, suppose that in step (1) we only find an approximate intersection $\tilde{x}$ instead of an exact intersection x, with $|\tilde{x} - x| < \varepsilon$ determined by machine precision. According to the above Lemma 4.1.2, if we do locate the exact intersection x, determining whether this point x is a vertex of the polyhedral complex $\mathcal{C}(F)$ relies only on determining the sign of $F_{ij}(x)$ for indices (i, j) corresponding to the bent hyperplanes which were not intersected. Under the supertransversality assumptions, $F_{ij}(x)$ is strictly nonzero in these (i, j) coordinates. We also note that F_{ij} is continuous, so there is an open neighborhood N containing x where, for all $\tilde{x} \in N$, we have $F_{ij}(\tilde{x}) = F_{ij}(x)$ for all relevant i, j. We can therefore expect that, despite possible numerical error in computing $\tilde{x}$, the computation of the sign sequence of the corresponding vertex is stable under these perturbations, since by recording which bent hyperplanes were intersected via signs

we have eliminated the unstable operation of determining which signs were exactly 0.

We contrast the numerical stability of this algorithm to the naive approach of intersection of two polyhedra P_1 and P_2 belonging to $\mathcal{C}(F)$ which were computed without tracking information about activation patterns or sign sequences. We expect the polyhedra of $\mathcal{C}(F)$ to intersect in either a lower-dimensional set, or not at all. If using a computational approach such as [32], P_1 and P_2 are stored in H-representations as $A_1x \leq b_1$ and $A_2x \leq b_2$ then the question of whether P_1 and P_2 intersect, and in what dimension they intersect, is determined by redundancy elimination on the union of the two systems of inequalities. If the polyhedra share a face, we expect all points of P_1 to satisfy $wx \leq b$ and all points of P_2 to satisfy $wx \geq b$ for some set of weights w. Under redundancy elimination, this will resolve to $wx = b$. However, once computed and stored, the H-representations P_1 and P_2 may not have the same entries for the row w: numerical error, arising from matrix multiplication on different matrices (some with zeroes, and others without). This could lead to P_1 satisfying the inequality $w_1x \leq b$ and P_2 satisfying the inequality $w_2x \geq b$ instead, with slightly different weights w_1 and w_2. Even if $|w_1 - w_2| < \varepsilon$ is small, these equations could have drastically different solution sets if $|w_1|$ and $|w_2|$ are also small, leading to numerical instability in the problem of determining the rank of intersection of P_1 and P_2, requiring thresholding.

Avoiding Singular Matrices

Another possible problem which can occur during the computation of $\mathcal{C}(F)$ is if the algorithm attempts to solve for an intersection of bent hyperplanes which should not exist at all. It is possible for the matrix corresponding to the linear map

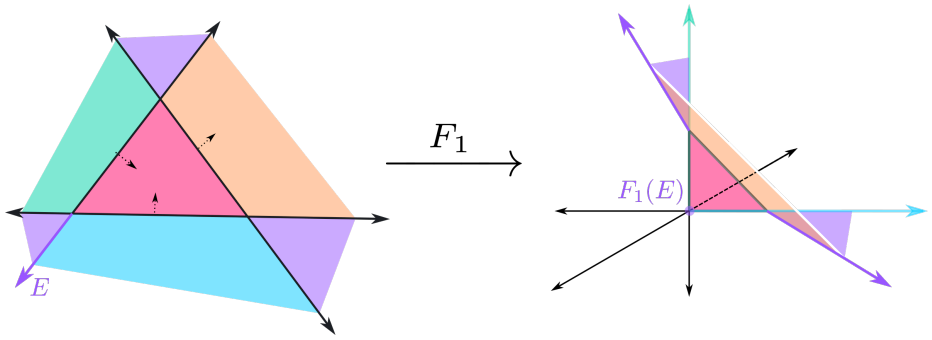


FIGURE 23. If F_1 is the pictured layer map, it is almost impossible for any bent hyperplane from a layer map given by F_2 or later to intersect edge E, as $F_1(E)$ is the origin.

F_i restricted to a region to be less than full rank. In this case, sometimes the region or the boundary of the region is collapsed. For example, in Figure 23 we observe that generically the preimage of any plane in $\mathbb{R}^3$ cannot intersect the edge E due to its image under F_1 being a point. Furthermore, any hyperplane intersecting image of the the blue region under F_1 must have a preimage which is parallel to E, and generically no hyperplane will intersect the image of the purple region incident to E. However, it may be the case that a linear algebra solver might not identify the system as singular, and thereby seek an intersection between the bent hyperplane and E. Once found, the intersection may even have the correct sign sequence! However, it is possible to use sign sequences to avoid this problem to begin with, by directing the algorithm to skip solving this system of equations.

To describe when we should not attempt to find the intersection of bent hyperplanes in a region, we use the following lemma.

Lemma 4.2.3. *Suppose $F_{(\ell)}$ is generic and supertransversal. Let D be a k-cell of $\mathcal{C}(F_{(\ell)})$. Let $M_i(D)$ be the count $\#\{j \mid s_{ij}(D) = 1\}$. Call $m_\ell(D) = \min_{i \leq \ell}\{M_i(D)\}$.*

1. *If $m_\ell(D) < k$, then a set of $n_0 - k$ hyperplanes in $R^{(\ell+1)}$ may intersect $F_{(\ell)}(D)$ and each other nontrivially only in a measure-zero subset of parameter space of $F_{\ell+1}$.*

2. *If $m_\ell(D) \geq k$, then it is generic for a set of $n_0 - k$ hyperplanes in $R^{(\ell)}$ to intersect each other in the affine span of $F_\ell(D)$ in one point.*

Proof. Observe that $M_i(D)$ gives an upper bound on the rank of $F_i|_{F_{(i-1)}(D)}$. The composite $F_{(\ell)|_D}$ must have rank bounded above by their minimum m_ℓ. Generically in the parameter space of $F_{(\ell)}$, this rank is equal to $\min(m_\ell, k)$.

For the first statement, if the dimension of the affine span of $F_{(\ell)}|(D)$ is less than k, then a set of k hyperplanes intersecting this affine span will generically not have a shared intersection within this affine span.

For 2. if $m_\ell \geq k$ then with probability 1 the rank of $F_(\ell)|_D$ is equal to k, so $\dim F_{(\ell)}(D) = k$ and a set of k hyperplanes intersecting the affine span of $F_{(\ell)}(D)$ generically intersect in a single point therein. □

Therefore by computing $m_i(D)$ we may evaluate the sign sequences of a k-cell D to determine whether intersections with $n_0 - k$ hyperplanes could exist before proceeding with their computation.

Remaining Numerical Error

In practice the floating point errors which are observed to occur when following this algorithm appear to be when a different number of bent hyperplanes appear to converge at a single point than should under supertransversality, due to insufficient resolution; that is, genericity failures. Analyses of when this may occur are given in section 5.1.

4.3. An Alternative Algorithm

The given algorithm in Section 4.1 is not completely efficient, as it computes the possible location of intersections multiple times. Each vertex is contained in 2^{n_0} regions, so searching all regions for each vertex locates that vertex 2^{n_0} times. Leveraging polyhedral geometry, we may reduce the multiplicity of solutions, thereby leading to an exponential improvement in time complexity. The process given by this algorithm is illustrated in Figure 24.

The strategy here is to only look for intersections of bent hyperplanes which are possible to occur according to sign polyhedral geometry. The step in 2a limits the computation of intersections of single new bent hyperplanes with edges from the previous layer by sign sequence. Then in order for two or more new bent hyperplanes to intersect in a region, they must also have intersected the boundary of that region. So, in step 2b we work upwards in dimension on the boundaries of regions to avoid investigating all regions.

To obtain the vertices of $\mathcal{C}(F)$:

1. Compute the intersections of the hyperplanes from the first layer. Obtain the sign sequences of the vertices by evaluating F on each intersection.

2. For each subsequent layer i, loop through dimensions $1 \leq k \leq n_0$:

   (a) For $k = 1$, obtain the sign sequence of the set of edges in $\mathcal{C}(F_{(n)})$ through the coboundary operation on vertices. For each edge E, if its vertices differ at sign (i, j), compute the intersection of the hyperplane corresponding to F_{ij} and that edge. These are vertices found as the intersection of $k = 1$ new hyperplane with $n_0 - 1$ old hyperplanes.

   (b) Then for each dimension $k \geq 2$, do the following:

i. Obtain the sign sequences of vertices found as the intersection of $k-1$ new hyperplanes with $n_0 - k + 1$ old hyperplanes.

ii. Obtain the all k-dimensional regions R which are incident to the most recently added vertices, through the coboundary operation. Any intersection of k new hyperplanes with $n_0 - k$ old hyperplanes must occur within these regions R.

iii. For each R, determine whether it is possible for k new hyperplanes to intersect within this region by Lemma 3.9.3. Then obtain the sign sequence of an n_0-dimensional region C incident to the k-dimensional region by replacing all remaining 0 entries with -1. For each j in which a new vertex incident to R has a 0, obtain the node map within C corresponding to F_{ij}; this gives an equation defining a hyperplane intersecting the boundary of R.

iv. For each set of k distinct new hyperplanes which intersect the boundary of R, find the intersections of these k new bent hyperplanes with the $n_0 - k$ bent hyperplanes corresponding to the region R. Evaluate whether they occur within R via sign sequences as before. These new vertices are the intersection of k new hyperplanes with $n_0 - k$ old hyperplanes.

v. If there are no new intersections, terminate the loop and move to the next layer.

Using this algorithm, the presence of a vertex with a given sign sequence is evaluated at most once. In particular, the intersection of k hyperplanes from the next layer is only computed if all k hyperplanes intersect the boundary of the region they are supposed to intersect on. An empirical comparison of the time

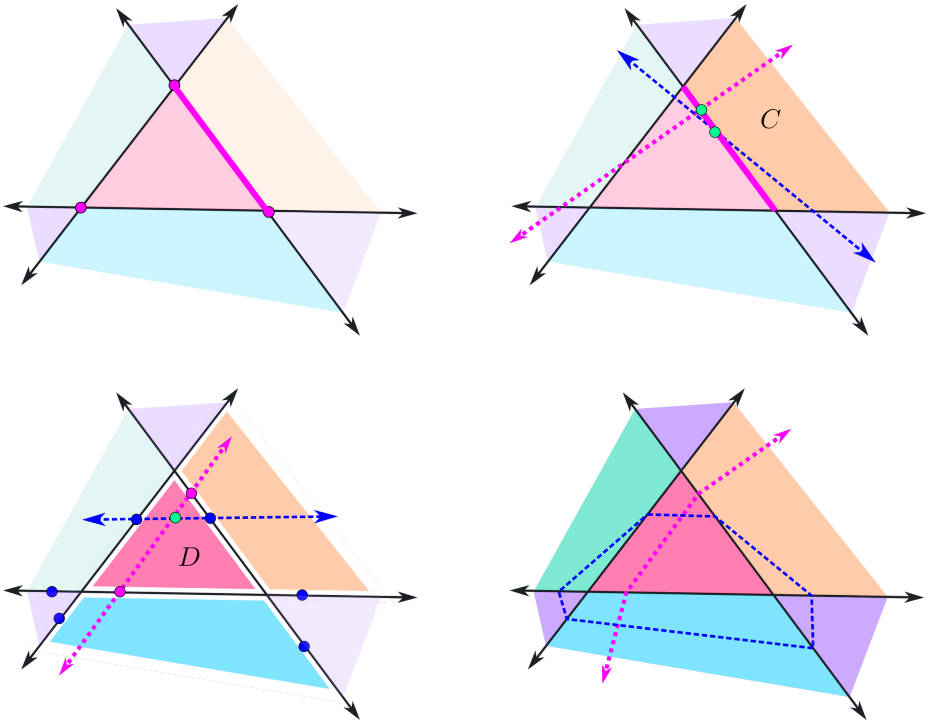


FIGURE 24. The improved algorithm for obtaining the vertices of $\mathcal{C}(F)$. Upper left: Step 2a, selecting an edge. Upper right: Step 2a, finding the location of the intersection of each bent hyperplane and that edge, if it exists, via computations on region C. Lower left: Step 2b for $k = 2$, specifically on the region D, with the other two regions to investigate highlighted. Bottom right: Final $\mathcal{C}(F)$.

required to compute the vertices of $\mathcal{C}(F)$ using this algorithm compared with the algorithm in Section 4.1 for input dimensions 2 and 3 is presented in Figure 25.

Explicit implementation of this algorithm is provided in the code repository included in the supplementary materials.

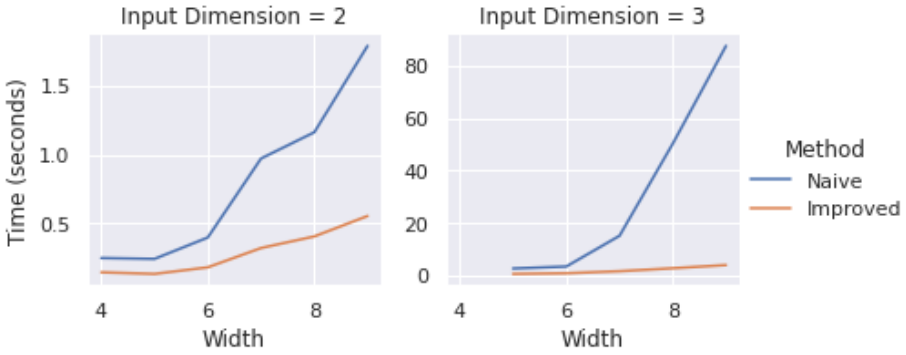


FIGURE 25. Time required to compute $\mathcal{C}(F)$ for randomly-initialized neural networks of input dimensions $n_0 = 2$ and 3 and two hidden layers, for the naïve and improved algorithms.

CHAPTER V

EXPERIMENTAL OBSERVATIONS OF NEURAL NETWORKS' TOPOLOGICAL PROPERTIES

We proceed from theory to implementation to make experimental observations. The first two sets of experiments in this section are about randomly-initialized neural networks, and the third is about neural networks' decision boundaries during training.

5.1. Empirical Measurements about Vertices of $\mathcal{C}(F)$

To understand when algorithmic failure is likely to occur, we measure the empirical distribution of properties of vertices in $\mathcal{C}(F)$ for different architectures. The canonical polyhedral complex associated to a single layer has a fixed number of vertices due to direct correspondence with hyperplane arrangements. However, deeper neural networks have a variable number of vertices, and distributional information about the number of vertices as well as the minimum paired distance between inequal vertices are reported herein.

Under standard normal weights and biases, we initialize 50 neural networks of input dimensions $n_0 = 2, 3, 4$ and widths from $n_0 + 1$ to 29, or the highest value computationally accessible.

First, we discuss the number of vertices present in $\mathcal{C}(F)$ at initialization, by architecture. The number of *top-dimensional regions* of $\mathcal{C}(F)$ for a ReLU neural network at initialization is, on average, $O(\frac{N^{n_0}}{n_0!})$ [17]. Each region may contain a very high number of vertices, and each vertex is incident to 2^{n_0} regions, so measuring the distribution of number of vertices is still beneficial. indeed, without

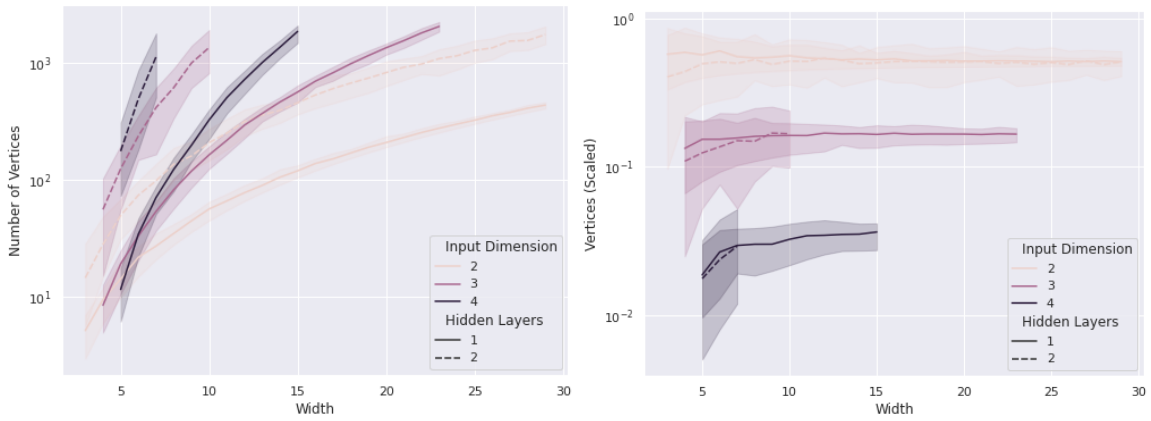


FIGURE 26. Left: The number of vertices of $\mathcal{C}(F)$ increases subexponentially with width. Right: Dividing the number of vertices by N^{n_0} demonstrates $O(N^{n_0})$ scaling.

knowing the number of unbounded regions, the number of vertices cannot be determined from the number of regions directly. We observe that the number of vertices, also, scales as $O(N^{n_0})$ (Figure 26).

We additionally measure the distance between the closest pair of vertices for each neural network, as seen in Figure 27. This scaling provides a way to obtain numerical estimates for the accuracy necessary to solve for vertices of $\mathcal{C}(F)$ under supertransversality assumptions. While the distance between vertices is not a lower bound on the distance a vertex could move before its measured sign sequence changes, these measurement may be used as a proxy in the following way. Let D be the minimum distance between vertices of $\mathcal{C}(F)$. If D is smaller than machine epsilon or the accuracy of the numerical solver, then two vertices v_1 and v_2 which satisfy $d(v_1, v_2) = D$ will numerically satisfy $F_{ij}(v_1) = F_{ij}(v_2)$, and have numerically equal sign sequences. Any (i, j) coordinates for which $s_{ij}(v_1) \neq s_{ij}(v_2)$ will resolve under evaluation as $\tilde{s}_{ij}(v_1) = \tilde{s}_{ij}(v_2) = 0$, making it appear as if more bent hyperplanes intersect in a single point than are predicted in general.

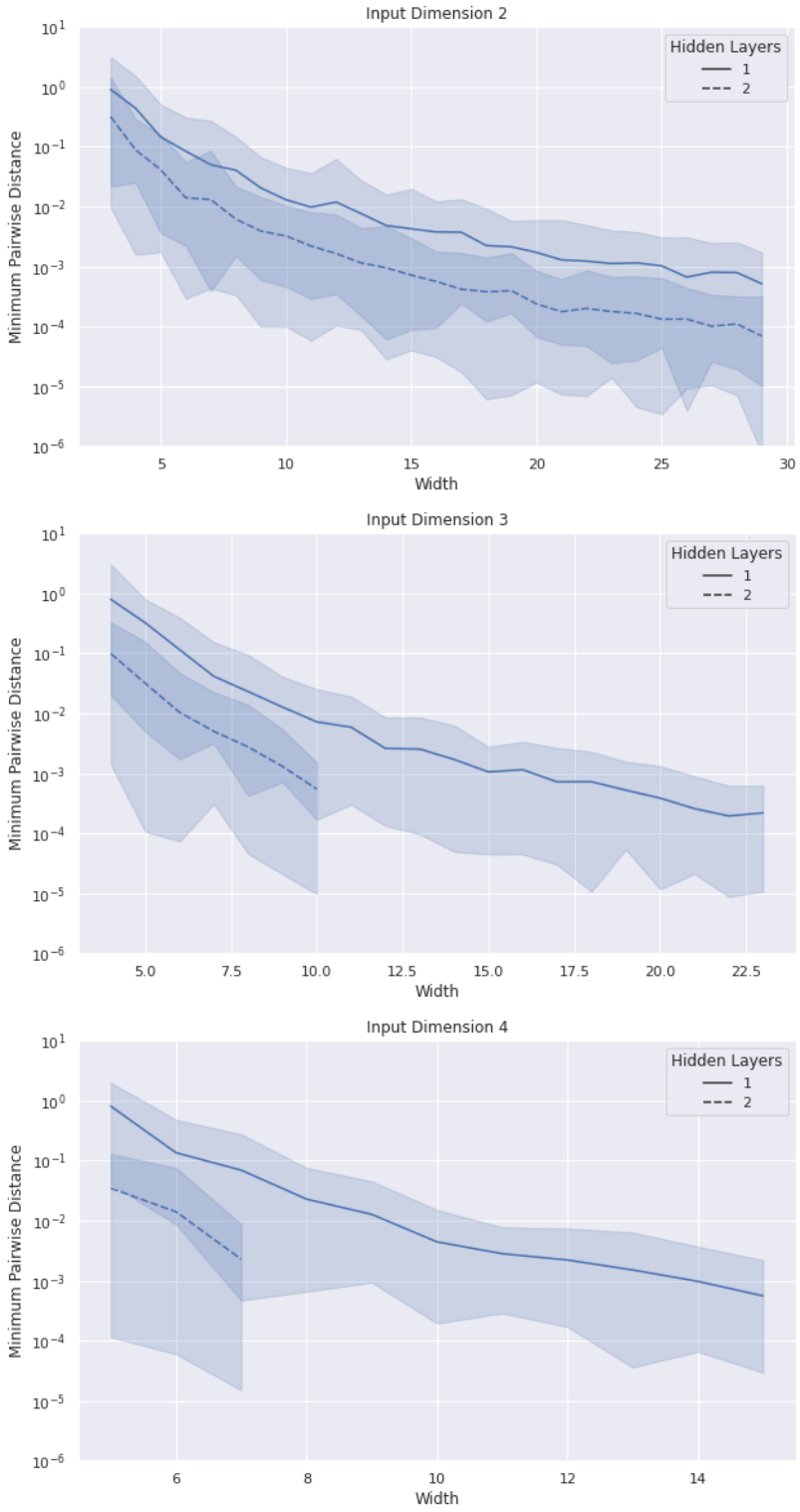


FIGURE 27. Distribution of minimum distance between vertices of $\mathcal{C}(F)$ by the input dimension and width of F. The width is the 95th percentile of minimum distances.

In practice, this form of numerical error occurred qualitatively most often when neural networks with insufficient architecture were trained on topologically complex tasks.

5.2. Decision Boundaries of Randomly-Initialized Neural Networks

We obtain statistics about the decision boundaries of binary classification networks, and find stark differences in the behaviors of shallow and deeper network architectures. To our knowledge, this is the first experimental determination of the *exact* topology of a large collection of decision boundaries with input dimension greater than two while having more than one hidden layer.

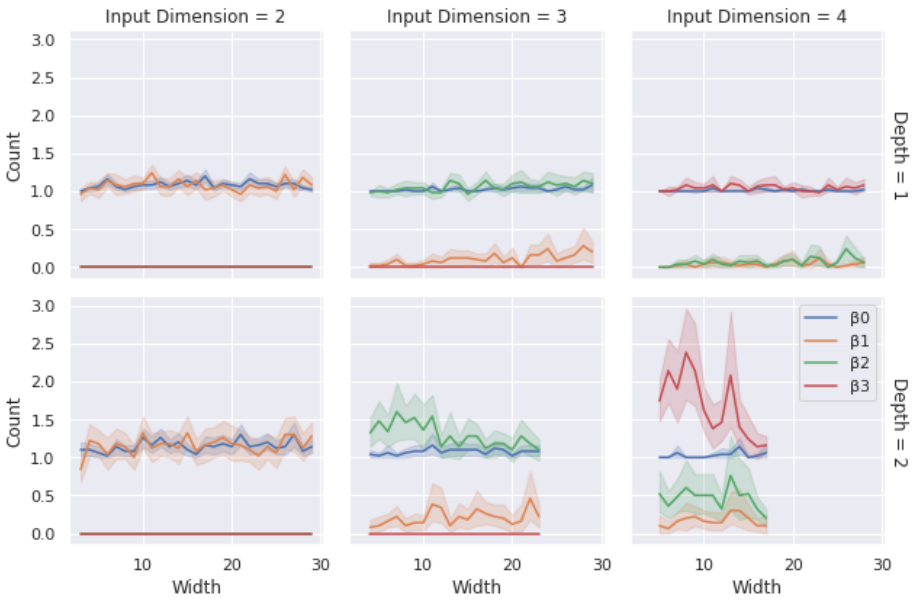


FIGURE 28. Average Betti numbers of the network decision boundary at initialization, with 95% confidence interval shown.

Experimental design

We randomly initialize 50 networks of fully-connected architectures $(k, n, 1)$ and $(k, n, n, 1)$ for $2 \leq k \leq 4$ with standard normal weights and biases (See Figure 17). We will call the networks of architecture $(k, n, 1)$ "shallow" and those of architecture $(k, n, n, 1)$ "deep" for the purposes of comparison in this section. We compute the canonical polyhedral complex $\mathcal{C}(F)$ for each network using an implementation of the algorithm described in Section 4.1 using Pytorch linear algebra solver [24]. We then obtain the Betti numbers β_i for $0 \leq i \leq k-1$ of the decision boundary of the network at initialization, by constructing the boundary map determined in Lemma 3.5.3, with additional point at infinity. The Betti numbers were obtained using a Sage implementation of general chain complexes [27]. The resulting Betti numbers provide a measure of topological complexity of the decision boundary at initialization.

Results and Discussion

We observe that the topology of the decision boundary for shallow networks, regardless of input dimension, remains constant over the range of dimensions investigated. Figure 28 depicts a range of networks of shallow architecture (top) with deep architecture (bottom), together with standard error. In contrast, for deep networks, there is both greater variability in the distribution of the topology of the decision boundary, and increasing width seems to, at least for $n_0 > 2$, lead to the the topological properties of the decision boundary appearing to change in distribution as the width n increases. We conjecture that a plausible explanation is that deep networks appear to require greater width before their network functions converge to Gaussian processes in distribution [7].

Two of these Betti numbers measure the number of bounded and unbounded components of the decision boundary. Since all unbounded components are compactified by attaching them to the same point, in the compactification they correspond to $(n_0 - 1)$-cycles belonging to the same connected component, which are counted by β_{n_0-1}. So the number of bounded and unbounded components can be computed by $\beta_0 - 1$ and $\beta_{n_0-1} - \beta_0 + 1$, respectively.

We observe that *bounded* connected components of the decision boundary at initialization are rare, with frequency decreasing with input dimension in both shallow and deep networks: For example, 8.1% of networks of the form $(2, n, 1)$ contain at least one bounded component, whereas only 0.05% of networks of the form $(4, n, 1)$ contain as much at initialization. Furthermore, regarding the number of unbounded components, across all shallow networks investigated, the largest number of unbounded components observed at initialization was 3, with a mode of 1 (average 1.0, 1.03 and 1.04, for $n_0 = 2, 3$, and 4 respectively). In deeper networks, in contrast, the number of unbounded components at initialization appears to be on average greater $(1.1, 1.4, 1.4$, respectively) reaching maximum observed values of 5, 7 and 12 for $n_0 = 2, 3$, and 4 respectively. However, the most common observation is still that a network at initialization has one unbounded connected component, and whether there is any trend associated with width is unclear. These observations lend additional credence to the notion that depth has a stronger influence than width on the topological complexity that a network can be easily trained to express.

Additional Distributional Information

Here we report additional distributional information about the topology of decision boundaries of randomly initialized ReLU networks with different architectures. Table 2 summarizes distributional information about the Betti numbers of the decision boundary, and Table 3 summarizes information about the connected components of the decision boundary. Figure 30 gives additional distributional information for selected architectures. While shallow architectures again have a very constant distribution of the number of unbounded components even across input dimension, the number of unbounded components seen at initialization in deeper architectures is much greater, and the distributional variability with width is apparent.

TABLE 2. Betti numbers of the compactified decision boundary dependent on architecture, across the range of widths studied (50 networks of each architecture). In β_{n_0-1}, deeper architectures exhibit greater variability and greater apparent change with width across the range of widths studied. This information is seen in Figure 28

Shallow Architectures				
	β_0		β_{n_0-1}	
	Average	SD	Average	SD
$(2, 5, 1)$	1.06	0.24	1.02	0.55
$(2, 15, 1)$	1.14	0.40	1.06	0.58
$(3, 5, 1)$	1.00	0.00	1.00	0.29
$(3, 15, 1)$	1.00	0.00	0.96	0.28
$(4, 5, 1)$	1.00	0.00	1.00	0.00
$(4, 15, 1)$	1.00	0.00	1.00	0.20
Deep Architectures				
	β_0		β_{n_0-1}	
	Average	SD	Average	SD
$(2, 5, 5, 1)$	1.06	0.24	1.18	0.63
$(2, 15, 15, 1)$	1.10	0.30	1.32	0.71
$(3, 5, 5, 1)$	1.00	0.00	1.48	0.88
$(3, 15, 15, 1)$	1.00	0.00	1.28	0.73
$(4, 5, 5, 1)$	1.00	0.00	1.74	1.01
$(4, 15, 15, 1)$	1.00	0.00	1.24	0.96

TABLE 3. Average number of bounded and unbounded components of the decision boundary dependent on architecture. Bounded components became exceedingly rare with increased input dimensions, to the extent that measured variability is 0 for some of the given architectures.

Shallow Architectures				
	Unbounded		Bounded	
	Average	SD	Average	SD
$(2, 5, 1)$	0.96	0.60	0.06	0.23
$(2, 15, 1)$	0.92	0.53	0.14	0.40
$(3, 5, 1)$	1.00	0.28	0.00	0.00
$(3, 15, 1)$	0.96	0.28	0.00	0.00
$(4, 5, 1)$	1.00	0.00	0.00	0.00
$(4, 15, 1)$	1.00	0.20	0.00	0.00
Deep Architectures				
	Unbounded		Bounded	
	Average	SE	Average	SD
$(2, 5, 5, 1)$	1.12	0.68	0.06	0.24
$(2, 15, 15, 1)$	1.22	0.76	0.10	0.30
$(3, 5, 5, 1)$	1.46	0.91	0.02	0.14
$(3, 15, 15, 1)$	1.18	0.83	0.10	0.30
$(4, 5, 5, 1)$	1.74	1.00	0.00	0.00
$(4, 15, 15, 1)$	1.24	0.95	0.00	0.00

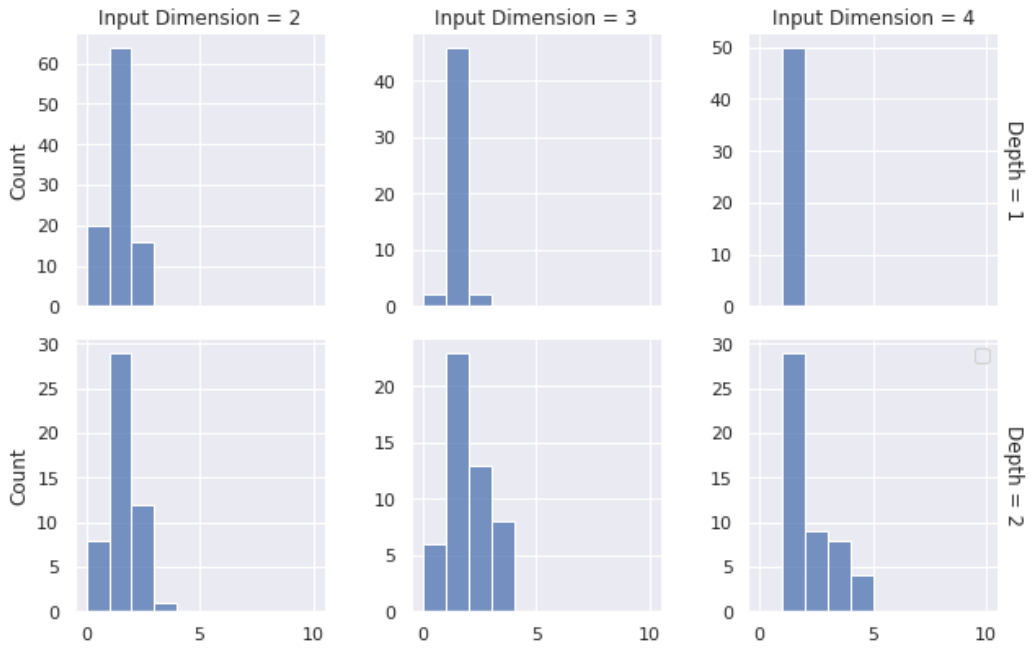


FIGURE 29. Distribution of the total number of connected components of the decision boundary for architectures of width 5.

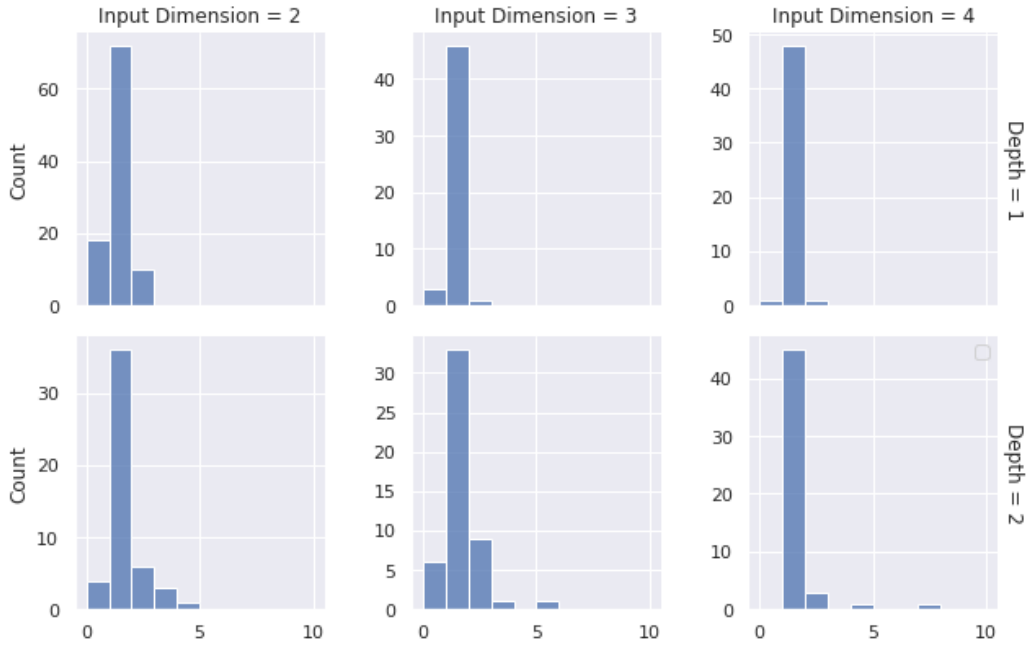


FIGURE 30. Distribution of the total number of connected components of the decision boundary for architectures of width 15.

5.3. Neural Networks During Training

The previous two experimental sections computed the topological properties of a neural network when randomly initialized. We next focus on how the topology of the decision boundary of a neural network changes as the neural network is trained on a given dataset. This may give insight into whether neural networks **topologically generalize**, as discussed in the introduction. Furthermore, the changes which take place in the decision boundary's topology during training may shed light on whether there is topological bias in the algorithm of stochastic gradient descent.

In this experiment, we train neural networks of a range of architectures to classify what we are calling the XOR task (Figure 31) and the Torus task (Figure 32).

Experimental Setup

Neural networks of different architectures were trained on the XOR and Torus tasks via stochastic gradient descent, using binary crossentropy loss, until apparent validation loss convergence. The weights of the trained neural networks were recorded during the training process. Betti numbers of the decision boundary of the neural network were then computed at evenly spaced time steps throughout training. Descriptions of the specific methods for each task follow.

XOR Task

The XOR task is a classical machine learning task which is used to illustrate the necessity of at least one hidden layer for neural networks to successfully learn certain tasks, and is an exemplar for the role of topology in neural network

architecture selection. The classical XOR task is to train a multilayer perceptron to classify the points $(1, 1)$ and $(-1, -1)$ as belonging to one class and the single points $(-1, 1)$ and $(1, -1)$ as belonging to a second class. To make this task compatible with the manifold hypothesis, we define two classes each sampled from mixed Gaussian density functions, with centers at the given points, as illustrated in Figure 31.

Because the true ideal decision boundary of this task is nonmanifold (See Figure 31, right), to evaluate topological generalization we note that the true decision boundary of a neural network which has topologically generalized on this task is given by a generic perturbation of the ideal decision boundary, pictured in the fourth image. Under the distinguished-point compactification, this decision boundary is a wedge sum of two circles, so we expect the decision boundary of a topologically-generalized neural network will satisfy $\beta_0 = 1$, $\beta_1 = 2$.

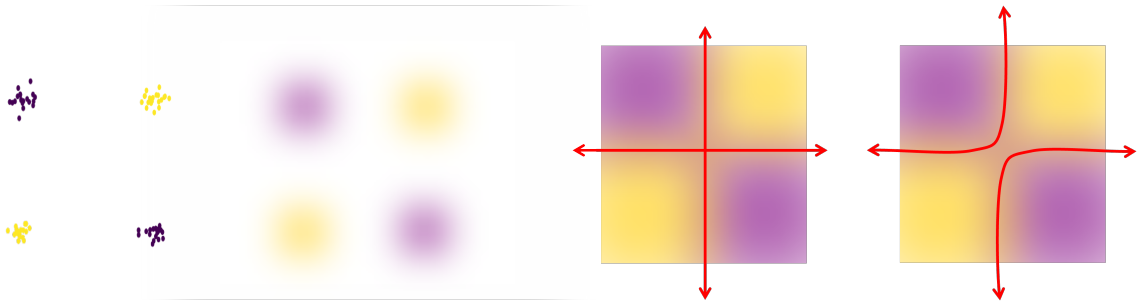


FIGURE 31. The XOR task. Left: Sample data. Left middle: Density function for the two data distributions. Right middle: Ideal classification function with true ideal decision boundary superimposed. Right: Generically perturbed ideal decision boundary.

Thirty different shallow (one hidden layer) and deep (two hidden layer) neural networks with input dimension 2 and widths 5, 10, 15 20 and 25 were initialized using the same scheme as in Section 5.2. Following initialization, the neural networks were trained using stochastic gradient descent. Each training step was

performed on a sample of 20 points sampled from each class, with a learning rate of 10^{-3}. The neural networks were trained for 500 training steps, and the weights were saved every 50 training steps. The Betti numbers of the decision boundary were computed for each of the recorded weights.

Torus Task

The Torus task consists of the classification task of separating a torus generated through the standard parametrization from an annulus at its center, as pictured in Figure 32.

Thirty different shallow (one hidden layer) and deep (two hidden layer) neural networks with input dimension 3 and widths ranging from 5 to 20 were initialized using the same scheme as in Section 5.2. Following initialization the neural networks were trained using stochastic gradient descent against binary crossentropy loss. Each training step was performed on a sample of 50 data points sampled from each class, with a batch size of 50, learning rate of 5×10^{-4}, and trained for a total of $500,000$ training steps.

Results and Discussion

XOR Task

All architectures selected were capable of learning the XOR task. As expected due to their greater number of parameters, deep neural networks reached a lower loss, on average, than shallow neural networks with the same width. We compare the effects of increasing width and increasing depth on topological generalization.

Neural networks with greater width generally showed improved topological generalization, on average. As networks get wider, a greater proportion of trained

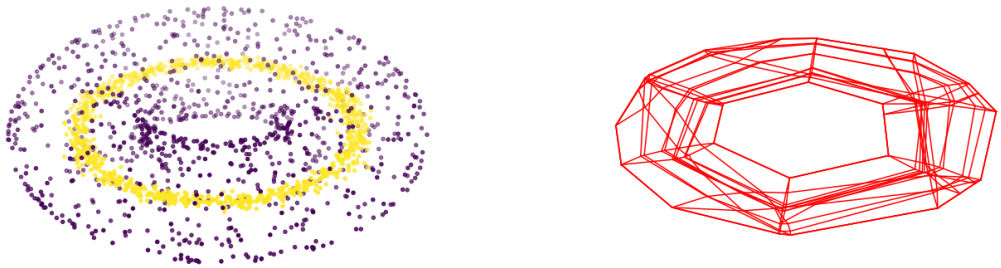


FIGURE 32. The Torus task. Left: Labeled data drawn from the two density functions. Right: A wireframe skeleton of the decision boundary of a neural network trained on the sample data, with edges from $\mathcal{C}(F)$.

networks exhibit Betti numbers consistent with topological generalization (Figure 33). Additionally, wider neural networks approached topological generalization sooner in their training path (Figure 34). We conclude that increasing parameters by widening the neural network appears to improve the ability of the neural network to topologically generalize. This is somewhat surprising: Classical machine learning wisdom is that increasing the complexity of a hypothesis class reduces the generalization capacity of a model (the bias-variance tradeoff). Here the hypothesis class has increased complexity, and a greater topological complexity is capable of being expressed by these wider neural networks, but the increased width does not lead to increased variability in the learned decision boundary; in fact, the opposite has occurred.

In contrast to increasing width, increasing depth does appear to increase the observed topological complexity of learned decision boundaries. First of all, as seen in Figure 34, wider networks required more training steps to approach the expected $\beta_1 = 2$ on average. While it is plausible that this is an artifact of

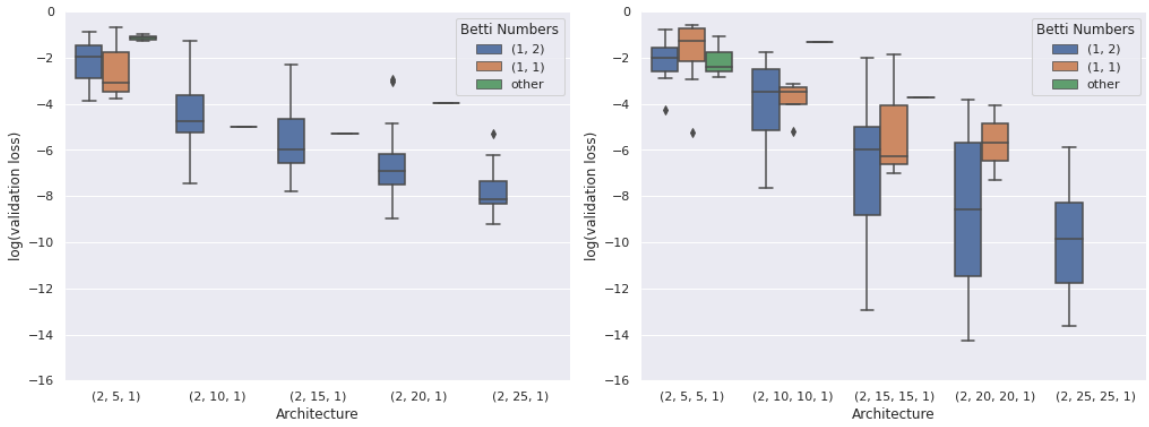


FIGURE 33. A comparison between shallow (left) and deep (right) architectures' topological generalization on the XOR task.

learning rate, the pictured training paths indicate that apparent convergence in topological complexity was reached in the training paths investigated, and that the final topological complexity was typically more variable even after apparent convergence. Secondly, after training convergence, even though deep neural networks had lower loss in comparison to the shallow neural networks of the same width, they exhibited worse topological generalization, with over 92% of shallow neural networks exhibiting topological generalization, whereas only 84% of deep neural networks exhibited such generalization. The greater topological variability exhibited by deeper neural networks at initialization was carried through to their training behavior.

Torus Task

Evaluation showed that the networks trained on the Torus task, despite the hyperparameter search, were relatively underfit, performing subjectively poorly on the task. Though training loss remained relatively constant, these networks failed to reach a trained state, as evidenced by validation loss remaining relatively high

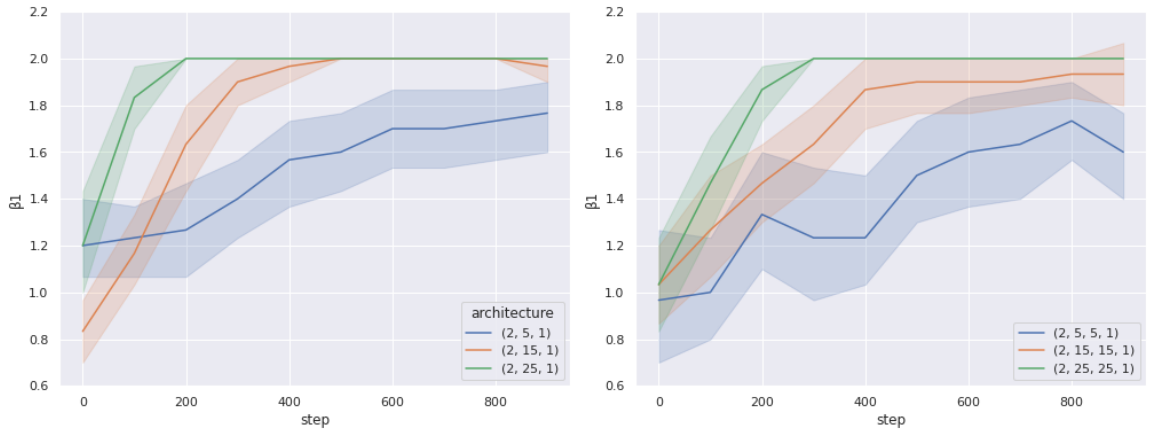


FIGURE 34. The average value of β_1 approaches 2 as the neural networks train on the XOR task. Wider neural networks approach topological generalization faster and exhibit less variability in learned topological features.

(above 10^{-3}). Higher learning rates corresponded with higher validation loss at convergence, and continuing to decrease learning rate led to slower training but apparent validation loss values remaining minimized around 10^{-3}.

While a systematic assay of network dynamics was not performed, we observe that different choices in training regime led to different network topological properties during training. To reach a point of comparison, an alternative training regime was selected for the $(3, 15, 15, 1)$ architecture, adding a momentum term to the gradient descent process. Twenty of these networks were trained with batch size 50, learning rate .01 and momentum factor 0.9. These networks, with the same architecture but different training regime, reached final validation loss below 10^{-4}. A contrast between the underfit networks trained with stochastic gradient descent and the successfully-trained networks follows.

We first observe the successfully-trained architectures' training path and final measures of topological generalization. The training behavior of neural networks trained with a $(3, 15, 15, 1)$ architecture using momentum, as well as the plot of the Betti numbers of the final decision boundary vs. loss, are pictured in Figure 35.

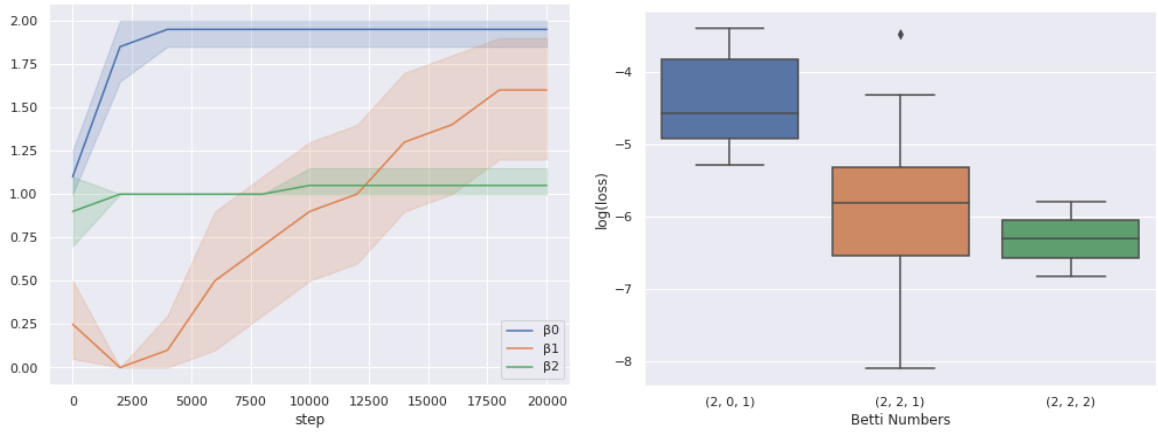


FIGURE 35. Left: The Betti numbers of the decision boundary of $(3, 15, 15, 1)$ neural networks during training on the Torus task using momentum. The width of the bar is a 95% confidence interval for the average β_i at that time step. Right: A boxplot of log(loss) by topology for the same task, late training.

We observe many of the same trends in convergence behavior of the average Betti numbers of the neural networks at the end of training: overall, the neural networks appear to converge towards the desired $\beta_0 = 2$, $\beta_1 = 2$ and $\beta_2 = 1$, with β_1 requiring the longest number of training steps to converge. The network training additionally appears biased to low values of β_1.

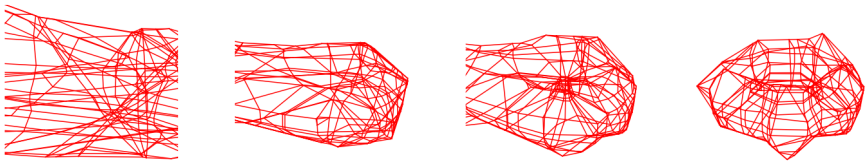


FIGURE 36. An example, and typical, training path of a neural network successfully trained on the torus.

At the end of training, only three different sets of topological invariants were observed, as pictured in the boxplot. The neural networks which exhibited the lowest loss typically had appropriate topological generalization, but there were some neural networks which exhibited $\beta_2 = 2$ insted of $\beta_2 = 1$; this corresponds

to an additional unbounded portion of the decision boundary, likely one which is far away from the bulk of the support of the density function. The $(2, 0, 1)$ signature corresponds to a decision boundary which has the same homology groups as a 2-sphere. Since the most common decision boundary at initialization is topologically a plane, we observe qualitatively that the general training path the decision boundary changes in topology from a noncompact plane, to a compact sphere, to a torus (by merging two sides of the sphere and "punching a hole" in it), as pictured in Figure 36.

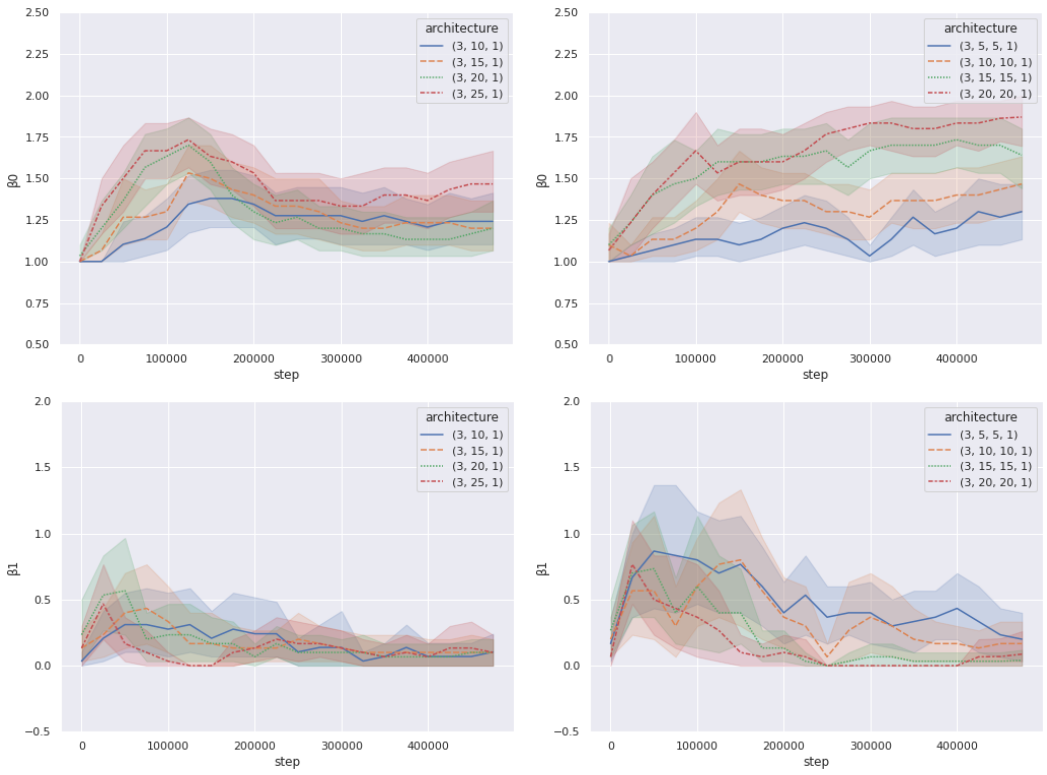


FIGURE 37. The Betti numbers of the decision boundaries of neural networks as they trained on the Torus task with pure SGD, separated by architecture.

In contrast, the neural networks which were underfit exhibit clear differences in their training behavior with respect to the Betti numbers of the decision boundary during training. In Figure 37, we observe that shallow networks appeared

to fail to approach, on average, the desired $\beta_0 = \beta_1 = 2$, but surprisingly, β_0 did exhibit a temporary average increase during the training regime. This phenomenon is not observed for deeper networks with respect to β_0, but is observed to some extent in the deep networks' failure to learn β_1.

Furthermore, while loss appeared to reach a constant value, the average β_0 appears to continue to be changing at the end of the training. It may be that the training was terminated at a gradient plateau, and that further training would lead to successful topological generalization. Lastly, the collapse in variability of β_1 could hypothetically correspond to the same phenomenon occurring much faster in the network trained with momentum which occurs around step 2500.

When we observe the Betti numbers of the decision boundaries at the end of training for the underfit networks, we see much greater variety in topological features, with the neural networks with the lowest loss within each architecture typically having the decision boundary topology of a sphere (Figure 38). Overall, the neural networks trained on this task with stochastic gradient descent failed to learn the appropriate β_1.

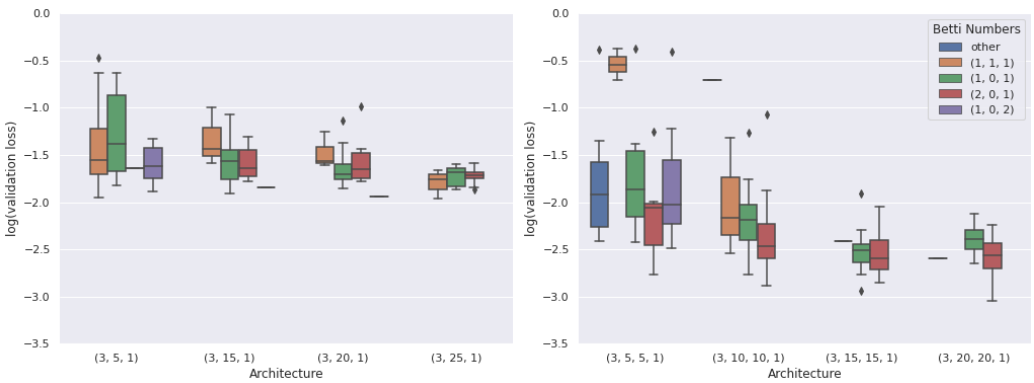


FIGURE 38. The loss of neural networks trained on the torus task, by final Betti numbers.

We hypothesize that another possible explanation for the failure of the neural networks trained with stochastic gradient descent is that there may be an implicit bias within stochastic gradient descent towards minimizing local absolute curvature of the underlying function. It is known that in low-dimensional settings, regularized stochastic gradient descent has an implicit bias towards interpolating discrete curvature [15]. The results observed here are consistent with this case for higher-dimensional inputs.

APPENDIX A

EXPLICIT WEIGHTS AND BIASES FOR COUNTEREXAMPLES

The explicit pair of counterexamples constructed in Theorem 3.3.5 were constructed by applying small perturbations around a single nongeneric neural network with weights,

$$W_1 = \begin{bmatrix} 0 & 1 \\ \frac{1}{2} & -\frac{\sqrt{3}}{2} \\ \frac{-1}{2} & \frac{-\sqrt{3}}{2} \\ 1 & 0 \end{bmatrix} \qquad b_1 = [-1, -1, -1, 5]$$

$$W_2 = [1, 1, 1, 0] \qquad b_2 = -0.1$$

The parameters of the first neural network model are given approximately by:

$$W_1 = \begin{bmatrix} 5.197 \cdot 10^{-3} & 1.004 \\ 4.984 \cdot 10^{-1} & -8.465 \cdot 10^{-1} \\ -4.849 \cdot 10^{-1} & -8.746 \cdot 10^{-1} \\ 1.006 & -7.160 \cdot 10^{-3} \end{bmatrix} \qquad b_1 = \begin{bmatrix} -1 & -1 & -1 & 5 \end{bmatrix}$$

$$W_2 = \begin{bmatrix} 9.972 \cdot 10^{-1} & 9.972 \cdot 10^{-1} & 9.972 \cdot 10^{-1} & -5.275 \cdot 10^{-2} \end{bmatrix} \qquad b_2 = -0.1$$

The parameters of the second neural network model are given approximately by:

$$W_1 = \begin{bmatrix} 5.301 \cdot 10^{-4} & 1.000 \\ 4.996 \cdot 10^{-1} & -8.604 \cdot 10^{-1} \\ -5.003 \cdot 10^{-1} & -8.612 \cdot 10^{-1} \\ 1.001 & 1.448 \cdot 10^{-3} \end{bmatrix} \qquad b_1 = \begin{bmatrix} -1 & -1 & -1 & 5 \end{bmatrix}$$

$$W_2 = \begin{bmatrix} 1 & 1 & 1 & -0.5 \end{bmatrix} \qquad b_2 = -0.1$$

APPENDIX B

IMPLEMENTATIONS OF ALGORITHMS

The code developed for this dissertation is available in the zipped file in the suppl ementary materials. This zipped file is formatted as a repository, and contains the code for the algorithms described in Sections 3.6 in Sage, as well as code for the algorithms in Sections 4.1 and 4.3 in Python.

APPENDIX C

LICENSE INFORMATION

PyTorch [24] is under a Modified BSD license, permitting use in other projects and requiring its licensing information repackaged when its source code is redistributed. We do not redistribute its source code in our work.

Sage [27] is licensed under the GNU General Public License (GPL). It is free to use and distribute. We do not redistribute its source code in our work, but it is necessary to run the decision boundary topology computations.

REFERENCES CITED

[1] M. Aguiar and S. Mahajan. *Topics in Hyperplane Arrangements*. American Mathematical Society, Providence, RI, 2017.

[2] M. Alfarra, A. Bibi, H. Hammoud, M. Gaafar, and B. Ghanem. On the decision boundaries of deep neural networks: A tropical geometry perspective. *CoRR*, abs/2002.08838, 2020.

[3] B. Anders, M. Las Vergnas, B. Sturmfels, N. White, and G. M. Ziegler. *Oriented Matroids (Encyclopedia of Mathematics and its Applications, Series Number 46)*. Cambridge University Press, paperback edition, 1 2000.

[4] R. Balestriero, R. Cosentino, B. Aazhang, and R. Baraniuk. The geometry of deep networks: Power diagram subdivision. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[5] M. Bianchini and F. 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, Aug. 2014.

[6] C. Curto, A. Veliz-Cuba, and N. Youngs. *Analysis of Combinatorial Neural Codes: An Algebraic Approach*, pages 213–240. 01 2019.

[7] A. G. de G. Matthews, J. Hron, M. Rowland, R. E. Turner, and Z. Ghahramani. Gaussian process behaviour in wide deep neural networks. In *International Conference on Learning Representations*, 2018.

[8] A. Fawzi, S.-M. Moosavi-Dezfooli, P. Frossard, and S. Soatto. Empirical study of the topology and geometry of deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[9] C. Fefferman, S. Mitter, and H. Narayanan. Testing the manifold hypothesis. *Journal of the American Mathematical Society*, 29(4):983–1049, feb 2016.

[10] J. E. Grigsby and K. Lindsey. On transversality of bent hyperplane arrangements and the topological expressiveness of relu neural networks, 2020. Available at `https://arxiv.org/abs/2008.09052`.

[11] J. E. Grigsby, K. Lindsey, and M. Masden. Local and global topological complexity measures of relu neural network functions. arXiv, 2022.

[12] R. Grunert. *Piecewise Linear Morse Theory*. PhD thesis, 2017.

[13] V. Guillemin, V. Guillemin, A. Pollack, V. GUILLERMIN, and P. Alan. *Differential Topology*. Mathematics Series. Prentice-Hall, 1974.

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

[15] B. Hanin. Ridgeless interpolation with shallow relu networks in $1d$ is nearest neighbor curvature extrapolation and provably generalizes on lipschitz functions. *arXiv preprint arXiv:2109.12960*, 2021.

[16] B. Hanin and D. Rolnick. Complexity of linear regions in deep networks. *ArXiv*, abs/1901.09021, 2019.

[17] B. Hanin and D. Rolnick. Deep relu networks have surprisingly few activation patterns. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[18] V. Itskov, A. Kunin, and Z. Rosen. Hyperplane neural codes and the polar complex. In N. A. Baas, G. E. Carlsson, G. Quick, M. Szymik, and M. Thaule, editors, *Topological Data Analysis*, pages 343–369, Cham, 2020. Springer International Publishing.

[19] M. Jordan, J. Lewis, and A. G. Dimakis. Provable certificates for adversarial examples: Fitting a ball in the union of polytopes. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[20] N. Lei, D. An, Y. Guo, K. Su, S. Liu, Z. Luo, S.-T. Yau, and X. Gu. A geometric understanding of deep learning. *Engineering*, 6(3):361–374, mar 2020.

[21] W. Li, G. Dasarathy, K. Natesan Ramamurthy, and V. Berisha. Finding the homology of decision boundaries with active learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 8355–8365. Curran Associates, Inc., 2020.

[22] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio. On the number of linear regions of deep neural networks. *Advances in neural information processing systems*, 27, 2014.

[23] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 807–814, Madison, WI, USA, 2010. Omnipress.

[24] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[25] T. Serra and S. Ramalingam. Empirical bounds on linear regions of deep rectifier networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5628–5635, 2020.

[26] T. Serra, C. Tjandraatmadja, and S. Ramalingam. Bounding and counting linear regions of deep neural networks. In *ICML*, 2018.

[27] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 9.0)*, 2018. `https://www.sagemath.org`.

[28] H. Xiong, L. Huang, M. Yu, L. Liu, F. Zhu, and L. Shao. On the number of linear regions of convolutional neural networks. In *International Conference on Machine Learning*, pages 10514–10523. PMLR, 2020.

[29] X. Yang, H.-D. Tran, W. Xiang, and T. Johnson. Reachability analysis for feed-forward neural networks using face lattices. 2020.

[30] X. Yang, T. Yamaguchi, H.-D. Tran, B. Hoxha, T. T. Johnson, and D. Prokhorov. Reachability analysis of convolutional neural networks. 2021.

[31] L. Zhang, G. Naitzat, and L.-H. Lim. Tropical geometry of deep neural networks. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5824–5832. PMLR, 10–15 Jul 2018.

[32] X. Zhang and D. Wu. Empirical studies on the properties of linear regions in deep neural networks. In *International Conference on Learning Representations*, 2020.