University of California

Santa Barbara

Universal Approximation for Neural Nets on Sets

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy

in

Mathematics

by

Christian Bueno

September 2021

The dissertation of Christian Bueno is approved.

Professor Mihai Putinar

Professor William Wang

Professor John R. Gilbert

Professor Paul J. Atzberger, Chair

September 2021

Universal Approximation for Neural Nets on Sets

A mis padres, mi hermano y mis abuelos.

Acknowledgements

I'd like to first thank my advisor and committee members for their time, patience, guidance, and support in this process as well as the many other professors who have taught me so much. In particular, I would like to thank the following grants and support from Professor Atzberger which included funding from the grants DOE Grant ASCR PHILMS DE-SC0019246 and NSF Grant DMS-1616353.

I'd also like to thank NASA GRC and SCaN for their support and for giving me the unique opportunity to work on many fascinating projects I wouldn't have been able to work on otherwise. I'd specifically like to thank Alan G. Hylton for the being an excellent mentor, collaborator, and friend during my time at NASA and beyond.

Thanks to the many great friends that I've made over the years. Spending time with you all has been the greatest part of this journey. Arturo, thank you for helping me navigate the world of deep learning and for pointing me to the paper on *Deep Sets*. It wasn't all champagne and caviar, so your help and encouragement along the way was truly invaluable. Garo, thanks for always making time to sanity check my math, chat, and travel to far off countries. Your positivity and discipline are something I continuously look up to. Steve, thanks for being awesome and a great help in understanding spaces of finite subsets and basically anything topological. Joey, thank you for entertaining endless math and physics discussion and for sharing your great differential geometry intuition. Nadir and Massy, thanks for your friendship, the laughs, and amazing discussions. Monika, thanks for your love and endless support during the hardest times of my studies.

Thanks to my wonderful large extended family which have always had my back. Thanks to my brother Alex who reminded me, that like a drill, we advance little by little with each turn and eventually pierce through. Thanks to my late abuelito Jesús who would challenge me to large computations and debates as a child. And lastly, thanks to my incredible loving parents, without whom none of this would have been possible.

Christian Bueno

University of California (814) 232-4089
Santa Barbara, CA 93106-3080 christianbueno@ucsb.edu

EDUCATION

UC Santa Barbara— Santa Barbara, CA Fall 2014 - Present
Ph.D. in Mathematics (Expected: September 2021).

Florida International University— Miami, Florida 2007 - 2013
Master's of Science in Mathematics in 2013.
B.S. in Physics & B.S. in Mathematics in 2011.

PUBLICATIONS

Christian Bueno, Kristina Collins, Alan G. Hylton, Robert Short. "A Geometric Approach to the Kinematics of the Canfield Joint." 2021. *arXiv*: 2105.05955 [cs.RO].

Christian Bueno and Alan G. Hylton. "On the Representation Power of Set Pooling Networks." 2021. (Under review).

WORK & RESEARCH EXPERIENCES

PhD Research 2016-Present
Advised by Dr. Paul J. Atzberger
◦ Established new universality and impossibility results for recent methods of deep learning on sets and point clouds.
◦ Developed a manifold learning pipeline (via Diffusion Maps) for the for the creation of low-dimensional surrogate models for high-dimensional SDEs.

Teaching at UCSB 2014-2020
◦ Teaching Assistant for classes ranging from Single and Multivariate Calculus, Differential Equations, Linear Algebra, and Mathematical Proofs. Taught Calculus as instructor of record.
◦ Mentored 3 math undergrads on a 3D deep learning project for point clouds and meshes for the Directed Reading Program (DRP).

GRIPS Internship in Machine Learning & Fraud Detection Summer 2019
Jointly ran by IPAM, Freie Universität and Deloitte
Summer Research Internship in Berlin
◦ Used techniques from data science and machine learning to develop and evaluate fraud detection methods that would be robust to changing fraud strategies.

NASA Glenn Research Center Summers 2016, 2017, 2018
Space Communications and Navigation (SCaN)
Applied Math Summer Internships
○ Investigated data-driven approaches to star tracking via TDA and independent development of a permutation-invariant neural network for this task.
○ Solved unresolved inverse kinematic problem for failure mode of the Canfield joint (CJ), a novel robotic manipulator for a future Mars communication satellite.
○ Discovered novel singularities of the CJ via computational algebraic geometry.

Civis Analytics Summer 2014
Data Science Internship
○ Utilized data sets ranging from the census and polls to internal data and client data. Then used SQL, Python, and R to clean, analyze, and model the data.

Analytical Flavor Systems 2013
Lead Data Scientist
○ Developed recommendation engine, data cleaning algorithms, as well as statistical modeling of flavor perception in Matlab and R.

REU in Representation Theory of String Links Summer 2011
Math/CS Department at Ohio Wesleyan University
Under Dr. Craig Jackson - Funded by NSF
○ Built evidence for the conjectured equivalence of the Lin and skein representations on string links and showed a novel way to compute the Alexander polynomial of knots conditional on the conjecture being true.

Undergraduate Research on Pebble Motion Problems Summer 2010
Math Department at Florida International University
Under Dr. Miroslav Yotov - Funded by FIU McNair program
○ Developed a group-theoretic invariant for pebble motion problems and proved various properties such as every finite abelian group being achievable.

Undergraduate Research in Quantum Optics Fall 2009 - Spring 2010
Physics Department, Florida International University
Under Dr. Yifu Zhu - Funded by NSF
○ Implemented Runge-Kutta to solve a system of 16 coupled complex ordinary differential equations and assisted in the laboratory with laser calibration.

PRESENTATIONS

○ *IEEE Cognitive Communications for Aerospace Applications Workshop (June 2021)*: Cardinality-Agnostic Universal Approximation for Neural Networks on Point Clouds.
○ *MIT (March 2021)*: Universal Approximation on Sets.
○ *MIT Center for Brains, Minds, and Machines (Sept 2020)*: Nonlinear Dimensionality Reduction.

○ *NASA Glenn Research Center (June 2020)*: An Introduction to Artificial Neural Networks.
○ *NeurIPS Workshop on Sets and Partitions (Dec 2019):* Limitations of Deep Learning on Point Clouds. Accepted as contributed talk, paper and poster.
○ *SIAM Algebraic Geometry (July 2019):* The Configuration Space and Kinematics of the Canfield Joint *(poster)*.
○ *SIAM @ UCSB (Feb 2019):* Clustering and the Mean Shift Algorithm.
○ *SIAM @ UCSB (Nov 2018):* Kernel Methods and Unsupervised Learning.
○ *Discrete Geometry Seminar @ UCSB (Oct 2018):* Hocus Locus - The Algebraic Geometry of the Canfield Joint.
○ *Meshfree and Particle Methods Workshop (2018):* Meshfree Manifold Learning Methods for Dimension Reduction of Stochastic Dynamical Systems *(poster presentation)*.
○ *Southern California Applied Mathematics (2018):* Manifold Learning for Data-Driven Dimension Reduction of Stochastic Systems *(poster presentation)*.
○ *SIAM @ UCSB (Feb 2018):* Neural Networks and the Universal Approximation Theorem.
○ *Discrete Geometry Seminar (Feb 2018):* The Configuration Space of the Canfield Joint.
○ *SIAM @ UCSB (Jan 2018):* A Topological Vector Space Safari.
○ *Southern California Applied Mathematics Symposium (2017):* Methods for Data-Driven Dimension Reduction of Stochastic Systems using Diffusion Maps *(poster presentation)*.
○ *NASA Glenn Research Center (2017):* Configuration Space and Singularities of the Canfield Joint.
○ *NASA Glenn Research Center (2017):* An Introduction to Diffusion Maps with Applications to Stochastic Dynamics.
○ *Discrete Geometry Seminar (Jun 2017):* Spectral Graph Drawing and Manifold Learning.
○ *SIAM @ UCSB (Feb 2017):* Topological Data Analysis.
○ *SIAM @ UCSB (Oct 2016):* Algebraic Geometry, Groebner Bases, and Robots.
○ *NASA Glenn Research Center (2016):* Introduction to R and Topological Data Analysis.
○ *Applied Topology Seminar (May 2016)*: Computing Persistent Homology Pt. II.
○ *Applied Topology Seminar (Apr 2016)*: Introduction to Topological Data Analysis.
○ *Graduate Algebra Seminar (May 2015):* Tropical Varieties of Amoebas and Other Tropical Adventures.

Awards

○ NASA internship award ($2,000) to attend Space Generations Congress in Bremen, Germany (Summer 2018).
○ 2^{nd} Place Research Paper amongst McNair Fellows (Fall 2010).
○ McNair Fellowship (Spring 2010)
○ Florida Bright Futures Scholarship (2007-2011)

Abstract

Universal Approximation for Neural Nets on Sets

by

Christian Bueno

Point clouds and sets are ubiquitous, unusual and unstructured data-types which present unique problems to machine learning when used as inputs. Since sets are inherently unaffected by permutations, the input space for these problems naturally forms a non-Euclidean space which is oftentimes not even a manifold. Moreover, similar inputs can have wildly varying cardinalities and so the input space is in general infinite-dimensional. Despite these mathematical difficulties, *PointNet* [40] and *Deep Sets* [62] form two foundational contributions for deep learning in this area. In this thesis we study the expressive power of such networks. To that end, we prove theorems about their Lipschitz properties, extend them to well-studied infinite-dimensional spaces, and prove new cardinality-agnostic universality results for point clouds. These results completely characterize the approximable functions and so can be used to compare the representational strength of the underlying model classes. In particular, a normalized version of the DeepSets architecture cannot uniformly approximate the diameter function but can uniformly approximate the center-of-mass function whereas PointNet can uniformly approximate the former but not the latter. Additionally, even when limited to a fixed input cardinality, PointNet cannot uniformly approximate the average value of a continuous function over sets of more than two points. We additionally obtain explicit error lower-bounds for this error of approximation and a present a simple geometric method to produce arbitrarily many examples of this failure-mode. Along the way, we also prove various general purpose universal approximation theorems, one of which is for generalized neural networks whose input space is a set of unknown or nonexistent topology.

Contents

List of Figures

List of Tables

Chapter 1

Overview

The architectures introduced in *PointNet* [40] and *Deep Sets* [62] are foundational contributions to the direct analysis of point clouds and sets via deep learning. These works provided two of the earliest such methods and continue to be of theoretical interest since both provide architectures which are inherently *permutation-invariant*. In addition to this, a single invariant model of either approach can handle sets of differing sizes and can theoretically scale to any cardinality with enough computational resources. These methods achieve this through careful application of pooling and weight sharing with their only difference being their choice of pooling function.

In either case the procedure to process a point cloud A in $\mathbb{R}^n$ is simply as follows: (1) apply a network $\boldsymbol{\varphi}$ to each element of A, (2) apply a permutation-invariant pooling operation to aggregate these point-features into a global feature for A (e.g. max-pooling for PointNet, sum-pooling for DeepSets), and lastly (3) pass this global feature to the second network $\boldsymbol{\rho}$ to obtain the final output. For general PointNet and DeepSets permutation-invariant models, this can be concisely expressed mathematically as

$$\boldsymbol{\psi}_{PointNet}(A) = \boldsymbol{\rho}\left(\max_{\boldsymbol{a}\in A} \boldsymbol{\varphi}(\boldsymbol{a})\right), \quad \boldsymbol{\psi}_{DeepSets}(A) = \boldsymbol{\rho}\left(\sum_{\boldsymbol{a}\in A} \boldsymbol{\varphi}(\boldsymbol{a})\right),$$

respectively. Depending on the output layer of $\boldsymbol{\rho}$ this can be used for either set-regression or set-classification. Note that arbitrarily large point clouds can be passed through and that the permutation-invariance of the max/sum-pooling operations ensure that rearrangement of the elements of A do not alter the output.

Importantly, each of these prior works provide their own universal approximation theorems (UATs) to support the empirical success of their methods. In this thesis we will prove substantial extensions of these results motivated by the following points:

Unified Approach and Minimal Universality: The pre-existing universality results use very different approaches. Additionally, these results do not explore the minimum architecture requirements needed for universality. In this thesis we present a proof method that applies to more than one model and provides universal "shallow" examples.

Cardinality Limitations: The universality results in [40, 62] both assume that the cardinality of the inputs are fixed to some size n and thus do not make use of the cardinality-agnostic nature of these models. This gap is of worthy of investigation because (1) real-world point cloud datasets can have heterogeneous cloud sizes and (2) it may happen that the deployed model encounters point clouds with cardinalities that did not exist in the training data (e.g. if better sensors become available). It is not immediately clear whether one should expect these model classes to have enough power to universally approximate the functions of interest when allowing for such changes in set size.

Infinite Input-Width Limits: One approach for learning with mesh inputs is to sample the mesh and feed the resulting point cloud into a suitable neural network [40, 41, 20]. Although PointNet and DeepSets can readily accept samples of any size and via any sampling method, the computation graph at the input layers necessarily widens as the point clouds get larger. Understanding the expressiveness and consistency as sampling cardinality (and hence input layer size) grows to infinity may provide theoretical insight on this approach to learning from meshes. Models such as Neural ODEs [10] and Neural Tangent Kernels [28] have benefited from similar considerations (infinite depth and infinite hidden-layer width limits respectively).

DeepSets Extension Conjecture: It was conjectured in the supplementary materials of [62] (below the proof Theorem 9) that the DeepSets invariant model should be extendable to input sets of countably infinite or even unaccountably infinite size and retain universality in that setting. However, they note that there are fundamental topological obstructions to answering this question. We provide partial resolutions to this conjecture by showing that a change in the pooling function allows for inputs beyond finite sets (Section 7.3).

Comparison of Representational Power: In this work we slightly modify the permutation-invariant DeepSets model by use of average-pooling instead of sum-pooling. The only change is re-scaling by the set size before applying $\boldsymbol{\rho}$ so we call this version **normalized-DeepSets**. We find that:

- PointNet (normalized-DeepSets) can uniformly approximate only the functions which are uniformly continuous with respect to the Hausdorff (Wasserstein) metric (Theorem 7.2).
- Only the constant functions can be simultaneously uniformly approximated by both PointNet and normalized-DeepSets when input sets are allowed to be arbitrarily large (Theorem 7.4).
- Even when cardinality is fixed to size k there is a substantial difference in approximation power. In particular, PointNet cannot uniformly approximate averages over sets such as the center-of-mass. We prove an explicit error lower bound for these tasks for PointNet and provide simple method for generating examples of this failure-mode (Theorem 7.5).

Along the way we also prove generalizations of the classical UAT for compact Hausdorff spaces, metric spaces, and in the most general setting, plain sets. To do this we expand our concept of what a neural network is so as to adapt to the non-Euclidean and even non-topological setting. These results then help us prove the point cloud related results and also helps us understand why we could not obtain similar results for the un-normalized version of DeepSets.

Related Work. Besides the base invariant Pointnet model introduced in [40], the authors also include a variant for point cloud segmentation, a subnetwork to learn alignment, and later introduce a hierarchical variant of PointNet in [41]. The authors of [62] also introduce a distinct permutation-*equivariant* model where permutation of the inputs results in a corresponding permutation of the output. Though such equivariant networks are also commonly referred to as DeepSets networks, they are fundamentally different and are generally not permutation-*invariant*. We will always refer to the invariant models in the context of this thesis and thus we do not consider the equivariant case here. Besides these models see [20] for a comprehensive survey of deep learning methods for point clouds.

There has also been much activity regarding the representational power of permutation-invariant models. Whether permutation-invariant functions can be *exactly* represented by sum-decomposition – i.e. as $\rho(\sum_{a \in A} \varphi(a))$ – has been addressed by [62, 54, 60] with positive and negative results depending on whether the point clouds come from a countable or uncountable universe. In particular, [62] proves that sum-decomposition is possible when the universe is countable and [54] shows that there are continuous permutation-invariant functions on $\mathbb{R}$ that are not sum-decomposable when the latent space is too small. Additionally, [60] considers similar questions but in the context of graph neural networks and also assesses the relative ability of max-pooling, average-pooling, and sum-pooling to distinguish multisets. Regarding approximation of permutation-invariant functions [61] also proves a universal approximation result for fixed cardinality and also establishes a universality results for general G-invariant functions by appealing to Stone-Weierstrass and Hilbert's finiteness theorems from classical invariant theory.

Much of the work in this thesis is an evolution and expansion of the work communicated in a workshop paper [8] and oral presentation for the *NeurIPS 2019 Sets & Partitions* workshop. A later version of this work also appears on OpenReview as a submission for ICLR 2020 [7] which was ultimately not accepted. Some of the ideas and results which do no appear in those earlier manuscripts include the various UATs for metric spaces and the general UAT for sets in Chapter 5. The Lipschitz properties proven in Theorem 6.4 are also new. Additionally, the error lower bound in Theorem 7.5 of those earlier manuscripts was substantially generalized beyond just the center-of-mass and now extends to an error lower bound for averages of continuous functions.

Structure of Thesis. Chapter 2 focuses on neural networks and contains a brief history of neural networks, introduces fundamental concepts in deep learning, illustrates the importance of the universal approximation theorem, and sets the context for the rest of the thesis. The concepts therein should be accessible to a wide audience and does not assume mathematics beyond linear algebra and multivariate calculus.

In Chapter 3 we quickly review higher level mathematics such as point-set topology, metric spaces, functional analysis, and measure theory. The definitions, conventions, and notations introduced in that chapter will be heavily used throughout the remainder of the thesis but is fairly standard graduate level mathematics.

In Chapter 4 we introduce various ways to topologize and metrize the set of finite subsets on a topological/metric space. In particular, there we introduce the Hausdorff metric d_H and Wasserstein metric d_W as well as the metric spaces $(\mathcal{K}(\Omega), d_H)$ and $(\mathcal{P}(\Omega), d_W)$ and many of their most important properties.

In Chapter 5 we introduce notation to generalize neural networks to novel settings and notation to compactly describe the families of functions which they form. These notations will be used in later chapters but we also immediately use those concepts to prove a variety of novel generalizations of the classical UAT culminating in a UAT for input spaces without a prescribed topology (Theorem 5.6). This and the remaining chapters all contain novel contributions.

In Chapter 6 we finally begin to look closely at the neural network models of the type introduced in *PointNet* and *Deep Sets* which we call *set pooling networks*. After spending some pages setting up the necessary mathematical groundwork, we then prove a variety of results on the mathematical properties of these models as well as properties for continuous extension of those models.

Finally, in Chapter 7 we prove results on what kinds of functions these set pooling networks can express, with special attention to what kinds of functions they can uniformly approximate. There we will heavily draw upon the preceding chapters to prove the UATs that lie therein. In there we also directly compare the approximation capabilities of PointNet and normalized-DeepSet and present an impossibility theorem for PointNet which is encapsulated quantitatively in error lower bound theorem Theorem 7.5. Lastly, we reinforce this latter result via a direct numerical experiment, empirically validating the theory.

Chapter 2

Neural Network Preliminaries

Over the past decades, the range of applications for neural networks has exploded, and along with this success has come an explosion of diversity in neural network architectures. Though the trend has generally been towards greater complexity, many of the great advances have come from careful consideration of how the network and its neurons are organized.

In support of better understanding the neural networks of interest in this thesis, this chapter will review some of the foundations of deep learning and its history. We will only assume linear algebra, multivariable calculus, and elementary probability in this portion.

2.1 Perceptrons and XOR

In the middle of the 20th century, efforts to mathematically model biological neurons birthed the ancestors of modern-day artificial neural networks. The first notable fruit of this effort to formalize the computational capabilities of the biological neuron came in 1943 with the Boolean logic based McCulloch-Pitts model [33]. This was later generalized by Frank Rosenblatt in 1957 to allow for continuous inputs and introduced tunable weights in what he called the *perceptron* [44]. This line of research ultimately culminated in what we now call an *artificial neuron*, which as a function can be simply expressed as

$$f(\boldsymbol{x}) = \sigma(\boldsymbol{w} \cdot \boldsymbol{x} + b),$$

where $\boldsymbol{w} \cdot \boldsymbol{x} = \sum_{i=1}^{n} w_i x_i$ and $\sigma : \mathbb{R} \to \mathbb{R}$ is a univariate (nonlinear) function called the *activation function* or just simply the *activation*.[1] The vector $\boldsymbol{w} \in \mathbb{R}^n$ we call the *weight vector* or *weights* and the scalar $b \in \mathbb{R}$ we call the *bias*. Though somewhat confusing, it is common in machine learning to refer to bias as one of the weights as well.[2]

Today, artificial neurons are recognized to be inaccurate models of their biological counterparts for various reasons, but the name has nevertheless stuck. In fact, when the context is clear, the adjective "artificial" is often even dropped in the machine learning literature. Likewise, we will also frequently refer to artificial neurons as simply *neurons* and instead use the adjective "biological" if we need to refer the neuroscientific counterpart.

From this modern perspective, Rosenblatt's perceptron is a special case of the artificial neuron. Its main use-case was to perform binary classification of Euclidean data[3] by outputting a "0" for examples belonging to one class and a "1" for examples belonging to the alternate class. Mathematically, an n-input perceptron can be expressed as

$$f(\boldsymbol{x}) = \sigma_H\Big(\boldsymbol{w} \cdot \boldsymbol{x} - b\Big),$$

where the activation function σ_H is the Heaviside step function

$$\sigma_H(x) = \begin{cases} 1 & \text{if } x > 0, \\ 0 & \text{if } x \leq 0. \end{cases}$$

[1]Other names which are sometimes used include *threshold function*, and *transfer function*.

[2]This language can be made consistent by letting $f(\boldsymbol{x}) = \sigma(\widetilde{\boldsymbol{w}} \cdot \widetilde{\boldsymbol{x}})$ where $\widetilde{\boldsymbol{x}} = (x_1, \ldots, x_n, 1)$ and $\widetilde{\boldsymbol{w}} = (w_1, \ldots, w_n, b)$.

[3]By Euclidean data, we mean data that can be encoded as equidimensional vectors of real numbers, i.e. data that can be interpreted as elements of $\mathbb{R}^n$ for some n.

Unlike the earlier McCulloch-Pitts model, Rosenblatt's perceptron could be automatically tuned through an interactive learning process called *training* which improves the model by changing the weights. The first training algorithm for the perceptron was inspired by the biological theory of Hebbian learning [23] and was introduced by Rosenblatt himself, but there have since been other approaches that were developed later on [57, 16].

In the time of its introduction, there was a lot of excitement and optimism regarding Rosenblatt's perceptron. Pretty quickly a physical incarnation of the perceptron was realized and this was written in the New York Times [37]:

> The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence. Later Perceptrons will be able to recognize people and call out their names and instantly translate speech in one language to speech and writing in another language, it was predicted.

This level of optimism soon came to be seen as overzealous. In 1969, Marvin Minsky and Seymour Papert published a book titled *Perceptrons: An Introduction to Computational Geometry* in which they demonstrated various limitations of the perceptron model. A key point within *Perceptrons* was that Rosenblatt's perceptrons could only learn to correctly classify *linearly separable* data (i.e. data for which the two classes can be separated by a hyperplane). To see why, we can rewrite the perceptron like so,

$$f(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{w} \cdot \boldsymbol{x} > b, \\ 0 & \text{if } \boldsymbol{w} \cdot \boldsymbol{x} \leq b. \end{cases}$$

From here we can easily see that regardless of the values of weights, the set of points classified as "1" lie in the open half-plane given by $\boldsymbol{w} \cdot \boldsymbol{x} > b$ and those classified as "0" lie in the closed half-plane given by $\boldsymbol{w} \cdot \boldsymbol{x} \leq b$. Thus, it is a prerequisite for perceptron-learnability that the dataset's two classes be possible to separate via a hyperplane, otherwise, mistakes must be tolerated.

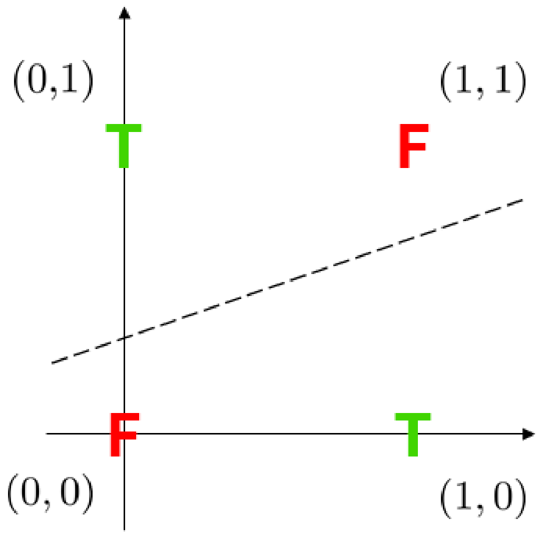

Figure 2.1: The "True" and "False" values of XOR cannot be linearly separated.

In particular, this observation meant that even a simple logical operation such as XOR (exclusive-or) could not be simulated by a perceptron. For such an important and basic logical operation to pose a problem was certainly bad news. The impossibility of this task can be easily seen geometrically by representing XOR's inputs as ordered pairs in the plane given by (0,0), (1,1), (1,0), and (0,1) as in Fig. 2.1. From this plot, it is visually clear that there is no line which separates the 0-class $C_0 = \{(0,0),(1,1)\}$ from the 1-class $C_1 = \{(1,0),(0,1)\}$. More formally, if there was such a separating line, then the line segment L_0 formed by C_0 and the line segment L_1 formed by C_1 would lie in opposite half-planes. However, this is impossible since these two line segments intersect at a common point $(\frac{1}{2}, \frac{1}{2})$. So we must conclude no such separating line exists, and hence no perceptron can simulate the XOR function.

x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0

Table 2.1: Truth table for the XOR operation.

It is worth noting that if we replace the step function in Rosenblatt's perceptron with a different function, it then becomes possible to resolve the XOR problem. For example, if we use a modified activation function $\sigma_M(x) = \sigma_H(x) - \sigma_H(x-1)$ so that

$$\sigma_M(x) = \begin{cases} 1 & \text{if } 0 < x \leq 1, \\ 0 & \text{otherwise,} \end{cases}$$

then it is possible to solve the XOR problem using weights $\boldsymbol{w} = (1, 1)$ and bias $b = 0$.

x_1	x_2	$f(\boldsymbol{x}) = \sigma_M(x_1 + x_2)$
0	0	0
0	1	1
1	0	1
1	1	0

Table 2.2: Modified perceptron easily solves the XOR problem.

It is not too surprising that by allowing for a broader family of activation functions, the capabilities of the perceptron also broadens. Naturally then, one would wonder if this is enough additional flexibility to allow us to represent any function we wish, even if just approximately. Unfortunately, regardless of what activation function σ we use, the function represented by the neuron $f(\boldsymbol{x}) = \sigma(\boldsymbol{w} \cdot \boldsymbol{x} + b)$ will always be constant on the hyperplanes $\boldsymbol{w} \cdot \boldsymbol{x} + b = c$. So even generalizing Rosenblatt's perceptrons to general artificial neurons will not allow us to represent even a simple rotationally symmetric bivariate function such as $f(\boldsymbol{x}) = x_1^2 + x_2^2$.

The observations in *Perceptrons* were a serious blow which, by some accounts, directly led to a period of reduced research activity in AI known as the "AI Winter." It would take some time for research activity to pick up again and for the neural network approach to regain favor. We will keep in mind this important piece of AI history as we explore the properties of more complex neural network models.

2.2 Feed-Forward Neural Networks and Universality

Using artificial neurons as a building blocks, we can create a dizzying diversity of more complex models which are collectively known as *neural networks*. This is done by having the outputs of an individual neuron used as an input for other neurons. If the flow of data in such a network of neurons never forms a cycle[4] then we can think of the information as strictly flowing forward from start of the network (inputs) to the end of the network (outputs). Such a network is aptly named a *feed-forward neural network*.

A special and important case of such networks involves arranging the neurons into groups of neurons known as *layers* in such a way that the outputs of the i-th layer directly feed into the $(i+1)$-th layer and so on.[5] Such networks are often called *multi-layer perceptrons* (MLPs), though this is a bit of a misnomer as they are generally comprised of neurons with continuous activations and not perceptrons (which have Heaviside activation).

The layers of an MLP come in three flavors: input, hidden, and output layers. The input layer is comprised of just the unaltered input values, the output layer spits out the outputs, and the hidden layers are all the layers in between. Every layer can be thought of as the composition of an affine function and followed by an activation function[6] (see Fig. 2.2). We say a layer is *fully-connected* if every neuron in that layer is connected to every neuron in the preceding layer. If all the (non-input) layers of an MLP are fully-connected, we say it is a fully-connected MLP, or just simply a *fully-connected network*.

More formally, an MLP with h hidden-layers is a function $f : \mathbb{R}^{d_{in}} \to \mathbb{R}^{d_{out}}$ which can be expressed as an alternating composition of affine transformations and (nonlinear)

[4]Formally we mean that the neural network's computation graph forms a directed acyclic graph (DAG). In particular, no neuron can feed into itself if the computation graph is a DAG.

[5]Networks such as ResNet [22] which connect non-consecutive layers with so-called "skip-connections" are also considered feed-forward neural networks, but we will not consider them to be MLPs.

[6]The input layer can be thought of as the identify function followed by the identity activation

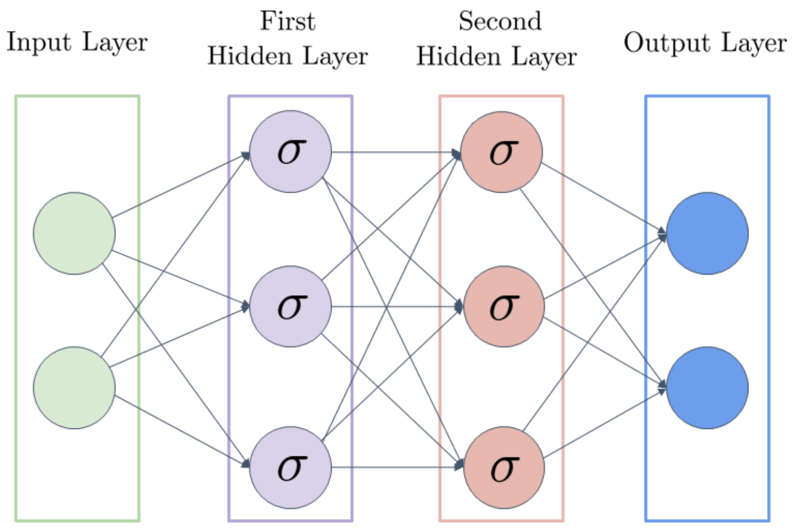

Figure 2.2: A fully-connected MLP with two hidden layers using activation functions σ and linear output layer.

activation functions:

$$\begin{aligned} \boldsymbol{y}_0 &= \boldsymbol{x}, & \boldsymbol{x} &\in \mathbb{R}^{d_{in}}, \\ \boldsymbol{y}_{i+1} &= \boldsymbol{\sigma}_i(W_i \boldsymbol{y}_i + \boldsymbol{b}_i), & \boldsymbol{y}_i &\in \mathbb{R}^{d_i}, \\ f(\boldsymbol{x}) &= \boldsymbol{y}_{h+1}, & \boldsymbol{y}_{h+1} &\in \mathbb{R}^{d_{out}}, \end{aligned}$$

where $W_i \in \mathbb{R}^{d_{i+1} \times d_i}$ are matrices, $\boldsymbol{b}_i \in \mathbb{R}^{d_i}$ are the bias vectors, and the vector activation $\boldsymbol{\sigma}_i$ just applies the univariate activation $\sigma_i : \mathbb{R} \to \mathbb{R}$ to each component. Oftentimes when discussing approximation theory a linear output layer is used, i.e. final activation is the identity $\sigma_h(\boldsymbol{x}) = \boldsymbol{x}$.

Despite the additional complexity and expressiveness that MLPs afford, XOR's spectre provokes a doubt – are MLPs sufficiently general to express all the functions we care about?

Fortunately, in the late 1980's and early 1990's, various authors approached this questions and provided affirmative answers which are now collectively called the *universal approximation theorem* (UAT). The focus of many of these investigations was on whether or not continuous functions could be uniformly approximated with single hidden-layer

MLPs, more specifically via linear combinations of neurons such as

$$f(\boldsymbol{x}) = \sum_{i=1}^{d} a_i \sigma(\boldsymbol{w}_i \cdot \boldsymbol{x} + b_i).$$

Notably, in the year 1989, four papers came out on this topic [9, 11, 15, 26] which proved UATs under different assumptions on the properties of the activation function σ. In the ensuing years more work was done to further clarify the picture. The interested reader can find a comprehensive overview of the work done in this period in [39]. Ultimately, the answer as to which kinds of continuous activation functions lead to universal approximation was quite simple: any non-polynomial continuous activation function will work. This was proven in [32] by Leshno et. al. in 1993, and in fact, they even addressed the possibilities for discontinuous activation functions as well. We state a version of the continuous case of their UAT here.

Theorem 2.1 (Leshno et al.). *Let $\Omega \subseteq \mathbb{R}^d$ be compact and $\sigma : \mathbb{R} \to \mathbb{R}$ continuous. Then for any continuous function $F : \mathbb{R}^d \to \mathbb{R}$ and any $\epsilon > 0$, there exists an integer $n \geq 1$, weights $a_i, b_i \in \mathbb{R}$ and $\boldsymbol{w}_i \in \mathbb{R}^d$ so that*

$$\left| F(\boldsymbol{x}) - \sum_{i=1}^{n} a_i \sigma(\boldsymbol{w}_i \cdot \boldsymbol{x} + b_i) \right| < \epsilon$$

for all $x \in \Omega$ if and only if σ is not a polynomial.

Caution: With the UAT we can at least feel assured that an extremely broad class of functions can be approximately represented by neural networks. That said, it is important to note that the UAT has many limitations and is often misinterpreted. First of all, universality is a property of the function class formed by these neural networks, but any particular neural network will not be universal.[7] Another problem with the above statement of the UAT is that it is non-constructive in nature. It does not tell us

[7]For any fixed architecture, the set of all possible functions that can be formed is parametrized by finitely many degrees of freedom (the weights) whereas the space of all continuous functions is infinite-dimensional.

how to find the architecture (i.e. number of hidden neurons) which will allow for the ϵ-approximation. Some interpret the UAT as saying that you need to take the number of neurons to infinity to get universality, though making this limit rigorous is nontrivial (see [28] for one way to study infinite-width limits). Moreover, even if an oracle told us how many hidden neurons were needed, we would still be left with the highly non-trivial task of finding the weights that would yield a tolerable error. Lastly, the result may be a bit misleading to a practitioner as it turns out the deeper networks tend to be better in real world applications than the shallow MLPs used in the UAT.

Today this result is taken as a given and is often seen as being of little practical value. However, it is worth stressing that before the UAT was established it was not clear whether we would one day be surprised by a simple function akin to XOR that would defy description (approximate or exact) via MLPs. If it had turned out that there were indeed important blind spots, then that would have been very important to know. Thus, the UAT gives us one less thing to worry about in an already complicated field.

2.3 Loss Functions and Training

As mentioned in the previous section, even if we choose an appropriate neural architecture with which to learn our intended target function, the universal approximation theorem will not tell us how to obtain good weights. Moreover, usually we do not even know which function we are trying to learn. Instead, we only see a silhouette of its true form through the lens of our data.

In this context, we need a judge of some sort to tell us how well our model is performing. Ideally, this judge should be a single real number which aids us in our quest to improve our model for the given task. Such a judge is called a *loss function* because by minimizing the "loss" we shall optimize our model.

Though loss functions come in a large variety of flavors, they generally all take the form of a non-negative real-valued function $l(\hat{y}, y)$ which gets smaller as the prediction $\hat{y}$ becomes more similar to the true outcome y. Ideally, we would want our model to minimize the average loss that would occur after encountering all possible data at their naturally occurring real-world frequencies. However, since we only have a finite amount of data, we in practice only hope to minimize the average loss across the dataset. That is, for a dataset $D = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n$ we would like our parametrized model $f_\theta : \mathbb{R}^m \to \mathbb{R}^k$ to minimize the so-called *empirical risk* which is given by

$$L(f_\theta; D) = \frac{1}{|D|} \sum_{(\boldsymbol{x},\boldsymbol{y}) \in D} l(f_\theta(\boldsymbol{x}), \boldsymbol{y}) = \frac{1}{k} \sum_{i=1}^{n} l(f_\theta(\boldsymbol{x}_i), \boldsymbol{y}_i)$$

One of the main flavors of loss functions for regression tasks are the ℓ_p loss functions given by $l(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \|\hat{\boldsymbol{y}} - \boldsymbol{y}\|_p$ where

$$\|\boldsymbol{v}\|_p = \begin{cases} \left(\sum_{j=1}^k |v_j|^p\right)^{\frac{1}{p}} & 1 \leq p < \infty, \\ \max\{|v_1|, \ldots, |v_k|\} & p = \infty. \end{cases}$$

The most commonly used value is $p = 2$ as it is a smooth function, but $p = 1$ is also quite common.

For classification tasks, neural networks are usually designed so that the output vector $f(\boldsymbol{x})$ is a probability vector i.e. $\sum_{j=1}^k f(\boldsymbol{x})_j = 1$ and $0 \leq f(\boldsymbol{x})_j \leq 1$ for all $j = 1, \ldots, k$. The predicted class of the input $\boldsymbol{x}$ is then chosen to be the class corresponding to the index with the largest probability.[8] In this setting, a commonly used loss function is the *cross-entropy loss* [19].

Notably, even if we know exactly what function f we are trying to learn with our neural network F, we do not attempt to minimize the uniform error over a compact

[8]These generally cannot be treated as "honest" probabilities which come from a probabilistic model, but it can be useful to think of the entries as probabilities nonetheless.

region Ω as in the UAT. That is, we *do not* try to minimize

$$\max_{\boldsymbol{x}\in\Omega} |F(\boldsymbol{x}) - f(\boldsymbol{x})| .$$

The reason for this is that outside of extremely special circumstances, it is intractable to ensure that the neural network F approximately matches the desired function f everywhere.

Once an appropriate loss function is chosen, the next step is to find weights that minimize the empirical risk. Ideally, we would do this via a technique that is ensured to minimize the empirical risk, but this is in general intractable. So instead, we seek an approximate solution which produces a satisfactory model. Since many of the loss functions and activation functions used in practice are differentiable, this means that $L(f_\theta; D)$ is differentiable with respect to the parameters $w \in \theta$. And even when the loss and activation functions are not differentiable, they are usually differentiable almost everywhere.

These properties allow us to try various techniques from optimization theory to iteratively minimize $L(f_\theta; D)$ with respect to θ. One popular method is Gradient Descent (GD)[9] which iteratively updates the weights by the scheme

$$\theta_{new} = \theta_{current} - \gamma \nabla_\theta L(f_\theta; D)\big|_{\theta=\theta_{current}},$$

for sufficiently small $\gamma > 0$. This continues until some level of convergence is reached or until some predetermined number of iterations occur. If there are multiple local minima, this may not converge to a globally minimizing solution and instead may get stuck at a local minimum.

On a more computational note, this scheme has to take the derivative of the entire empirical risk function and so must process the *entire* dataset before the update step

[9]Also known as Batch Gradient Descent.

can occur. This may require holding the whole dataset in memory which can become a problem if the dataset is extremely large. This is one of the reasons Stochastic Gradient Descent (SGD) is usually preferred to full GD. In SGD, the iterative scheme only considers the gradient one observation at a time and is given by

$$\theta_{new} = \theta_{current} - \gamma \nabla_\theta l(f_\theta(\boldsymbol{x}_i), \boldsymbol{y}_i)\big|_{\theta=\theta_{current}}$$

where i ranges from $i = 1, \ldots, n$, after which the process can be repeated. Usually the observation points $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ are randomly shuffled as well. One pass through all the data is called an **epoch** and SGD is stopped once some sort of convergence is reached or after a predetermined number of epochs.

The reason this is known as *stochastic* gradient descent (besides the random shuffling) is because $\nabla_\theta l(f_\theta(\boldsymbol{x}_i), \boldsymbol{y}_i)$ can be seen as a noisy estimate of the gradient of the empirical risk $\nabla_\theta L(f_\theta; D) = \frac{1}{n}\sum_{i=1}^{n} \nabla_\theta l(f_\theta(\boldsymbol{x}_i), \boldsymbol{y}_i)$. This noisiness can be helpful in escaping local minima. However, SGD can sometimes be *too* noisy and lead to erratic updates. A compromise to between SGD and GD is provided by Mini-Batch Gradient Descent for which the update rule is given by

$$\theta_{new} = \theta_{current} - \frac{\gamma}{|B|} \sum_{(\boldsymbol{x},\boldsymbol{y})\in B} \nabla_\theta l(f_\theta(\boldsymbol{x}), \boldsymbol{y})\big|_{\theta=\theta_{current}}$$

where $B \subseteq D$ is a subset of data points called a *mini-batch* of *batch size* $|B|$. As with SGD, the order of the data is randomized before partitioning. If the batch size $b = |B|$ evenly divides the size of the data set $n = |D|$ then we will have $\frac{n}{b}$ updates per epoch, and each update step will see b elements of the data. Intuitively, the term $\frac{1}{|B|}\sum_{(\boldsymbol{x},\boldsymbol{y})\in B} \nabla_\theta l(f_\theta(\boldsymbol{x}), \boldsymbol{y})$ is a noisy estimate of the full empirical risk gradient, but one that is less noisy than the gradient term in SGD. This desirable behavior is why Mini-Batch Gradient Descent is one of the most popular optimization methods.[10]

[10]Note that Mini-Batch Gradient Descent with $b = 1$ and $b = n$ is just SGD and GD respectively. Moreover, adding to the terminological confusion, Mini-Batch Gradient Descent is commonly just re-

There are many more gradient-based optimizations schemes such as Momentum [42], AdaGrad [12], RMSprop [25], Adam [30], and more. However, we will not delve further into their details here.

Throughout all this, there has been one big issue that has been unaddressed: How do we compute the gradient of our complicated multi-layer neural network? This seems like a trivial multivariate calculus exercise, after all, we can just use the multivariate chain rule the usual way and be done with it. However, taking the gradient of the loss with respect to each weight individually would lead to expressions that would rapidly become unwieldy for large and complex neural network models with multiple layers and many weights. Fortunately, there is a lot of redundancy in these expressions, and with clever organization via dynamic programming one can make the gradient computation about as simple as evaluating the function itself. The tool to do this is called *backpropagation*. Though the general idea has been discovered multiple times in multiple settings, within the world of neural networks it is usually attributed to David E. Rumelhart, Geoffrey E. Hinton and Ronald J. Williams [47]. By computing the error between the predicted output and the truth, then propagating this error backwards through the network from the last layer to the first via the chain rule, backpropagation provides an efficient way of computing the gradient.

With all these pieces in place (loss functions, optimization scheme, and backpropagation) we can iteratively go from an initial neural network model with random weights to improved models by progressively minimizing empirical risk. This process is called *training* the neural network. When things go well, trained models generalize and perform well on new unseen examples. However, things can go awry for multiple reasons. It could be that the network's capacity to model the needed function is too limited and so we may need a more complex model. It can also be the case that the model has

ferred to as SGD with batch size b.

overfit to the data and memorized it, leading to poor predictions when presented with unseen examples. The latter case can sometimes be remedied with more data, better architecture choice, or regularization. Regardless of outcome, training is a crucial part of creating useful neural network models for real data.

2.4 Novel Input Types and Inductive Bias

So far we have been looking at neural networks as functions/algorithms which take generic fixed-dimensional Euclidean vector inputs and produce fixed-dimensional Euclidean vector output (or produce class labels in the case of classification). However, there are many data types for which this is an overly simplistic perspective (e.g. time-series, images, video) or for which this perspective does not directly apply (e.g. sets, graphs, meshes). As the field progressed, the neural network approach was adapted to many such inputs in various ways. One of the most important methods is via incorporating *inductive bias* into the network architecture, i.e. building into the network our assumptions about how the network *should* behave.

The first notable case of this was that of images. A fixed resolution (e.g. 32x32 pixels) grey scale image can be seen as fixed dimensional vector (e.g. 1024 dimensional) and so can be readily fed into a fully-connected feed-forward neural network (with 1024 dimensional input layer). However, this completely ignores the grid structure of an image in which neighboring pixels should be more closely related than distant pixels. This also does not account for certain obvious human phenomenon such as our perceptual insensitivity to simple transformations such as translations. To address these problems, convolutional neural networks (CNN) were developed by LeCun et al. [31], taking inspiration from its biologically inspired predecessor, Fukushima's Neocognitron [14]. The most important feature in a CNN is the convolutional layer from which it gets its name in which a small filter (e.g. 5x5) get convolved[11] with the image. Mathematically, the convolutional operation is just a special linear operation in which some of the weights are set to be equal to each other (a.k.a. “shared”) and many of the weights are set to zero (so that hidden neurons have localized receptive fields). Hence, a CNN is really just a special case of feed-forward neural network. Though the UAT may have led us to choose a simple single hidden-layer MLP for image processing, this would be a sub-optimal choice as CNNs routinely and dramatically outperform fully-connected models. This example demonstrates the value of inductive biases such as translation invariance into architecture choice.

To apply neural networks to arbitrary length sequences, new neural network models needed to be considered, namely *recurrent neural networks* (RNNs). These models allow for loops to exist in the neural network computation graph, i.e. it allows neurons to feed into themselves. By “unrolling” such networks, it is possible to process arbitrary length sequence inputs, as well as train them through backpropagation.

The principles of incorporating inductive bias and being more flexible with network structure makes it possible to deal very diverse data types. As a result, there are now neural networks for sets [20, 40, 41, 62, 1, 55, 51], graphs (GNNs) [59, 63, 60], meshes [21, 49, 35, 64, 27], and more. Much of these techniques often get placed under the umbrella of Geometric Deep Learning [6] because they often time incorporate geometric/topological inductive bias of some sort.

With the growth of such a diverse class of neural network types, it once again becomes important to understand exactly what sorts of functions they can or can’t represent. Generally, such theoretical questions must be addressed case-by-case. In Chapter 7 we analyze PointNet [40] and DeepSets [62] invariant models on the space of point clouds then set-up and prove universality results in that context (Theorem 7.2). We do this by

[11]Strictly speaking, the operation we are doing is cross-correlation not convolution, but the name has stuck.

proving various properties about those networks and by using the results of Chapter 5. It is in Chapter 5 where we attempt to provide general purpose tools for neural network universal approximation for non-Euclidean data, with Theorem 5.6 being the broadest generalization of the classical UAT that the author has seen and which can hopefully be a useful guide when creating neural networks for novel input data-types.

Chapter 3

Mathematical Preliminaries

As much of this thesis is centered on proving theorems, we will be making use of standard mathematical notations from logic such as $\Rightarrow, \Leftrightarrow, \forall, \exists$, etc. and we will use "iff" as shorthand for "if and only if." We will also be frequently working with sets and using standard notation such as $\varnothing, \in, \subseteq, \subsetneq, \cup, \cap$, etc. We include 0 in the naturals $\mathbb{N}$. We will use the convention of denoting set difference by $A \setminus B := \{a \in A \mid a \notin b\}$. Cartesian products of sets are denoted by $A \times B = \{(a, b) \mid a \in A, b \in B\}$. For a (potentially infinite) family of set $\{A_i\}_{i \in I}$ we denote Cartesian products by $\Pi_{i \in I} A_i$. When A is a set, we will use $|A|$ to denote the cardinality of A, otherwise $|\cdot|$ should be interpreted as the absolute value.

For a set X, let 2^X denote its power set (i.e. the set of all subsets of X) and let $\mathcal{F}(X) \subseteq 2^X$ denote the set of all finite nonempty subsets of X. We will also use $\mathcal{F}^{\leq k}(X)$ to denote the set of nonempty subsets of size $\leq k$, and $\mathcal{F}^k(X)$ to denote the set of k-point subsets. Note that if $|X| = k$ then $\mathcal{F}^{\leq k}(X) = \mathcal{F}(X) = 2^X \setminus \{\varnothing\}$.

Our main motivation for this chapter will be to develop the mathematical background necessary to study $\mathbb{R}$-valued function on $\mathcal{F}(X)$ and the associated approximation theory via neural networks. As a result, we will need to introduce notions of closeness both for the functions on $\mathcal{F}(X)$ and for $\mathcal{F}(X)$ itself.

The most general notion of closeness we will encounter is that of a *topology* on a set. This makes the set into a topological space and many of the most important properties of functions and the spaces they live on can be captured with this structure. However, this notion is sometimes too broad and unwieldy and so we will oftentimes work with the more rigid and numerical notion of a *metric* also known as a *distance function*. Every metric space is a topological space but not necessarily the other way around. Moreover, two nonequivalent metrics can produce the same underlying topology. Sitting halfway between these two concepts is the less common notion of a *uniform space*. Every metric space induces a uniform structure, and every uniform structure induces a topology. Once again, two distinct metrics can produce identical uniform structures and two distinct uniform structures can produce identical topologies. Though it is good to be aware of the existence of uniform spaces, we will only explicitly focus on topological spaces and metric spaces.

3.1 Topological Spaces

We shall quickly review some fundamental definitions and results in elementary topology. Virtually all that will be described in this section can be found in standard textbooks on topology such as [36, 58, 29].

In the Euclidean space $\mathbb{R}^n$, the set of all open sets exhibits some notable properties: (i) $\varnothing$ and $\mathbb{R}^n$ are open sets, (ii) the union of arbitrarily many open sets is open, and (iii) the intersection of finitely many open sets is open. These collections of open sets of $\mathbb{R}^n$ is known as the Euclidean topology. Abstracting these three simple properties leads to the abstract notion of a topology for a set X. Specifically, given a set X, a **topology on** X is a collection of subsets $\tau \subseteq 2^X$ such that:

(i) Both $\varnothing$ and X are members of τ.

(ii) Any union of elements of τ, is itself an element of τ.

(iii) Any intersection of finitely many elements of τ, is itself an element of τ.

The pair (X, τ) is called a **topological space**. When the topology τ is understood from context, we often refer to simply X as the topological space. Surprisingly, a great deal of useful concepts can be built from these three humble axioms.

Given two topologies τ_1 and τ_2 on X such that $\tau_1 \subseteq \tau_2$, we say that τ_1 is **coarser** than τ_2, and conversely, we say that τ_2 is **finer** than τ_1. The finest possible topology on a set X is the power set 2^X and the coarsest possible topology is $\{\varnothing, X\}$. These are called the **discrete topology** and **indiscrete topology** respectively.

For a given topology τ on X, we refer to the elements of τ as the **open sets** of X. An **open neighborhood** of a point $p \in X$ is an open set U such that $p \in U$. Note that a set $A \subseteq X$ may be open with respect to one topology but not another. Alternatively, the **closed sets** of X are the complements of open sets, i.e. they are precisely the sets which can be written as $U^c := X \setminus U$ for an open set U. Note, that it is possible for a set to be both open *and* closed, in which case they are called **clopen**. Similarly, it is possible for a set to be neither open or closed.

The **closure** of a set A is defined to be the smallest[1] closed set containing A and is denoted by $\overline{A}$. The set $\overline{A}$ is equivalently the set of all points $p \in X$ which have the property that every open neighborhood of p contains a point of A. On the other hand, a point $p \in A$ which has an open neighborhood $U \subseteq X$ such that $U \cap A = \{p\}$ is called an **isolated point** of A. A space without isolated points is necessarily infinite. If a set A is such that $\overline{A} = X$ then we say that A is **dense** in X. If X contains a countable dense set, then we say that X is **separable** (e.g. $\mathbb{Q}$ is dense in $\mathbb{R}$ in the Euclidean topology and hence $\mathbb{R}$ is separable).

A function $f : X \to Y$ between two topological spaces is said to be **continuous** if the inverse image of open sets is open i.e. $f^{-1}(B)$ is open in X whenever B is an open subset

[1] This set always exists because one can just take the intersection of all closed sets containing A, and by the axioms this will also be a closed set.

of Y. This coincides with the usual notion of continuity for functions $f : \mathbb{R}^n \to \mathbb{R}^m$. An immediate consequence of this definition is that if we have functions $f : X \to Y$ and $g : Y \to Z$ where and both f and g are continuous, then the composition $g \circ f : X \to Z$ is also continuous. A function $f : X \to Y$ is a **homeomorphism** if it is a continuous bijection whose inverse function $f^{-1} : Y \to X$ is also continuous. We say two topological spaces are **homeomorphic** if there exists a homeomorphism between them. For all intents and purposes, two homeomorphic spaces are the same in the eyes of topology.

Given a family of functions $f_i : X \to Y_i$ for $i \in I$ where all Y_i are topological spaces, it is natural to ask if one can choose a topology on X so that all f_i are continuous. Indeed, it is possible to create the coarsest such topology by constructing the set of all possible finite intersections and arbitrary unions that can be made from the sets $f_i(U)$ where $i \in I$ and $U \subseteq Y_i$ is open. This topology is called the **initial topology** (or weak topology) for the family of functions $\{f_i\}_{i \in I}$.

Two important examples of initial topologies are the subspace topology and the product topology. Given a topological space X and a subset $A \subseteq X$, there is a natural inclusion map $i : A \to X$ given by $i(a) = a$. The **subspace topology** on A induced from X is the initial topology with respect to $i : A \to X$. Now consider a family of topological spaces X_i for $i \in I$ and their Cartesian product $X = \Pi_{i \in I} X_i$. The initial topology on X with respect to the i-th coordinate projection maps $\pi_i : X \to X_i$ given by $\pi_i(\boldsymbol{x}) = x_i$ is called the **product topology** on $X = \Pi_{i \in I} X_i$.

Two very important properties when studying general topological spaces are the Hausdorff property and compactness. A topological space X is said to be **Hausdorff** if for any two arbitrary distinct points $x, y \in X$ there exists open neighborhoods $x \in U_x$ and $y \in U_y$ such that $U_x \cap U_y = \varnothing$.[2] We say $A \subseteq X$ is **compact** if for every collection of open sets $\{U_i\}_{i \in I}$ such that $A \subseteq \bigcup_{i \in I} U_i$ (such collections are called **open covers** of A), there

[2]A popular mneumonic for this property is that "a space is Hausdorff if any two distinct points can be 'housed-off' by open sets."

exists a finite subset of indices $\{i_1, \ldots, i_n\} \subseteq I$ such that $A \subseteq U_{i_1} \cup \ldots \cup U_{i_n}$.[3] We say a topological space X is compact if it is compact as a subset of itself. As an example, finite subsets are always compact and moreover, in a Hausdorff space finite sets are always closed. As a further example, Euclidean spaces are always Hausdorff, and a subset A of Euclidean space is compact iff it is closed and bounded (Heine-Borel theorem). Both compactness and the Hausdorff property are invariant under homeomorphism.

There are many nice interplays between the above definitions. Closed subsets of compact spaces are always compact, compact subsets of Hausdorff spaces are always closed, and continuous images of compact spaces are always compact. If $f : X \to Y$ is a continuous bijection from a compact space X to a Hausdorff space Y then it has continuous inverse $f^{-1} : Y \to X$ and hence f is a homeomorphism. A continuous function $f : X \to \mathbb{R}$ where X is compact always achieves its maximum and minimum for some points in the domain (the extreme value theorem). An arbitrary product of Hausdorff spaces is Hausdorff. An arbitrary product of compact spaces is compact (Tychonoff's theorem).

There are many more concepts (e.g. nets, connectedness, local compactness, separation axioms, etc.) and results but these will be sufficient for our purposes.

3.2 Metric Spaces

Though topology provides a very general way of dealing with 'closeness,' it will be advantageous to introduce a more numerically-oriented analogue of this notion. The most popular way to do this is by introducing a function which allows us to measure distances. As usual, it will be beneficial to do this in a very general way. Thus, we intend to retain just enough properties of the Euclidean distance $\|\boldsymbol{x} - \boldsymbol{y}\|$ so as to have a powerful

[3]Alternatively, the compactness property is often phrased "as every open cover has a finite subcover."

tool, but no more properties than necessary. As with the topology section, all of what is discussed here can be readily found in standard sources [36, 58, 29].

Let X be a set. A bivariate function $d : X \times X \to [0, \infty)$ is called a **metric** if it satisfies the following properties for all $x, y, z \in X$:

(i) $d(x, x) = 0$,

(ii) $d(x, y) = 0$ implies $x = y$,

(iii) $d(x, y) = d(y, x)$,

(iv) $d(x, y) \leq d(x, z) + d(z, x)$.

The pair (X, d) is called a **metric space**. When it is clear from context what the metric is, we often just refer to X as the metric space. If we allow d to take values of $+\infty$ then it is an **extended metric**. If we do not assume property (ii) then we have a **pseudo-metric**. In particular, every metric is both a pseudo-metric and an extended metric.

The classic example of a metric is the Euclidean metric on $\mathbb{R}^n$ given by $d(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|$. This is a special case ($p = 2$) of the ℓ_p metrics on $\mathbb{R}^n$ given by

$$d_p(\boldsymbol{x}, \boldsymbol{y}) = \sqrt[p]{\left(\sum_{i=1}^{n} |x_i - y_i|^p\right)}, \qquad p \geq 1.$$

A less familiar, more general, yet simpler example of a metric is the discrete metric. Given any nonempty set X we can define the **discrete metric** $d : X \times X \to [0, \infty)$ by $d(x, y) = 0$ iff $x = y$ and $d(x, y) = 1$ iff $x \neq y$.

Given a metric space (X, d), we can define **open balls** of radius $r > 0$ centered at a point $x \in X$ as $B_r(x) = \{y \in X \mid d(x, y) < r\}$. This lets us define the **metric topology** τ_d by declaring a set to be open iff it is the empty set or it can be expressed as a union of open balls. It is easy to check a metric topology is always Hausdorff. As a simple example, the metric topology induced by the discrete metric is the discrete topology. The analogous construction for a pseudo-metric d is Hausdorff iff d is a metric.

We can turn a subset A of metric space (X, d) into a metric space in its own right by simply restricting the domain of the metric to $A \times A$, i.e. equip A with the metric $d_A := d|_{A\times A} : A \times A \to [0, \infty)$. As one would hope, the subspace topology induced on A by the metric topology of (X, d) and the metric topology on A given by the restricted metric d_A are actually one and the same.

Now we will consider functions $f : X \to Y$ between metric spaces (X, d_X) and (Y, d_Y). Using the metric topology, we can determine whether such a function is continuous. However, in the context of metric spaces, we can be even more precise and give a local notion of continuity. We say the function $f : X \to Y$ is **continuous at** $x_0 \in X$ if

$$(\forall \epsilon > 0)\,(\exists \delta > 0)\,(\forall y \in X) \quad d_X(x_0, y) < \delta \implies d_Y(f(x_0), f(y)) < \epsilon.$$

As one would hope, it turns out that $f : X \to Y$ being continuous in the topological sense is equivalent to $f : X \to Y$ being continuous at every $x \in X$. That is, $f : X \to Y$ is continuous iff

$$(\forall x \in X)\,(\forall \epsilon > 0)\,(\exists \delta > 0)\,(\forall y \in X) \quad d_X(x, y) < \delta \implies d_Y(f(x), f(y)) < \epsilon.$$

With metric spaces, we can also introduce a more stringent notion of continuity known as *uniform* continuity. A function $f : X \to Y$ is defined to be **uniformly continuous** if

$$(\forall \epsilon > 0)\,(\exists \delta > 0)\,(\forall x, y \in X) \quad d_X(x, y) < \delta \implies d_Y(f(x), f(y)) < \epsilon.$$

Note that in the definition of uniform continuity that the '$\forall x \in X$' has been moved to the inside and now occurs after '$\exists \delta > 0$'. This means that the choice of δ can no longer depend on x (as was allowed in the definition of metric continuity) and can now only depend on the choice of ϵ, i.e. the choice of δ must work *uniformly* across all of X for a given $\epsilon > 0$. Like with topological continuity, if we have three metric spaces $(X, d_X), (Y, d_Y)$, and (Z, d_Z) and uniformly continuous functions $f : X \to Y$ and $g : Y \to Z$ then the composition $g \circ f : X \to Z$ is also uniformly continuous. Also, if $f : X \to Y$ is continuous mapping from a compact metric space to a metric space, then f is uniformly continuous.

Another important class of functions worth reviewing are the Lipschitz function. We say a function $f : X \to Y$ between metrics spaces (X, d_X) and (Y, d_Y) is **K-Lipschitz** for $K \geq 0$ if for all $x, y \in X$ we have

$$d_Y(f(x), f(y)) \leq K d_X(x, y).$$

We say f is a **Lipschitz function** so long as there exists some $K \geq 0$ for which f is K-Lipschitz. Intuitively the above formula means a K-Lipschitz function can only expand the distance between points $x, y \in X$ by at most a factor of K. A 0-Lipschitz function is necessarily constant, and a 1-Lipschitz function is sometimes called a **non-expanding map** or a **short map**. If f is K-Lipschitz with $K < 1$ then f is called a **contraction mapping**. We use $\mathrm{Lip}_K(X, Y)$ to denote the set of all K-Lipschitz maps from X to Y and $\mathrm{Lip}(X, Y) := \bigcup_{K \geq 0} \mathrm{Lip}_K(X, Y)$. If $Y = \mathbb{R}$ with the Euclidean metric, we use the simpler notation $\mathrm{Lip}_K(X)$ and $\mathrm{Lip}(X)$. In these notations, the metrics on X and Y need to be declared or understood from context. It is easy to check that if $f \in \mathrm{Lip}_K(X, Y)$ and $g \in \mathrm{Lip}_L(Y, Z)$ then $g \circ f \in \mathrm{Lip}_{LK}(X, Z)$.

Finally, a function $f : X \to Y$ between metric spaces (X, d_X) and (Y, d_Y) is an **isometry** if for all $x, y \in X$ we have

$$d_Y(f(x), f(y)) = d_X(x, y).$$

By definition, an isometry is always an injective map because if $f(x) = f(y) = 0$ then $0 = d_Y(f(x), f(y)) = d_X(x, y)$ and so $x = y$. On the other hand a surjective isometry has an inverse function which is also an isometry – such maps are called **isometric isomorphisms**. Note that an isometry is always in $\mathrm{Lip}_1(X, Y)$ but the converse is not true. However, if $f \in \mathrm{Lip}_1(X, Y)$ is bijective with $f^{-1} \in \mathrm{Lip}_1(Y, X)$ then f is an isometric isomorphism.

Like in Euclidean space, it is possible to define the notion of a limit of a sequence for a general metric space (X, d). Given a sequence of points $\{x_m\}_{m\in\mathbb{N}} \subseteq X$, we say that $\lim_{m\to\infty} x_m = x$ if for all $\epsilon > 0$ there exists an $N \in \mathbb{N}$ such that for all $n \geq N$ we have that $x_n \in B_\epsilon(x)$. In this case we say x is the **limit** of the sequence x_m as m goes to ∞. If there does not exist an $x \in X$ so that $\lim_{m\to\infty} x_m = x$, we say the sequence does not converge. We say a sequence $\{x_m\}_{m\in\mathbb{N}} \subseteq X$ is a **Cauchy sequence** if for all $\epsilon > 0$ there exists an $N \in \mathbb{N}$ such that for all $n, m \geq N$ we have $d(x_n, x_m) < \epsilon$. In other words, a sequence is Cauchy if the elements of the sequence get arbitrarily close to each other when we go deep enough into the sequence.

In Euclidean space Cauchy sequences always converge, but this is not true for general metric spaces. For example, in the metric space $(\mathbb{Q}, d)$ where $d(x, y) = |x - y|$ the sequence $x_0 = 3, x_1 = 3.1, x_2 = 3.14, \ldots$ given by the decimal truncations of π is a Cauchy sequence in $\mathbb{Q}$, but it definitely does not converge to any member of $\mathbb{Q}$ since π is irrational. This can be remedied by a construction known as the **completion of a metric space**. Essentially what is done is that additional points are added to the metric space in order to fill in the gaps so that every Cauchy sequence converges. This is done in a minimalistic and universal way which we will not detail here (see any of the standard references that were provided for the details). We will denote the completion of a metric space (X, d) by $(\overline{X}, \bar{d})$. As an example, $\mathbb{R}$ is the metric completion of $\mathbb{Q}$ with respect to the Euclidean metric (and with the ℓ_p metrics as well). Note that we also use $\bar{A}$ to denote the closure of $A \subseteq X$, but this will rarely lead to confusion when working with metric spaces since it turns out that the closure of a subset of a complete metric space (X, d) is isometrically isomorphic to the metric completion of that subset. In particular, for a complete metric space, a subset A is closed iff it is complete. Another nice property of the completion is that if $f : X \to Y$ is uniformly continuous then it uniquely extends to the completion, i.e. there exists a unique continuous function $\overline{f} : \overline{X} \to \overline{Y}$ such that

$\overline{f}|_X = f$. A very useful property is that a compact metric space is always complete.[4]

Lastly, it is worth a word of warning to point out that a complete metric space can be homeomorphic to an incomplete metric space. The classic example is that the open interval $(-\frac{\pi}{2}, \frac{\pi}{2}) \subseteq \mathbb{R}$ is homeomorphic to all of $\mathbb{R}$ via $f(x) = \arctan(x)$. The latter is a complete metric space, but the former is not as it is not even closed. However, if two metric spaces are isometrically isomorphic then they are either both complete or both incomplete. That is to say, completeness is not a topological property, but a metric property.

3.3 Functional Analysis

The material in this section can be found among many standard books on measure theory and functional analysis such as [46, 45, 56]. For a deeper comprehensive dive into these subjects we refer the reader to [5] for topological vector spaces and [3, 4] for measure theory. From now on, we will only use the real numbers $\mathbb{R}$ as our scalars unless otherwise specified (i.e. not $\mathbb{C}$) and in this section in particular, we will only consider real-valued functions (i.e. not vector-valued) for our function spaces unless otherwise stated.

A **normed vector space** is a vector space V equipped with a function $\|\cdot\| : V \to [0, \infty)$ called a **norm** which has the properties that for all $\alpha \in \mathbb{R}$ and $x \in V$ we have,

$$\text{(i) } \|x\| = 0 \text{ iff } x = 0, \qquad \text{(ii) } \|\alpha x\| = |\alpha| \, \|x\|, \qquad \text{(iii) } \|x + y\| \leq \|x\| + \|y\|.$$

The norm induces a natural distance function on V given by $d(x, y) = \|x - y\|$ which turns V into a metric space. A **Banach space** is a normed vector space which is a *complete* metric space with respect to the above induced distance function. For simple

[4]In fact, more can be said. A metric space is compact iff it is complete and totally bounded. This is a generalization of the Heine-Borel theorem for Euclidean space.

examples, the ℓ_p norms on $\mathbb{R}^n$ given by

$$\|\boldsymbol{v}\|_p = \begin{cases} \left(\sum_{j=1}^{k} |v_j|^p\right)^{\frac{1}{p}} & 1 \leq p < \infty, \\ \max\{|v_1|, \ldots, |v_k|\} & p = \infty. \end{cases}$$

all turn $\mathbb{R}^n$ into a Banach space. If we try to extend the concept to $p < 1$, then $\|\cdot\|_p$ is no longer even a norm. Whenever context is clear that we are working in Euclidean space, we will frequently omit the subscript in $\|\cdot\|_2$ and simply write $\|\cdot\|$.

Function spaces also provide a rich source of examples for functional analysis. We say a real-valued function f is bounded if there exists an R such that $|f(x)| \leq R$ everywhere. Let $\mathcal{B}(A)$ be the set of bounded functions on a set A, let $\mathcal{C}(X)$ be the set of continuous functions on a topological space X, and let $\mathcal{U}(M)$ be the uniformly continuous functions on a metric space (M, d). Additionally, we can define the bounded versions of these functions spaces as $\mathcal{C}_b(X) = \mathcal{C}(X) \cap \mathcal{B}(X)$ and $\mathcal{U}_b(M) = \mathcal{U}(M) \cap \mathcal{B}(M)$.

We can equip the space of bounded functions $\mathcal{B}(A)$ with the supremum norm

$$\|f\|_{\infty,A} = \sup_{a \in A} |f(a)|,$$

and if the domain is understood we just write $\|f\|_\infty$. This in turn defines the uniform distance $d_\infty(f, g) = \|f - g\|_\infty$. This makes all the bounded families $\mathcal{U}_b(M), \mathcal{C}_b(X), \mathcal{B}(A)$ into Banach spaces. We can equip the other spaces with $\|\|\cdot\|\|$ as well but we may obtain infinite-values and hence we will not have a true norm in general, but at least this will induce a topology on these spaces since then d_∞ is an extended metric. Conveniently, if X is compact then $\mathcal{C}(X) = \mathcal{C}_b(X)$, and similarly, if M has compact metric completion then $\mathcal{U}(M) = \mathcal{U}_b(M)$.

A key concept we will be using throughout the later chapters is the **uniform closure** of a family of real-valued functions S. This is just the closure in the topology given by the uniform distance. Equivalently, the uniform closure of S is just the set of all functions which can be uniformly approximated via members of the family S. That is to say, f is in the uniform closure of S iff there exists a sequence of functions $f_n \in S$ such that $d_\infty(f_n, f) \to 0$. Thus, the question of which functions a family of $\mathbb{R}$-valued neural network can uniformly approximate and which functions it cannot uniformly approximate is completely answered by determining that family's uniform closure. From this point of view, we can see that the classical UAT in Theorem 2.1 is telling us about the uniform closure of the family of single hidden layer neural networks under consideration.

A famous and far-reaching result known as the Stone-Weierstrass theorem gives us sufficient conditions for determining when a family of continuous functions $S \subseteq \mathcal{C}(\Omega)$ has uniform closure $\overline{S} = \mathcal{C}(\Omega)$. To state the theorem, we will need to introduce the concept of a unital algebra and what it means for a family of functions to separate points. A collection $S \subseteq \mathcal{C}(\Omega)$ is a unital subalgebra of $\mathcal{C}(\Omega)$ if it contains the constant 1 function (unital) and is both closed under linear combinations point-wise products (algebra). A collection of real-valued functions S on Ω is said to **separate points** of Ω if for all $x, y \in X$ where $x \neq y$ there exists an $f \in S$ such that $f(x) \neq f(y)$. With these notions in place the theorem is as follows.

Theorem 3.1 (Stone-Weierstrass). *Let Ω be a compact Hausdorff space and $S \subseteq \mathcal{C}(\Omega)$ a unital subalgebra. Then $\overline{S} = \mathcal{C}(\Omega)$ iff S separates points.*

Next, we will also need to go beyond real-valued functions and consider spaces of measures on Ω. To do this carefully would require a great deal of exposition and a slew of technical definitions (σ-algebra, measures, Lebesgue integration, regularity, etc.) so instead we will only convey the key ideas here. The interested reader may to the references mentioned at the beginning of this section.

Intuitively, given a set X, a measure μ on X is just a way to assign a "size" to subsets of X that play well with basic set operations. For example, if $A, B \subseteq X$ and $A \cap B = \varnothing$ we would expect that $\mu(A \cup B) = \mu(A) + \mu(B)$ and that $\mu(\varnothing) = 0$. A cumbersome technicality that occurs in the process of defining a measure is that we need to declare which subsets are "measurable."

Fortunately for our purposes in which we deal (Ω, τ) compact Hausdorff, we will only need to involve the space of finite signed regular Borel measures $\mathcal{M}(\Omega)$.[5] Though the laundry list of adjective may make $\mathcal{M}(\Omega)$ seem like an extremely esoteric collection mathematical objects, it turns out that all those conditions on the measures allow us to see $\mathcal{M}(\Omega)$ as a sort of mirror-image of the space $\mathcal{C}(\Omega)$. Specifically, the famous Riesz-Markov theorem tells us that $\mathcal{M}(\Omega)$ is the topological dual space of $\mathcal{C}(\Omega)$, i.e. $\mathcal{M}(\Omega)$ can be thought of as the space of all continuous linear functionals[6] on $\mathcal{C}(\Omega)$. The way an element $\mu \in \mathcal{M}(\Omega)$ acts as a linear functional on $\mathcal{C}(\Omega)$ is via integration. Specifically, given a $\mu \in \mathcal{M}(\Omega)$ and an $f \in \mathcal{C}(\Omega)$, they act on each other via (Lebesgue[7]) integration like so,

$$\langle \mu, f \rangle := \int_\Omega f \, \mathrm{d}\mu.$$

This is sometimes called the **duality pairing**. The Riesz-Markov theorem also ensures us that if we concoct a continuous linear functional $L : \mathcal{C}(\Omega) \to \mathbb{R}$ then there will exist a $\mu \in \mathcal{M}(\Omega)$ such that $L(f) = \int_\Omega f \, \mathrm{d}\mu$. Thus, whenever in doubt as to what the members of $\mathcal{M}(X)$ are like, one can without loss of generality think of them as continuous linear functionals. The simplest such continuous linear functional is the evaluation map $\mathcal{C}(\Omega) \ni f \mapsto f(a)$. By Riesz-Markov theorem we know there is an associated measure we can use to represent the evaluation map, and it is the Dirac delta measure $\delta_a \in \mathcal{M}(\Omega)$. As a measure, the Dirac delta $\delta_a \in \mathcal{M}(\Omega)$ also has a very simple definition given by

$$\delta_a(A) = \begin{cases} 1 & \text{if } a \in A, \\ 0 & \text{if } a \notin A, \end{cases}$$

where A is any Borel set.[8]

[5]These are sometimes known as Radon measures.

[6]A functional is just scalar valued map. A linear functional is just a linear scalar-valued map. The scalar here being $\mathbb{R}$.

[7]The definition of Lebesgue integration is crucial for this to work out correctly but is very technical, so we leave that to the references.

[8]We won't delve into the precise meaning, but know that every open and closed set is a Borel set.

A **Borel probability measure** on our compact Hausdorff space (Ω, τ) is a member $\mu \in \mathcal{M}(\Omega)$ such that $\mu(A) \geq 0$ for all Borel sets A and $\int_\Omega \mathrm{d}\mu = 1$. That is to say, μ is non-negative everywhere and has total mass 1. The set of all Borel probability measures on (Ω, τ) is denoted by $\mathcal{P}(\Omega)$.

One thing that $\mathcal{M}(\Omega)$ is missing is a topology. We can give it a metric topology by using the so-called *total variation distance* d_{TV}, but this is not the best path for us. Our preferred topology is the **weak-* topology**. The weak-* topology on $\mathcal{M}(\Omega)$ is the coarsest topology on $\mathcal{M}(\Omega)$ that ensures all the linear functionals $\langle -, f \rangle : \mathcal{M}(\Omega) \to \mathbb{R}$ are continuous. In other words, the weak-* topology is the initial topology with respect to the family of functions $\{\langle -, f \rangle \mid f \in \mathcal{C}(\Omega)\}$.

One great benefit of the weak-* topology is that every closed and bounded set in $\mathcal{M}(\Omega)$ will be compact. This ultimately implies that $\mathcal{P}(\Omega)$ is compact since Ω is compact. On the other hand, if we used the total variation distance d_{TV} we would find that not even the closed unit ball would be compact. Another benefit of the weak-* topology is that if the sequence $a_n \in \Omega$ converges to $a \in \Omega$ then δ_{a_n} converges to δ_a as expected. On the other hand, the total variation distance has $d_{TV}(\delta_a, \delta_b) = 2$ for all $a \neq b$. From this we see that the weak-* topology actually respects the underlying topology of Ω.

And the countable union/intersection or complements of Borel sets are also Borel. And so on.

Chapter 4

Spaces of Finite Subsets

Let $\mathcal{F}(\Omega)$ denote the set of nonempty finite subsets of a set Ω. If Ω has a topological (or metric) structure, it would be ideal to find a topological (or metric) structure for $\mathcal{F}(\Omega)$. There is more than one way to do this and, perhaps somewhat surprisingly, there is more than one useful choice.

We will be focusing on equipping the space $\mathcal{F}(\Omega)$ in its entirety with a metrics that are compatible with the underlying metric space (Ω, d). For an interesting purely topological approach to the bounded cardinality subset spaces of the unit circle $\mathcal{F}^{\leq k}(S^1)$ see [52].

4.1 Set of Subsets and the Discrete Metric

As a first example, we show that for any set Ω it is possible to endow the power set 2^Ω with a metric structure in a trivial way. This can be done via the discrete metric $d : 2^\Omega \times 2^\Omega \to [0, \infty)$ given by

$$d_{2^\Omega}(A, B) = \begin{cases} 1 & \text{if } A \neq B, \\ 0 & \text{if } A = B. \end{cases}$$

It is clear this is a metric. Notably, we did not require Ω to have a topology before defining this metric. Thus, when Ω comes with a topology we are left in an undesirable position since we would like closeness of points in Ω to translate to notions of closeness in $\mathcal{F}(\Omega) \subseteq 2^{\Omega}$. Indeed if Ω came equipped with a metric d, the only time $d_{2^{\Omega}}(\{x\},\{y\}) = d(x,y)$ is when d is itself the discrete metric on Ω. Thus, outside of this special case, this approach to metrizing $\mathcal{F}(\Omega)$ will mainly serve as low-hanging trivial example.

4.2 Compact Subsets and the Hausdorff Metric

Let $\mathcal{K}(\Omega)$ denote the set of all nonempty compact subsets of a space Ω. A singleton set $\{x\}$ is automatically compact in any topological space and so $\mathcal{F}(\Omega) \subseteq \mathcal{K}(\Omega)$. Formally we will denote this inclusion map by $i_{\mathcal{K}} : \mathcal{F}(\Omega) \to \mathcal{K}(\Omega)$.

When working with a metric space (Ω, d) it is possible to metrize $\mathcal{K}(\Omega)$ in a useful way. The **Hausdorff metric** (not to be confused with the Hausdorff property) is a metric d_H on $\mathcal{K}(\Omega)$ which can be expressed in many different ways. The most useful version for us will be

$$d_H(A,B) = \inf\{\epsilon \geq 0 \mid A \subseteq B^{\epsilon}, B \subseteq A^{\epsilon}\}$$

where $A^{\epsilon} = \bigcup_{a\in A}\{x \in \Omega \mid d(x,a) \leq \epsilon\}$ is the **ϵ-fattening** of A (the set of points within ϵ of A). In plain terms, $d_H(A,B) \leq \epsilon$ if and only if the ϵ-fattening of A is large enough to completely engulf B and the ϵ-fattening of B is large enough to completely engulf A. Minimizing the choice of $\epsilon \geq 0$ which satisfy the mutually-engulfing criteria leads us at the Hausdorff distance. See Fig. 4.1 for an example of this concept.

To each compact set $A \in \mathcal{K}(\Omega)$, there corresponds a well-defined distance functional $d(A,-) : \Omega \to \mathbb{R}$ given by $d(A,x) = \min_{a\in A} d(A,x)$ (the minimizer exists due to compactness of A). It turns out that by comparing distance functionals via the uniform distance, we can arrive at the following equivalent formulation of the Hausdorff metric,

$$d_H(A,B) = \sup_{q\in\Omega} \big|d(A,q) - d(B,q)\big|.$$

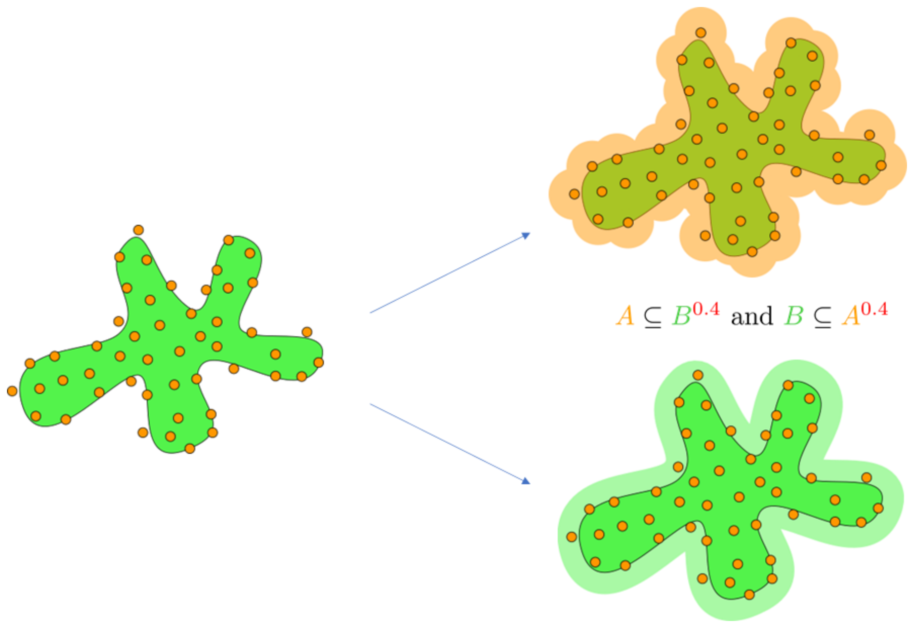

Figure 4.1: On the left we have two compact sets, the finite collection of orange points A, and the green region B. On the right we show two ϵ-fattenings for $\epsilon = 0.4$. In this example, the fattening $A^{0.4}$ contains B and the fattening $B^{0.4}$ contains A. Thus, $\epsilon = 0.4$ satisfies the mutual engulfment criteria and so we know that $d_H(A, B) \leq 0.4$. To find the exact value of $d_H(A, B)$ we would need to take the infimum over all valid ϵ.

(example 4.13 in [43]). For some other alternative formulas and additional background refer to [36, 34, 2, 43].

The following are some well-known remarkable properties of the Hausdorff metric that we shall need.[1]

Theorem 4.1. *If (Ω, d) is compact, then so is $(\mathcal{K}(\Omega), d_H)$. Moreover, $\mathcal{F}(\Omega)$ forms a dense subset of $\mathcal{K}(\Omega)$ i.e. every compact set can be d_H-approximated by finite sets.*

Proof. See [24] for why (Ω, d) compact implies $(\mathcal{K}(\Omega), d_H)$ is compact. For density of $\mathcal{F}(\Omega)$, consider an arbitrary compact set A and let $\epsilon > 0$ be arbitrary. Note that $\mathcal{U} = \{B_\epsilon(x) \mid x \in A\}$ is an open cover of A. By compactness there's a finite subcover $\mathcal{V} =$

[1]The topology induced on $\mathcal{K}(\Omega)$ by d_H depends only on the topology of (Ω, d) and not on the choice of metric d. Indeed, we could have proceeded without metrics at all and used the topology of Ω to directly define the topology of $\mathcal{K}(\Omega)$. This topology is called the Vietoris topology [34]. However, metric spaces are sufficient generality for the work in this thesis, so we take advantage of its existence.

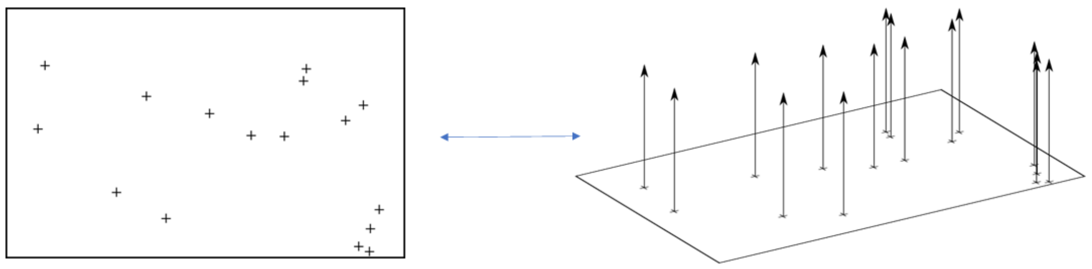

Figure 4.2: Given the finite set $A \subseteq \mathbb{R}^2$ on the left, we can create its empirical measure $\mu_A = \frac{1}{|A|}\sum_{a\in A}\delta_a$ on the right. Conversely, we can get A from μ_A by taking the support.

$\{B_{\epsilon_1}(x_1), \ldots, B_{\epsilon_n}(x_n)\}$. Thus the set of centers of these balls $B = \{x_1, \ldots, x_n\}$ are within ϵ of every point of A i.e. $A \subseteq B^\epsilon$. Because $B \subseteq A$ we also have $B \subseteq A^\epsilon$. Thus $d_H(A, B) < \epsilon$ and $B \in \mathcal{F}(\Omega)$. Since ϵ was arbitrary, $\mathcal{F}(\Omega)$ is dense in $\mathcal{K}(\Omega)$. □

4.3 Probability Measures and the Wasserstein Metric

Let (Ω, d) be a compact metric space and $\mathcal{P}(\Omega)$ its set of Borel probability measures. One natural probability measure that we always have available is the **Dirac delta at** x given by

$$\delta_x(A) = \begin{cases} 0 & x \notin A \\ 1 & x \in A \end{cases}.$$

This measure can also be seen as an evaluation map, since for $f \in \mathcal{C}(\Omega)$ we have

$$\mathbb{E}_{X\sim\delta_x}[f(X)] = \int_\Omega f(\omega)\,\mathrm{d}\delta_x(\omega) = f(x).$$

The Dirac delta provides us a way to associate to any $\{x\} \in \mathcal{F}(\Omega)$ a unique probability measure. We can extend this idea to $\mathcal{F}(\Omega)$ by using the concept of an **empirical measure**. Namely, to each $A \in \mathcal{F}(\Omega)$ we associate the empirical measure $\mu_A := \frac{1}{|A|}\sum_{a\in A}\delta_a$ (see Fig. 4.2). Formally, we denote this mapping by $i_{\mathcal{P}} : \mathcal{F}(\Omega) \to \mathcal{P}(\Omega)$. Since the map $i_{\mathcal{P}}$ is injective, any topology or metric on $\mathcal{P}(\Omega)$ will induce such a structure on $\mathcal{F}(\Omega)$

A useful metric we can place on $\mathcal{P}(\Omega)$ is the **1-Wasserstein metric** or simply the **Wasserstein metric** for short. It can be given via the Kantorovich-Rubinstein formula

$$d_W(\mu,\nu) = \sup_{\substack{f:M\to\mathbb{R}\\ \text{1-Lipschitz}}} \left| \int_M f\,d\mu - \int_M f\,d\nu \right| = \sup_{\substack{f:M\to\mathbb{R}\\ \text{1-Lipschitz}}} \left| \mathbb{E}_{X\sim\mu}[f(X)] - \mathbb{E}_{X\sim\nu}[f(X)] \right|$$

This is not the only way to express this metric. Indeed, a common alternative definition in terms of transport plans and couplings can be extended to give the p-Wasserstein (the "primal formulation"). When expressed that way, it is a bit easier to understand what Wasserstein metrics are measuring (see Fig. 4.3). For further details and theoretical properties of these metrics see one of [53, 48] and for a reference with a more computation perspective see [38].

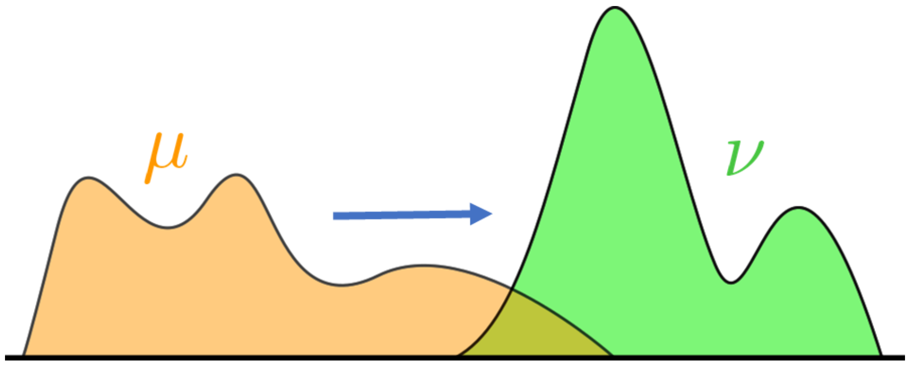

Figure 4.3: In general, the 1-Wasserstein distance $d_W(\mu,\nu)$ represents the minimum effort it would take to reshape the distribution μ into the shape of the distribution ν. For this reason, it is also often called the "earth mover's distance."

One important property of the Wasserstein metric on $\mathcal{P}(\Omega)$ is that it metrizes the weak-* convergence of probability measures. That is to say that, that for a compact metric space (Ω, d), that $d_W(\mu_n,\mu) \to 0$ iff $\int f\,\mathrm{d}\mu_n \to \int f\,\mathrm{d}\mu$ for all $f \in \mathcal{C}(\Omega)$. And so, like in the case of d_H, the topology induced on $\mathcal{P}(\Omega)$ by d_W depends only on the topology of (Ω, d) and not on the choice of metric d.

Finally, another parallel to d_H is the following key property of d_W:

Theorem 4.2. *If (Ω, d) is compact, then so is $(\mathcal{P}(\Omega), d_H)$. Moreover, $i_{\mathcal{P}}(\mathcal{F}(\Omega))$ is dense in $\mathcal{P}(\Omega)$ i.e. every probability measure can be d_W-approximated by empirical measures.*

Proof. The first part follows from the Riesz-Markov Theorem and Banach-Aloaglu. For the second part see Theorem 11.4.1 of [13]. $\square$

4.4 Signed Measures and Weak-* Topology

Once again we let (Ω, d) be a compact metric space. Another way to induce a topology on $\mathcal{F}(\Omega)$ using measure theory is to embed it not into $\mathcal{P}(\Omega)$, but into the larger superset space $\mathcal{M}(\Omega)$. We do this via the mapping $i_{\mathcal{M}}(A) : \mathcal{F}(\Omega) \to \mathcal{M}(\Omega)$ given by $i_{\mathcal{M}}(A) = \mu^A := \sum_{a\in A} \delta_a$. The key difference here between $i_{\mathcal{P}}$ and $i_{\mathcal{M}}$ is of course the absence of the cardinality-dependent normalization factor. Though this difference seems small, it has a dramatic effect on the topology of $\mathcal{F}(\Omega)$.

As mentioned in Section 3.3, it is preferrable to use the weak-* topology over the total variation norm topology on $\mathcal{M}(\Omega)$. If we chose to use the total variation norm topology instead, then the induced topology on $\mathcal{F}(\Omega)$ would actually be the discrete topology, which of course can be metrized with the discrete metric. As we mentioned in Section 4.1, this is undesirable unless the underlying metric space (Ω, d) was already a discrete space. On the other hand, using the weak-* topology actually ensures that the singleton space $\mathcal{F}^1(\Omega)$ actually is homeomorphic Ω itself.

Unfortunately, it is only possible to metrize the weak-* topology on all of $\mathcal{M}(\Omega)$ when Ω is a finite set. One thing that may help, is noticing the $\mu_A = \frac{1}{|A|}\mu^A$. Indeed, if $|A| = k$ then μ^A lies in the sphere of radius k (with respect to the total variation distance). This suggests the following piecewise definition:

$$d_M(\mu^A, \mu^B) = \begin{cases} |A|\, d_W(\mu_A, \mu_B) & \text{if } |A| = |B|, \\ \infty & \text{otherwise.} \end{cases}$$

This turns out to be an extended metric on $i_{\mathcal{M}}(\mathcal{F}(\Omega))$. It is clear that for fixed cardinality k, this is just a multiple of d_W and so it creates the same topology on $\mathcal{F}^k(\Omega)$, namely the induced weak-* topology. Though ad-hoc extended metric naturally shares some similarities with the Wasserstein metric, it naturally has some important differences. One the most important differences is that for infinite Ω, neither $i_{\mathcal{M}}(\mathcal{F}(\Omega))$ or its completion are compact. An intuitive reason why is that we can create a sequence of sets A_k with $|A_k| = k$ and see that $d_M(\mu^{A_m}, \mu^{A_n}) = \infty$ for all $m \neq n$. This would be impossible in metric space with compact completion.

4.5 Metrizing $\mathcal{F}(\Omega)$

The injective maps $i_{\mathcal{K}} : \mathcal{F}(\Omega) \to \mathcal{K}(\Omega)$ and $i_{\mathcal{P}} : \mathcal{F}(\Omega) \to \mathcal{P}(\Omega)$ allow us to induce the d_H and d_W metrics on $\mathcal{F}(\Omega)$. Similarly, the injective map $i_{\mathcal{M}} : \mathcal{F}(\Omega) \to \mathcal{M}(\Omega)$ let us induce an extended metric on $\mathcal{F}(\Omega)$. By a slight abuse of notation, we will interpret d_W and d_M on $\mathcal{F}(\Omega)$ as

$$d_W(A, B) = d_W(\mu_A, \mu_B), \qquad d_M(A, B) = d_M(\mu^A, \mu^B).$$

Note $d_H(A, B)$ already makes sense.

We will denote the (extended) metrized versions of $\mathcal{F}(\Omega)$ by $\mathcal{F}_H(\Omega)$, $\mathcal{F}_W(\Omega)$, and $\mathcal{F}_M(\Omega)$ and similarly we use the same subscript convention for the bounded cardinality spaces $\mathcal{F}^{\leq k}(\Omega), \mathcal{F}^k(\Omega) \subseteq \mathcal{F}(\Omega)$.

Chapter 5

Extending the Universal Approximation Theorem

5.1 Revisiting the Classical UAT

In this chapter we will generalize the classical UAT in various directions. To do so we will need to generalize the idea of a neural network to allow for inputs which are not finite-dimensional Euclidean vectors. We do this in a very general way, allowing us to apply the neural network framework whenever we have a input space X (possibly without topology) and are provided a collection of real-valued functions S which lets us extract features. The family of functions S may itself be comprised of some kind of neural network as in Point Cloud UAT (Theorem 7.2).

To the lay the foundation for this chapter, we will introduce a series of notations to help us describe the families of functions we care about in a compact way.

5.2 Notation for Constructing Families of Functions

In this section we naturally generalize the notion of a neural network to arbitrary input domains and introduce some notation to help study the various classes of neural networks that result.

Composition of families: For two collections of functions $\mathcal{A}$ and $\mathcal{B}$, we use $\mathcal{A} \circ \mathcal{B}$ to denote the set of all *well-defined* composite functions i.e. functions of the form $f \circ g$ where $f \in \mathcal{A}$ and $g \in \mathcal{B}$. If $f \in \mathcal{A}$ and $g \in \mathcal{B}$ cannot be composed (due to incompatibility of their domain and codomains) then $f \circ g$ does not make sense and will not appear in $\mathcal{A} \circ \mathcal{B}$. For singleton sets such as $\mathcal{A} = \{\sigma\}$ we can write $\sigma \circ \mathcal{B}$ instead of $\mathcal{A} \circ \mathcal{B}$.

Sum of families: If $\mathcal{A}$ and $\mathcal{B}$ are families of functions with common domain X, then we let $\mathcal{A} + \mathcal{B} := \{f + g \mid f \in \mathcal{A}, g \in \mathcal{B}\}$. For singleton sets such as $\mathcal{A} = \{f\}$ we can write $f + \mathcal{B}$ instead of $\mathcal{A} + \mathcal{B}$. We also may interpret $\mathbb{R}$ itself as a family of constant functions so that when $\mathcal{A}$ is a family of $\mathbb{R}$-valued functions we have $\mathcal{A} + \mathbb{R} = \{f + c \mid f \in \mathcal{A}, c \in \mathbb{R}\}$.

Powers of families: If $\mathcal{A}$ is a family of functions with common domain X, we use $\mathcal{A}^n$ to denote set of maps of the form $\boldsymbol{f}(x) = (f_1(x), \ldots, f_n(x))$ where $f_i \in \mathcal{A}$. We use $\mathcal{A}^\bullet$ to denote $\mathcal{A}^\bullet := \bigcup_{n=1}^\infty \mathcal{A}^n$. In particular, if $\mathcal{A}$ is a family of $\mathbb{R}$-valued functions, then $\mathcal{A}^\bullet$ is the set of all functions from X to some Euclidean space where every component function belongs to $\mathcal{A}$.

Linear span of a family: If $\mathcal{A}$ is a family of $\mathbb{R}$-valued functions on X, then we use $\operatorname{span} \mathcal{A}$ to denote the set of all finite linear combinations i.e. functions of the form $\sum_{i=1}^n a_i f_i$ for $f_i \in \mathcal{A}$. Note that trivially $\mathcal{A} \subseteq \operatorname{span} \mathcal{A}$ and if all members of $\mathcal{A}$ are continuous, the same is true for $\operatorname{span} \mathcal{A}$.

Algebra generated by a family: If $\mathcal{A}$ is a family of $\mathbb{R}$-valued functions on X, then we use $\operatorname{Alg}(\mathcal{A})$ to denote the $\mathbb{R}$-algebra generated by $\mathcal{A}$ i.e. the set of all functions that can be obtained via finitely many applications of multiplication, scalar multiplication, and addition of elements of $\mathcal{A}$ and $\mathbb{R}$. Equivalently, $\operatorname{Alg}(\mathcal{A})$ is the set of all $\mathbb{R}$-valued functions that can be written as $p(f_1, \ldots, f_n)$ for some multivariate polynomial p and where all $f_i \in \mathcal{A}$. In particular, note that $\mathbb{R} \subseteq \operatorname{Alg}(\mathcal{A})$ and $\mathcal{A} \subseteq \operatorname{span} \mathcal{A} \subseteq \operatorname{Alg}(\mathcal{A})$. Additionally, if all members of $\mathcal{A}$ are continuous, the same is true for $\operatorname{Alg}(\mathcal{A})$.

5.3 Generalized Neural Networks

Using these tools, we can describe certain families of neural networks with concise notation. For the purposes of studying expressivity and function approximation, we shall consider two neural networks to be equivalent if they represent the same underlying *function* even if they have distinct architectures (computation graph).

Simple σ-networks: A **simple σ-network** is any single hidden-layer neural network with 1D linear output-layer and activation function σ i.e. any network of the form $f(\boldsymbol{x}) = \sum_{i=1}^{n} a_i \sigma(\boldsymbol{w}_i \cdot \boldsymbol{x} + b_i)$. The set of all simple σ-networks is denoted by $\mathcal{N}^\sigma$ and the ones with input dimension n by $\mathcal{N}^\sigma_n$. Note that both of these cases only contain scalar-valued networks but they can be of arbitrarily large width.

Though we will not be focusing on these as much, to define deeper neural network classes we do the following. The ℓ hidden layer networks on $\mathbb{R}^n$ are denoted by $\mathcal{N}^{\boldsymbol{\sigma}}_n$ where $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_\ell)$ is a list of ℓ-many activation functions where σ_1 is the activation for the first hidden layer, σ_2 the activation for the second hidden layer, and so on. Formally, the family of neural networks $\mathbb{R}^n$ with $(\ell + 1)$ hidden-layers can be expressed recursively by $\mathcal{N}_n^{(\sigma_1,\ldots,\sigma_\ell,\sigma_{\ell+1})} := \operatorname{span}(\sigma_{\ell+1} \circ \mathcal{N}^{(\sigma_1,\ldots,\sigma_\ell)})$. If we have activations $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_\ell)$ and another activation τ, we can also use the shorthand $(\boldsymbol{\sigma}, \tau) := (\sigma_1, \ldots, \sigma_\ell, \tau)$.

Finally, we note here that Leshno et al.'s classical UAT can be reformulated with this notation as follows:

Theorem 5.1 (Classical UAT rephrased)**.** *Let $\Omega \subseteq \mathbb{R}^n$ be compact and $\sigma \in \mathcal{C}(\mathbb{R})$. Then, the uniform closure of $\mathcal{N}^\sigma_n$ is $\mathcal{C}(\Omega)$ iff σ is not a polynomial.*

Generalized neural networks: To adapt the notion of neural networks to more general domains than Euclidean space we need ways of producing real numbers for each possible input. Let S be a family of $\mathbb{R}$-valued functions on a set X. Then we define $\mathcal{N}^\sigma(S) := \mathcal{N}^\sigma \circ S^\bullet$. Similarly, deeper analogues are denoted by $\mathcal{N}^{\boldsymbol{\sigma}}(S) := \mathcal{N}^{\boldsymbol{\sigma}} \circ S^\bullet$. More concretely, $F \in \mathcal{N}^{\boldsymbol{\sigma}}(S)$ if and only if we can write $F = \eta \circ (f_1, \ldots, f_n)$ for $\eta \in \mathcal{N}^{\boldsymbol{\sigma}}$ and

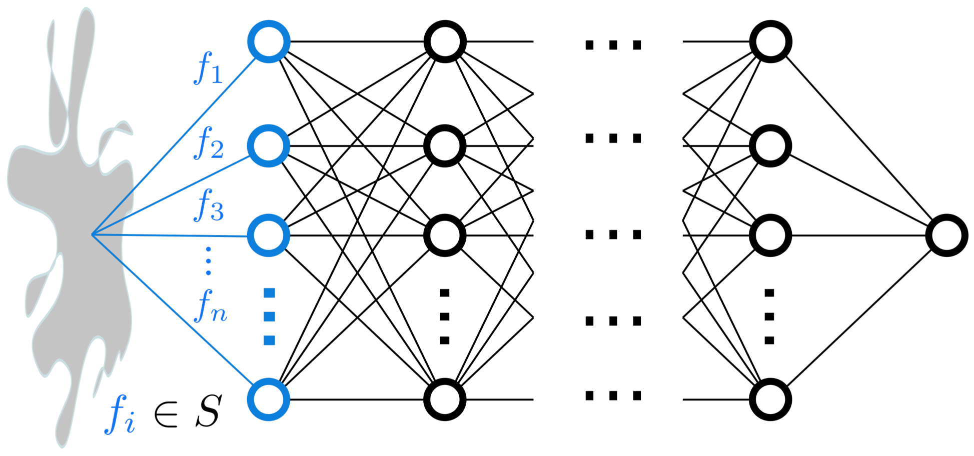

Figure 5.1: A schematic example of a deep generalized neural network. For an input $x \in X$, the functions $f_i \in S$ produce real features $f_i(x)$ (blue) which are then fed as inputs into a classical deep neural network (black).

all $f_i \in S$. See Fig. 5.1 for an example.

5.4 Topological UAT

In [32], Leshno et al. prove that $\mathcal{N}_n^\sigma$ with $\sigma \in \mathcal{C}(\mathbb{R})$ has universal approximation property iff σ is not a polynomial. Using this theorem and Stone-Weierstrass we prove a UAT for generalized neural networks on an abstract compact Hausdorff space. Though we independently arrived at this theorem, we later discovered this result was essentially proven by Stinchcombe in [50] (Theorem 5.1) for a different context. However, our approach is slightly different and so we provide a detailed proof here for completeness and the benefit of the reader.

Recall that a family of functions S on Ω separates points if for any $x \neq y$ there is an $f \in S$ so that $f(x) \neq f(y)$.

Theorem 5.2 (Topological-UAT). *Let X be compact Hausdorff and $\sigma \in \mathcal{C}(\mathbb{R})$ non-polynomial. If $S \subseteq \mathcal{C}(X)$ separates points, then the uniform closure of $\mathcal{N}^\sigma(S)$ is $\mathcal{C}(X)$.*

Proof. Let X, S, and σ satisfy the above. Let $\mathrm{Alg}(S)$ denote the $\mathbb{R}$-algebra generated by S. Then $\mathrm{Alg}(S)$ is unital subalgebra of $\mathcal{C}(X)$ that separates points. By the Stone-

Weierstrass theorem $\mathrm{Alg}(S)$ is dense in $\mathcal{C}(X)$. Now let $F \in \mathcal{C}(X)$ and $\epsilon > 0$ be arbitrary. By density there is a $G \in \mathrm{Alg}(S)$ such that $|F(a) - G(a)| < \epsilon/2$ for all $a \in X$. Since $G \in \mathrm{Alg}(S)$ there is an N-variable polynomial p and $\boldsymbol{s} = (s_1, \dots, s_N)$ where $s_i \in S$, so that $G = p \circ \boldsymbol{s}$. Since all $s_i \in \mathcal{C}(X)$ and X is compact, the image $\boldsymbol{s}(X) \subseteq \mathbb{R}^N$ is compact. By the classical UAT [32], there exists an $\eta \in \mathcal{N}^\sigma$ such that $|p(\boldsymbol{x}) - \eta(\boldsymbol{x})| < \epsilon/2$ for all $\boldsymbol{x} \in \boldsymbol{s}(X)$. Thus, for every $a \in X$ we have

$$|F(a) - (\eta \circ \boldsymbol{s})(a)| \leq |F(a) - p(\boldsymbol{s}(a))| + |p(\boldsymbol{s}(a)) - \eta(\boldsymbol{s}(a))| < \epsilon/2 + \epsilon/2 = \epsilon.$$

Since $\eta \circ \boldsymbol{s} \in \mathcal{N}^\sigma(S)$ and $F \in \mathcal{C}(X)$ was arbitrary, this this means that the uniform closure of $\mathcal{N}^\sigma(S)$ contains $\mathcal{C}(X)$. Conversely, the uniform limit of continuous functions is continuous and so $\mathcal{N}^\sigma(S)$ can only uniformly approximate members of $\mathcal{C}(X)$. Thus the uniform closure of $\mathcal{N}^\sigma(S)$ is precisely $\mathcal{C}(X)$. □

5.5 Metric UAT

Since metric spaces are automatically Hausdorff, we can use the topological-UAT whenever we have a compact metric space equipped with a separating family of continuous functions. This leads to a special application of the result.

Corollary 5.3. *Let (X, d) be a compact metric space and $\sigma \in \mathcal{C}(\mathbb{R})$ non-polynomial. If $S \subseteq \mathcal{C}(X)$ separates points, then $\mathcal{N}^\sigma(S)$ is dense in $\mathcal{C}(X)$. Additionally, if $D \subseteq X$ is dense in X, then the uniform closure of $\mathcal{N}^\sigma(\,S\big|_D)$ is $\mathcal{U}(D)$.*

Proof. Since X is a compact metric space, it is compact Hausdorff. Since $S \subseteq \mathcal{C}(X)$ separates points of X, it follows that $\mathcal{C}(X)$ is the uniform closure of $\mathcal{N}^\sigma(S)$ by Theorem 5.2.

For the second part, recall that $\mathcal{C}(X) = \mathcal{U}(X)$ when X is compact. Thus $S \subseteq \mathcal{U}(X)$ and $\mathcal{N}^\sigma(S) \subseteq \mathcal{U}(X)$. Restricting the domain doesn't change this so both $S\big|_D$ and $\mathcal{N}^\sigma(\,S\big|_D) = \mathcal{N}^\sigma(S)\big|_D$ are subsets of $\mathcal{U}(D)$. By the uniform limit theorem this means that the uniform closure of $\mathcal{N}^\sigma(\,S\big|_D)$ is a subset of $\mathcal{U}(D)$. For the reverse inclusion, let

$f \in \mathcal{U}(D)$ be arbitrary. Since $f \in \mathcal{U}(D)$ and $\overline{D} = X$, there is a unique $\widetilde{f} \in \mathcal{C}(X)$ such that $\widetilde{f}\big|_D = f$. This $\widetilde{f}$ can be uniformly approximated by a sequence $g_n \in \mathcal{N}^\sigma(X)$. Thus $g_n\big|_D$ is a sequence in $\mathcal{N}^\sigma(S)\big|_D = \mathcal{N}^\sigma(\,S\big|_D)$ that uniformly approximates f on D. This completes the reverse inclusion and so the uniform closure of $\mathcal{N}^\sigma(\,S\big|_D)$ equals $\mathcal{U}(D)$. □

We can state this result another way. Recall that a precompact metric space is a metric space whose metric completion is compact.

Theorem 5.4 (Metric-UAT)**.** *Let (X, d) be a precompact metric space and $\sigma \in \mathcal{C}(\mathbb{R})$ non-polynomial. If $S \subseteq \mathcal{C}(X)$ extends to a separating family of continuous functions on the completion, then the uniform closure of $\mathcal{N}^\sigma(S)$ is $\mathcal{U}(X)$.*

Proof. The completion of X is compact by definition. Apply Corollary 5.3 using the extension of S ensured by the hypothesis. □

In abstract, it is nice to have these theorems which provide sufficient conditions for universality on precompact metric spaces. However, it would also be nice to have some concrete examples. This amounts to finding a suitable S for $\mathcal{N}^\sigma(S)$.

Since we are working with metric spaces, we are naturally blessed with a large family of continuous functions. For any metric space X we define the family of distance functionals with respect to A as $S_d^A := \{d(a, -) : X \to \mathbb{R} \mid a \in A\}$. This is always a continuous family. Additionally, if A is dense in X then S_d^A separates points of X. These properties allow us to prove the following result.

Corollary 5.5. *Let (X, d) be a precompact metric space and $\sigma \in \mathcal{C}(\mathbb{R})$ non-polynomial. Then the uniform closure of $\mathcal{N}^\sigma(S_d^X)$ is $\mathcal{U}(X)$.*

Proof. Since (X, d) is precompact, its completion $(\overline{X}, \bar{d})$ is compact. The family S_d^X extends to the continuous family $S_{\bar{d}}^X = \left\{\bar{d}(a, -) : \overline{X} \to \mathbb{R} \mid a \in X\right\}$. By the density of X in $\overline{X}$, for any $x, y \in \overline{X}$ we can find an $a \in X$ so that $\bar{d}(a, x) < \bar{d}(x, y)/2$. This means $\bar{d}(a, y) > \bar{d}(x, y)/2$ and so $\bar{d}(a, x) \neq \bar{d}(a, y)$. Thus $S_{\bar{d}}^X \subseteq \mathcal{C}(\overline{X})$ separates points of $\overline{X}$ and so the uniform closure of $\mathcal{N}^\sigma(S_d^X)$ is $\mathcal{U}(X)$ by Theorem 5.4. □

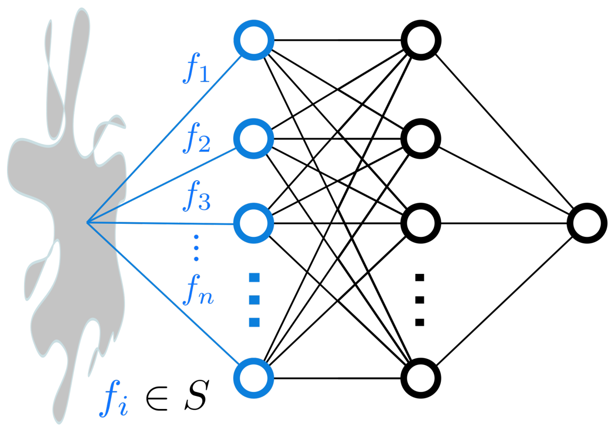

Figure 5.2: A schematic example of an element of $\mathcal{N}^\sigma(S)$ as used in Theorem 5.6.

Note that if X is compact itself, then $\mathcal{U}(X) = \mathcal{C}(X)$ and so $\mathcal{N}^\sigma(S_d^X)$ can uniformly approximate all the continuous functions.

5.6 Generalized UAT

Now we will generalize the UATs so that we no longer assume a metric or even a topology on the input space. That is to say, we treat the domain X as merely a set.

In this setting, we shall also be provided a set of real-valued functions S on the set X. This family of functions S will provide us a way to extract real features for any input element from X. From S, we can always construct the family $\mathrm{Alg}(S)$ by taking all possible products and affine combinations. We always have that $S \subseteq \mathrm{Alg}(S)$ and typically this inclusion will be strict leading richer ways of extracting features. Alternatively, we have also demonstrated that one can build generalized neural networks from S and create the class of functions $\mathcal{N}^\sigma(S)$. The following theorem shows that under very mild conditions, the latter family is equally as powerful as the former in terms of uniform approximation.

Theorem 5.6 (Set-UAT)**.** *Let $S \subseteq \mathcal{B}(X)$ and $\sigma \in \mathcal{C}(\mathbb{R})$ non-polynomial. Then $\mathrm{Alg}(S)$ and $\mathcal{N}^\sigma(S)$ have the same uniform closure.*

Proof. For each $f \in S$ define the compact interval $I_f := [\inf_{x\in X} f(x), \sup_{x\in X} f(x)]$.

These intervals are well-defined because we are given that each $f \in S$ is bounded. By Tychonoff's theorem [36] we know that arbitrary products of compact sets are compact, thus the so-called Tychonoff cube $\prod_{f \in S} I_f$ is compact. Additionally, by definition of the product topology, the coordinate projection maps $\pi_g : \prod_{f \in S} I_f \to I_g$ are continuous. Moreover, the family of all projection maps $\widetilde{S} = \{\pi_g \mid g \in S\}$ is a family of real-valued bounded functions which separate points of the Tychonoff cube (since members of the product space are completely determined by their components).

Given these properties, we can consider the function families on the compact Hausdorff space $\prod_{f \in S} I_f$ given by $\mathrm{Alg}(\widetilde{S})$ and $\mathcal{N}^\sigma(\widetilde{S})$. By the Stone-Weierstrass theorem we have that the uniform closure of $\mathrm{Alg}(\widetilde{S})$ is $\mathcal{C}(\prod_{f \in S} I_f)$. Similarly, by Theorem 5.2 we have the uniform closure of $\mathcal{N}^\sigma(\widetilde{S})$ is also $\mathcal{C}(\prod_{f \in S} I_f)$.

Lastly, define $e_S : X \to \prod_{f \in S} I_f$ as $e_S(x) = (f(x))_{f \in S}$ i.e. the function which maps x to the tuple whose components are the functions of S evaluated at x. Note that $g(x) = \pi_g(e_S(x))$. Thus $\mathrm{Alg}(S) = \mathrm{Alg}(\widetilde{S}) \circ e_S$ and $\mathcal{N}^\sigma(S) = \mathcal{N}^\sigma(\widetilde{S}) \circ e_S$. Thus, the uniform closures of $\mathrm{Alg}(S)$ and $\mathcal{N}^\sigma(S)$ are the same. $\square$

In essence, this result shows that when faced with an unfamiliar input space X of unknown topology along with a collection of real-valued functions S, nothing is lost by considering generalized neural networks in place of $\mathrm{Alg}(S)$, at least from the perspective of uniform approximation. Looking at this another way, this result gives us a way to determine the uniform approximation power of generalized neural networks by instead determining what $\mathrm{Alg}(S)$ can approximate. Since the latter family is much more amenable to standard tools of functional analysis (e.g. Stone-Weierstrass), this can prove to be very helpful. Indeed, if we use the set-UAT as a starting point, all the other UATs we discussed can be derived as a special case.

It is natural to wonder, if this result could be even further generalized. In particular, it is possible to get the same result while removing the boundedness assumption on S? The answer is unfortunately no. To see this one need only look at the following counterexample. Let $X = \mathbb{R}$, let S contain only the identity map $\mathrm{id}(x) = x$, and let $\sigma(x) = \max\{x, 0\}$ (the popular ReLU nonlinearity). Immediately we can check that $\mathrm{Alg}(S)$ is exactly the set of all polynomials on $\mathbb{R}$ and that $\mathcal{N}^\sigma(S)$ is comprised solely of continuous piecewise linear functions which have only finitely many linear pieces. This is a problem because this means the functions in $\mathcal{N}^\sigma(S)$ can have at best a linear growth rate, which is insufficient to uniformly approximate functions such as $f(x) = x^2$ which live in $\mathrm{Alg}(S)$. Thus $\mathrm{Alg}(S)$ and $\mathcal{N}^\sigma(S)$ do not uniformly approximate the same things on all of $\mathbb{R}$.

With that avenue closed, how else could we strengthen this result? One possibility would be to determine a converse: If $\mathrm{Alg}(S)$ and $\mathcal{N}^\sigma(S)$ have the same uniform closure, what can we say about σ (assuming $S \subseteq \mathcal{B}(X)$ and $\sigma \in \mathcal{C}(\mathbb{R})$)? Note that if activation function σ is a polynomial, we immediately have that $\mathcal{N}^\sigma(S) \subseteq \mathrm{Alg}(S)$. However, unlike the Leshno et al.'s classical UAT in Euclidean space, it is sometimes possible (though not always) for $\mathrm{Alg}(S)$ and $\mathcal{N}^\sigma(S)$ to be equally expressive even when σ is a polynomial. As an example, let $X = \mathbb{R}$ and $S = \{\sigma_H\}$ where σ_H is the Heaviside step function. It's easy to see that $\sigma_H(x)^2 = 1$, and so

$$\sigma_H(x)^n = \begin{cases} \sigma_H(x) & n \text{ is odd,} \\ 1 & n \text{ is even.} \end{cases}$$

This lets us simplify any element of $\mathrm{Alg}(S) = \mathrm{Alg}(\{\sigma_H\})$ to the form $a\sigma_H + b$. Now by choosing the identity map $\mathrm{id}(x) = x$ as the activation function for our neural networks, we have that $\mathcal{N}^{\mathrm{id}}(S) = \mathcal{N}^{\mathrm{id}}(\{\sigma_H\})$ is also exactly comprised of functions of the form $a\sigma_H + b$. Thus, despite $\mathrm{id}(x)$ being a polynomial, we have that $\mathrm{Alg}(S)$ and $\mathcal{N}^{\mathrm{id}}(S)$ have the same approximation power, and in fact we even have $\mathrm{Alg}(S) = \mathcal{N}^{\mathrm{id}}(S)$.

A key property in the above example seems to be that σ_H satisfies the polynomial equation $x^2 - 1 = 0$. Perhaps to appropriately describe the converse in general, one first needs to understand what polynomial equations the elements of S satisfy and from there deduce which activation functions would yield $\mathrm{Alg}(S) = \mathcal{N}^\sigma(S)$. We will not delve into this any further but comment that it appears that techniques from commutative algebra or algebraic geometry may be helpful in resolving this question.

Chapter 6

Properties of Set Pooling Networks

In this chapter we focus on understanding which functions on $\mathcal{F}(\Omega)$ can be represented via neural network models such as those of *PointNet*[40] and *Deep Sets* [62]. We will consider these architectures as functions of the form $F : \mathcal{F}(\Omega) \to \mathbb{R}^n$ and so will necessarily be *permutation-invariant* and *cardinality-agnostic*. Such functions can be used for point cloud regression and can be used for classification if composed with a softmax layer at the end. Tasks such as point cloud segmentation or point cloud generation are not directly addressed, but some of the results here may have relevance depending on architecture details.

6.1 PointNet, DeepSets, and Normalized-DeepSets

In addition to the max-pooling of PointNet and sum-pooling of DeepSets, we also take special interest in the case of average-pooling. For a point cloud $A \in \mathcal{F}(\mathbb{R}^n)$, we shall focus primarily on neural networks that have the following form,

$$\boldsymbol{\psi}_{\max}(A) = \boldsymbol{\rho}\left(\max_{\boldsymbol{a}\in A} \boldsymbol{\varphi}(\boldsymbol{a})\right), \quad \boldsymbol{\psi}_{\text{ave}}(A) = \boldsymbol{\rho}\left(\frac{1}{|A|}\sum_{\boldsymbol{a}\in A} \boldsymbol{\varphi}(\boldsymbol{a})\right), \quad \boldsymbol{\psi}_{\text{sum}}(A) = \boldsymbol{\rho}\left(\sum_{\boldsymbol{a}\in A} \boldsymbol{\varphi}(\boldsymbol{a})\right).$$

The first and last are just the invariant PointNet and DeepSets architectures respectively, and the middle one is what we will call *normalized-DeepSets*.

Here the function $\boldsymbol{\varphi} : \mathbb{R}^n \to \mathbb{R}^m$ is applied to each point of the point cloud A to produce features for each of them. These point-features then get aggregated by a permutation-invariant pooling function e.g. max/ave/sum-pooling (maximums are taken component-wise along the feature vectors). This global point cloud feature finally gets passed to $\boldsymbol{\rho} : \mathbb{R}^m \to \mathbb{R}^\ell$ to produce the output (we will usually take $\ell = 1$). In practice we want both $\boldsymbol{\rho}$ and $\boldsymbol{\varphi}$ to be neural networks, though we may occasionally consider them to be general functions. It may also be of interest to use some other sort of parametrized model with learnable parameters, but we do not explore such options here.

A key consequence of using these permutation-invariant pooling functions is that they ensure the input to $\boldsymbol{\rho}$ does not depend on the ordering of the point cloud elements. As a result, the output of $\boldsymbol{\rho}$ and hence the whole network is also permutation-invariant. Another consequence of this architecture design is that because the pooling operations scale to arbitrarily large finite cardinalities, the size of the input point cloud is not an issue. The fact that these models can accept arbitrarily large point clouds does not mean that cardinality is ignored. In fact, it is possible for two point clouds of drastically differing sizes two have nearly identical outputs or extremely different outputs. Understanding how these model's outputs vary with respect to their inputs will be one of the key focuses of this chapter.

6.2 Refactoring and Simplifying Notation

It will help to introduce some simplifying notation. As usual we let $\mathcal{F}(\Omega)$ denote the set of all nonempty finite subsets of a set Ω (i.e. point clouds in Ω), and we also let $\mathcal{F}^{\leq k}(\Omega)$ be the set of nonempty subsets of size $\leq k$, and $\mathcal{F}^k(\Omega)$ the set of k-point subsets. For a function $f : \Omega \to \mathbb{R}$ we define $\max_f, \operatorname{ave}_f, \operatorname{sum}_f : \mathcal{F}(\Omega) \to \mathbb{R}$ as the set-functions given by

$$\max_f(A) = \max_{\boldsymbol{a}\in A} f(\boldsymbol{a}), \quad \operatorname{ave}_f(A) = \frac{1}{|A|}\sum_{\boldsymbol{a}\in A} f(\boldsymbol{a}), \quad \operatorname{sum}_f(A) = \sum_{\boldsymbol{a}\in A} f(\boldsymbol{a}),$$

respectively. We make sense of this in the natural way if we use vector-valued $\boldsymbol{f}$ by operating component-wise. When convenient, we refer to these operations as max-neurons, averaging-neurons, and sum-neurons respectively.

With these conventions, we can refactor the PointNet, normalized-DeepSets, and DeepSets models respectively into the following concise form,

$$\boldsymbol{\psi}_{\text{max}} = \boldsymbol{\rho} \circ \max_{\boldsymbol{\varphi}}, \quad \boldsymbol{\psi}_{\text{ave}} = \boldsymbol{\rho} \circ \text{ave}_{\boldsymbol{\varphi}}, \quad \boldsymbol{\psi}_{\text{sum}} = \boldsymbol{\rho} \circ \text{sum}_{\boldsymbol{\varphi}}.$$

For simplicity of notation and to avoid redundancy when the same idea works for all three of these pooling cases, we will use pool to denote a generic pooling function such as max, ave, and sum. Similarly, we will use pool_f to stand for pool_f to denote a generic pooling function such as $\max_f$, ave_f, and sum_f. With this our neural network models take on the unified notation of,

$$\boldsymbol{\psi}_{\text{pool}} = \boldsymbol{\rho} \circ \text{pool}_{\boldsymbol{\varphi}}.$$

Collectively, we call such architectures **set pooling networks**.

6.3 Stability and Limits

Looking at ψ_{ave} and ψ_{sum}, it may at first seem that the use of average-pooling instead of sum-pooling makes no difference. Indeed, this is true when cloud cardinality is fixed since in that case the factor of $|A|^{-1}$ is constant and can be absorbed into the first layer of $\boldsymbol{\rho}$. However, it does make a difference when cardinality is free to change.

To see this, imagine a finer and finer even sampling of the unit circle given by a sequence of point clouds A_n. If we use average-pooling, then $\text{ave}_{\boldsymbol{\varphi}}(A_n)$ will converge to the average value of $\boldsymbol{\varphi}$ on the circle. Similarly, if we use max-pooling then $\max_{\boldsymbol{\varphi}}(A_n)$ will converge to the (component-wise) maximum of $\boldsymbol{\varphi}$ on the circle. But if we use sum-pooling, then the components of $\text{sum}_{\boldsymbol{\varphi}}(A_n)$ can potentially blow-up to $\pm\infty$. This can pose some theoretical issues. For example, if the components of $\boldsymbol{\varphi}$ are strictly positive functions and $\boldsymbol{\rho}$ is a non-constant linear mapping then $\psi_{\text{sum}}(A_n)$ can diverge. If $\boldsymbol{\rho}$ is oscillatory then the limit of $\psi_{\text{sum}}(A_n)$ may even be undefined. As a result, the subtle change introduced by cardinality-normalization will prove to be extremely helpful in the ensuing analysis.

For this reason and others, the mathematical theory for the un-normalized DeepSets model $\boldsymbol{\rho} \circ \text{sum}_{\boldsymbol{\varphi}}$ will prove difficult to treat in the same manner as the others, hence it will get a separate treatment.

6.4 Continuity and Extensions

From now on we will assume (Ω, d) is a compact metric space and when $\Omega \subseteq \mathbb{R}^n$ it will be compact and equipped with the Euclidean metric. Recall from Chapter 4 that the compactness of (Ω, d) implies that both $(\mathcal{K}(\Omega), d_H)$ and $(\mathcal{P}(\Omega), d_W)$ are compact metric spaces. They also contain copies of $\mathcal{F}(\Omega)$ as dense subsets via the injective mapping $i_{\mathcal{K}} : \mathcal{F}(\Omega) \to \mathcal{P}(\Omega)$ and $i_{\mathcal{P}} : \mathcal{F}(\Omega) \to \mathcal{P}(\Omega)$.

The following lemma that will show that the max-neurons, sum-neurons, and averaging-neurons continuously extend to $\mathcal{K}(\Omega)$, $\mathcal{M}(\Omega)$, and $\mathcal{P}(\Omega)$ respectively. As a result, so will the three architectures we are studying. As a result, this will let us analyze the PointNet and normalized-DeepSets architectures as continuous functions on compact metric spaces, which is mathematically a much better situation than studying them as set-theoretic functions on an un-metrized $\mathcal{F}(\Omega)$.

For $f \in \mathcal{C}(\Omega)$, define $\text{Max}_f : \mathcal{K}(\Omega) \to \mathbb{R}$, $\text{Sum}_f : \mathcal{M}(\Omega) \to \mathbb{R}$, and $\text{Ave}_f : \mathcal{P}(\Omega) \to \mathbb{R}$ as the functions given by

$$\text{Max}_f(K) = \max_{x \in K} f(x), \quad \text{Sum}_f(\mu) = \int_\Omega f \, \mathrm{d}\mu, \quad \text{Ave}_f(\mu) = \mathbb{E}_{x \sim \mu}[f(x)]$$

Of course, if $\mu \in \mathcal{P}(\Omega)$ then $\text{Ave}_f(\mu) = \int_\Omega f \, \mathrm{d}\mu = \text{Sum}_f(\mu)$ but $\text{Ave}_f(\mu)$ is undefined when μ is not a probability measure. We now prove the following lemma.

Lemma 6.1. *Let* (Ω, d) *be compact,* $f \in \mathcal{C}(\Omega)$. *Then*

a) $\mathrm{Max}_f \in \mathcal{C}(\mathcal{K}(\Omega))$ *and* $\mathrm{Max}_f \circ i_{\mathcal{K}} = \max_f$,

b) $\mathrm{Sum}_f \in \mathcal{C}(\mathcal{M}(\Omega))$ *and* $\mathrm{Sum}_f \circ i_{\mathcal{P}} = \mathrm{sum}_f$,

c) $\mathrm{Ave}_f \in \mathcal{C}(\mathcal{P}(\Omega))$ *and* $\mathrm{Ave}_f \circ i_{\mathcal{M}} = \mathrm{ave}_f$.

As a consequence, PointNet, DeepSets, and normalized-Deepsets continuously extend to $\mathcal{K}(\Omega)$, $\mathcal{M}(\Omega)$, *and* $\mathcal{P}(\Omega)$ *respectively. In particular, we can say that PointNet and normalized-DeepSets are uniformly continuous on* $\mathcal{F}_H(\Omega)$ *and* $\mathcal{F}_W(\Omega)$ *respectively.*

Proof. Assume (Ω, d) is compact and $f \in \mathcal{C}(\Omega)$. By definition, we immediately see the following.

a) Since $i_{\mathcal{K}}(A) = A$ for all $A \in \mathcal{F}(\Omega)$ we have

$$\mathrm{Max}_f(i_{\mathcal{K}}(A)) = \mathrm{Max}_f(A) = \max_{a \in A} f(a) = \max_f(A).$$

b) By linearity of integration and the Dirac delta property $\int f \, \mathrm{d}\delta_x = f(x)$ we get

$$\mathrm{Sum}_f(i_{\mathcal{M}}(A)) = \int f \, \mathrm{d}\left(\sum_{a \in A} \delta_a\right) = \sum_{a \in A} \int f \, \mathrm{d}\delta_a = \sum_{a \in A} f(a) = \mathrm{sum}_f(A).$$

c) Similarly we obtain

$$\mathrm{Ave}_f(i_{\mathcal{P}}(A)) = \int f \, \mathrm{d}\left(\frac{1}{|A|}\sum_{a \in A} \delta_a\right) = \frac{1}{|A|}\sum_{a \in A} \int f \, \mathrm{d}\delta_a = \frac{1}{|A|}\sum_{a \in A} f(a) = \mathrm{ave}_f(A).$$

The main work will be in establishing continuity of Max_f. For that case we will appeal directly to the Hausdorff metric. For the latter two cases we appeal to the weak-* topology.

a) Let $\epsilon > 0$. Since Ω is compact, f is uniformly continuous and so there is a $\delta > 0$ so that $|f(x) - f(y)| < \epsilon/2$ whenever $d(x, y) < 2\delta$. Now let $A, B \in \mathcal{K}(\Omega)$ and suppose

$d_H(A, B) < \delta$. By definition this means $A \subseteq B_\delta$ and $B \subseteq A_\delta$. So by the triangle inequality we have

$$|\text{Max}_f(A) - \text{Max}_f(B)| \leq |\text{Max}_f(A) - \text{Max}_f(A_\delta)| + |\text{Max}_f(A_\delta) - \text{Max}_f(B)|\,.$$

Since A_δ is compact there is a $p \in A_\delta$ so that $\text{Max}_f(A_\delta) = f(p)$. Observe that if $q \in K \subseteq A_\delta$ with $d(p, q) < 2\delta$ then $|f(p) - f(q)| < \epsilon/2$ and $f(p) = \text{Max}_f(A_\delta) \geq \text{Max}_f(K) \geq f(q)$. This implies $|\text{Max}_f(A_\delta) - \text{Max}_f(K)| < \epsilon/2$ whenever we can find such a $q \in K$. For $K = A$, note that since $p \in A_\delta$ there is an $a \in A \subseteq A_\delta$ such that $d(p, a) < \delta < 2\delta$ so, $|\text{Max}_f(A_\delta) - \text{Max}_f(A)| < \epsilon/2$. For $K = B$, note that B is compact so there is a $b \in B \subseteq A_\delta$ closest to p and so,

$$d(p, b) \leq d_H(A_\delta, B) \leq d_H(A_\delta, A) + d_H(A, B) < 2\delta,$$

which means $|\text{Max}_f(A_\delta) - \text{Max}_f(B)| < \epsilon/2$. Thus, $|\text{Max}_f(A) - \text{Max}_f(B)| < \epsilon$ as desired.

b) Recall that the topology we are using for $\mathcal{M}(\Omega)$ is the weak-* topology for measures. Since the weak-* topology is the coarsest topology such that that all maps $\mu \mapsto \int f \,\mathrm{d}\mu$ are continuous when $f \in \mathcal{C}(\Omega)$, it follows that $\text{Sum}_f(\mu) = \int f \,\mathrm{d}\mu$ is continuous on $\mathcal{M}(\Omega)$.

c) Recall that since (Ω, d) is compact that the topology induced by d_W on $\mathcal{P}(\Omega)$ coincides with the weak-* topology (Section 4.3). Thus for the same reason as for the previous case, we have Ave_f is continuous on $\mathcal{P}(\Omega)$

Lastly, it is clear that PointNet, DeepSets, and normalized-DeepSets continuously extend to $\mathcal{K}(\Omega)$, $\mathcal{M}(\Omega)$, and $\mathcal{P}(\Omega)$ by simply replacing max_f, sum_f, and ave_f with their corresponding extensions using the above. Since (Ω, d) compact implies both $\mathcal{K}(\Omega)$ and $\mathcal{P}(\Omega)$ are compact, and continuous functions on compact metric spaces are uniformly continuous, we can also deduce that PointNet and normalized-DeepSets are uniformly continuous on $\mathcal{F}_H(\Omega)$ and $\mathcal{F}_W(\Omega)$. $\square$

This result will be helpful in applying the UATs from Chapter 5 to our set pooling networks.

6.5 Lipschitz Properties

By refining the continuity arguments for Max_f and Ave_f, we can say more. In particular, we can deduce Lipschitz properties of these maps when f is Lipschitz. Before being able to do so for the case of Max_f it will help to establish an analogue of the Kantorovich-Rubinstein formula for the Hausdorff metric (for which the author could not find a reference for in the literature).

Proposition 6.2. *Let (X, d) be a metric space, and $A, B \in \mathcal{K}(X)$. Then,*

$$d_H(A,B) = \sup_{\substack{f:X\to\mathbb{R}\\ f\in\mathrm{Lip}_1(X)}} \left|\mathrm{Max}_f(A) - \mathrm{Max}_f(B)\right|$$

Proof. Let $D_q(x) = d(x,q)$ and note that $-D_q$ is 1-Lipschitz for any $q \in X$. Thus,

$$\begin{aligned}
\sup_{f\in\mathrm{Lip}_1(X)} \left|\mathrm{Max}_f(A) - \mathrm{Max}_f(B)\right| &\geq \sup_{\substack{g=-D_q\\ q\in X}} \left|\mathrm{Max}_g(A) - \mathrm{Max}_g(B)\right| \\
&= \sup_{q\in X} \left|\mathrm{Min}_{D_q}(A) - \mathrm{Min}_{D_q}(B)\right| \\
&= \sup_{q\in X} \left|\min_{a\in A} d(a,q) - \min_{b\in B} d(b,q)\right| \\
&= \sup_{q\in X} \left|d(A,q) - d(B,q)\right| \\
&= d_H(A,B)
\end{aligned}$$

To show the reverse inequality, consider any $f \in \mathrm{Lip}_1(X)$ and suppose without loss of generality that $\mathrm{Max}_f(A) \geq \mathrm{Max}_f(B)$. By compactness, we know there exist an $a^* \in A$

and $b^* \in B$ so that $f(a^*) = \mathrm{Max}_f(A) \geq \mathrm{Max}_f(B) = f(b^*)$. This and $f \in \mathrm{Lip}_1(X)$ imply that for any $b \in B$ we have

$$\big|\mathrm{Max}_f(A) - \mathrm{Max}_f(B)\big| = f(a^*) - f(b^*) \leq f(a^*) - f(b) \leq d(a^*, b)$$

Finally observe that if we replace f with the negative distance functional $g = -D_{a^*} \in \mathrm{Lip}_1(X)$ then we can see that we have not decreased the absolute difference,

$$\big|\mathrm{Max}_g(A) - \mathrm{Max}_g(B)\big| = \big|\mathrm{Min}_{D_{a^*}}(A) - \mathrm{Min}_{D_{a^*}}(B)\big| \geq |0 - d(a^*, b)| = d(a^*, b)$$

Since $f \in \mathrm{Lip}_1(X)$ was arbitrary, this completes the required reverse inequality

$$\sup_{f \in \mathrm{Lip}_1(X)} \big|\mathrm{Max}_f(A) - \mathrm{Max}_f(B)\big| \leq \sup_{\substack{g = -D_q \\ q \in X}} \big|\mathrm{Max}_g(A) - \mathrm{Max}_g(B)\big|.$$

□

With this proposition we can, we now readily obtain the following result.

Lemma 6.3. *Suppose (Ω, d) is compact and that $f : \Omega \to \mathbb{R}$ is L-Lipschitz. Then $\mathrm{Max}_f \in \mathcal{C}(\mathcal{K}(\Omega))$ and $\mathrm{Ave}_f \in \mathcal{C}(\mathcal{P}(\Omega))$ are also L-Lipschitz.*

Proof. Assume $f : \Omega \to \mathbb{R}$ is L-Lipschitz with $L > 0$ and note that $\widetilde{f} = \frac{1}{L} f$ is 1-Lipschitz. So for $\mu, \nu \in \mathcal{P}(\Omega)$ the Kantorovich-Rubinstein formula tells us that

$$\begin{aligned}
|\mathrm{Ave}_f(\mu) - \mathrm{Ave}_f(\nu)| &= L \left|\mathrm{Ave}_{\widetilde{f}}(\mu) - \mathrm{Ave}_{\widetilde{f}}(\nu)\right| \\
&= L \left| \int \widetilde{f}\, \mathrm{d}\mu - \int \widetilde{f}\, \mathrm{d}\nu \right| \\
&\leq L \sup_{g \in \mathrm{Lip}_1(X)} \left| \int g \, \mathrm{d}\mu - \int g \, \mathrm{d}\nu \right| \\
&= L\, d_W(\mu, \nu).
\end{aligned}$$

Similarly, for $A, B \in \mathcal{K}(\Omega)$ we have by Proposition 6.2 that

$$\begin{aligned}
|\mathrm{Max}_f(A) - \mathrm{Max}_f(B)| &= L\left|\mathrm{Max}_{\widetilde{f}}(A) - \mathrm{Max}_{\widetilde{f}}(B)\right| \\
&\leq L \sup_{g \in \mathrm{Lip}_1(X)} \left|\mathrm{Max}_g(A) - \mathrm{Max}_g(B)\right| \\
&= L\, d_H(A, B).
\end{aligned}$$

Lastly, $L = 0$ iff f is constant. Thus, if $L = 0$ then Ave_f and Max_f are constant and so both are 0-Lipschitz. □

Because of this, we can now show that a PointNet or normalized-DeepSets architecture must be Lipschitz whenever all its activation functions are Lipschitz.

Theorem 6.4. *Let $\Omega \subseteq \mathbb{R}^N$ be compact. Then PointNet and normalized-DeepSets networks with Lipschitz activation functions are Lipschitz on $\mathcal{F}_H(\Omega)$ and $\mathcal{F}_W(\Omega)$ respectively.*

Proof. As usual, assume all Euclidean spaces involved have the Euclidean metric. Recall that the networks we are considering are of the form $\boldsymbol{\psi} = \boldsymbol{\rho} \circ \mathrm{pool}_{\boldsymbol{f}}$ where $\boldsymbol{f} : \Omega \to \mathbb{R}^k$ and $\boldsymbol{\rho} : \mathbb{R}^k \to \mathbb{R}^m$ are feed-forward neural networks. Since $\boldsymbol{f}$ and $\boldsymbol{\rho}$ are feed-forward neural networks, they are composition of affine transforms (which are always Lipschitz) and nonlinear transforms. The nonlinear transform must have the form

$$\boldsymbol{\sigma}(x_1, x_2, \ldots, x_n) = (\sigma_1(x_1), \sigma_2(x_2), \ldots, \sigma_n(x_n))$$

and by assumption all the activation functions σ_i are L_i-Lipschitz for some $0 \leq L_i < \infty$.

Let $L = \max_i L_i$. It is easy to check that these nonlinear transforms are also Lipschitz:

$$\begin{aligned}
\|\boldsymbol{\sigma}(\boldsymbol{x}) - \boldsymbol{\sigma}(\boldsymbol{y})\| &= \sqrt{\sum_{i=1}^{n} |\sigma_i(x_i) - \sigma_i(y_i)|^2}, \\
&\leq \sqrt{\sum_{i=1}^{n} L^2 |x_i - y_i|^2}, \\
&= L \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2}, \\
&= L \|x - y\| .
\end{aligned}$$

Since compositions of Lipschitz functions are Lipschitz, it follows that both $\boldsymbol{f}$ and $\boldsymbol{\rho}$ are Lipschitz. By Lemma 6.3, the components f_i of $\boldsymbol{f}$ being Lipschitz implies $\max_{f_i}$ and ave_{f_i} are Lipschitz. By a similar argument to the above, we can show the vector-valued versions $\max_{\boldsymbol{f}}$ and $\mathrm{ave}_{\boldsymbol{f}}$ are also Lipschitz. Thus $\boldsymbol{\psi} = \boldsymbol{\rho} \circ \mathrm{pool}_{\boldsymbol{f}}$ is Lipschitz for the cases of $\mathrm{pool} \in \{\max, \mathrm{ave}\}$ since it is a composition of Lipschitz functions. $\square$

With knowledge of the weights of $\boldsymbol{f}$ and $\boldsymbol{\rho}$ one could compute Lipschitz constants for the affine maps. If one also knew Lipschitz constant for the activation functions, then it would be possible to directly compute Lipschitz constants for $\boldsymbol{f}$ and $\boldsymbol{\rho}$. This would be especially simple for networks with only ReLU activation functions as ReLU trivially has Lipschitz constant 1. From there, one could then follow the proof to compute Lipschitz constants for the whole set pooling network.

This results is essentially a more general quantitative alternative to Theorem 2 of [40] since the latter shows how to find $A, C \in \mathcal{F}(\Omega)$ so that the output of PointNet is constant for any $A \subseteq B \subseteq C$. On the other hand, Theorem 6.4 works not only for PointNet but also normalized-DeepSets. Moreover, it allows one to bound how different the outputs can be based solely on the network's Lipschitz constant and how similar the points clouds are with respect to the corresponding metrics. Furthermore, since $\mathcal{F}(\Omega)$ is dense in $\mathcal{K}(\Omega)$

and $\mathcal{P}(\Omega)$, we can actually extend Theorem 6.4 to apply the extension networks on $\mathcal{K}(\Omega)$ and $\mathcal{P}(\Omega)$ while keeping the same Lipschitz constant.

One implication of this is that if one were to sample a mesh sufficiently well and knew the Lipschitz constant K of the whole PointNet/normalized-DeepSets network, then it would be possible to bound how big of a difference in the outputs two different samplings of the same object would produce. This bound would not depend on the cardinality, but on the quality of the sampling with respect to the metrics d_H and/or d_W.

Finally, one of the problems that has plagued deep learning models since their explosion in popularity is the problem of adversarial examples [18]. Adversarial examples are inputs which are modified in a virtually imperceptible way (to humans) that result in extremely erroneous output. One recent approach to dealing with this issue is to compute and then take advantage of the Lipschitz constant of the network (computed by taking the product of layer-wise Lipschitz constants). With Lemma 6.3 and Theorem 6.4, this opens up the possibility of applying a similar technique for point clouds in way that works for variable cardinality.

6.6 Separation Properties

As we saw in Chapter 5, it is valuable property for a family of functions to separate points, particularly for questions of function approximation. In this section we will explore the separation capabilities of the set pooling networks which we are studying.

Lemma 6.5 (Separation Lemma)**.** *Let $\Omega \subseteq \mathbb{R}^n$ be compact and $\sigma \in \mathcal{C}(\mathbb{R})$ non-polynomial. Then the set of functions $S_{\text{Max}} = \{\text{Max}_f \mid f \in \mathcal{N}_n^\sigma\}$ and $S_{\text{Ave}} = \{\text{Ave}_f \mid f \in \mathcal{N}_n^\sigma\}$ separate points of $\mathcal{K}(\Omega)$ and $\mathcal{P}(\Omega)$ respectively and contain constants.*

Proof. Let d denote the Euclidean distance. First note that by choosing weights correctly, we can find a constant function $h = \sigma(c) \in \mathcal{N}_n^\sigma$ for some $c \in \mathbb{R}$. Since σ is not a polynomial, there is a choice of c for which $\sigma(c) \neq 0$. This means $\text{Max}_h \in S_{\text{Max}}$ and

$\mathrm{Ave}_h \in S_{\mathrm{Ave}}$ are both constant. Now we just need to show that S_{Max} and S_{Ave} separate points.

(S_{Max} separates points): Let $A, B \in \mathcal{K}(\Omega)$ with $A \neq B$. Without loss of generality, $A \setminus B \neq \varnothing$ so choose $a \in A \setminus B$. Let $f(x) = \min\{1, d(x,B)/d(a,B)\}$ and note that $f(a) = 1$, $f(B) = \{0\}$ and $f(\Omega) = [0,1]$. By the classical UAT [32], $\mathcal{N}_n^\sigma$ is uniform dense in $\mathcal{C}(\Omega)$, so there is a $g \in \mathcal{N}_n^\sigma$ so that $|f(x) - g(x)| < 1/2$ for all $x \in \Omega$. Note $\mathrm{Max}_g \in S_{\mathrm{Max}}$ and that $\mathrm{Max}_g(A) > 1/2$ and $\mathrm{Max}_g(B) < 1/2$. Since A and B were arbitrary, this shows S_{Max} separates point in $\mathcal{K}(X)$.

(S_{Ave} separates points): Given $\mu_1, \mu_2 \in \mathcal{P}(\Omega)$ with $\mu_1 \neq \mu_2$, by the Hahn-Banach separation theorem there exists a weak-* continuous linear functional $L : M(\Omega) \to \mathbb{R}$ that separates them. Let $\delta = |L(\mu_1) - L(\mu_2)|$. The topological dual of $M(\Omega)$ with the weak-* topology is equivalent to $\mathcal{C}(\Omega)$ and so there is an $f \in \mathcal{C}(\Omega)$ so that $L(\eta) = \int f \, \mathrm{d}\eta$ for all $\eta \in M(\Omega)$. Since $\mathcal{N}_n^\sigma$ is uniform dense in $\mathcal{C}(\Omega)$ there is a $g \in \mathcal{N}_n^\sigma$ so the that $|f(x) - g(x)| < \delta/2$ for all $x \in \Omega$. Define $J(\eta) = \int g \, \mathrm{d}\eta$. Then for all $\eta \in \mathcal{P}(\Omega)$ we have $|L(\eta) - J(\eta)| \leq \int |f - g| \, \mathrm{d}\eta < \frac{\delta}{2} \int \mathrm{d}\eta = \delta/2$. Applying the triangle inequality we obtain,

$$\delta \leq \underbrace{|L(\mu_1) - J(\mu_1)|}_{<\delta/2} + |J(\mu_1) - J(\mu_2)| + \underbrace{|J(\mu_2) - L(\mu_2)|}_{<\delta/2}$$

Thus $0 < |J(\mu_1) - J(\mu_2)|$ and so $J = \mathrm{Ave}_g \in S_{\mathrm{Ave}}$ separates μ_1 and μ_2. Since μ_1 and μ_2 were arbitrary, it follows that S_{Ave} separates points in $\mathcal{P}(\Omega)$. $\square$

Chapter 7

Universality and Limitations of PointNet and DeepSets

In this chapter study what functions $F : \mathcal{F}(\Omega) \to \mathbb{R}$ can and cannot be approximated by our set pooling networks of interest.

7.1 Notation for Set Pooling Network Classes

The set pooling networks we described can have varying degrees of complexity depending on the architecture used in their subnetworks. As with classical feed-forward neural networks, one expects that with more neurons comes increased expressive power and that more layers increase practical effectiveness. To aid in our study of neural network representations of functions $F : \mathcal{F}(\Omega) \to \mathbb{R}$, it well help to introduce some compact notation for encapsulating families of set pooling networks of varying complexity.

Let $\text{pool}_S = \left\{\text{pool}_f \mid f \in S\right\}$. Using this notation, we make the following inductive families of functions based off of set pooling networks:

$$\begin{aligned}
&\text{height 1:} \qquad \mathcal{N}^{\boldsymbol{\sigma}}_{n,\text{pool}} := \mathcal{N}^{\boldsymbol{\sigma}}_{n},\\
&\text{height 2:} \qquad \mathcal{N}^{\boldsymbol{\sigma};\boldsymbol{\tau}}_{n,\text{pool}} := \mathcal{N}^{\boldsymbol{\tau}}\left(\text{pool}_{\mathcal{N}^{\boldsymbol{\sigma}}_{n,\text{pool}}}\right),\\
&\text{height 3:} \qquad \mathcal{N}^{\boldsymbol{\sigma};\boldsymbol{\tau};\boldsymbol{\rho}}_{n,\text{pool}} := \mathcal{N}^{\rho}\left(\text{pool}_{\mathcal{N}^{\boldsymbol{\sigma};\boldsymbol{\tau}}_{n,\text{pool}}}\right),\\
&\qquad\qquad\qquad\qquad \vdots
\end{aligned}$$

Note that the first (base case) family of neural networks $\mathcal{N}^{\boldsymbol{\sigma}}_{n,\text{pool}}$ are just typical feed-forward networks and accept inputs which live in $\mathbb{R}^n$. The second family $\mathcal{N}^{\boldsymbol{\sigma};\boldsymbol{\tau}}_{n,\text{pool}}$ is comprised of networks that accept inputs which live in $\mathcal{F}(\mathbb{R}^n)$. The networks from the third family $\mathcal{N}^{\boldsymbol{\sigma};\boldsymbol{\tau};\boldsymbol{\rho}}_{n,\text{pool}}$ accept inputs which live in $\mathcal{F}(\mathcal{F}(\mathbb{R}^n))$. And so on. As a result, each subsequent neural network class has fundamentally different capabilities compared to the last. As height increases, the complexity of the input structures increases as well. As a result, we say we are building "higher" networks.

The notation is such that "depth" of a "strata" of the network is distinct from the "height" of the overall network. This is emphasised by the usage of a semi-colons in the superscript to separate the different "strata" of the networks, whereas the number of activation functions denoted between colons indicates the depth of that particular "stratum." For example, the class below is of height 3 and has depths of 3, 2, and 1 in each stratum respectively.

$$\mathcal{N}^{\boldsymbol{\sigma};\boldsymbol{\tau};\boldsymbol{\rho}}_{n,\text{pool}} = \mathcal{N}^{\sigma_1,\sigma_2,\sigma_3;\tau_1,\tau_2;\rho}_{n,\text{pool}}$$

It is worth pointing out that with this nomenclature, the works of PointNet [40] and DeepSets [62] focus on height 2 (though they do not use this language).

As we saw with Lemma 6.1, we can extend all the operations of our neural networks to larger spaces in a natural way. Let Pool_f be shorthand for Max_f, Sum_f, or Ave_f. As before we let $\text{Pool}_S = \{\text{Pool}_f \mid f \in S\}$ and build a hierarchy of network classes.

$$
\begin{aligned}
&\text{height 1:} \qquad \mathcal{N}^{\boldsymbol{\sigma}}_{n,\text{Pool}} := \mathcal{N}^{\boldsymbol{\sigma}}_n\,, \\
&\text{height 2:} \qquad \mathcal{N}^{\boldsymbol{\sigma};\boldsymbol{\tau}}_{n,\text{Pool}} := \mathcal{N}^{\boldsymbol{\tau}}\left(\text{Pool}_{\mathcal{N}^{\boldsymbol{\sigma}}_{n,\text{Pool}}}\right), \\
&\text{height 3:} \qquad \mathcal{N}^{\boldsymbol{\sigma};\boldsymbol{\tau};\boldsymbol{\rho}}_{n,\text{Pool}} := \mathcal{N}^{\boldsymbol{\rho}}\left(\text{Pool}_{\mathcal{N}^{\boldsymbol{\sigma};\boldsymbol{\tau}}_{n,\text{Pool}}}\right), \\
&\qquad\qquad\qquad\vdots
\end{aligned}
$$

These extensions have a slightly more complicated meaning. For example, networks from $\mathcal{N}^{\boldsymbol{\sigma};\boldsymbol{\tau}}_{n,\text{Max}}$ take compact subsets of $\mathbb{R}^n$ as inputs but networks from $\mathcal{N}^{\boldsymbol{\sigma};\boldsymbol{\tau};\boldsymbol{\rho}}_{n,\text{Max}}$ take compact collections of compact subsets of $\mathbb{R}^n$ i.e. elements of $\mathcal{K}(\mathcal{K}(\mathbb{R}^n))$. Similarly, $\mathcal{N}^{\boldsymbol{\sigma};\boldsymbol{\tau}}_{n,\text{Ave}}$ takes elements of $\mathcal{P}(\mathbb{R}^n)$ as inputs but networks from $\mathcal{N}^{\boldsymbol{\sigma};\boldsymbol{\tau};\boldsymbol{\rho}}_{n,\text{Max}}$ takes elements of $\mathcal{P}(\mathcal{P}(\mathbb{R}^n))$.

7.2 Point Cloud UAT

The following theorems show that one hidden layer in the inner network and one hidden layer in the outer network suffice to prove the universal approximation theorems for PointNet and normalized-DeepSets.

Theorem 7.1. *Let $\Omega \subseteq \mathbb{R}^n$ be compact and $\sigma, \tau \in \mathcal{C}(\mathbb{R})$ non-polynomial. Then $\mathcal{N}^{\sigma;\tau}_{n,\text{Max}}$ and $\mathcal{N}^{\sigma;\tau}_{n,\text{Ave}}$ are uniform dense in $\mathcal{C}(A)$ and $\mathcal{C}(B)$ respectively, whenever $A \subseteq \mathcal{K}(\Omega)$ and $B \subseteq \mathcal{P}(\Omega)$ are closed subsets.*

Proof. Since Ω is compact, we have that $\mathcal{K}(\Omega)$ and $\mathcal{P}(\Omega)$ are compact metric spaces. Since A and B are closed subsets of a compact Hausdorff set, they must also be compact Hausdorff. By Lemma 6.1 and Lemma 6.5 we know $\text{Max}_{\mathcal{N}^{\boldsymbol{\sigma}}_n}$ and $\text{Ave}_{\mathcal{N}^{\boldsymbol{\sigma}}_n}$ are comprised of continuous functions and separate points on their domains. Those properties are preserved when restricting to the subsets A and B. Finally, the topological-UAT (Theorem 5.2) applied to $\mathcal{N}^{\tau}(\text{Max}_{\mathcal{N}^{\boldsymbol{\sigma}}_n})$ on A and $\mathcal{N}^{\tau}(\text{Ave}_{\mathcal{N}^{\boldsymbol{\sigma}}_n})$ on B yields the desired result. □

For $f : \mathcal{F}(\Omega) \to \mathbb{R}$ the following result tells us that under mild hypothesis that (1) PointNets can uniformly approximate f iff f is d_H-uniformly continuous and (2) normalized-DeepSets can uniformly approximate f iff f is d_W-uniformly continuous.

Theorem 7.2 (Point-Cloud-UAT)**.** *Let $\Omega \subseteq \mathbb{R}^n$ be compact. If $\sigma, \tau \in \mathcal{C}(\mathbb{R})$ non-polynomial, then the uniform closure of $\mathcal{N}^{\sigma;\tau}_{n,\max}$ and $\mathcal{N}^{\sigma;\tau}_{n,\text{ave}}$ within $\mathcal{B}(\mathcal{F}(\Omega))$ is $\mathcal{U}(\mathcal{F}_H(\Omega))$ and $\mathcal{U}(\mathcal{F}_W(\Omega))$ respectively.*

Proof. The metric completions of $\mathcal{F}_H(\Omega)$ and $\mathcal{F}_W(\Omega)$ are isometrically isomorphic to the compact metric spaces $(\mathcal{K}(\Omega), d_H)$ and $(\mathcal{P}(\Omega), d_W)$ by Theorems 4.1 and 4.2. Hence $\mathcal{F}_H(\Omega)$ and $\mathcal{F}_W(\Omega)$ are precompact. By Lemma 6.1 we have that $\max_{\mathcal{N}^\sigma_n}$ and $\text{ave}_{\mathcal{N}^\sigma_n}$ continuously extend to $\text{Max}_{\mathcal{N}^\sigma_n}$ and $\text{Ave}_{\mathcal{N}^\sigma_n}$ which are separating families on the completions of $\mathcal{F}_H(\Omega)$ and $\mathcal{F}_W(\Omega)$ (resp.) by Lemma 6.5. Finally, by Theorem 5.4 we have that the uniform closures of $\mathcal{N}^\tau(\max_{\mathcal{N}^\sigma_n})$ and $\mathcal{N}^\tau(\text{ave}_{\mathcal{N}^\sigma_n})$ are $\mathcal{U}(\mathcal{F}_H(\Omega))$ and $\mathcal{U}(\mathcal{F}_W(\Omega))$ respectively. □

It's worth noting that for the universality of normalized-DeepSets we could have directly used Stinchcombe's Corollary 5.1.2 from [50] which is a UAT for neural networks on locally convex spaces. However, this result would not directly work for PointNets so we chose the above route for consistency of technique and to be self-contained.

We now prove as a corollary a refinement of the universal approximation theorems of [40] and [62], both of which were for the case of k-point point clouds (for fixed k). In this version of the theorem, we are able to restrict the depth of the neural network to just three hidden layers (when counting the pooling). Additionally, as with the above, this result simultaneously establishes which functions *cannot* be uniformly approximated by these architectures.

Corollary 7.3. *Let $\Omega \subseteq \mathbb{R}^n$ be compact. If $\sigma, \tau \in \mathcal{C}(\mathbb{R})$ non-polynomial, then the uniform closure of $\mathcal{N}^{\sigma;\tau}_{n,\max}$ and $\mathcal{N}^{\sigma;\tau}_{n,\text{ave}}$ within $\mathcal{B}(\mathcal{F}^k(\Omega))$ are $\mathcal{U}(\mathcal{F}^k_H(\Omega))$ and $\mathcal{U}(\mathcal{F}^k_W(\Omega))$ respectively.*

Proof. Apply Theorem 7.2 and restrict from $\mathcal{F}_H(\Omega)$ and $\mathcal{F}_W(\Omega)$ to $\mathcal{F}_H^k(\Omega)$ and $\mathcal{F}_W^k(\Omega)$. □

7.3 Theoretical Issues with Standard DeepSets

To use the universal approximation results developed here, it was key for there to exists an underlying compact Hausdorff space on which to study the networks. In the case of PointNet and normalized-DeepSets, the compact Hausdorff space was not obvious, but we identified the appropriate extension spaces $\mathcal{K}(\Omega)$ and $\mathcal{P}(\Omega)$ to which we could unambiguously and continuously extend the networks. This gave a partial answer to the DeepSets extension conjecture of [62] in the sense that if we replace sum-pooling with max-pooling then these networks can be made to theoretically accept sets of uncountably infinite cardinality (so long as they are compact i.e. members of $\mathcal{K}(\Omega)$). Similarly, for sufficiently nice sets like compact smooth manifolds $M \subseteq \mathbb{R}^n$ one can feed the manifold's uniform probability measure U_M of M to the extension of normalized-DeepSets to $\mathcal{P}(\Omega)$.

However, we were not able to do this for standard DeepSets. This is because, generally speaking, there is no compact Hausdorff space to which we can continuously extend sum_f for all reasonable neural networks f. This is because a continuous function on a compact space is necessarily a bounded function but $\text{sum}_f : \mathcal{F}(\Omega) \to \mathbb{R}$ is generally unbounded. To illustrate this, let $\Omega = [0,1] \subseteq \mathbb{R}$ and let f be a continuous function, e.g. a neural network, which is not identically zero. Then there must be a closed interval K on which $|f| > \epsilon$ for some $\epsilon > 0$. From this we can see that $\text{sum}_f(A)$ for $A \subseteq K$ grows arbitrarily large in absolute value as the size of A increases. Thus, in this case, every sum_f except sum_0 is unbounded on $\mathcal{F}(\Omega)$ meaning there is no compact extension space. Moreover, because of the unboundedness of generic $\sum_f$, we cannot even use the Theorem 5.6.

One potential way to address this issue would be to only consider bounded non-linearities such as the sigmoid or arctan as a way to control the wild behavior as cloud size gets large. Even so, it is not immediately clear what the natural underlying compact Hausdorff space for such a model would be and it may very well depend on the activation functions. Alternatively, alternative tools may be needed that don't depend on compactness.

7.4 Limitations

Unlike the classical universal approximation theorem, we should not expect to be able uniformly approximate all of $\mathcal{C}(\mathcal{F}_H(\Omega))$ or $\mathcal{C}(\mathcal{F}_W(\Omega))$. When $\Omega \subseteq \mathbb{R}^n$ is compact, we know by Theorem 7.2 that PointNet and normalized-DeepSets can only approximate the uniformly continuous functions with respect to d_H and d_W. since their elements might not even be bounded functions. For example, $\alpha_K(A) = d_H(A, K)^{-1}$ is unbounded but continuous on $\mathcal{F}_H(\Omega)$ whenever $K \in \mathcal{K}(\Omega)$ but $K \notin \mathcal{F}(\Omega)$. Subtler still, we do not even obtain all elements of $\mathcal{C}_b(\mathcal{F}_H(\Omega))$ and $\mathcal{C}_b(\mathcal{F}_W(\Omega))$. As an example of this, observe that $\beta_K = \sin \circ \alpha_K$ is bounded and is also continuous on $\mathcal{F}_H(\Omega)$ because α_K is. However, β_K cannot be *uniformly* continuous because it becomes catastrophically oscillatory as we approach K. So both α_K and β_K are not uniformly d_H-continuous and hence cannot be uniformly approximated by PointNets. Analogous problematic constructions can be made for d_W and normalized-DeepSets by letting $\alpha_\mu(A) = d_W(A, \mu)^{-1}$ where μ not an empirical measure.

We'll now compare the representational power of these two architectures. Let $\Omega \subseteq \mathbb{R}^N$ be compact. We define the point cloud diameter function $\operatorname{Diam} : \mathcal{F}(\Omega) \to \mathbb{R}$ and point cloud center-of-mass function $\operatorname{Cent} : \mathcal{F}(\Omega) \to \mathbb{R}^N$ by $\operatorname{Diam}(A) = \max_{\boldsymbol{x},\boldsymbol{y}\in A} d(\boldsymbol{x}, \boldsymbol{y})$ and $\operatorname{Cent}(A) = \operatorname{ave}_{\mathrm{id}}(A) = \frac{1}{|A|}\sum_{\boldsymbol{x}\in A} \boldsymbol{x}$.

For the following theorem, as with many others, the reader may assume Ω is a the closed unit ball or closed cube $[0, 1]^N$ if convenient. However, we shall formulate and prove the result in the general setting.

Theorem 7.4. *Let (Ω, d) be an infinite compact metric space with no isolated points. Then a function $f : \mathcal{F}(\Omega) \to \mathbb{R}^n$ is continuous with respect to both d_H and d_W iff it is constant. As a corollary,* Diam *is uniformly approximable by PointNet networks but not by normalized-DeepSets networks and* Cent *is uniformly approximable by normalized-DeepSets networks but not PointNet networks.*

Proof. Assume $f : \mathcal{F}(\Omega) \to \mathbb{R}^n$ is both d_H-continuous and d_W-continuous. Let $A \in \mathcal{F}(\Omega)$ and let $p \in A$. For each $n = 1, 2, \ldots$ choose $A'_n \in \mathcal{F}(\Omega)$ to be an n-point set contained within the $1/n$-ball around p. We can do this because Ω is infinite without isolated points. Now let $A_n = A'_n \cup (A \setminus \{p\})$.

Observe that $A_n \overset{d_H}{\to} A$ and $A_n \overset{d_W}{\to} \{p\}$. Thus by continuity,

$$f(A) = f\left(\overset{d_H}{\lim_{n\to\infty}} A_n\right) = f\left(\overset{d_W}{\lim_{n\to\infty}} A_n\right) = f(\{p\})$$

Note that $A \in \mathcal{F}(\Omega)$ was arbitrary, hence f must always give A and any $\{p\} \subseteq A$ the same value. Now if $B, C \in \mathcal{F}(\Omega)$ with $q \in B$ and $r \in C$, then

$$f(B) = f(\{q\}) = f(\{q, r\}) = f(\{r\}) = f(C).$$

Thus f must be constant. Thus, since constant maps are always continuous, we conclude that a functions is continuous on $\mathcal{F}(\Omega)$ with respect to both metrics iff it is constant.

It can be shown that $\mathrm{Diam} : \mathcal{K}(\Omega) \to \mathbb{R}$ is 2-Lipschitz [17] i.e. it satisfies

$$|\mathrm{Diam}(A) - \mathrm{Diam}(B)| \leq 2d_H(A, B)$$

for compact A and B. Hence, $\mathrm{Diam} \in \mathcal{U}(\mathcal{F}_H(\Omega))$.

Next we show that $\mathrm{Cent} \in \mathcal{U}(\mathcal{F}_W(\Omega))$ by showing that each of its components are 1-Lipschitz. To see why, note that the i-th coordinate projection map π_i is clearly 1-Lipschitz, hence ave_{π_i} is also 1-Lipschitz (by Lemma 6.3), and that the i-th component of $\mathrm{Cent}(A)$ is given by ave_{π_i}.

Thus, Diam and Cent are uniformly continuous on $\mathcal{F}_H(\Omega)$ and $\mathcal{F}_W(\Omega)$ (resp.) and so the result follows from the above and Theorem 7.2.

□

While it is interesting to know that these neural networks describe fundamentally different kinds of functions when we allow for unbounded cloud cardinality, in practice there is always a bound due to computational resource limitations. Nevertheless, the above result sheds some light on the radically different nature of the PointNet and normalized-DeepSets models in the limit of large cloud size. However, even when limiting attention to point clouds of fixed cardinality it turns out that there are simple functions which PointNet cannot approximate.

In [54] it was shown in the proof of Theorem B.1 that $\mathrm{sum}_{\mathrm{id}}(A) = \sum_{x \in A} x$ cannot be *exactly* represented by PointNet models $\rho \circ \max_{\boldsymbol{f}}$ on $\mathcal{F}^k(\mathbb{R})$ if the output dimension of $\boldsymbol{f}$ is smaller than k. They do not address if exact representation is possible with a larger output dimension for $\boldsymbol{f}$ or whether such functions can be approximated by PointNet models instead. Of course, the same issue will apply to the center-of-mass function $\mathrm{ave}_{\mathrm{id}}$ as it differs by a constant factor from $\mathrm{sum}_{\mathrm{id}}$ when cloud size is held fixed.

The next result shows that even when we bound the cloud cardinality, PointNet still cannot uniformly approximate many simple functions such as the center-of-mass function. In fact, under mild assumptions, PointNet cannot in general uniformly approximate averages of continuous functions on point clouds either. The minimum approximation error for such tasks can actually be bounded and this bound is described by the following theorem (which we will prove later).

Theorem 7.5. *Let (Ω, d) be compact with no isolated points and $f \in \mathcal{C}(\Omega, \mathbb{R}^N)$. Then for every d_H-continuous $F : \mathcal{F}^{\leq k}(\Omega) \to \mathbb{R}^d$ there exists $A \in \mathcal{F}^k(\Omega)$ such that*

$$\|F(A) - \mathrm{ave}_f(A)\| \geq \frac{k-2}{2k} \mathrm{Diam}(f(\Omega)).$$

Since PointNet architectures are d_H-continuous they must suffer from the above uniform-norm error lower-bound. This means that when cloud cardinality is at least three, they cannot uniformly approximate functions of the form ave_f for continuous f on e.g. the unit ball of $\mathbb{R}^n$. In particular, they cannot even uniformly approximate the center-of-mass function because this corresponds to the case where $\Omega \subseteq \mathbb{R}^N$ and f is the identity map.

It is possible to see the intuition for why PointNet cannot uniformly approximate Cent by considering the following. For a k-element point cloud, continuously move $k-2$ of those points towards either of the remaining two points p or q. This ultimately produces the same 2-element set $\{p, q\}$ in a d_H-continuous way. By d_H-continuity, this means PointNet must output similar predictions for the center-of-mass as we approach $\{p, q\}$ along either of these two paths. However, this leads to conflicting estimates for the center of mass along the way.

To obtain the explicit error lower bound in Theorem 7.5, we need a slightly more detailed result.

Lemma 7.6. *Suppose (Ω, d) has no isolated points and $f \in \mathcal{C}(\Omega, \mathbb{R}^N)$ with $f(p) \neq f(q)$ for some $p, q \in \Omega$. If $k \geq 3$ and $F : \mathcal{F}^{\leq k}(\Omega) \to \mathbb{R}^N$ is d_H-continuous, then for every $0 < \tau < 1$ there exists a k-point set A such that $\{p, q\} \subseteq A \subseteq \Omega$ and*

$$\|F(A) - \mathrm{ave}_f(A)\| > (1-\tau)\left(\frac{k-2}{2k}\right)\|f(p) - f(q)\|.$$

In particular, PointNet architectures satisfy this error bound since they are d_H-continuous.

Proof. Let $C_p := \frac{f(p)+f(q)}{k} + (\frac{k-2}{k})f(p)$ and $C_q := \frac{f(p)+f(q)}{k} + (\frac{k-2}{k})f(q)$ and observe that

$$\|C_p - C_q\| = \frac{k-2}{k}\|f(p) - f(q)\| \neq 0.$$

Let $\epsilon := \frac{\tau}{4}\|C_p - C_q\| > 0$. Since F is d_H-continuous there exists a $\delta_{\{p,q\}} \in (0, \epsilon]$ such that whenever $d_H(A, \{p, q\}) < \delta_{\{p,q\}}$ we have that $\|F(A) - F(\{p, q\})\| < \epsilon$. Similarly, since f is continuous, there exists $\delta_p, \delta_q \in (0, \epsilon]$ so that $d(a, p) < \delta_p$ implies $\|f(a) - f(p)\| < \epsilon$ and $d(a, q) < \delta_q$ implies $\|f(a) - f(q)\| < \epsilon$. Define $\delta := \min\{\delta_{\{p,q\}}, \delta_p, \delta_q\}$

Because Ω has no isolated points, its open balls must have infinitely many points, in particular, $B_\delta(p)$ and $B_\delta(q)$. This ensures the existence of sets $\widetilde{A}_p^\delta \subseteq B_\delta(p) \setminus \{p\}$ and $\widetilde{A}_q^\delta \subseteq B_\delta(q) \setminus \{q\}$ with cardinality $k-2$. We can then define k-point supersets of $\{p,q\}$ given by

$$A_p^\delta := \widetilde{A}_p^\delta \cup \{p,q\}, \qquad A_p^\delta := \widetilde{A}_p^\delta \cup \{p,q\}.$$

It's easy to see that both A_p^δ and A_q^δ are δ-close to $\{p,q\}$ with respect to d_H. Thus, by definition of δ we have,

$$\left\|F(A_p^\delta) - F(\{p,q\})\right\| < \epsilon, \quad \left\|F(A_q^\delta) - F(\{p,q\})\right\| < \epsilon.$$

Next observe that,

$$\text{ave}_f(A_p^\delta) = \frac{1}{k}\sum_{a \in A_p^\delta} f(a) = \frac{f(p)+f(q)}{k} + \frac{1}{k}\sum_{a \in \widetilde{A}_p^\delta} f(a),$$
$$\text{ave}_f(A_q^\delta) = \frac{1}{k}\sum_{a \in A_q^\delta} f(a) = \frac{f(p)+f(q)}{k} + \frac{1}{k}\sum_{a \in \widetilde{A}_q^\delta} f(a).$$

Since $f\left(\widetilde{A}_p^\delta\right) \subseteq B_\epsilon(f(p))$ and $f\left(\widetilde{A}_q^\delta\right) \subseteq B_\epsilon(f(q))$, the triangle inequality implies

$$\left\|\text{ave}_f(A_p^\delta) - C_p\right\| = \frac{1}{k}\left\|\sum_{a \in \widetilde{A}_p^\delta}\Big(f(a) - f(p)\Big)\right\| \leq \frac{1}{k}\sum_{a \in \widetilde{A}_p^\delta}\|f(a) - f(p)\| < \left(\frac{k-2}{k}\right)\epsilon < \epsilon,$$

$$\left\|\text{ave}_f(A_q^\delta) - C_q\right\| = \frac{1}{k}\left\|\sum_{a \in \widetilde{A}_q^\delta}\Big(f(a) - f(p)\Big)\right\| \leq \frac{1}{k}\sum_{a \in \widetilde{A}_q^\delta}\|f(a) - f(p)\| < \left(\frac{k-2}{k}\right)\epsilon < \epsilon.$$

Now we can consider the triangle in $\mathbb{R}^N$ formed by C_p, C_q, and $F(\{p,q\})$. By basic geometry, we know that one of the two side lengths $\|F(\{p,q\}) - C_p\|$ or $\|F(\{p,q\}) - C_q\|$ must greater than or equal to half the third side length i.e. greater than $\|C_p - C_q\|/2$. Without loss of generality, let $\|F(\{p,q\}) - C_q\| \geq \|C_p - C_q\|/2$ and then apply the triangle inequality in conjunction with our ϵ bounds to yield,

$$\begin{aligned}\frac{\|C_p - C_q\|}{2} &\leq \left\|F(\{p,q\}) - F(A_q^\delta)\right\| + \left\|F(A_q^\delta) - \mathrm{ave}_f(A_q^\delta)\right\| + \left\|\mathrm{ave}_f(A_q^\delta) - C_q\right\|, \\ &< \left\|F(A_q^\delta) - \mathrm{ave}_f(A_q^\delta)\right\| + 2\epsilon.\end{aligned}$$

Finally, by rearranging and substituting $\epsilon = \tau \|C_p - C_q\| / 4$ we get,

$$\begin{aligned}\left\|F(A_q^\delta) - \mathrm{ave}_f(A_q^\delta)\right\| &> \frac{\|C_p - C_q\|}{2} - 2\epsilon, \\ &= \frac{\|C_p - C_q\|}{2} - \tau\frac{\|C_p - C_q\|}{2}, \\ &= \frac{(1-\tau)}{2}\|C_p - C_q\|, \\ &= (1-\tau)\left(\frac{k-2}{2k}\right)\|f(p) - f(q)\|.\end{aligned}$$

Thus A_q^δ is the promised set which achieves the desired uniform-norm error. $\square$

proof of Theorem 7.5. The case of $k = 1, 2$ are trivial since the right-hand side becomes non-positive. For $k \geq 3$, note that by compactness of Ω, we can choose $p, q \in \Omega$ so that the diameter is achieved i.e. $\|f(p) - f(q)\| = \mathrm{Diam}(f(\Omega))$. If $\mathrm{Diam}(f(\Omega)) = 0$ then the inequality holds trivially, if not then there exist then we have $f(p) \neq f(q)$ and so we apply Lemma 7.6 and take $\tau \to 0$. $\square$

7.5 Numerical Experiments

The proof of Lemma 7.6 not only establishes the error bound but also suggest an algorithmic approach to finding point clouds that exhibit the failure of uniform approximation. This lets us produce (in an architecture and weight dependent way) problematic examples for the centor-of-mass regression with PointNet. When Ω is additionally convex (e.g. the unit disk D in $\mathbb{R}^2$) it becomes fairly easy to construct many examples of A_p^δ and A_q^δ from the proof of Lemma 7.6 for a given PointNet model, allowing us to empirically verify the uniform-norm error lower bound. In the following experiment, we train a simple PointNet architecture to learn the center-of-mass for 10-element point clouds in D. We train on a synthetic dataset of 1 million point clouds labeled with their center-of-mass (each element uniformly sampled from D). The PointNet architecture has ∼500K trainable parameters. The network has the form $F(A) = \boldsymbol{\rho}(\max_{\boldsymbol{a} \in A} \boldsymbol{\varphi}(a))$ where $\boldsymbol{\varphi}$ has 2-D input layer, 500-D hidden layer (ReLU activation), and 500-D linear output layer, and $\boldsymbol{\rho}$ has 500-D input layer, 500-D hidden layer (ReLU activation) and 2-D linear output layer (this means this network is an element of $\mathcal{N}_{2,\max}^{\text{ReLU;ReLU}}$ to which Theorem 7.2 applies). Since it is fundamentally not possible to train with respect to the uniform-norm, we opt to train with the traditional ℓ_2 loss.

To form our problematic examples, we pick a nonzero $\tau = 0.01$ and two distinct points $p, q \in D$ at random. We set $\epsilon = \tau \|p - q\| / 4$. We then initialize $A_p^\delta = A_q^\delta$ to be the set $\{p, q\}$ union a set of 8 randomly sampled points from the unit disk D. To determine problematic point cloud, we then update A_p^δ and A_q^δ by iteratively pulling the extra 8 points closer towards p and q (resp.) until $F(A_p^\delta)$ and $F(A_q^\delta)$ are within distance ϵ of $F(\{p, q\})$ so that the criteria in the proof of Lemma 7.6 is satisfied. This must eventually happen due to d_H-continuity of the PointNet F. We are theoretically ensured one of these two will have error larger than our error lower bound. In Fig. 7.1 we plot the the produced error versus the distance between the seed points p and q that were used to make A_p^δ and A_q^δ. As predicted, all the errors for the problematic examples lie above the line representing the uniform-norm error lower bound.

This experiment was chosen because the task is very simple (computing the mean), not very high-dimensional (ten 2D points per cloud), and yet PointNet fails to eliminate its errors through training on virtually unlimited synthetic data. Moreover, the simple to produce problematic examples show that PointNet never violates the error lower bound.

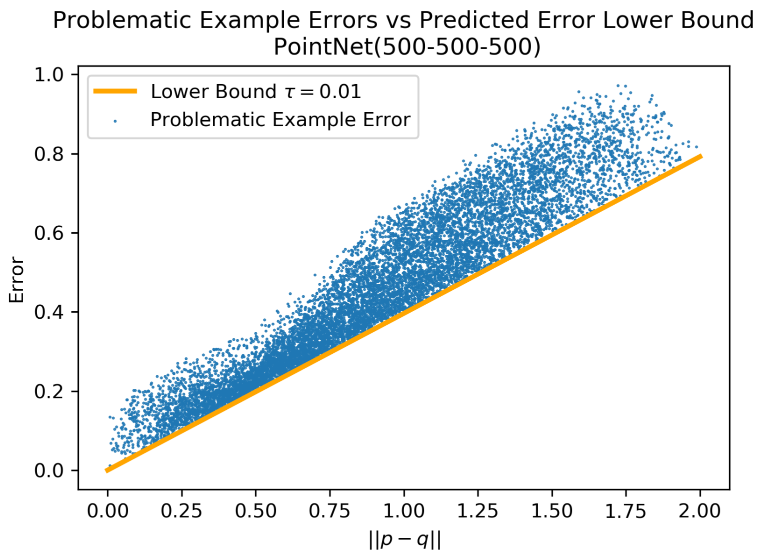

Figure 7.1: In the experimental tests of the error lower bound we see that we are always able to find problematic examples (blue dots) for PointNet on the task of computing the center-of-mass. The method always produces errors at least as large as the theoretical guarantee (orange line).

Bibliography

[1] Matan Atzmon, Haggai Maron, and Yaron Lipman. "Point convolutional neural networks by extension operators". In: *arXiv preprint arXiv: 1803.10091* (2018).

[2] Michael Fielding Barnsley. *Superfractals*. Cambridge University Press, 2006.

[3] Vladimir I Bogachev. *Measure theory*. Vol. 1. Springer Science & Business Media, 2007.

[4] Vladimir I Bogachev. *Measure theory*. Vol. 2. Springer Science & Business Media, 2007.

[5] Vladimir I Bogachev, Oleg Georgievich Smolyanov, and VI Sobolev. *Topological vector spaces and their applications*. Springer, 2017.

[6] Michael M Bronstein et al. "Geometric deep learning: going beyond euclidean data". In: *IEEE Signal Processing Magazine* 34.4 (2017), pp. 18–42.

[7] Christian Bueno and Alan G. Hylton. "Limitations for Learning from Point Clouds". In: *OpenReview* (2020). URL: https://openreview.net/forum?id=r1x63grFvH.

[8] Christian Bueno and Alan G. Hylton. "Limitations of Deep Learning on Point Clouds". In: *NuerIPS Sets & Partitions Workshop* (2019). URL: https://drive.google.com/file/d/10z-9_hYJdb9mxBb4JJupVaVL33lYy9zn/view (visited on 08/24/2021).

[9] S Carroll. "Construction of neural networks using the Radon transform." In: *IEEE International Conference on Neural Networks*. Vol. 1. IEEE. 1989, pp. 607–611.

[10] Ricky TQ Chen et al. “Neural ordinary differential equations”. In: *arXiv preprint arXiv: 1806.07366* (2018).

[11] George Cybenko. “Approximation by superpositions of a sigmoidal function”. In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314.

[12] John Duchi, Elad Hazan, and Yoram Singer. “Adaptive subgradient methods for online learning and stochastic optimization.” In: *Journal of machine learning research* 12.7 (2011).

[13] Richard M Dudley. *Real analysis and probability.* CRC Press, 2018.

[14] K Fukushima. “Neocognitron: A Self Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position”. In: *Biological cybernetics* 36.4 (1980), pp. 193–202.

[15] Ken-Ichi Funahashi. “On the approximate realization of continuous mappings by neural networks”. In: *Neural networks* 2.3 (1989), pp. 183–192.

[16] Stephen I Gallant et al. “Perceptron-based learning algorithms”. In: *IEEE Transactions on neural networks* 1.2 (1990), pp. 179–191.

[17] GCD. *Diameter and Hausdorff Distance.* Mathematics Stack Exchange. (version: 2013-04-28). URL: https://math.stackexchange.com/q/375312 (visited on 08/25/2021).

[18] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. “Explaining and harnessing adversarial examples”. In: *arXiv preprint arXiv: 1412.6572* (2014).

[19] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning.* MIT press, 2016.

[20] Yulan Guo et al. “Deep learning for 3d point clouds: A survey”. In: *IEEE transactions on pattern analysis and machine intelligence* (2020).

[21] Rana Hanocka et al. "Meshcnn: a network with an edge". In: *ACM Transactions on Graphics (TOG)* 38.4 (2019), pp. 1–12.

[22] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2016, pp. 770–778.

[23] Donald Olding Hebb. *The organization of behavior: A neuropsychological theory.* Psychology Press, 2005.

[24] Jeff Henrikson. "Completeness and total boundedness of the Hausdorff metric". In: *MIT Undergraduate Journal of Mathematics* 1.69-80 (1999), p. 10.

[25] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent". In: (2012).

[26] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators". In: *Neural networks* 2.5 (1989), pp. 359–366.

[27] Ran Ben Izhak, Alon Lahav, and Ayellet Tal. "AttWalk: Attentive Cross-Walks for Deep Mesh Analysis". In: *arXiv preprint arXiv: 2104.11571* (2021).

[28] Arthur Jacot, Franck Gabriel, and Clément Hongler. "Neural tangent kernel: Convergence and generalization in neural networks". In: *arXiv preprint arXiv: 1806.07572* (2018).

[29] John L Kelley. *General topology.* Courier Dover Publications, 2017.

[30] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv: 1412.6980* (2014).

[31] Yann LeCun et al. "Backpropagation applied to handwritten zip code recognition". In: *Neural computation* 1.4 (1989), pp. 541–551.

[32] Moshe Leshno et al. “Multilayer feedforward networks with a nonpolynomial activation function can approximate any function”. In: *Neural networks* 6.6 (1993), pp. 861–867.

[33] Warren S McCulloch and Walter Pitts. “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.

[34] Ernest Michael. “Topologies on spaces of subsets”. In: *Transactions of the American Mathematical Society* 71.1 (1951), pp. 152–182.

[35] Francesco Milano et al. “Primal-dual mesh convolutional neural networks”. In: *arXiv preprint arXiv: 2010.12455* (2020).

[36] James Raymond. Munkres. *Topology*. Prentice Hall, 2000.

[37] Mikel Olazaran. “A sociological study of the official history of the perceptrons controversy”. In: *Social Studies of Science* 26.3 (1996), pp. 611–659.

[38] Gabriel Peyré, Marco Cuturi, et al. “Computational optimal transport: With applications to data science”. In: *Foundations and Trends® in Machine Learning* 11.5-6 (2019), pp. 355–607.

[39] Allan Pinkus. “Approximation theory of the MLP model in neural networks”. In: *Acta numerica* 8 (1999), pp. 143–195.

[40] Charles R. Qi et al. “Pointnet: Deep learning on point sets for 3d classification and segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 652–660.

[41] Charles Ruizhongtai Qi et al. “PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space”. In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 5099–5108.

[42] Ning Qian. “On the momentum term in gradient descent learning algorithms”. In: *Neural networks* 12.1 (1999), pp. 145–151.

[43] R Tyrrell Rockafellar and Roger J-B Wets. *Variational analysis*. Vol. 317. Springer Science & Business Media, 2009.

[44] Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton (Project Para)*. Cornell Aeronautical Laboratory, 1957.

[45] W. Rudin. *Functional Analysis*. International series in pure and applied mathematics. Tata McGraw-Hill, 1974. ISBN: 9780070995581.

[46] Walter Rudin. *Real and complex analysis*. Tata McGraw-hill education, 2006.

[47] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. “Learning representations by back-propagating errors”. In: *nature* 323.6088 (1986), pp. 533–536.

[48] Filippo Santambrogio. “Optimal transport for applied mathematicians”. In: *Birkäuser, NY* 55.58-63 (2015), p. 94.

[49] Jonas Schult et al. “DualConvMesh-Net: Joint Geodesic and Euclidean Convolutions on 3D Meshes”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020), pp. 8609–8619.

[50] M.b. Stinchcombe. “Neural network approximation of continuous functionals and continuous functions on compactifications”. In: *Neural Networks* 12.3 (1999), pp. 467–477. DOI: 10.1016/s0893-6080(98)00108-7.

[51] Hugues Thomas et al. “Kpconv: Flexible and deformable convolution for point clouds”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 6411–6420.

[52] Christopher Tuffley. “Finite subset spaces of S1”. In: *Algebraic & Geometric Topology* 2.2 (2002), pp. 1119–1145.

[53] Cedric Villani. *Optimal transport: old and new*. Springer, 2009.

[54] Edward Wagstaff et al. "On the limitations of representing functions on sets". In: *arXiv preprint arXiv: 1901.09006* (2019).

[55] Yue Wang et al. "Dynamic graph cnn for learning on point clouds". In: *Acm Transactions On Graphics (tog)* 38.5 (2019), pp. 1–12.

[56] Nik Weaver. *Measure theory and functional analysis*. World Scientific Publishing Company, 2013.

[57] A Wendemuth. "Learning the unlearnable". In: *Journal of Physics A: Mathematical and General* 28.18 (1995), p. 5423.

[58] Stephen Willard. *General topology*. Courier Corporation, 2012.

[59] Zonghan Wu et al. "A comprehensive survey on graph neural networks". In: *IEEE transactions on neural networks and learning systems* 32.1 (2020), pp. 4–24.

[60] Keyulu Xu et al. "How Powerful are Graph Neural Networks?" In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=ryGs6iA5Km.

[61] Dmitry Yarotsky. "Universal approximations of invariant maps by neural networks". In: *arXiv preprint arXiv: 1804.10306* (2018).

[62] Manzil Zaheer et al. "Deep Sets". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 3391–3401. URL: http://papers.nips.cc/paper/6931-deep-sets.pdf.

[63] Jie Zhou et al. "Graph neural networks: A review of methods and applications". In: *AI Open* 1 (2020), pp. 57–81.

[64] Yi Zhou et al. "Fully convolutional mesh autoencoder using efficient spatially varying kernels". In: *arXiv preprint arXiv: 2006.04325* (2020).