Deep Learning and Approximation theory 2023

Nadav Dym

September 21, 2023

Preface

These are lecture notes from the course ‘Deep Learning and Approximation Theory’ which I taught in the Technion in the spring of 2023. The course is intended for graduate and advanced undergraduate students. Its goal is to introduce students to modern research which focuses on understanding the approximation power of neural networks. The course is divided into two main chapters: the first chapter deals with approximation results pertaining to fully connected neural networks. The second chapter discusses approximation results for neural network architectures which are invariant to permutations or other group actions.

I am making these notes public with the hope that they will be of interest to people outside the Technion. However, note that wile I did some minimal proof reading of these notes, they have not undergone very serious editing and will tend to contain more errors than, say, a textbook. Use at your own risk... Please do let me know via email of any errors which you find.

Chapter 0

Some math preliminaries

Lesson 1

The first goal of this lesson is to show some fundamental facts about approximation: basically that polynomials are dense in the space of continuous functions, while polynomials of finite degrees are not. The secondary goal is to clarify some of the math background necessary for this course, and help close the gaps to those who need to. For additional reading see [Simmons, 1963].

Definition 0.1. Let X be a vector space over $\mathbb{R}$. A norm is a function $\|\cdot\| : X \to \mathbb{R}$ satisfying

1. Positivity: For every $x \in X$, $\|x\| \geq 0$, with equality if and only if $x = 0$.
2. Triangle inequality: $\|x + y\| \leq \|x\| + \|y\|$ for every $x, y \in X$.
3. Homogeneity: $\|\lambda x\| = |\lambda|\|x\|$ for all $x \in X$ and $\lambda \in \mathbb{R}$.

Example 0.2. In $\mathbb{R}^n$, we have the norms

1. the standard 2 norm $\|x\|_2 = \sqrt{\sum_{i=1}^n (x_i)^2}$.
2. the 1 norm $\|x\|_1 = \sum_{i=1}^n |x_i|$
3. In general, for $p \geq 1$ we can define the norm $\|x\|_p = \left(\sum_{i=1}^n |x_i|^p\right)^{1/p}$
4. The ∞ norm $\|x\|_\infty = \max_{1 \leq i \leq n} |x_i|$.

We also say that $(X, \|\cdot\|)$ is a normed space. Later today we will use the reverse triangle inequality:

$$|\|x\| - \|y\|| \leq \|x - y\|$$

which follows from the triangle inequality (check).

Definition 0.3. Let X be a non-empty set. A metric d is a function $d : X \times X \to \mathbb{R}$ which satisfies the following conditions

1. Positivity: for every $x, y \in X$, $d(x, y) \geq 0$ with equality if and only if $x = y$.
2. Symmetry: for every $x, y \in X$, $d(x, y) = d(y, x)$.
3. Triangle inequality: for every $x, y, z \in X$

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

We also say that (X, d) is a metric space.

Example 0.4.

1. For any non-empty set X, we can define the 'discrete metric' $d(x,x) = 0$ and $d(x,y) = 1$ if $x \neq y$.
2. On $X = \mathbb{R}$ we have the metric $d(x,y) = |x - y|$.
3. On $X = \mathbb{R}^n$ we have the metric $d_2(x,y) = \|x - y\|_2 = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$, and more generally $d_p(x,y) = \|x - y\|_p$
4. If (X,d) is a metric space, and $Y \subseteq X$, then (Y,d) is a metric space.

As examples 2-3 suggest, norms induce metrics:

Proposition 0.5. *If $(X, \|\cdot\|)$ is a normed space then $d(x,y) = \|x - y\|$ is a metric on X.*

Problem 0.6. The discrete metric on $\mathbb{R}$ is not induced from a norm. Explain why.

Definition 0.7 (Convergence)**.** Let (X,d) be a metric space. We say that a sequence $x_n, n = 1,2,\dots$ of elements in X converges to x if

$$d(x_n, x) \to 0$$

Definition 0.8 (Cauchy sequences)**.** Let (X,d) be a metric space. We say that a sequence $x_n, n = 1,2,\dots$ of elements in X is Cauchy if for every $\epsilon > 0$ there exists N such that for all $n,m \geq N$ we have $d(x_n, x_m) < \epsilon$.

A converging sequence is always a Cauchy sequence: If $x_n \to x$ then for all $\epsilon > 0$ there exists N such that for all $n > N$ we have $d(x_n, x) < \epsilon/2$, and so for $n,m > N$ we use the triangle inequality to obtain

$$d(x_n, x_m) \leq d(x_n, x) + d(x_m, x) < \epsilon/2 + \epsilon/2 = \epsilon$$

A metric space is *complete* if ever Cauchy sequence has a limit. For example $\mathbb{R}^d$ and closed subsets of $\mathbb{R}^d$ are complete, while $[0,1)$ or the rational numbers are not complete.

Definition 0.9. Let (X,d) be a metric space. For $x_0 \in X$ and $r > 0$, the open ball $B_r(x_0)$ is the set

$$B_r(x_0) = \{x \in X \,|\, d(x, x_0) < r\}$$

Definition 0.10 (Open sets)**.** Let (X,d) be a metric space. We say that $U \subseteq X$ is an open set if for all $x \in U$ there exists some $r > 0$ such that $B_r(x) \subseteq U$.

Example 0.11.

1. An open ball is an open set.
2. In $\mathbb{R}^d$, the hyper-cube $(0,1)^d$ is an open set.

Definition 0.12 (Closed sets)**.** Let (X,d) be a metric space. We say that a set $C \subseteq X$ is closed if for every sequence $x_n \in C$ which converges to a point $x \in X$, we have that $x \in C$.

Problem 0.13. Show that a set $C \subseteq X$ is closed if and only if its complement is open.

Definition 0.14 (Compactness)**.** Let (X,d) be a metric space. We say that a set $K \subseteq X$ is compact if every sequence $x_n \subseteq K$ has a subsequence which converges to a limit in K.

Example 0.15. The set $(0,1]$ is not a compact subset of $\mathbb{R}$ because the sequence $1/n$, and all its subsequences, converge to a point which is not in the set. The set $[0,1]$ is compact. In general in $\mathbb{R}^n$ any closed and bounded set is compact.

Definition 0.16 (Continuity)**.** Let (X, d_X) and (Y, d_Y) be metric spaces. We say that a function $f : X \to Y$ is continuous if for every sequence $x_n \to x$ in X, we have that $f(x_n) \to f(x)$ in Y.

Proposition 0.17. *Let (X, d_X) and (Y, d_Y) be metric spaces. If $f : X \to Y$ is continuous and $K \subseteq X$ is compact, then $f(K)$ is compact.*

Proof. We need to show that $f(K)$ is compact. Let y_n be a sequence in $f(K)$. There exist $x_n \in K$ such that $f(x_n) = y_n$. By compactness of K, the sequence x_n has a subsequence x_{n_k} which converges to some $x \in K$. By continuity

$$y_{n_k} = f(x_{n_k}) \to f(x) \in f(K)$$

and so we showed that y_n has a converging subsequence, and thus $f(K)$ is compact. □

Corollary 0.18. *Let (X, d) be a metric space. $K \subseteq X$ compact, and $f : X \to \mathbb{R}$ a continuous function. Then f obtains a maximum and minimum of K.*

Proof. By the previous proposition $f(K)$ is a compact subset of $\mathbb{R}$ and so it is bounded: there exists some $M > 0$ such that $|f(x)| < M$. It follows that $S = \sup_{x\in K} f(x)$ is finite. There exists a sequence $x_n \subseteq K$ such that $f(x_n) \to S$. By compactness x_n has a subsequence which converges to $x \in K$ and by continuity $f(x) = S$ so x is a maximum. The same argument can be used to show the existence of a minimum. □

Let V be a vector space over the reals and $K \subseteq V$ be a subset. Denote

$$C(K) = \{f : K \to \mathbb{R} | \, f \text{ is continuous}\}.$$

If we additionally assume that K is compact, we can define a norm on $C(K)$ by

$$\|f\| = \max_{x\in K} |f(x)|$$

by the previous corollary this is well defined. We can verify that this is indeed a norm. Positivity and homogeneity are rather obvious. As for the triangle inequality: For $f, g \in C(K)$ and for all $x \in K$

$$|f(x) + g(x)| \leq |f(x)| + |g(x)| \leq \|f\| + \|g\|$$

and since this inequality is true for all x it is also true for the x which maximize $|f(x) + g(x)|$.

Problem 0.19. Give an example of f, g for which (i) the inequality is strict and (ii) the inequality is not strict.

The norm on $C(K)$ induces the metric $\|f - g\| = \max_{x\in K} |f(x) - g(x)|$ on functions. This metric space will be central in this course. We will consider questions such as: given a subset of function $P \subseteq C(K)$, can any function $f \in C(K)$ be approximated by functions in P? That is, can we, for every given $\epsilon > 0$ find some $p \in P$ such that

$$\|f - p\| = \max_{x\in K} |f(x) - p(x)| < \epsilon$$

or equivalently, can we find a sequence $p_n \subseteq P$ such that $\|p_n - f\| \to 0$? When this happens we say that p_n *uniformly converges* to f, and we say that P is *dense* in $C(K)$.

We state Weierstrass's theorem

Theorem 0.20 (Weierstrass)**.** *Every continuous function $f : [a, b] \to \mathbb{R}$ can be approximated uniformly by polynomials.*

we will later discuss this theorem's generalization, the Stone-Weierstrass theorem. For now, we want to show that we cannot approximate a continuous function by polynomials of degree $\leq D$. Denote

$$\mathbb{R}_D[x] = \{ \text{ polynomials of degree } \leq D\}$$

This is a vector space of dimension $D + 1$, with the norm $\|f\|_{C[a,b]]} = \max_{x\in[a,b]} |f(x)|$. We will show that all finite dimensional normed spaces are 'topologically equivalent' and through this show that they cannot approximate all functions. In contrast, note that the space of *all polynomials* $\mathbb{R}[x]$ is infinite dimensional.

Definition 0.21. Let $(X, \|\cdot\|_X)$ and $(Y, \|\cdot\|_Y)$ be normed spaces over $\mathbb{R}$. We say that a map $T : X \to Y$ is an isomorphism of normed spaces if T is linear, onto, and there exists $0 < m < M$ such that

$$m\|x\|_X \leq \|Tx\|_Y \leq M\|x\|_X$$

Problem 0.22. Show that the identity map $Ix = x$ is a isomorphism between $(\mathbb{R}^n, \|\cdot\|_\infty)$ and $(\mathbb{R}^n, \|\cdot\|_1)$. What are the constants? Can you do the same when ∞ is replaced with 2?

Solution. We have that

$$\|x\|_1 = \sum_{i=1}^n |x_i| \geq \max_i |x_i| = \|x\|_\infty,$$

and on the other hand

$$\|x\|_1 = \sum_{i=1}^n |x_i| \leq \sum_{i=1}^n |x|_\infty = n|x|_\infty,$$

and so we obtained

$$m\|x\|_\infty \leq \|x\|_1 \leq M\|x\|_\infty.$$

with $m = 1$ and $M = n$. Note that these constants aren't unique: we can make M larger of m smaller. They are however the optimal constants one can choose, as can be seen by taking

$$x = [1, 0, 0, \ldots, 0] \text{ and } x = [1, 1, \ldots, 1]$$

□

We say that $(X, \|\cdot\|_X)$ and $(Y, \|\cdot\|_Y)$ are isomorphic if there exists an isomorphism between them.

As it turns out, there is always an isomorphism between normed spaces of the same finite dimension.

Theorem 0.23. *Let $(V, \|\cdot\|)$ be a vector space over $\mathbb{R}$ of dimension n. Then there is a isomorphism between V and $(\mathbb{R}^n, \|\cdot\|_2)$.*

Proof. Let $v_1, \ldots, v_n$ be a basis for V, and $e_1, \ldots, e_n$ denote the standard basis for $\mathbb{R}^n$. We define a linear map $T : \mathbb{R}^n \to V$ by

$$T(x) = \sum_{i=1}^n x_i v_i.$$

The map T is linear and onto. It is also one-to-one, for $T(x) = 0$ implies that all x_i are zero. It remains to show that the norm $\|Tx\|$ can be bounded from above and below by $\|x\|$ up to some constant. Define

$$M = \sqrt{\sum_{i=1}^n \|v_i\|^2}$$

we have, using the properties of the norm and Cauchy-Schwarz, that

$$\|Tx\| = \|\sum_{i=1}^n x_i v_i\| \leq \sum_{i=1}^n \|x_i v_i\| \leq \sum_{i=1}^n |x_i|\|v_i\| \leq \sqrt{\sum_{i=1}^n |x_i|^2}\sqrt{\sum_{i=1}^n \|v_i\|^2} = M\|x\|_2$$

this gives us the first inequality we need. We can also use it to get the other direction: the inequality we just proved shows that the map

$$N : \mathbb{R}^n \to \mathbb{R}, N(x) = \|Tx\|$$

is continuous. For if $x_n \to x$ then, using the reverse triangle inequality:

$$|N(x_n) - N(x)| = |\|Tx_n\| - \|Tx\|| \leq \|T(x_n - x)\| \leq M\|x_n - x\|_2 \to 0$$

The continuous function N obtains a minimum on the compact set $S_1 = \{x \in \mathbb{R}^n | \|x\|_2 = 1\}$. This minimum is clearly non-negative. Moreover, we saw that N is injective and so doesn't assume zero values on S_1, which then implies that since $\|\cdot\|$ is a norm, $N(x) = \|Tx\|$ is never zero on S_1, and so there is a positive minimum $m > 0$. It follows that for all $x \in S_1$

$$\|Tx\| \geq m = m\|x\|_2$$

this inequality holds for all x. If $x = 0$ then both sides are zero. Otherwise we have

$$\|Tx\| = \|x\|_2 \cdot \|T\left(\frac{x}{\|x\|_2}\right)\| \geq \|x\|_2 \cdot m$$

□

Corollary 0.24. *Every real finite dimensional normed spaces is complete.*

sketch of proof. Let $(V, \cdot)$ be a real normed space of dimension n and let $T : V \to \mathbb{R}^n$ be an isomorphism. One can verify that $v_n \subseteq V$ converges to $v \in V$ if and only if Tv_n converges to Tv, and similarly v_n is a Cauchy sequence if and only if Tv_n is a Cauchy sequence. Since v_n is Cauchy, so is Tv_n, and since $\mathbb{R}^n$ is complete, Tv_n has a limit x, and so v_n has a limit $T^{-1}x$. □

Corollary 0.25. *Let $(V, \|\cdot\|)$ be a real normed vector space, and $P \subseteq V$ is a finite dimensional subspace, and assume that $p_n \subseteq P$ converges to some limit $f \in V$. Then $f \in P$.*

Proof. Since $p_n \to f$ the sequence p_n is a Cauchy sequence in P and so it has a limit in P. Thus f is in P. □

As mentioned above, this corollary shows in particular that polynomials of degree $\leq D$ cannot approximate *any* continuous functions which are not polynomials of degree $\leq D$.

0.0.1 The Stone Weierstrass Theorem

In the previous section we saw that polynomials are dense in $C[a, b]$. Now we will discuss the Stone-Weierstrass theorem which gives an anologous result for $C(K)$ where K is any compact metric space (it also hold for compact Hausdorf spaces but we do not discuss non-metric topologies in this course). We start by defining multivariate polynomials:

A monomial function $m : \mathbb{R}^n \to \mathbb{R}$ is a function of the form

$$x = (x_1, \ldots, x_n) \mapsto x_1^{\alpha_1} \cdot x_2^{\alpha_2} \ldots \cdot x_n^{\alpha_n}$$

where $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}_0^n$ (here $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$). We use the shortened notation x^α to described this function. The degree of a monomial is the sum of all indices of α which is denoted by $|\alpha| = \|\alpha\|_1$. A polynomial function $p : \mathbb{R}^n \to \mathbb{R}$ is a finite linear combination of monomials. The degree of a polynomials is the maximal degree of a monomial in the linear combination which has a non-zero coefficient.

The Stone-Weierstrass theorem shows that multi-variate polynomials are dense in $C(K)$ for compact $K \subseteq \mathbb{R}^n$. In fact, it applies to a wider family of functions called sub-algebras:

Definition 0.26. Let X be a metric space. We say that a subset $\mathcal{A} \subseteq C(X)$ is a subalgebra of $C(X)$ if for all $f, g \in \mathcal{A}$ and $c \in \mathbb{R}$

1. $f + g \in \mathcal{A}$
2. $cf \in \mathcal{A}$
3. $f \cdot g \in \mathcal{A}$

Example 0.27.
1. Polynomials $p : \mathbb{R} \to \mathbb{R}$ are a sub-algebra.
2. Polynomials of degree $\leq D$ are not a sub-algebra.
3. Multi-variate polynomials $p : \mathbb{R}^d \to \mathbb{R}$ are a sub-algebra.
4. The trigonometric polynomials: functions which are finite linear combinations of the constant function 1, and the functions $\cos(nx)$ and $\sin(mx)$ for $n, m \in \mathbb{N}$, are a sub-algebra. This can be seen using the trigonometric identities

$$\begin{aligned} 2\sin(x)\sin(y) &= \cos(x - y) - \cos(x + y) \\ 2\sin(x)\cos(y) &= \sin(x + y) + \sin(x - y) \\ 2\cos(x)\cos(y) &= \cos(x - y) + \cos(x + y) \end{aligned}$$

5. Linear combinations of functions $f_w : \mathbb{R}^n \to \mathbb{R}$ of the form

$$f_w(x) = e^{\langle w,x \rangle}$$

are a subalgebra.

Definition 0.28. An algebra $\mathcal{A} \subseteq C(K)$ separates points in K, if for any points $x, y \in K$ with $x \neq y$ there exists some $f \in \mathcal{A}$ such that $f(x) \neq f(y)$.

Theorem 0.29 (Stone-Weierstrass)**.** *Let K be a compact metric space, and let $\mathcal{A}$ be an algebra which contains a non-zero constant function and separate points, then $\mathcal{A}$ is dense in $C(K)$.*

We will not prove this theorem, but we will practice using it.

Problem 0.30. Using the Stone-Weierstrass Theorem, answer the following questions:

1. Show that the univariate polynomials are dense on every closed interval $[a, b]$ (that is, that Weierstrass follows from Stone Weirstrass)?
2. Are the univariate polynomials dense on all of $\mathbb{R}$?
3. Are the multivariate polynomials dense on any compact subset of $\mathbb{R}^d$?
4. Are the trigonometric polynomials dense on every compact subset of $\mathbb{R}$?
5. Are functions of the form

$$\mathbb{R}^d \ni x \mapsto \sum_{n=1}^{N} c_n e^{\langle w_n, x \rangle}$$

dense on any compact subset of $\mathbb{R}^d$?

Chapter 1

Fully Connected Neural Networks

Lesson 2

1.1 Introduction

1.1.1 Supervised Learning

The motivation for most questions discussed in this course come from a *supervised learning* problem: Let $K \subseteq \mathbb{R}^d$ be a compact set, and $f : K \to \mathbb{R}^\ell$ a continuous function. We have 'no access' to f, except via its samples $f(x_i), i = 1, \ldots, N$ at some points $x_i, i = 1, \ldots, N$ in K. Our aim is to find a good approximation for f, by choosing an appropriate hypothesis class of parametric functions

$$\mathcal{H} = \{h(x; \theta) : K \times \mathbb{R}^p \to \mathbb{R}^\ell\}$$

and searching for the best approximation of f by h in terms of the given data, by solving an optimization problem such as

$$\min_{\theta \in \mathbb{R}^p} L(\theta) = \sum_{i=1}^{N} \left[h_\theta(x_i) - f(x_i)\right]^2 .$$

Typically $L(\theta)$ is differentiable (at least at 'most' points), and the methods used for searching for the minimizer of $L(\theta)$ are variations of the 'gradient descent' method. In its simple form, in gradient descent we are given an initial guess θ_0 for the minimum of L, and we iteratively update our guess for θ by going in the direction in which L is reduced most:

$$\theta_{t+1} = \theta_t - \nabla L(\theta_t).$$

Once this procedure is terminated and some final θ_* is reached, we choose h_{θ_*} as our approximation of f. There are several theoretical (and practical) issues that can arise:

1. **Optimization** Can gradient descent (or a different algorithm) find a solution θ_* which is a good approximation of the true minimum?
2. **Generalization** Will a good approximation of f on the data x_i give a good approximation for other points in K?
3. **Approximation** Does our hypothesis class H contain a good approximation for f?

In this course we focus only on the question of approximation. We would like to show that the hypothesis classes we construct can approximate any continuous function. This property is called 'universality'. In the previous chapter we saw some examples of parametric function classes which can approximate all continuous functions, at the limit where the number of parameters goes to infinity. In practice, we will focus mostly on hypothesis classes called neural networks, which are popular in modern applications. Our first goal will be to show that they too, like polynomials, can approximate all continuous functions. But before we do this, we will start by defining these hypothesis classes:

1.1.2 Neural Network Functions

We refer to functions which are compositions of affine functions and *activation functions* as *neural network functions*. Recall that an affine function $h : \mathbb{R}^{w_0} \to \mathbb{R}^{w_1}$ is a function of the form

$$h(x) = Ax + b, \text{ where } A \in \mathbb{R}^{w_1 \times w_0}, x \in \mathbb{R}^{w_0} \text{ and } b \in \mathbb{R}^{w_1}$$

An activation function is basically any function $\sigma : \mathbb{R} \to \mathbb{R}$. We extend σ to a function from $\mathbb{R}^w \to \mathbb{R}^w$ via

$$\sigma(x_1, \dots, x_w) = (\sigma(x_1), \sigma(x_2), \dots, \sigma(x_w))$$

Fixing some activation function σ, and natural numbers $w_0, \dots, w_L, w_{L+1}$, a *Neural Network Function (NNF)* with widths $w_0, \dots, w_L, w_{L+1}$ is defined to be a function $h : \mathbb{R}^{w_0} \to \mathbb{R}^{w_{L+1}}$ of the form

$$h_{L+1} \circ \sigma \circ h_L \dots \circ \sigma \circ h_1 \text{ where } h_i : \mathbb{R}^{w_{i-1}} \to \mathbb{R}^{w_i} \text{ is affine } \forall i = 1, \dots, L. \tag{1.1}$$

Choices of activation function Possibly the most popular choice of activation function today is

$$\sigma(x) = \mathit{ReLU}(x) = \max(x, 0).$$

A very popular choices of activation function in the past was the analytic sigmoid function

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}}.$$

A smooth version of the ReLU can be obtained by considering

$$\mathit{SiLU}(x) = x \cdot \text{sigmoid}(x) = \frac{x}{1 + e^{-x}}.$$

Another common option is the hyperbolic tangent

$$\mathit{tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

In the literature discontinuous functions like

$$\text{sign}(x) = \begin{cases} 1 & \text{if } x > 0 \\ -1 & \text{if } x \le 0 \end{cases}$$

are also discussed though they are difficult to work with using gradient descent.

1.1.3 Neural Network Architectures

Neural Network Architectures We define a *Neural Network Architecture* to be a set of functions which are all neural network functions with respect to some fixed activation function. Here are some examples

1. **Fully connected neural networks** Let $\mathcal{X} = \mathbb{R}^{w_0}$ be our domain. Let $\sigma : \mathbb{R} \to \mathbb{R}$ be some function, which we will call an *activation function*. Let

   $$\vec{w} = (w_0, w_1, \dots, w_L, w_{L+1}) \in \mathbb{N}^{L+1}$$

   for some $L \in \mathbb{N}$. For fixed $\vec{w}, \sigma$ this defines a *hypothesis class*

   $$\mathcal{FC}\left(\vec{w}; \sigma\right) = \{h(x) = h_{L+1} \circ \sigma \circ h_L \dots \circ \sigma \circ h_1(x) \text{ where } h_i : \mathbb{R}^{w_{i-1}} \to \mathbb{R}^{w_i} \text{ is affine } \forall i = 1, \dots, L+1\}$$

   The number L is called the depth of the network, and each number w_ℓ is called the width of the ℓ-th layer. We usually think of the vector $\vec{w}$ (and the activation function σ) as *hyper-parameters*: once they are fixed, we can use gradient descent to find the parameters which define the affine functions h_i.

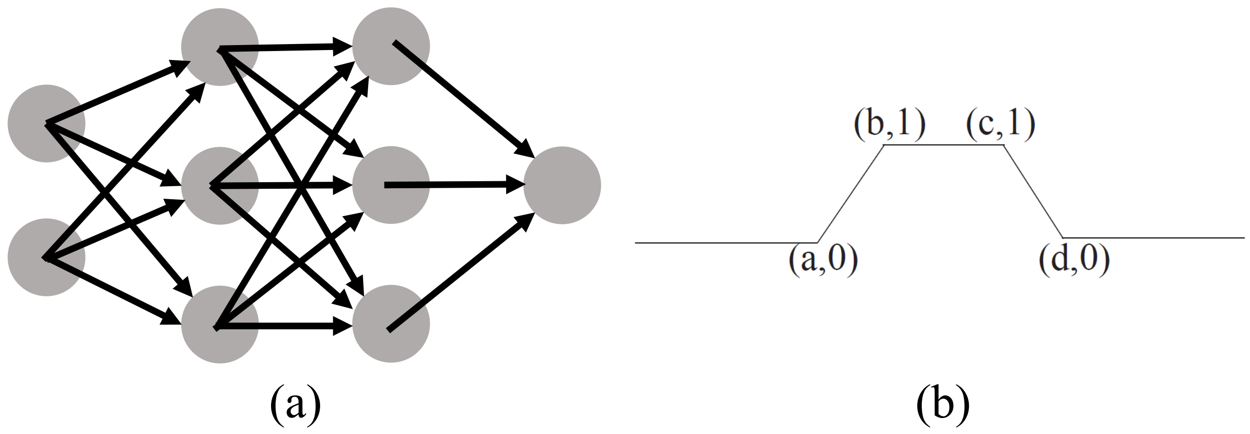

Figure 1.1: (a) The directed graph corresponding to the architecture $\mathcal{FC}(3,2;2,1,\sigma)$. (b) The 'trapezoid function' $h_{a,b,c,d}$.

2. **Fully connected neural networks with fixed width** When studying the approximation power of fully connected neural networks we will usually simplify things by assuming that all hidden layer have the same width. The width vector $\vec{w}$ is then described by four hyper-parameters

$$\vec{w}\left(W,L,w_{in},w_{out}\right)=\left(w_0=w_{in},w_1=W,w_2=W,\ldots,w_L=W,w_{L+1}=w_{out}\right)$$

and accordingly these hyper-parameters define the hypothesis class

$$\mathcal{FC}\left(W,L;w_{in},w_{out},\sigma\right)=\mathcal{FC}\left(\vec{w}\left(W,L,w_{in},w_{out}\right);\sigma\right)$$

3. **Convolutional Neural Networks** are another example (which we will not discuss in this course) of functions which are compositions of affine and activation functions. The affine functions are convolutions rather than general linear transformations.

1.1.4 The 'Neural' Terminology

The source of terms such as neural networks, and activation functions comes from the origin of neural networks as an attempt to create learning models which imitate the way the human brain work. For this reason the neural networks we discuss in this course are sometimes called Artificial Neural Networks, to differentiate them from the neural networks which compose our brain.

The general idea is that humans process information through a network of neurons which communicate between themselves through electrical signals. Based on the electrical signals a neuron obtains from its neighbors, the neuron decides whether to 'fire' or not. This decision in turn influences its neighbors etc.

Consider a neural network architecture of the form $\mathcal{FC}(W,L;w_{in},w_{out},\sigma)$. We can associate with this function a directed graph (V,E): the vertices in this graph are indexed by

$$v_i^{(\ell)},\ell=0,\ldots,L \text{ and } i=1,\ldots,w_\ell$$

Each vertex is called a *neuron*, and the collection of all vertices with superindex ℓ are called the ℓ-th layer. The 0-th layer is called the input layer, the last layer is called the output layer, and the intermediate layers are called 'hidden layers'.

For a given vertex=neuron v_j^ℓ, the incoming edges in the graph connect the neuron to all neurons in the previous layer

$$v_i^{(\ell-1)},i=1\ldots w_{\ell-1}$$

while the outgoing edges connect v_j^ℓ to all neurons in the next layer

$$v_i^{(\ell+1)},i=1\ldots w_{\ell+1}$$

when $\ell = 0$ there are no incoming edges and when $\ell = L$ is maximal there are no outcoming edges. In Figure 1.1(a) we visualize the directed graph associated with the architecture $\mathcal{FC}(W = 3, L = 2; w_{in} = 2, w_{out} = 1, \sigma)$.

The function in (1.1) can be rewritten as a composition of functions, where we denote the input $x \in \mathbb{R}^{w_0}$ by $x^{(0)}$ and then recursively define

$$x^{(\ell)} = \sigma\left(h_\ell(x^{(\ell-1)})\right) = \sigma\left(A^{(\ell)} x^{(\ell-1)} + b^{(\ell)}\right). \tag{1.2}$$

The final vector we obtain by this process $x^{(L)}$ is the output of the neural network function defined in (1.1). Looking at the j-th coordinate of $x^{(\ell)}$ in (1.2) we obtain

$$x_j^{(\ell)} = \sigma\left(\sum_{k=1}^{w_{\ell-1}} A_{jk}^{(\ell)} x_k^{(\ell-1)} + b_k^{(\ell)}\right)$$

That is, the neuron $v_j^{(\ell)}$ aggregates the signals it obtains from its neighbors according to weights $A_{jk}^{(\ell)}$ (which are defined per edges) and b_k^ℓ (which is defined per vertex), and then decides whether to fire. In this context the sigmoid and sign activation functions seem more natural. When using the sign function, the neuron decides to fire (or activate) if the signal it obtained from its neighbors is above a certain threshold. with sigmoid a smoother approach is taken, where the amount of activation varies continuously with the input. Of course, in practice we are interested in which functions perform best in terms of learning, irrespective to how well they resemble our understanding of 'natural neural networks'.

1.1.5 Universality of Neural Network

Universality theorem We now turn to discuss the approximation power of neural networks. We'd like to have an analogue of the Weierstrass theorem for polynomials- that is- we'd like to say that for every compact set $K \subseteq \mathbb{R}^d$, some neural network function space will be dense (with respect to the 'uniform convergence' norm defined in the first lesson) in

$$C(K) = \{f : K \to \mathbb{R} \text{ is continuous }\}$$

Recall that for a normed vector space C, we say that a subspace P is dense in C if for every $f \in C$ and $\epsilon > 0$ there is some $p \in P$ such that $\|f - p\| < \epsilon$. Equivalently, this means that for every f there is a sequence of $p_n \in P$ such that $p_n \to f$ (that is, $\|f - p_n\| \to 0$).

Problem 1.1. Two of the function classes below *are not* dense in $C(K, \mathbb{R})$. Which are they?

1. The function class
$$\bigcup_{(W,L)\in\mathbb{N}^2} \mathcal{FC}\left(W, L; w_{in} = d, w_{out} = 1, \sigma\right)$$
with activation function $\sigma(x) = x$.

2. The function class
$$\bigcup_{(W,L)\in\mathbb{N}^2} \mathcal{FC}\left(W, L; w_{in} = d, w_{out} = 1, \sigma\right)$$
with activation function $\sigma(x) = ReLU(x)$.

3. The function class
$$\bigcup_{W\in\mathbb{N}} \mathcal{FC}\left(W, L = 1; w_{in} = d, w_{out} = 1, \sigma\right)$$
with activation function $\sigma(x) = ReLU(x)$. These networks are called shallow neural networks.

4. The function class
$$\bigcup_{W\in\mathbb{N}} \mathcal{FC}\left(W, L = 1; w_{in} = d, w_{out} = 1, \sigma\right)$$
with activation function $\sigma(x) = x^2$.

Solution: The first is not dense as it contains only linear functions. The last is not dense as it contains only degree two polynomials. The second and third function classes are dense as we will now discuss.

As it turns out, for ReLU activation functions, a single hidden layer is sufficient to obtain universality(=denseness), as the width is taken to infinity. Which other activation functions have this property? Clearly this cannot occur for polynomial activation functions of some degree D because then $\mathcal{FC}\left(W, L=1; w_{in}, w_{out}, \sigma\right)$ only contains polynomials $p : \mathbb{R}^{w_{in}} \to \mathbb{R}^{w_{out}}$ of degree $\leq D$. The space of such polynomials is finite dimensional, and by Corollary 0.25 functions from this space cannot approximate any functions from outside that space.

Remarkably, this is the only restriction on the activation function. The well known universality theorem for neural networks (perhaps the most well known result in neural network theory...) states

Theorem 1.2 ([Pinkus, 1999])**.** *Let $K \subseteq \mathbb{R}^d$ be a compact set, and $\sigma : \mathbb{R} \to \mathbb{R}$ be a continuous function which is not a polynomial. Then*

$$\bigcup_{W \in \mathbb{N}} \mathcal{FC}\left(W, L=1; w_{in}=d, w_{out}=1, \sigma\right)$$

is dense in $C(K)$.

Due to this theorem, neural networks are often called 'universal approximators'. That is: for any given unknown continuous f, there is hope to approximate it using a neural network hypothesis class, as such hypothesis classes can approximate *any* continuous function.

A standard reference for the theorem and it historical development, is in the review article by Allan Pinkus [Pinkus, 1999], (who by the way is a retired professor from our very own Technion Math department). In this course we will not present the full proof, but we will use parts of his proof to prove Theorem 1.2 for *anlaytic* non-polynomial activations. Later on, we will present a different proof for ReLU activations, where we will allow the depth of the network to grow and not only the width.

1.2 Proof of Universality of Shallow Neural Networks with analytic activations

Our goal in this section is to prove Theorem 1.2. We begin with some preliminaries.

First, let us reformulate out goal. In general, we would like to prove denseness of functions of the form $h_2 \circ \sigma \circ h_1$ where $h_1 : \mathbb{R}^d \to \mathbb{R}^W$ and $h_2 : \mathbb{R}^W \to \mathbb{R}$ are affine and W can be arbitrarily large. In the proof we will always only consider h_2 which are linear, that is $h_2(0) = 0$. We can then write

$$y = \sigma \circ h_1(x) = \sigma(Ax + b)$$
$$h_2(y) = \sum_{w=1}^{W} c_w y_w$$

Denoting the rows of A by $a^{(1)}, \dots, a^{(W)}$, we have that

$$y_w = \sigma(\langle a^{(w)}, x\rangle + b_w)$$
$$h_2(y) = \sum_{w=1}^{W} c_w y_w = \sum_{w=1}^{W} c_w \sigma(\langle a^{(w)}, x\rangle + b_w)$$

We conclude

Lemma 1.3. *Let $K \subseteq \mathbb{R}^d$ be a compact set, and $\sigma : \mathbb{R} \to \mathbb{R}$ be some function, if*

$$\operatorname{span}\{\sigma\left(\langle a, x\rangle + b\right) \mid a \in \mathbb{R}^d, b \in \mathbb{R}\} \tag{1.3}$$

is dense in $C(K)$, then

$$\bigcup_{W \in \mathbb{N}} \mathcal{FC}\left(W, L=1; w_{in}=d, w_{out}=1, \sigma\right)$$

is dense in $C(K)$.

Closure Next, recall that if C is a normed space and $P \subseteq C$ is a subspace, then the *closure* of P is denoted by $\bar{P}$, and defined as the set

$$\bar{P} = \{f \in C | \exists (p_n)_{n\in\mathbb{N}} \subseteq P \text{ such that } \|p_n - f\| \to 0\}.$$

Note that saying that P is dense in C is equivalent to saying that $C = \bar{P}$. One can verify that

Proposition 1.4. *Let C be a normed space and $P \subseteq C$ a subspace, then*

1. *$\bar{P}$ contains P, is closed, and is the smallest set having these two properties.*
2. *$\bar{P}$ is a subspace of C.*

Analytic functions Recall that if a function $\sigma : \mathbb{R} \to \mathbb{R}$ is analytic then it is continuous differentiable ∞ times, and all its derivatives are analytic. If σ is not a polynomial then all its derivatives will not be the zero function. We then have

Lemma 1.5. *If $\sigma : \mathbb{R} \to \mathbb{R}$ is analytic and non-polynomial, then there exists a point $b \in \mathbb{R}$ such that none of the derivatives of σ at b vanish, that is*

$$\sigma(b) \neq 0, \sigma'(b) \neq 0, \sigma^{(2)}(b) \neq 0, \ldots \tag{1.4}$$

Proof. For analytic functions, if there is a sequence t_n which converges to some t, for which $\sigma(t_n) = 0$ for all n, then $\sigma(t) = 0$ for all $t \in \mathbb{R}$. It follows that if σ is not zero, then in the interval $[0, 1]$ it can have a finite number of zeros (otherwise, we can find a sequence t_n on which σ is zero, and then take a converging subsequence which would imply that $\sigma = 0$). We deduce that for all $k = 0, 1, \ldots$ the set

$$Z_k = \{b \in [0, 1] | \sigma^{(k)}(b) = 0\}$$

is finite and therefore the set $\cup_{k\in\{0\}\cup\mathbb{N}} Z_k$ is countable, and so there exists some $b \in [0, 1]$ which is not in this set, and for this b we have (1.4). □

We are now ready to prove Theorem 1.2 for the case $d = 1$. The general case $d \geq 1$ will then follow relatively easily.

Lemma 1.6 (Shallow universality for $d = 1$)**.** *Let $K \subseteq \mathbb{R}$ be a compact set, and $\sigma : \mathbb{R} \to \mathbb{R}$ be an analytic function which is not a polynomial. Then*

$$\bigcup_{W\in\mathbb{N}} \mathcal{FC}\left(W, L = 1; w_{in} = 1, w_{out} = 1, \sigma\right)$$

is dense in $C(K)$.

Proof. By Lemma 1.3, it is sufficient to show that the space

$$P = \text{span}\{\sigma(at + b) | \, a, b \in \mathbb{R}\}$$

is dense, that is that $\bar{P} = C(K)$. Let us fix some b for which σ and all its derivatives do not vanish as in (1.4). We claim by induction that for every $a \in \mathbb{R}$ and every $k = 0, 1, \ldots$ the function

$$f_a^{(k)}(t) = \frac{d^k}{da^k}\sigma(at + b) = t^k \sigma^{(k)}(at + b)$$

is in $\bar{P}$. For $k = 0$ this is obvious.

Let us now assume correctness for k and prove for $k + 1$. Using the induction hypothesis and the fact that $\bar{P}$ is a subspaces, we have for every $h \neq 0$ that $\frac{1}{h}\left(f_{a+h}^{(k)}(t) - f_a^{(k)}(t)\right)$ is in $\bar{P}$. We need to show that they converge uniformly on K to $f_a^{(k+1)}(t)$ and therefore since $\bar{P}$ is closed this function is also in $\bar{P}$. Indeed we have for every $t \in K$ that there exists some $\hat{a} \in [a-1, a+1]$ such that

$$\begin{aligned} |\frac{1}{h}\left(f_{a+h}^{(k)}(t) - f_a^{(k)}(t)\right) - f_a^{(k+1)}(t)| &= |\frac{1}{h}\left(f_{a+h}^{(k)}(t) - f_a^{(k)}(t) - hf_a^{(k+1)}(t)\right)| \\ &= |\frac{1}{h}\frac{1}{2}h^2 f_{\hat{a}}^{(k+2)}(t)| \\ &\leq \frac{|h|}{2} \max_{(\hat{a},t)\in[a-1,a+1]\times K} |f_{\hat{a}}^{(k+2)}(t)|, \end{aligned}$$

where the existence of the maximum follows from the fact that all derivatives of σ exists and are continuous, and the set $[a-1, a+1] \times K$ is compact. It follows that $\frac{1}{h}\left(f_{a+h}^{(k)}(t) - f_a^{(k)}(t)\right)$ converges unifomly on K to $f_a^{(k+1)}(t)$ as required, and so $f_a^{(k+1)} \in \bar{P}$ for all $a \in \mathbb{R}$ and all k. In particular, this holds for $a = 0$, in which case we see that for all k the function $t^k\sigma^{(k)}(b)$ is in $\bar{P}$. Since $\sigma^{(k)}(b) \neq 0$ for all k we see that every polynomial can be obtained as a linear combination of these functions, and since $\bar{P}$ is a subspace of $C(K)$ it follows that all polynomials are contained in $\bar{P}$. From the denseness of polynomials it follows that $\bar{P} = C(K)$ and so we are done. □

We can now conclude the proof of the theorem for general $d \geq 1$.

Proof of Theorem 1.2. By Lemma 1.3 it is sufficient to show that the closure of

$$P = \text{span}\{\sigma\left(\langle a, x\rangle + b\right) \,|\, a \in \mathbb{R}^d, b \in \mathbb{R}\}$$

is equal to $C(K)$. In Problem 0.30 we used Stone-Weierstrass to show that linear combinations of functions of the form

$$E_a(x) = \exp\left(\langle a, x\rangle\right)$$

are dense in $C(K)$. It is thus sufficient to show that all such functions are in $\bar{P}$. By the previous lemma, For fixed a, and every $\epsilon > 0$ we can approximate the continuous exponential function on the compact set $K_a = \{\langle a, x\rangle | x \in K\}$ to ϵ accuracy by an expression of the form $\sum_{n=1}^N c_n\sigma(a_n t + b_n)$, and so for all $x \in K$

$$\|\exp(\langle a, x\rangle) - \sum_{n=1}^N c_n\sigma(a_n\langle a, x\rangle + b_n)\| \leq \epsilon$$

which implies that $\exp(\langle a, x\rangle)$ is in $\bar{P}$ and so we are done. □

The full proof of Theorem 1.2 We proved Theorem 1.2 for the special case where the activation function is analytic and non-polynomial. The same proof works for $C^\infty(\mathbb{R})$ non-polynomial functions, the only difference being that proving that there exists a point on which all derivative do not vanish is more challenging. This can then be extended to all continuous non-polynomial functions, essentially by approximating the smooth activations by continuous activations. The details can be found in [Pinkus, 1999].

Lesson Three

1.3 Universality of (deep) ReLU Neural Networks

Recall that for a fixed activation function $\sigma : \mathbb{R} \to \mathbb{R}$ and integer vector

$$\vec{w} = (w_0, w_1, \ldots, w_L, w_{L+1}) \in \mathbb{N}^{L+2},$$

we defined a *hypothesis class*

$$\mathcal{FC}\left(\vec{w}; \sigma\right) = \{h(x) = h_{L+1} \circ \sigma \circ h_L \ldots \circ \sigma \circ h_1(x) \text{ where } h_i : \mathbb{R}^{w_{i-1}} \to \mathbb{R}^{w_i} \text{ is affine } \forall i = 1, \ldots, L+1\}.$$

We also said that we will often focus on the case where all hidden layers have the same width. The width vector $\vec{w}$ is then described by four hyper-parameters

$$\vec{w}\left(W, L, w_{in}, w_{out}\right) = (w_0 = w_{in}, w_1 = W, w_2 = W, \ldots, w_L = W, w_{L+1} = w_{out})$$

and accordingly these hyper-parameters define the hypothesis class

$$\mathcal{FC}\left(W, L, w_{in}, w_{out}, \sigma\right) = \mathcal{FC}\left(\vec{w}\left(W, L, w_{in}, w_{out}\right); \sigma\right)$$

From now on we will focus mostly on the case where $\sigma = ReLU$. For short, we will denote $\rho := ReLU$, and when referring to the architectures above with ReLU activation we will drop the activation from the notation, so that our architectures will just be denoted by $\mathcal{FC}\left(\vec{w}\right)$ and $\mathcal{FC}\left(W, L, w_{in}, w_{out}\right)$.

In this section we will prove the universality of neural networks with ReLU activations. What we will do now is discuss in general some basic properties of ReLU activation neural networks functions. Once we establish these, proving universality will be rather straightforward. The properties we will discuss will be useful not only for this theorem but for our understanding of ReLU networks and various other results we will see later on.

1.3.1 Basic approximation properties of ReLU networks

We now discuss some elementary useful properties of fully connected neural networks. For example, it would be natural to hope that smaller networks will be contained in bigger networks. This is indeed the case (up to a small caveat...)

Composition, concatenation, and linear combinations

1. **Going wider can't hurt** Given $\vec{w} = (w_0, \ldots, w_{L+1}) \in \mathbb{N}^{L+2}$, every NNF which is in $\mathcal{FC}(\vec{w})$ is also in $\mathcal{FC}([w_0, w_1, \ldots, w_j + n, w_{j+1}, \ldots, w_{L+1}])$, for every $1 \leq j \leq L$ and every $n \in \mathbb{N}$. Let

$$h(x) = h_{L+1} \circ \rho \circ h_L \ldots \circ \rho \circ h_1(x) \text{ where } h_i : \mathbb{R}^{w_{i-1}} \to \mathbb{R}^{w_i} \text{ is affine } \forall i = 1, \ldots, L$$

be a function in $\mathcal{FC}(\vec{w})$. To see our claim is true, it is sufficient to show that there exist $\tilde{h}_{j+1} : \mathbb{R}^{w_j+n} \to \mathbb{R}^{w_{j+1}}$ and $\tilde{h}_j : \mathbb{R}^{w_{j-1}} \to \mathbb{R}^{w_j+n}$ such that

$$\tilde{h}_{j+1} \circ \rho \circ \tilde{h}_j = h_{j+1} \circ \rho \circ h_j$$

which can be obtained by setting

$$\tilde{h}_j(x) = \begin{pmatrix} 0_n \\ h_j(x) \end{pmatrix}, \text{ and } \tilde{h}_{j+1} \begin{pmatrix} y \\ z \end{pmatrix} = h_{j+1}(z).$$

We can deduce that if the first and last coordinates of $\vec{w}' \in \mathbb{N}^{L+2}$ and $\vec{w} \in \mathbb{N}^{L+2}$ are the same, and all other coordinates of $\vec{w}'$ are larger, then $\mathcal{FC}(\vec{w})$ is contained in $\mathcal{FC}(\vec{w}')$. In particular, for a given $\vec{w} \in \mathbb{N}^{L+2}$, we always have that $\mathcal{FC}(\vec{w}) \subseteq \mathcal{FC}(W, L, w_0, w_{L+1})$ where $W = \max_{1 \leq i \leq L} w_i$.

2. **Composition** If $h \in \mathcal{FC}(W, L, w_{in}, w_{out})$ and $h' \in \mathcal{FC}(W', L', w'_{in}, w'_{out})$ and $w'_{in} = w_{out}$, then
$$h' \circ h \in \mathcal{FC}(W'', L'', w_{in}, w'_{out})$$
where
$$L'' = L + L' \text{ and } W'' = \max\{W, W'\}$$
This is because h and h' are of the form
$$h(x) = h_{L+1} \circ \rho \circ h_L \ldots \circ \rho \circ h_1(x)$$
$$h'(x) = h'_{L'+1} \circ \rho \circ h'_{L'} \ldots \circ \rho \circ h'_1(x)$$
and so
$$h' \circ h = h'_{L'+1} \circ \rho \circ h'_{L'} \ldots \circ \rho \circ h'_1 \circ h_{L+1} \circ \rho \circ h_L \ldots \circ \rho \circ h_1$$
we get a fully connected network with 'width vector'
$$\vec{w} = (w_{in}, W, \ldots, W, W', \ldots, W', w'_{out})$$
and since 'going wider can't hurt' we can replace W, W' with $W'' = \max(W, W')$.

3. **Concatenation** If h_1 and h_2 are neural networks in $\mathcal{FC}(W_1, L, w_{in}, w_{out} = w_1)$ and $\mathcal{FC}(W_2, L, w_{in}, w_{out} = w_2)$ respectively, then
$$h(x) = \begin{pmatrix} h_1(x) \\ h_2(x) \end{pmatrix}, h : \mathbb{R}^{w_{in}} \to \mathbb{R}^{w_1 + w_2}$$
is in $\mathcal{FC}(W_1 + W_2, L, w_{in}, w_1 + w_2)$.

4. **Going deeper doesn't hurt (too much)** Given a NNF $h \in \mathcal{FC}(W, L; w_{in}, w_{out})$ with $L \geq 1$, so
$$h(x) = h_{L+1} \circ \rho \circ h_L \ldots \circ \rho \circ h_1(x)$$
we have for all $\tilde{L} \geq L$ that $h \in \mathcal{FC}(W, \tilde{L}; w_{in}, w_{out})$ since we can write
$$h(x) = h_{L+1} \circ \rho \circ I \ldots \circ \rho \circ I \circ \rho \circ I \circ \rho \circ h_L \ldots \circ \rho \circ h_1(x)$$
and we are using the fact that
$$\rho \circ \rho(x) = \begin{cases} \rho(x) & \text{if } x > 0 \\ \rho(0) & \text{if } x \leq 0 \end{cases} = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases} = \rho(x).$$
What happens if $L = 0$, so we are talking just of an affine function? This actually is not so straightforward. Let us focus only on the identity function $x \mapsto x$ which is in $\mathcal{FC}(W = n, L = 0, w_{in} = n, w_{out} = n)$ (the W is in fact meaningless since there are no hidden neurons at depth zero). This function cannot in fact be expressed exactly in $\mathcal{FC}(W = n, L = 1, w_{in} = n, w_{out} = n)$ as we saw in homework. There are two tricks to overcome this: the first is to increase the width: note that
$$\rho(x) - \rho(-x) = x$$
so the identity function is in $\mathcal{FC}(W = 2n, L = 1, w_{in} = n, w_{out} = n)$. By the composition rule, we can compose the identity function with itself L times for any $L \in \mathbb{N}$ and obtain that it is in $\mathcal{FC}(W = 2n, L, w_{in} = n, w_{out} = n)$.

An alternative trick does not require increasing width, but only applies when restricting to compact sets. That is, for any compact $K \subseteq \mathbb{R}^n$ and every $L \in \mathbb{N}$ we can find a function $h \in \mathcal{FC}(W = n, L, w_{in} = n, w_{out} = n)$ such that
$$h(x) = x, \forall x \in K.$$
This is done as follows: choose $b \in \mathbb{R}^n$ whose coordinates are all large enough so that
$$x_i + b_i \geq 0, \forall i = 1, \ldots, n, \forall x \in K.$$
and thus $\rho(x + b) = x + b$ for all $x \in K$. Then apply the identity transformation as much as necessary and conclude with the inverse translation $x \mapsto x - b$.

1.3.2 Expressing functions as ReLU networks: first examples

Some functions can be not only approximated by ReLU networks, but actually can be exactly *expressed* as neural networks. We will now see some simple but useful examples, which in particular will be useful for our universality proof later on.

Problem 1.7. 1. Write the function $(x_1, x_2) \mapsto \max(x_1, x_2)$ as a ReLU neural network. What are the width and depth?

2. For any $a < b < c < d$ show that the trapezoid function $h_{a,b,c,d}$ illustrated in Figure 1.1(b) can be realized by a neural network.

We have

$$\max(x_1, x_2) = \max(x_2 - x_1, 0) + x_1 = \max(x_2 - x_1, 0) + \max(x_1, 0) - \max(-x_1, 0)$$

so $\max \in \mathcal{FC}(W = 3, L = 1, n_{in} = 2, n_{out} = 1)$. In homework you will show this can be extended to finding the maximum of a d dimensional vector:

Lemma 1.8. *The function $h : \mathbb{R}^d \to \mathbb{R}$ defined by $h(x) = \max(x)$ is expressible as a neural network in $\mathcal{FC}(W, L, n_{in} = d, n_{out} = 1)$ with width $W = 3d$ and depth $L = \lceil \log_2(d) \rceil$.*

We note that we can also express the minimum function with a network of the same size. This follows from the fact that

$$\min(x) = -\max(-x), \forall x \in \mathbb{R}^d.$$

We now discuss how to create the trapezoid function (note there is more than one way to do this).

Lemma 1.9. *For any $a < b < c < d$ the trapezoid function $h_{a,b,c,d}$ illustrated in Figure 1.1(b) can be expressed as a neural network in $\mathcal{FC}(W, L, n_{in} = w, n_{out} = 1)$ with depth $L = 1$ and width $W = 4$.*

Proof. Let us relabel a, b, c, d as $a_1, \dots, a_4$, and choose some $a_5 > a_4$. We guess that there exist $\alpha_1, \dots, \alpha_4$ such that

$$h_\alpha(x) = \sum_{i=1}^{4} \alpha_i \rho(x - a_i)$$

is the trapezoid function. Note that for any choice of $\alpha_1, \dots, \alpha_4$ the obtained function is linear on the intervals

$$(-\infty, a_1], [a_4, \infty) \text{ and } [a_i, a_{i+1}], i = 1, \dots, 3,$$

Accordingly is is sufficient to find on each such interval two points on which the expected value is obtained. Note that by construction the function h_α is zero on $(-\infty, \alpha_1]$. It is thus sufficent to find $\alpha_1, \dots, \alpha_4$ such that

$$\begin{aligned} h_\alpha(a_2) &= 1 \\ h_\alpha(a_3) &= 1 \\ h_\alpha(a_4) &= 0 \\ h_\alpha(a_5) &= 0 \end{aligned}$$

By definition of the ReLU function ρ, this is equivalent to the equations

$$\begin{pmatrix} a_2 - a_1 & 0 & 0 & 0 \\ a_3 - a_1 & a_3 - a_2 & 0 & 0 \\ a_4 - a_1 & a_4 - a_2 & a_4 - a_3 & 0 \\ a_5 - a_1 & a_5 - a_2 & a_5 - a_3 & a_5 - a_4 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

The matrix on the left hand side is non-singular, as its determinant is just the product of the non-zero diagonal elements. Thus this equation has a unique solution and the h_α obtained by using the vector α solving the equation is the NNF we wanted. □

We'd like to think of the trapezoid function we just constructed as an approximation of an indicator function for the unit interval $[b, c]$ by a continuous function (think of the case $a = b - \epsilon, d = c + \epsilon$). We can now use this and the maximum function we constructed, to construct approximations of a indicator for a high-dimensional cube:

Lemma 1.10. *Let $\hat{a}, a, \hat{b}, b \in \mathbb{R}^k$ be such that $\hat{a}_i < a_i < b_i < \hat{b}_i$ for all $i = 1, \dots, k$, then there exists a neural network function h in $\mathcal{FC}(W = 4k, L = 1 + \lceil \log_2(k) \rceil, n_{in} = k, n_{out} = 1)$ such that*

1. *$0 \le h(x) \le 1$ for all $x \in \mathbb{R}^k$.*
2. *$h(x) = 1$ for all x in the cube $C = \prod_{i=1}^k [a_i, b_i]$.*
3. *$h(x) = 0$ for all x outside the cube $\hat{C} = \prod_{i=1}^k [\hat{a}_i, \hat{b}_i]$.*

We will say that such an h is a quasi-indicator of C with support in $\hat{C}$.

Proof. For all $i = 1, \dots, k$ let h_i denote the trapezoid function $h_i = h_{\hat{a}_i, a_i, b_i, \hat{b}_i}$. Each one of these functions can be realized as a neural network, as can be the minimum function, and therefore so can

$$h(x) = \min_{i=1,\dots,k} h_i(x_i)$$

which satisfies the conditions of the lemma. By Lemma 1.8 the minimum function is in $\mathcal{FC}(W, L, n_{in} = k, n_{out} = 1)$ with width $W = 3k$ and depth $L = \lceil \log_2(k) \rceil$ while each of the h_i can be realized with width four and depth 1 by Lemma 1.9. Using the concatenation and composition rules we see that the function

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{pmatrix} \mapsto \begin{pmatrix} h_1(x_1) \\ h_2(x_2) \\ \vdots \\ h_k(x_k) \end{pmatrix} \mapsto \min\{h_1(x_1), h_2(x_2), \dots, h_k(x_k)\}$$

is in $\mathcal{FC}(W = 4k, L = 1 + \lceil \log_2(k) \rceil, n_{in} = k, n_{out} = 1)$. $\square$

Universality We now want to prove universality of ReLU networks, using the fact that they can approximate cube indicators and minimum and maximum functions. For any set $C \subseteq \mathbb{R}^d$ we define its indicator function to be

$$1_C(x) = \begin{cases} 1 & \text{if } x \in C \\ 0 & \text{if } x \notin C \end{cases} .$$

For practice, let us prove the following theorem

Theorem (Universality of real indicator functions)**.** *Let $f : [0, 1]^d \to \mathbb{R}$ be a continuous, non-negative function. For every $\epsilon > 0$ there exist $n = n(\epsilon)$ cubes $C_1, \dots, C_n$ and numbers $c_1, \dots, c_n$ such that*

$$|f(x) - \max_{k=1,\dots,n} c_k 1_{C_k}(x)| < \epsilon, \forall x \in [0, 1]^d$$

Moreover, if f is ℓ Lipschitz then we can choose $n(\epsilon) \le \left(\frac{\sqrt{d}\ell}{2\epsilon} + 1\right)^d$.

Proof. Let $\epsilon > 0$. Since f is continuous on a compact set, it is uniformly continuous, and so there exists some $\delta > 0$ such that

$$\forall x, y \in [0, 1]^d \text{ if } |x - y| \le \delta \text{ then } |f(x) - f(y)| \le \epsilon.$$

Moreover, if f is ℓ Lipschitz then we can take $\delta = \epsilon/\ell$, because this implies that if $|x - y| < \delta$ then

$$|f(x) - f(y)| \le \ell |x - y| \le \ell \delta \le \epsilon.$$

Now, For every N we define a partition of $[0, 1]$ into N intervals of the from $I_k = [\frac{k-1}{N}, \frac{k}{N}]$ where $k = 1, \dots, N$. This partition induces a partition of all of $[0, 1]^d$ into N^d cubes of the general form

$$C_k = I_{k_1} \times I_{k_2} \times \dots \times I_{k_d}, \text{ where } k = (k_1, \dots, k_d) \in \{1, \dots, N\}^d$$

We choose N large enough so that $\frac{\sqrt{d}}{2N} < \delta$. This means that the distance of any point in the cube C_k from the center of the cube is less than δ. Let x_k denote the center of the cube C_k. It follows that for $c_k = f(x_k)$ we have that for all $x \in C_k$

$$|f(x) - c_k 1_{C_k}(x)| = |f(x) - f(x_k)| \leq \epsilon$$

while if $x \notin C_k$ we have that $c_k 1_{C_k}(x) = 0$. Thus we have that for all $x \in [0,1]^d$, the maximum of $c_k 1_{C_k}(x)$ over all k is obtained for some $k_* = k_*(x)$ for which $x \in C_{k_*}$. We obtain

$$|f(x) - \max_{k=1,\ldots,N} c_k 1_{C_k}(x)| = |f(x) - c_{k_*} 1_{C_{k_*}}(x)| < \epsilon.$$

We now consider the number of cubes needed for an ℓ-Lipschitz function. In out construction we used N^d cubes, and we required that $\frac{\sqrt{d}}{2N} < \delta = \epsilon/\ell$. By rearranging we find that this is equivalent to the requirement that $N > \frac{\sqrt{d}\ell}{2\epsilon}$. The nearest largest N to this number is smaller than $\frac{\sqrt{d}\ell}{2\epsilon} + 1$, and therefore in total we can achieve ϵ approximation with $\leq \left(\frac{\sqrt{d}\ell}{2\epsilon} + 1\right)^d$ cubes. □

Curse of Dimensionality Our proof achieved ϵ accuracy with approximately $(1/\epsilon)^d$ cubes. This type of phenomena where the complexity depends exponentially on the dimensionality is called the 'curse of dimensionality'.

We now prove universality for ReLU networks. The idea is to imitate the proof of the previous theorem, based on the fact that we can express the maximum function and approximate the cube indicator function used in the previous theorem.

Theorem 1.11 (Universality for ReLU networks)**.** *Let $f : [0,1]^d \to \mathbb{R}$ be a continuous function. For every $\epsilon > 0$ there exist a neural network function $h : \mathbb{R}^d \to \mathbb{R}$ such that*

$$|f(x) - h(x)| < \epsilon, \forall x \in [0,1]^d$$

Moreover, if f is ℓ-Lipschitz then we can choose h to be in $\mathcal{FC}(W, L, d, 1)$ where

$$W \leq 4d\left(\frac{\sqrt{d}\ell}{2\epsilon} + 1\right)^d, L \leq \lceil \log_2(d) \rceil + \lceil d \cdot \log_2(\left(\frac{\sqrt{d}\ell}{2\epsilon} + 1\right)) \rceil$$

Proof. Choose some $\epsilon > 0$, we want to construct a NNF h s.t.

$$\|f - h\|_{C(K)} < \epsilon. \tag{1.5}$$

Since f is continuous on a compact set it is bounded from below, so we can find some $b \geq 0$ such that the function $\tilde{f}(x) = f(x) + b$ is non-negative. We will find a NNF $\tilde{h}$ for which

$$\|\tilde{f} - \tilde{h}\|_{C(K)} < \epsilon$$

which implies that $h(x) = \tilde{h}(x) - b$ satisfies (1.5). Note that h will be a NNF with the same width and depth as $\tilde{h}$.

Since $\tilde{f}$ is continuous on a compact set it is uniformly continuous and so there exists some $\delta > 0$ such that

$$\forall x, y \in [0,1]^d \text{ if } |x - y| \leq \delta \text{ then } |f(x) - f(y)| \leq \epsilon.$$

Moreover, if f is ℓ Lipschitz then $\tilde{f}$ will be ℓ Lipschitz as well, and we can take $\delta = \epsilon/\ell$ as we saw before.

Again as before, for every N we define a partition of $[0,1]$ into N intervals of the form $I_k = [\frac{k-1}{N}, \frac{k}{N}]$ where $k = 1, \ldots, N$. This partition induces a partition of all of $[0,1]^d$ into N^d cubes of the general form

$$C_k = I_{k_1} \times I_{k_2} \times \ldots \times I_{k_d}, \text{ where } k = (k_1, \ldots, k_d) \in \{1, \ldots, N\}^d$$

We choose N large enough so that $\frac{\sqrt{d}}{2N} < \delta$, which means that the distance of x_k from any point in C_k will be less than δ.

Let x_k denote the center of C_k, and denote $c_k = f(x_k)$. Let ϕ_k be a quasi-indicator of C_k supported in $\hat{C}_k$, where $\hat{C}_k$ is chosen so that its center is also x_k, it contains C_k, and is close enough to C_k such that the distance of any point in $\hat{C}_k$ from x_k is still smaller than δ. We deduce that if $\phi_k(x) > 0$ then x is in $\hat{C}_k$ and so $|x - x_k| < \delta$ which implies that $|f(x) - c_k| = |f(x) - f(x_k)| < \epsilon$. It follows that for every $x \in [0,1]^d$ the maximum of $c_k\phi_k(x)$ over all k will be obtained by some $k_* = k_*(x)$ for which $\phi_k(x) > 0$, and we have that

$$\max_k(c_k\phi_k(x)) - f(x) = \phi_{k_*}(x)f(x_{k_*}) - f(x) \le f(x_{k_*}) - f(x) \le \epsilon.$$

On the other hand, since the cubes cover all the domain, there exists some k_{**} for which $x \in C_{k_{**}}$, and thus

$$\max_k(c_k\phi_k(x)) - f(x) \ge c_{k_{**}}\phi_{k_{**}}(x) - f(x) = f(x_{k_{**}}) - f(x) \ge -\epsilon$$

and so the function $x \mapsto \max_k(c_k\phi_k(x))$ is an ϵ approximation of f. This function can be realized as a neural network through the construction

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \mapsto \begin{pmatrix} c_1\phi_1(x_1) \\ c_2\phi_2(x_2) \\ \vdots \\ c_n\phi_n(x_n) \end{pmatrix} \mapsto \max\{c_1\phi_1(x_1), c_2\phi_2(x_2), \ldots, c_n\phi_n(x_n)\}$$

where $n = N^d$ denotes the number of cubes. By Lemma 1.10 the width of each ϕ_k is $4d$ and the depth is $\lceil \log_2(d) \rceil$, and so overall the width of the function we constructed would be $4nd$ and the depth would be $\lceil \log_2(d) \rceil + \lceil \log_2(n) \rceil$. As in our previous claim, when f is ℓ-Lipschitz we can take $n(\epsilon) \le \left(\frac{\sqrt{d}\ell}{2\epsilon} + 1\right)^d$ and so we can obtain ϵ approximation with width and depth bounded by

$$W \le 4d\left(\frac{\sqrt{d}\ell}{2\epsilon} + 1\right)^d, L \le \lceil \log_2(d) \rceil + \lceil d \cdot \log_2\left(\frac{\sqrt{d}\ell}{2\epsilon} + 1\right) \rceil$$

□

To conclude, we showed that ReLU networks can 'imitate' approximation by piecewise constant functions. The original piecewise constant construction, as well as the ReLU imitation, suffer from the curse of dimensionality. Later on in the course we will see that smoothness of f enables neural networks to 'overcome' the curse of dimensionality.

1.4 ReLU Networks, CPwL Functions and Depth Separation

1.4.1 ReLU networks are Continuous Piecewise Linear functions

A good reference for this topic is again [DeVore et al., 2021] and references therein. ReLU networks always product continuous functions, as they are compositions of affine functions and ReLU which are both continuous. In fact, ReLU networks always produce Continuous Piecewise Linear (CPwL) functions, since ReLU and affine functions are CPwL functions. Before we see this, we need to formally defined what CPwL functions are. For this we will need some more definitions...

Definition 1.12. A closed convex polytope $P \subseteq \mathbb{R}^d$ is a set of the form

$$P = \{x \in \mathbb{R}^d | \langle a_i, x\rangle + b_i \ge 0 | \, i = 1, \ldots, m \text{ where for all } i \; a_i \in \mathbb{R}^d, b_i \in \mathbb{R}\} \tag{1.6}$$

As there name suggests, closed convex polytopes are always closed and convex (verify this). Examples include

1. The whole space $P = \mathbb{R}^d$ (with $m = 0$ constraints, or with the constraint $\langle 0, x\rangle + 1 \ge 0$).

2. The empty set is a polytope defined by the constraint $\langle 0, x\rangle - 1 \ge 0$.

3. The positive quadrant
$$Q = \{x \in \mathbb{R}^d |\, x_i \geq 0, i = 1, \dots, d\}.$$
This expression is of the same form as in (1.6) where $a_i = e_i, b_i = 0$ and $m = d$.

4. The unit hyper cube
$$C = \{x \in \mathbb{R}^d | -1 \leq x_i \leq 1,\, i = 1, \dots, d\}$$
which can be rewritten as in (1.6)
$$C = \{x \in \mathbb{R}^d | x_i \geq -1 \text{ and } -x_i \geq -1, \quad i = 1, \dots, d\}$$
.

5. The plane
$$P = \{(x_1, x_2, x_3) |\, x_3 = 0\}$$
which can be rewritten as in (1.6)
$$P = \{x \in \mathbb{R}^3 |\, x_3 \geq 0 \text{ and } -x_3 \geq 0\}.$$
In general, using this same trick we see that closed convex polytopes can be defined by affine equalities as well as affine inequalities.

Remark 1.13. From now on we will use the term polytope to denote a closed convex polytope.

Problem 1.14. Show that all bounded polytopes in $\mathbb{R}$ are points or closed intervals.

Definition 1.15. A polytope covering $\mathcal{P}$ of $\mathbb{R}^d$ is a finite collection of polytopes $\mathcal{P} = \{P_1, \dots, P_R\}$ such that $\bigcup_{r=1}^R P_r = \mathbb{R}^d$

Problem 1.16. Which of the following is not a polytope covering of $\mathbb{R}^2$

1. The collection of polytopes
$$P_1 = \{(x, y)|x \geq 0\}, P_2 = \{(x, y)|x \leq 0\}, P_3 = \{(x, y)|y \geq 0\}, P_4 = \{(x, y)|y \leq 0\}$$

2. The collection of polytopes
$$P_1 = \{(x, y)|x \geq 0\}, P_2 = \{(x, y)|x \leq 0\}, P_3 = \{(x, y)|y \geq 0\}, P_4 = \{(x, y)|y < 0\}$$

3. The collection of polytopes
$$P_1 = \{(x, y)|x \geq 0\}, P_2 = \{(x, y)|x \leq 0\}, P_3 = \{(x, y)|y \geq 0\}, P_4 = \{(x, y)|y \leq 0\}, P_5 = \{(1, 1)\}$$

4. The collection of polytopes
$$P_1 = \{(x, y)|x \geq -1\}, P_2 = \{(x, y)|x \leq 1\}, P_3 = \{(x, y)|y \geq 0\}, P_4 = \{(x, y)|y \leq 0\}$$

Solution The second, because P_4 is not closed.
We can finally define a CPwL function

Definition 1.17. We say that $f : \mathbb{R}^d \to \mathbb{R}^\ell$ is a CPwL function subordinate to the polytope covering $\mathcal{P} = \{P_1, \dots, P_R\}$ if f is continuous, and the restriction of f to each P_i is an affine function.

We say that f is a CPwL function if it is CPwL function subordinate to some polytope partition.

Remark 1.18. We follow the definition in [DeVore et al., 2021] and require explicitly that f is continuous, though this follows from the other assumptions. On the other hand, we do not require as in [DeVore et al., 2021] that the polytope covering is a partition (i.e., that the interiors of the polytopes do not intersect). This makes the proof of Proposition 1.20 easier and does not hurt generality: if f is CPwL subordinate to some covering, then this covering can be refined to a partition which f will be subordinate to. We will see this soon.

Let us see some examples for CPwL functions. Perhaps the two most important examples for us are affine functions, which are CPwL with respect to the trivial partition $\mathcal{P} = \{\mathbb{R}^d\}$, and the ReLU function. The ReLU function is a linear function on each of the 2^d quadrants of $\mathbb{R}^d$. Formally, for every vector $s \in \{-1,1\}^d$ we can define the polytope

$$Q_s = \{x \in \mathbb{R}^d |\, s_i x_i \geq 0, \forall i = 1, \dots, d\}$$

The collection of all such polytopes form a polytope partition, and

$$ReLU(x) = \frac{1}{2}\begin{pmatrix} 1+s_1 & 0 & 0 & \dots & 0 \\ 0 & 1+s_2 & 0 & \dots & 0 \\ 0 & 0 & 1+s_3 & \dots & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & 1+s_d \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_d \end{pmatrix}, \forall x \in Q_s. \tag{1.7}$$

Thus, ReLU is a CPwL function.

Problem 1.19. Show that the maximum functions from $\mathbb{R}^d$ to $\mathbb{R}$ is CPwL.

As we saw that affine functions and ReLU functions are both CPwL, we can now deduce that every NNF is a CPwL function by showing that the composition of CPwL functions are also CPwL.

Proposition 1.20. *If $f : \mathbb{R}^d \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^m$ are CPwL functions then $g \circ f$ is a CPwL functions.*

Proof. By assumption f is subordinate to some polytope covering $\mathcal{P} = \{P_1, \dots, P_R\}$ and g is subordinate to some polytope covering $\mathcal{Q} = \{Q_1, \dots, Q_S\}$. The restriction of f to each P_r is an affine function f_r, and the restriction of g to each Q_s is an affine function g_s. We can now define a new polytope covering $\{T_{r,s}\}_{1\leq r\leq R, 1\leq s\leq S}$ by

$$T_{r,s} = \{x \in P_r \text{ and } f_r(x) \in Q_s\}$$

We note that these sets do indeed cover all of $\mathbb{R}^d$, that $g \circ f$ is continuous and its restriction to $T_{r,s}$ is the affine function $g_s \circ f_r$, and finally that each $T_{r,s}$ is a polytope, for if

$$P_r = \{x \in \mathbb{R}^d | \langle a_i, x\rangle + b_i \geq 0, i = 1, \dots, m\}, f_r(x) = Ax + b$$

and

$$Q_s = \{y \in \mathbb{R}^n | \langle c_j, y\rangle + d_j \geq 0, j = 1, \dots, \ell\}$$

then

$$\begin{aligned} T_{r,s} &= \{x \in P_r \text{ and } f_r(x) \in Q_s\} \\ &= \{x \in \mathbb{R}^d | \langle a_i, x\rangle + b_i \geq 0 \text{ and } \langle c_j, Ax + b\rangle + d_j \geq 0, \quad i = 1, \dots, m, j = 1, \dots \ell\} \\ &= \{x \in \mathbb{R}^d | \langle a_i, x\rangle + b_i \geq 0 \text{ and } \langle A^T c_j, x\rangle + [d_j + \langle c_j, b\rangle] \geq 0, \quad i = 1, \dots, m, j = 1, \dots \ell\} \end{aligned}$$

□

Lesson 4

1.4.2 From coverings to partitions

We saw that any ReLU network is a CPwL subordinate to some polytope covering. This covering is not unique. For example, the function $f(x) = |x|$ is subordinate to the covering $\mathcal{P} = \{(\infty, 0], [0, 1], \{1/2\}, [1/2, \infty)\}$ but also to the covering $\mathcal{P}' = \{(\infty, 0], [0, \infty)\}$. Clearly, the second covering is more natural as it is minimal, and forms a 'partition' of the space. We will now define the notion of a partition:

Definition 1.21. Let $P \subseteq \mathbb{R}^d$. The interior of P, denoted by $int(P)$, are all points $p \in P$ such that there exists some open ball B with $p \in B \subseteq P$.

For example, the interior of $[0, 1] \in \mathbb{R}$ is $(0, 1)$. More generally

Problem 1.22. Let $P = \{x \in \mathbb{R}^d | \langle a_i, x\rangle + b_i \geq 0, i = 1, \ldots, m\}$ be a polytope. Show that its interior is the set

$$P = \{x \in \mathbb{R}^d | \langle a_i, x\rangle + b_i > 0, i = 1, \ldots, m\}$$

Definition 1.23. Let $\mathcal{P} = \{P_1, \ldots, P_N\}$ be a finite collection of polytopes in $\mathbb{R}^d$. we say that $\mathcal{P}$ is a *partition* of $\mathbb{R}^d$ if

1. $\bigcup_i P_i = \mathbb{R}^d$.
2. Each P_i had non-empty interior.
3. $int(P_i) \cap int(P_j)$ is empty for all $i \neq j$.

In the example above, we see that $\mathcal{P}'$ is a partition, while $\mathcal{P}$ is a covering but not a partition since $\{1/2\}$ has empty interior, and since the intersection of the interior of two intervals in $\mathcal{P}$ is not empty.

Proposition 1.24. *If $f : \mathbb{R}^d \to \mathbb{R}^\ell$ is a CPwL function subordinate to a covering $\mathcal{P}$, then it is also subordinate to some partition $\mathcal{P}'$.*

Proof. We build $\mathcal{P}'$ from the given covering $\mathcal{P} = \{P_1, \ldots, P_N\}$.

First note that if some P_j has empty interior then it is contained in the union of all other sets and therefore it can be removed from $\mathcal{P}$. This is because every point $p \in P_j$ has a sequence of points p_n converging to it which are not in P_j, and so since all P_i cover $\mathbb{R}^d$, the sequence is in $\cup_{i:\, i\neq j} P_i$. Since this set is closed as a finite union of closed sets, we have that p is also in this set. Thus we can assume without loss of generality that all elements in $\mathcal{P}$ have non-empty interior.

Now, assume we have two $P_i \neq P_j$ whose interiors intersect. We can write

$$P_i = \{x \in \mathbb{R}^d | \langle a_i, x\rangle + b_i \geq 0, i = 1, \ldots, m\}$$

and

$$P_j = \{x \in \mathbb{R}^d | \langle c_j, x\rangle + d_j \geq 0, j = 1, \ldots, \ell\}.$$

We will show how to defined a new finite collection of polytopes whose union is exactly $P_i \cup P_j$ and who do not intersect.

The intersection $P_i \cap P_j$ is a polytope

$$P_i \cap P_j = \{x \in \mathbb{R}^d | \langle a_i, x\rangle + b_i \geq 0, i = 1, \ldots, m \text{ and } \langle c_j, x\rangle + d_j \geq 0, j = 1, \ldots, \ell\}$$

The set $P_i \setminus (P_i \cap P_j)$ is not necessarily a polytope, but it can be written as a finite union of the polytopes

$$P^i_s = \{x \in \mathbb{R}^d | \langle a_i, x\rangle + b_i \geq 0, i = 1, \ldots, m \text{ and } \langle s_j c_j, x\rangle + s_j d_j \geq 0, j = 1, \ldots, \ell\}$$

where $s \in \{-1, 1\}^\ell$. That is

$$P_i = \cup_{s\in\{-1,1\}^\ell} P^i_s$$

and the polytope corresponding to $s = (1, \ldots, 1)$ is $P_i \cap P_j$. Note that the interior of the different P^s_i do not intersect.

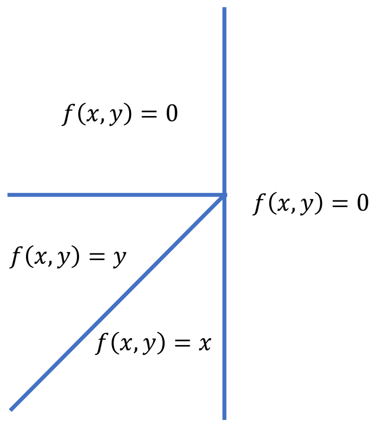

Figure 1.2: In our definition f has four (convex) linear region, though it has three non-convex linear regions.

We can do the same thing to P_j and write it as a union of polytopes P_t^j where $t \in \{-1,1\}^m$ and $P_i \cap P_j$ is the polytope corresponding to $t=(1,\dots,1)$, and such the interior of the P_t^j do not intersect. Finally we consider this new collection of polytopes of all P_t^j and P_s^i and we find that their interiors do not intersect. Thus, we have showed how we can reduce the number of intersections between polytopes by one, and by repeating this process recursively we can transform a covering into a partition. $\square$

Definition 1.25. The number of linear regions of a CPwL function f is the smallest m such that f is subordinate to a *partition* $\mathcal{P}=\{P_1,\dots,P_m\}$.

Example 1.26. The function $f(x)=|x|$ is subordinate to the partition $\mathcal{P}=\{(\infty,0],[0,1],[1,\infty)\}$ but also to the smaller partition $\mathcal{P}'=\{(\infty,0],[0,\infty)\}$ and therefore the number of linear regions is ≤ 2. Clearly the number of linear regions cannot be 1 since f is not linear, and so the number of linear regions is two.

Example 1.27. The function in Figure 1.2 has four linear regions, though if we would have allowed non-polytope partitions the number of regions would have been reduced to three.

1.4.3 Depth separation and counting linear regions

In this subsection we will address some of the questions mentioned in previous lessons: can we show that neural networks have some expressive advantage over standard function bases (e.g., standard CPwL function spaces)? Can we show that deep neural networks ($L \gg 1$) have an expressive advantage over shallow networks ($L=1$)? These questions are motivated by the successfulness of deep learning in the past years, in comparison with alternative function spaces or shallow networks.

In this subsection we will consider univariate neural networks with $n_{in}=1=n_{out}$. We will also restrict our attention to the behavior of these functions on $[0,1]$. Thus we are interested in the function space

$$\Upsilon(W,L)=\{f:[0,1]\to\mathbb{R} | f=g_{|[0,1]} \text{ for some } g\in\mathcal{FC}(W,L,1,1)\}$$

Most of the results described here come from [Daubechies et al., 2021] unless mentioned otherwise.

As we saw, any function $f\in\mathcal{FC}(W,L,1,1)$ is a CPwL function $f:\mathbb{R}\to\mathbb{R}$ subordinate to a polytope partition of $\mathbb{R}$. A non-empty polytope in $\mathbb{R}$ with a non-empty interior is a closed interval (which may or may not be bounded from the left or right). Based on this, it is not difficult to see that a univariate CPwL function always has a finite sequence of breakpoints $k_1<k_2\dots<k_n$, on which it is not differentiable, and the restriction of f to the intervals $(-\infty,k_1]$ and $[k_i,k_{i+1}, i=1,\dots,n-1]$ and $[k_n,\infty)$ is an affine univariate function. In our case we focus on the function value on $[0,1]$, so f will be linear on the intervals $[k_{i-1},k_i]$ where $i=1,\dots,n+1$ and $k_0=0, k_{n+1}=1$. The number of linear regions of f will be n in this case.

One natural question to ask is whether every CPwL univariate function on $[0,1]$ can be realized a neural network? The answer to this is affirmitive. We then want to compare neural networks with more standard CPwL bases:

Free knot splines For fixed n, let Σ_n be the set of CPwL functions determined by $2n+2$ parameters: n knots $0 < k_1 < \ldots < k_n < 1$, n assigned values y_i to each of the knots, and values y_0 and y_{n+1} assigned to 0 and 1. These values then uniquely define a CPwL function on $[0,1]$ which attains these values and is affine in the intervals defined by the knots. We refer to this function spaces as the space of free knot splines with n knots. Note that a function in Σ_n can have a breakpoint (a non-differentiable point) only at a knot, but a knot does not necessarily have to be a breakpoint. In fact Σ_n is exactly the set of all CPwL functions on $[0,1]$ with n or less breakpoints, and in particular $\Sigma_n \subseteq \Sigma_{n+1}$.

Counting parameters in $\Upsilon(W,L)$ We will compare function spaces Σ_n and $\Upsilon(W,L)$ in terms of the number of their parameters on the one hand, and their expressive power on the other hand. Recall that Σ_n is defined by $2n+2$ parameters. Let's now count the number of parameters in $\Upsilon(W,L)$:

For fixed W, L the number of parameters which determine functions in $\Upsilon(W,L)$ is

$$\begin{aligned} n(W,L) &= 2W + (L-1)(W^2+W) + W + 1 \\ &= 3W + 1 + (L-1)(W^2+W) \end{aligned}$$

When $L=1$ we have that $n(W,L) = 3W+1$. When $L \geq 2$ we see that $n(W,L)$ is proportional to LW^2, as for all integer W and $L \geq 2$

$$\begin{aligned} 3W + 1 + (L-1)(W^2+W) &\leq 3W^2 + W^2 + 2(L-1)W^2 = W^2(2L+2) \leq 3LW^2 \\ 3W + 1 + (L-1)(W^2+W) &\geq (L-1)W^2 \geq \frac{1}{2}LW^2 \end{aligned}$$

In the following we'd like to compare the function space Σ_n and $\Upsilon(W,L)$. We consider the following questions: can every CPwL univariate function be expressed as a NNF? If so, is it more efficient to parameterize CPwL functions as free knot splines or as neural networks? In the following we will show that for shallow networks these representations are more or less equivalent. We will then discuss deep networks where we will have more interesting results...

1.4.4 Shallow univariate neural networks

We begin by discussing shallow networks, that is networks with $L=1$. We prove the following

Proposition 1.28. *For every $W \in \mathbb{N}$ we have*

$$\Sigma_{W-1} \subseteq \Upsilon(W, L=1) \subseteq \Sigma_W$$

Recall that $n(W, L=1) = 3W+1$ while functions in Σ_n are determined by $2n$ parameters. Thus this theorem says that the number of parameters needed to express a CPwL functions via shallow neural networks or via free knot splines is roughly the same.

Proof. Let us first show that $\Upsilon(W, L=1) \subseteq \Sigma_W$. Recall that a function f in $\Upsilon(W, L=1)$ is of the form

$$f(x) = h_2 \circ ReLU \circ h_1$$

where $h_1 : \mathbb{R} \to \mathbb{R}^W$ and $h_2 : \mathbb{R}^W \to \mathbb{R}$ are affine, and can be written as

$$h_1(x) = x\vec{a} - \vec{b} \text{ and } h_2(\vec{y}) = \langle \vec{c}, \vec{y} \rangle + d$$

so

$$f(x) = d + \sum_{j=1}^{W} c_j \rho(a_j x - b_j) \tag{1.8}$$

the breakpoints of f are exactly the points x_j which solve the equation $a_j x - b_j$ and so there are at most W such solutions in $(0,1)$. Thus $f \in \Sigma_W$.

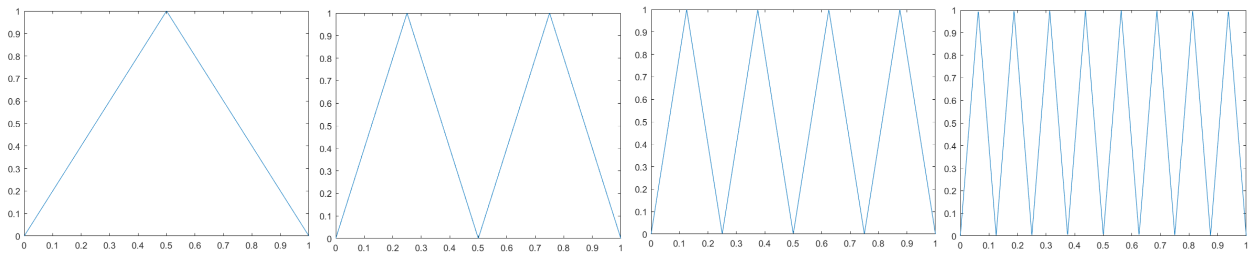

Figure 1.3: The function $H^{\circ s}$ for $s = 1, 2, 3, 4$.

In the other direction: Let $g \in \Sigma_{W-1}$. The function g has $W - 1$ knots $0 < k_1 < k_2 < \ldots < k_{W-1} < 1$. There exist some $c, d \in \mathbb{R}$ such that $g(x) = cx + d$ for all $x \leq k_1$. We claim that there exist $c_1, \ldots, c_{W-1}$ such that

$$g(x) = c\rho(x) + d + \sum_{i=1}^{W-1} c_i \rho(x - k_i)$$

and thus g is in $\Upsilon(W, 1)$. Thus, we need to show that we can select $c_1, \ldots, c_{W-1}$ such that

$$\tilde{g}(x) := g(x) - c\rho(x) - d = \sum_{i=1}^{W-1} c_i \rho(x - k_i), \quad \forall x \in [0, 1]$$

Note that for all selection of the c_i, the functions $\tilde{g}$ and the function on the right hand side both have knots at $k_1, \ldots, k_{W-1}$, and are zero on $[0, k_1]$. At this point we essentially repeat the argument we used in Lemma 1.9: it is sufficient to show that for an appropriate choice of the c_i, the two functions are equal on the points $k_2 < k_3 < \ldots < k_{W-1} < 1$. This gives us $W - 1$ linear equations in the c_i:

$$\begin{pmatrix} k_2 - k_1 & & & & \\ k_3 - k_1 & k_3 - k_2 & & & \\ k_4 - k_1 & k_4 - k_2 & k_4 - k_3 & & \\ \vdots & \vdots & & \ddots & \\ 1 - k_1 & 1 - k_2 & 1 - k_3 & \cdots & 1 - k_{W-1} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_{W-1} \end{pmatrix} = \begin{pmatrix} \tilde{g}(k_2) \\ \tilde{g}(k_3) \\ \tilde{g}(k_4) \\ \vdots \\ \tilde{g}(1) \end{pmatrix}$$

This equation has a unique solution since it is a lower diagonal matrix and all diagonal elements are non-zero. □

Problem 1.29. Give an example of a univariate CPwL function with W breakpoints that is not in $\Upsilon(W, 1)$.

1.4.5 Deep Univariate Neural Networks

We've seen in Proposition 1.28 that shallow neural networks of width W are essentially the same as free knot CPwL splines. What happens when $L > 1$? The following theorem, which is a simplified version of Theorem 3.1 in [Daubechies et al., 2021], shows that also when $L > 1$, we can express all functions with n breakpoints with $n \sim n(W, L)$:

Theorem 1.30. *[[Daubechies et al., 2021]] There exists an absolute constant $C > 0$, such that for every natural $W \geq 8$ and $n \geq \frac{1}{6}(W - 2)^2$, there exists an $L \in \mathbb{N}$ satisfying*

$$n(W, L) \leq Cn \text{ and } \Sigma_n \subseteq \Upsilon(W, L)$$

NNF with exponentially many knots Interestingly, for deep networks, it is possible to express *some* functions with $n \gg n(W, L)$ breakpoints as we will now see. The following question is in the homework:

Problem 1.31. Consider the function

$$H(x) = \begin{cases} 2x & \text{if } 0 \leq x \leq 1/2 \\ 2 - 2x & \text{if } 1/2 \leq x \leq 1 \end{cases}.$$

1. Show that $H \in \Upsilon(2,1)$ and that the function $H^{\circ s}$ (the composition of H with itself s times) is in $\Upsilon(2, L = s)$.

2. Define for every $s \in \mathbb{N}$ the points
$$k_0 = 0, k_1 = \frac{1}{2^s}, k_2 = \frac{2}{2^s} \ldots, k_{2^s} = 1.$$
Show that $H^{\circ s}(x)$ is the unique CPwL function which is affine on each interval $[k_{j-1}, k_j], j = 1, \ldots, 2^s$ and satisfies
$$H^{\circ s}(k_j) = \begin{cases} 0 & \text{if } j \text{ is even} \\ 1 & \text{if } j \text{ is odd} \end{cases}$$

3. How many breakpoints does $H^{\circ s}$ have?

We call the function H the 'hat function'. The composition of H with itself s times is called the sawtooth function(s), or sometimes Telgarsky's function. These are illustrated in figure 1.3. The function $H^{\circ s}$ are in $\Upsilon(2, s)$, so the number of parameters needed to express it grows linearly in s. However it has $2^s - 1$ breakpoints!!!

Here is a helpful way to see how many breakpoints H has easily:

Problem 1.32. say $f : [0,1] \to \mathbb{R}$ is CPwL and is affine in the intervals $[k_{j-1}, k_j]$ defined by the points $0 < k_1 < k_2 < \ldots < k_n < 1$. Show that $f \circ H$ is affine in the intervals defined by the points
$$0 < \frac{k_1}{2} < \ldots < \frac{k_n}{2} < \frac{1}{2} < 1 - \frac{k_n}{2} < \ldots < 1 - \frac{k_1}{2} < 1.$$

Using this argument we can compute the knots of $H^{\circ s}$. The calculation of $H^{\circ s}$ is then reduced to finding the value of $H^{\circ s}$ at these knots.

Bounding the number of breakpoints We saw that deep networks can have exponentially many breakpoints. The following gives an upper bound to the number of breakpoints

Proposition 1.33. *Every $f \in \Upsilon(W, L)$ has less than $(W+1)^L$ breakpoints.*

Note that this bound is not completely optimal. For example in [Raghu et al., 2017] it is shown that every $f \in \Upsilon(W, L)$ has no more than W^L breakpoints. Recall that $H^{\circ s}$ is in $\Upsilon(W = 2, L = s)$ and has $2^s - 1 = W^L - 1$ breakpoints, so this bound is very close to optimal.

Proof. We consider functions of the form
$$g = \rho \circ h_L \circ \rho \circ \ldots \circ \rho \circ h_1 \tag{1.9}$$
where $h_1 : \mathbb{R} \to \mathbb{R}^W$ and $h_i : \mathbb{R}^W \to \mathbb{R}^W, i = 1, \ldots L$ are affine. We denote the functions as in (1.9) by $\tilde{\Upsilon}(W, L)$. We denote the maximal number of breakpoints a function in $\tilde{\Upsilon}(W, L)$ can have by $m(L)$.

Note that any function $f \in \Upsilon(W, L)$ is a composition of an affine $h_{L+1} : \mathbb{R}^W \to \mathbb{R}$ with a function g as defined in (1.9). Since h_{L+1} is smooth $h_{L+1} \circ g$ will not have more breakpoints than g has and so it is sufficient to prove that $m(L) < (W+1)^L$. We prove this by induction.

When $L = 1$ we have that any function of the form $\rho \circ h_1$ can be written as
$$\rho \circ h_1(x) = \begin{pmatrix} \rho(a_1 x + b_1) \\ \rho(a_2 x + b_2) \\ \vdots \\ \rho(a_W x + b_W) \end{pmatrix}$$
the breakpoints of this function are the solutions of the equations $a_i x + b_i = 0$, and so $m(1) \leq W < W + 1$.

Now let us assume that we know for some given L that $m(L) < (W+1)^L$. Let g be some function in $\tilde{\Upsilon}(W, L+1)$ so $g = \rho \circ h_{L+1} \circ g_L$ where g_L is in $\tilde{\Upsilon}(W, L)$. We get

$$g(x) = \begin{pmatrix} \rho(\langle a_1, g_L(x)\rangle + b_1) \\ \rho(\langle a_2, g_L(x)\rangle + b_2) \\ \vdots \\ \rho(\langle a_W, g_L(x)\rangle + b_W) \end{pmatrix}$$

A function the the form $\rho(\langle a_i, g_L(x)\rangle + b_i)$ could have a breakpoint in every one of the original breakpoints of g_L. It could also have a 'new' breakpoint between every pair of consecutive breakpoints (if $g_L(k_i) > 0$ and $g_L(k_{i+1}) < 0$ or vice versa), or between 0 and the first breakpoint, or between 1 and the last breakpoint. All in all each one of the $\rho(a_i g_L(x) + b_i)$ can have up to $m(L)+1$ new breakpoints, and so altogether these functions could have up to $W(m(L)+1)$ new breakpoints, as well as the original $m(L)$ breakpoints of g_L. Altogether we will have no more than $W(m(L)+1) + m(L)$ breakpoints. Since $m(L)$ is an integer which is strictly smaller than $(W+1)^L$ we have that

$$m(L+1) \leq W(m(L)+1)+m(L) = m(L)(W+1)+W \leq \left[(W+1)^L - 1\right](W+1)+W = (W+1)^{L+1}-1 < (W+1)^{L+1}$$

□

Summary To summarize our discussion of univariate ReLU networks: we saw that shallow neural networks are more or less the same as free knot linear splines (Proposition 1.28). Deep networks can also express all free knot splines with $n(W, L)$ parameter (Theorem 1.30), however, they can also express some functions, such as 'sawtooth functions', with a number of breakpoints which is exponential in L (but not all such functions, see proof in [Dym et al., 2020]). In general, univariate networks with width W and depth L will have less that $(W+1)^L$ breakpoints.

The attractiveness of the latter results as that they point to an expressive advantage of deep neural networks over a 'standard function base'-free knot linear splines. It also points to an advantage over shallow neural networks- a network with depth one, or with fixed depth, would need its width to grow exponentially with s to express the function $H^{\circ s}$, rather than the linear growth we have observed. This type of results are sometimes called 'depth separation' results. There are many similar arguments in various papers, see e.g., [Telgarsky, 2016, Eldan and Shamir, 2016].

Possible criticisms of these arguments are that the functions for which deep neural networks have an advantage over shallow ones don't necessarily seem to be the functions we are interested in learning. Another argument which has been made is that highly oscillating functions such as the sawtooth function are difficult to learn using gradient descent (see e.g., [Malach and Shalev-Shwartz, 2019, Hanin and Rolnick, 2019]). One could also wonder to what extent the univariate case discussed here is representative of the multivariate case. We will discuss this next.

1.4.6 Linear regions of multivariate neural networks

Up to now we have focused on counting linear region of univariate neural networks, that is, functions in

$$\mathcal{FC}\left(W, L=1; w_{in}=d, w_{out}=1\right)$$

with $d=1$. We now want to focus on the case where $d>1$.

In terms of counting linear regions, the main difference between multivariate and univariate networks is that *shallow* multivariate networks can also have significantly more linear regions than parameters. In fact, we are already familiar with such an example:

Example 1.34. Recall that the function $x \mapsto ReLU(x)$ which is in $\mathcal{FC}(W=d, L=1, n_{in}=d, n_{out}=d)$, has 2^d linear regions.

The ReLU function is a linear function on each of the 2^d quadrants of $\mathbb{R}^d$. Formally, for every vector $s \in \{-1,1\}^d$ we can define the polytope

$$Q_s = \{x \in \mathbb{R}^d |\, s_i x_i \geq 0, \forall i = 1, \ldots, d\}$$

The collection of all such polytopes form a polytope partition, and

$$ReLU(x) = \frac{1}{2}\begin{pmatrix} 1+s_1 & 0 & 0 & \ldots & 0 \\ 0 & 1+s_2 & 0 & \ldots & 0 \\ 0 & 0 & 1+s_3 & \ldots & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & 1+s_d \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_d \end{pmatrix}, \forall x \in Q_s. \tag{1.10}$$

In the homework we saw that, if the restriction of a CPwL function f to each partition element is a different affine function, then there is no smaller polytope partition which f is subordinate to. In our context, since the restriction to each of the 2^d polytopes Q_s is a different affine function, the number of linear regions of $ReLU$ is exactly 2^d.

The number of linear regions of shallow univariate networks is strongly related to the hyperplane arrangement problem. Let us describe this problem:

Hyperplane arrangements Recall that a hyperplane in $\mathbb{R}^d$ is a set of the form

$$H = \{x \in \mathbb{R}^d | \langle a, x\rangle - b = 0\}$$

where $a \in \mathbb{R}^d$ is non-zero and $b \in \mathbb{R}$. A hyper-plane arrangement is a collection of m hyperplanes H_i in $\mathbb{R}^d$, defined by pairs $(a_i, b_i) \in \mathbb{R}^d \setminus \{0\} \times \mathbb{R}$. Now consider the open set

$$U = \mathbb{R}^d \setminus \bigcup_{i=1}^m H_i$$

We'd like to ask the following question: how many connected components does U have? Note that U is a union of a finite number of convex open sets U_s with empty intersection: For every $s \in \{-1,1\}^m$ define

$$U_s = \{x \in \mathbb{R}^d | s_i \left(\langle a_i, x\rangle - b_i\right) > 0, \forall i = 1, \ldots, m\}$$

all in all there are 2^m possible choices of s, but some of the sets U_s may be empty. The number of linear regions is the number of U_s which are not empty.

Example When $d=1$, consider the hyperplanes $H_i = \{x \in \mathbb{R} | x = i\}$ for $i = 1,2,3,4$. The set U has 5 connected components. In general we see that given m hyperplanes in $\mathbb{R}$, the complement will have $m+1$ connected component, unless some of the hyperplanes are equal in which case we will have less hyperplanes.

Problem 1.35. How many connected components does the complement of a hyper-plane arrangement with m hyperplanes in $\mathbb{R}^d$ have when

1. $d = 2$ and $m = 2$.
2. $d = 2$ and $m = 3$.

Answer Could always be as little as two if all hyperplanes are taken to be the same. When $d = 2, m = 2$ the maximal number is four, while when $m = 3$ the maximal number is seven, as we shall soon see.

In general, the number of connected components in a hyperplane arrangement are given by the following theorem (which we will not prove):

Theorem 1.36 ([Zaslavsky, 1975])**.** *A hyperplane arrangement with m hyperplanes in $\mathbb{R}^d$ has at most*

$$\nu(d,m) := \sum_{j=0}^{d} \binom{m}{j} \tag{1.11}$$

linear regions. Additionally, there are m hyperplanes in $\mathbb{R}^d$ with exactly $\nu(d,m)$ linear regions (in fact, this happens for Lebesgue almost every $(a_i, b_i), i = 1, \ldots, m$.)

Remark 1.37. When $d > m$, the summation in (1.11) includes expressions $\binom{m}{j}$ with $j > m$. This number is zero by definition.

Let us consider some examples: when $d = 1$ we saw previously that m hyperplanes will have

$$\nu(1,m) = \binom{m}{0} + \binom{m}{1} = m + 1$$

connected components, unless some hyperplanes are identical in which case there will be less connected components. When $d = 2$ and $m = 3$, the maximal number of connected components will be

$$\nu(2, m = 3) = \binom{m}{0} + \binom{m}{1} + \binom{m}{2} = 1 + 3 + 3 = 7,$$

while when $m = 2, d = 2$ there will be only four. Finally, when $m = d$, there can be at most

$$\nu(d, m = d) = \sum_{j=0}^{d} \binom{m}{j} = \sum_{j=0}^{d} \binom{d}{j} = 2^d$$

connected components. For example, the d hyperplanes $\{x \in \mathbb{R}^d | x_i = 0\}$ divide $\mathbb{R}^d$ into 2^d connected components.

The relationship between the number of linear regions in a hyperplane arrangement and the number of linear regions of a shallow ReLU network are given by the following proposition:

Proposition 1.38. *If $f \in \mathcal{FC}(W, L = 1, n_{in} = d, n_{out})$, then f has at most*

$$\nu(d, W) = \sum_{j=0}^{d} \binom{W}{j}$$

linear regions.

Note that the number of linear regions does not depend on n_{out} at all.

Proof. We can write $f = h_2 \circ \rho \circ h_1$. Denoting $h_1(x) = Ax + b$, and denoting the rows of A by $a_1, \ldots, a_W$, we have that

$$h_2 \circ \rho(Ax + b) = h_2 \circ \rho \begin{pmatrix} \langle a_1, x\rangle + b_1 \\ \vdots \\ \langle a_W, x\rangle + b_W \end{pmatrix}$$

Denoting $d = n_{in}$, we can define a polytope covering of $\mathbb{R}^d$, indexed by $s \in \{-1, 1\}^W$ by

$$P_s = \{x \in \mathbb{R}^d | s_i \left(\langle a_i, x\rangle + b_i\right) \geq 0, i = 1, \ldots, W\}$$

and the restriction of f to each P_s is a linear function. We can remove from the partition $\{P_s\}_{s\in\{-1,1\}^W}$ all polytopes P_s whose interior is empty, and still remain with a partition. The number of linear regions in this partition is exactly the number of connected components of $\mathbb{R}^d \setminus \cup_i \{x \in \mathbb{R}^d | \, \langle a_i, x\rangle + b_i = 0\}$. The result then follows from Theorem 1.36. $\square$

We've seen that for shallow networks, the number of linear regions can grow exponentially with the input dimension. However, it is still true that you can get much more linear regions with deep networks. For example, while Proposition 1.38 implies that $f \in \mathcal{FC}(W, L, n_{in}, n_{out})$ will have no more than $W^{n_{in}} + 1$ linear regions when $L = 1$ (you will verify this for homework), when we allow $L > 1$ there exist $f \in \mathcal{FC}(W, L, n_{in}, n_{out})$ with $\sim \left[\frac{W}{n_{in}}\right]^{(L-1)n_{in}} W^{n_{in}}$ linear regions (see [Montufar et al., 2014] for more details).

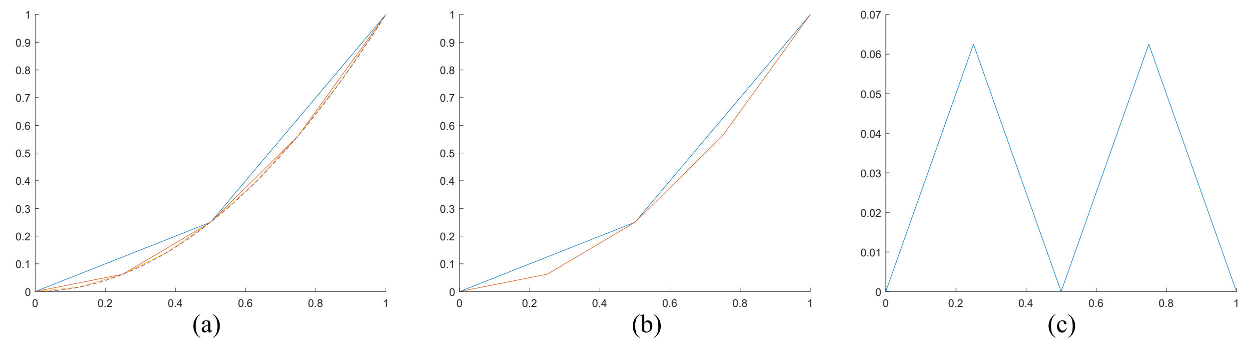

Figure 1.4: (a) Swift convergence of the approximations $f_m(x)$ of x^2 (b) the functions f_1 and f_2 and (c) the function $f_1 - f_2$ is a scaling of the sawtooth function $H \circ H$.

1.5 Approximating polynomials and smooth functions

We saw that functions like the sawtooth function can be expressed with deep neural networks significantly more efficiently than with shallow networks. We will now show how this can be leveraged to obtain similar results for approximation of the function x^2. We will then build upon this further to approximate monomials, polynomials, and smooth functions in general. This discussion is taken from [Yarotsky, 2017].

The sawtooth function will actually turn out to be a useful tool in our discussion. Recall that the sawtooth function $H^{\circ s}$ can be implemented with width 2 and depth s despite having $2^s - 1$ breakpoints. It is linear on the intervals $[k_i, k_{i+1}]$ where

$$k_0 = 0, k_1 = \frac{1}{2^s}, k_2 = \frac{2}{2^s} \ldots, k_{2^s} = 1$$

and $H^{\circ s}(k_\ell)$ is 0 if ℓ is even and one if ℓ is odd.

1.5.1 Approximating x^2

Let us consider the problem of approximating x^2 with a ReLU network. In Theorem 1.11 we showed that we can approximate a function $f : [0,1]^d \to \mathbb{R}$ which is ℓ-Lipschitz to ϵ accuracy (uniformly on $[0,1]^d$) by a function in $\mathcal{FC}(W, L, d, 1)$ where

$$W \leq 4d \left(\frac{\sqrt{d}\ell}{2\epsilon} + 1 \right)^d, L \leq \lceil \log_2(d) \rceil + \lceil d \cdot \log_2(\left(\frac{\sqrt{d}\ell}{2\epsilon} + 1 \right)) \rceil$$

In particular, if we want to approximate x^2 on $[0,1]$, we note that x^2 is Lipschitz with Lipschitz constant $\ell = 2$, and we can obtain an ϵ approximation with $\sim 1/\epsilon$ width and $\sim \log(1/\epsilon)$ depth. We now will see that we can do much better than this. As a first step, let us consider approximations with shallow networks, or equivalently, with functions in Σ_N.

Proposition 1.39. *Let g_N be the unique CPwL function which satisfies $g_N(k/N) = (k/N)^2$ for all $k = 0, \ldots, N$, and is affine on the intervals $[\frac{k-1}{N}, \frac{k}{N}], k = 1, \ldots N$. Then*

$$\max_{x \in [0,1]} |x^2 - g_N(x)| \leq \left(\frac{1}{2N} \right)^2$$

Proof. We will use the following lemma

Lemma 1.40. *Let $g(x) = g(x|a, b)$ be the linear interpolation of x^2 at a, b, that is*

$$g(x) = \frac{1}{b-a} \left[(b-x)a^2 + (x-a)b^2 \right]$$

then

$$\max_{x \in [a,b]} |g(x) - x^2| = g\left(\frac{a+b}{2} \right) - \left(\frac{a+b}{2} \right)^2 = \left(\frac{b-a}{2} \right)^2$$

Proof of the Lemma. The continuous function $q(x) = g(x) - x^2 = g(x) - x^2$ is maximized and minimize at a or b or at a point $x \in (a, b)$ with $q'(x) = 0$. We have that $q(a) = 0 = q(b)$. Solving the equation

$$0 = q'(x) = \frac{b^2 - a^2}{b - a} - 2x$$

we obtain that the derivative is zero at the midpoint $\frac{a+b}{2}$. At this point we have

$$q\left(\frac{a+b}{2}\right) = \frac{a^2 + b^2}{2} - \left(\frac{a+b}{2}\right)^2 = \frac{1}{4}\left(2a^2 + 2b^2 - a^2 - b^2 - 2ab\right) = \frac{1}{4}(b-a)^2$$

So the minimum of q is obtained at a and b, and the minimal value of q is zero, while the maximum is obtained at the center and the maximal values is $\frac{1}{4}(b-a)^2$. This concludes the proof of the lemma. $\square$

Returning to our definition of g_N we see that

$$\begin{aligned}\max_{x\in[0,1]} |f(x) - g_N(x)| &= \max_{k=0,\ldots,N-1} \max_{x\in[\frac{k}{N},\frac{k+1}{N}]} |f(x) - g(x|k/N, (k+1)/N)\\ &= \max_{k=0,\ldots,N-1} \left(\frac{1}{2N}\right)^2 = \left(\frac{1}{2N}\right)^2\end{aligned}$$

This concludes the proof of the proposition $\square$

We see that with N knots we can get an approximation of $\left(\frac{1}{2N}\right)^2$. Stated differently, to achieve ϵ accuracy we need to take N such that $\left(\frac{1}{2N}\right)^2 < \epsilon$ so $N > (1/(4\epsilon))^{1/2}$. Thus we can achieve the same approximation rate with a univariate shallow network with width $W = N + 1$ and depth $L = 1$ by Proposition 1.28. Thus we already get a better approximation rate than what we had from our standard Lipschitz bound ($\epsilon^{1/2}$ width and parameters vs. $1/\epsilon$ width and $(1/\epsilon)^2 \log(1/\epsilon)$ parameters.).

We will now see that we can do much better with deep networks: denote $f_m = g_{2^m}$: that is, f_m is the CPwL function with knots in $0, \frac{1}{2^m}, \frac{2}{2^m}, \ldots, 1$ and whose value at these points is

$$f_m\left(\frac{k}{2^m}\right) = \left(\frac{k}{2^m}\right)^2, \quad k = 0, 1, \ldots, 2^m.$$

By our previous discussion we have that

$$\max_{x\in[0,1]} |f_m(x) - f(x)| \le \frac{1}{2^{2m+2}}.$$

This is a great approximation rate but of course the function f_m have 2^m knots and so expensive to compute using free knot splines or shallow networks. However, this can be done much more efficiently with deep networks, using the sawtooth function we studied earlier:

Lemma 1.41. *Let f_m denote the Piecewise linear interpolation of x^2 at the points $k2^{-m}, k = 0, \ldots, 2^m$ as defined above, and let H denote the sawtooth function, then*

$$f_{m-1}(x) - f_m(x) = \frac{H^{\circ m}(x)}{2^{2m}}, \quad \forall x \in [0, 1]$$

Proof. Let us consider the difference $f_{m-1}(x) - f_m(x)$. This function has knots at the points $\frac{k}{2^m}$. The knots where k is even are also knots of f_{m-1} and therefore at these knots $f_{m-1}(\frac{k}{2^m}) = \left(\frac{k}{2^m}\right)^2 = f_m(\frac{k}{2^m})$, and the difference between the functions is zero at these points. When k is odd, we can repeat the computation we did in Lemma 1.40: Let us denote $a = \frac{k-1}{2^m}$ and $b = \frac{k+1}{2^m}$, then $\frac{k}{2^m} = \frac{1}{2}(a + b)$ and

$$f_{m-1}\left(\frac{k}{2^m}\right) - f_m\left(\frac{k}{2^m}\right) = \frac{1}{2}(a^2 + b^2) - \left(\frac{1}{2}(a+b)\right)^2 = \frac{1}{4}(b-a)^2 = \frac{1}{2^{2m}}$$

and so altogether we have

$$f_{m-1}\left(\frac{k}{2^m}\right) - f_m\left(\frac{k}{2^m}\right) = \begin{cases} 0 & \text{if } k \text{ is even} \\ \frac{1}{2^{2m}} & \text{if } k \text{ is odd} \end{cases}$$

This is exactly the sawtooth function $H^{\circ m}$ up to scaling, that is

$$f_{m-1}(x) - f_m(x) = \frac{H^{\circ m}(x)}{2^{2m}}. \tag{1.12}$$

An illustration of this result is shown in Figure 1.4. □

Applying (1.12) recursively, we obtain (note that $f_0(x) = x$)

$$f_m(x) = f_0(x) + (f_1(x) - f_0(x)) + \ldots + (f_m(x) - f_{m-1}(x)) = x - \sum_{s=1}^{m} \frac{H^{\circ s}(x)}{2^{2s}}$$

We can summarize our results as follows

Theorem 1.42. *There exists some natural numbers W_0, L_0, such that the function $f(x) = x^2$ can be approximated to ϵ accuracy in $[0,1]$ with a NNF in $\Upsilon(W_0, L_0 \log(1/\epsilon))$*

Remark Recall that a network with width W and depth L has $\sim W^2 L$ parameters, so the total number of parameters is linear in $\log(1/\epsilon)$.

Proof. We saw that $f_m(x)$ approximate x^2 with an error rate of $\frac{1}{2^{2m+2}}$, and we can reorder (1.12) to obtain $f_s = f_{s-1} - \frac{H^{\circ s}}{2^{2s}}$, and so we can recursively define

$$x \mapsto \begin{bmatrix} H(x) \\ f_0(x) \end{bmatrix} \overset{F_2}{\mapsto} \begin{bmatrix} H^{\circ 2}(x) \\ f_1(x) \end{bmatrix} \overset{F_3}{\mapsto} \ldots \overset{F_{m-1}}{\mapsto} \begin{bmatrix} H^{\circ m-1}(x) \\ f_{m-1}(x) \end{bmatrix} \mapsto f_m(x), \text{ where } F_s \begin{bmatrix} y \\ z \end{bmatrix} = \begin{bmatrix} H(y) \\ z - y/2^{2s} \end{bmatrix}$$

which shows that f_m can be constructed with a neural network with fixed width W_0 and depth mL_0. To obtain ϵ accuracy we need to have

$$2^{-2m-2} < \epsilon$$

or equivalently

$$m > 1/2 \log_2\left(\frac{1}{\epsilon}\right) - 1$$

so we can choose $m = 1/2 \log_2(1/\epsilon)$ to get width and depth as required by the theorem. □

New Lesson

1.5.2 Approximating multiplication

An important consequence of the previous result is that the product function $(x, y) \mapsto xy$ can also be approximated by neural networks with exponential accuracy: in general, this can be done by noting

$$xy = \frac{1}{2}\left((x+y)^2 - x^2 - y^2\right)$$

and then using our approximation of the squaring function. Recall that our approximation of x^2 was obtained on $[0, 1]$. We will want to obtain approximation for xy on $[-M, M]^2$. To do this note that we can rewrite the previous equation as

$$xy = \frac{M^2}{8}\left(\left(\frac{|x+y|}{2M}\right)^2 - \left(\frac{|x|}{2M}\right)^2 - \left(\frac{|y|}{2M}\right)^2\right). \tag{1.13}$$

The advantage of this is that we are now applying the squaring function to inputs in the interval $[0, 1]$.

Proposition 1.43. *There exists $(W_1, L_1) \in \mathbb{N}^2$, such that for all $M > 0$ and $\epsilon > 0$, there is an NNF $NN_\times : \mathbb{R}^2 \to \mathbb{R}$ satisfying*

1. *for all x, y with $|x|, |y| < M$ we have $|NN_\times(x, y) - xy| < \epsilon$.*
2. *if $x = 0$ or $y = 0$ then $NN_\times(x, y) = 0$.*
3. *$NN_\times$ is in $\mathcal{FC}(W_1, L_1(1 + \log(1/\epsilon) + \log(M)), 2, 1)$.*

Proof. Choose some $\epsilon > 0$ and $M > 0$. Recall that there exists $(W_0, L_0) \in \mathbb{N}^2$ such that for all $\tilde{\epsilon} > 0$ there exists $s \in \mathcal{FC}(W_0, L_0 \log(1/\tilde{\epsilon}), 1, 1)$ such that $|s(x) - x^2| < \tilde{\epsilon}$ for all $x \in [0, 1]$. Now set

$$\tilde{\epsilon} = \frac{8\epsilon}{3M^2}.$$

and let s be the function in $\mathcal{FC}(W_0, L_0 \log(1/\tilde{\epsilon}), 1, 1)$ such that $|s(x) - x^2| < \tilde{\epsilon}$. We replace the squares in (1.13) with s to obtain the function

$$NN_\times(x, y) := \frac{M^2}{8}\left(s\left(\frac{|x+y|}{2M}\right) - s\left(\frac{|x|}{2M}\right) - s\left(\frac{|y|}{2M}\right)\right)$$

We can verify directly that this function satisfied (1) due to (1.13), satisfies (2) since $s(0) = 0$, and satisfies (3) as $NN_\times$ is in $\mathcal{FC}(W_1, L_1(1 + \log(1/\epsilon) + \log(M)), 2, 1)$ and

$$|z| = \rho(z) + \rho(-z)$$

is a network of depth 1 and width 2. □

We can think of this result as follows: up to now we knew that if f, g were NNFs then we can take linear combinations of them and get another NNF. We can also concatenate or take compositions and get an NNF. This result means that the same is almost true for the product $f \cdot g$. While we cannot exactly express $f \cdot g$ as an NNF, we can approximate it extremely well.

We now prove a similar lemma for the multi-dimensional product function

Proposition 1.44. *For every $s \in \mathbb{N}$ there exist $W_2 = W_2(s), L_2 = L_2(s)$ such that for every $1 > \epsilon > 0$, there exists a function $m \in \mathcal{FC}(W_2, L_2(1 + \log(1/\epsilon)), d_{in} = s, 1)$ with*

$$|x_1 \cdot x_2 \cdot \ldots \cdot x_s - m(x)| < \epsilon, \quad \forall x = (x_1, \ldots, x_s) \in [0, 1]^s$$

Proof. For given $\epsilon > 0$, let $NN_\times$ be the function promised by Proposition 1.13 with respect to the parameters $\tilde{\epsilon} = \epsilon/(s-1)$ and $M = 2$. Define recursively the functions

$$m_1(x) = x_1, \quad m_2(x) = NN_\times(m_1(x), x_2), \quad p_3(x) = NN_\times(m_2(x), x_3) \quad \ldots, \quad m_s(x) = NN_\times(m_{s-1}(x), x_s).$$

We can think of each m_k as an approximation of the product of the first k entries of x and $m := m_s$ is the final approximation we need. Note that we can apply Proposition 1.43 recursively to see that:

Since for all $x \in [0,1]^s$ we have that $|m_2(x) - x_2 \cdot m_1(x)| < \tilde{\epsilon}$ and both x_2 and $m_1(x) = x_1$ are in $[0,1]$, we deduce that $|m_2(x)| < 1 + \tilde{\epsilon} < 1 + 1/(s-1) < 2$.

Since $|m_2(x)| < 2$ we know that for all $x \in [0,1]^s$ we have $|m_3(x) - x_3(x)m_2(x)| < \tilde{\epsilon}$, and therefore $|m_3(x)| < 1 + 2/(s-1) < 2$

continuing recursively with this process we see that for all $j = 1, \ldots, s-1$,

$$|m_{s+1}(x) - x_{s+1}m_s(x)| < \tilde{\epsilon}.$$

Using this we can see that the error of the approximation $m = m_s$ is bounded for all $x \in [0,1]^s$ by

$$\begin{aligned} |x_1 \cdot \ldots \cdot x_s - m_s(x) &\leq |m_s(x) - x_s m_{s-1}(x)| + |x_s m_{s-1}(x) - x_s x_{s-1} m_{s-2}(x)| + \ldots \\ &+ |x_s x_{s-1} \ldots x_3 m_2(x) - x_s x_{s-1} \ldots x_2 m_1(x)| \\ &\leq \sum_{i=1}^{s-1} |m_{i+1}(x) - x_{i+1} m_i(x)| \\ &\leq (s-1)\tilde{\epsilon} = \epsilon \end{aligned}$$

A simple way to construct m_s as a neural networks is through the architecture

$$x \mapsto \begin{pmatrix} m_1(x) = x_1 \\ x_2 \\ \vdots \\ x_s \end{pmatrix} \mapsto \begin{pmatrix} m_2(x) = NN_\times(m_1(x), x_2) \\ x_3 \\ \vdots \\ x_s \end{pmatrix} \mapsto \begin{pmatrix} m_3(x) = NN_\times(m_2(x), x_3) \\ x_4 \\ \vdots \\ x_s \end{pmatrix} \ldots \mapsto m_s(x)$$

The width of this construction does not depend on ϵ and the depth is proportional to $s\log(1/\tilde{\epsilon}) = s\log(s-1) + s\log(1/\epsilon)$. □

1.5.3 Approximation of univariate smooth functions

We now consider approximation of functions from the function space

$$\mathcal{F}_n = \{f : [0,1] \to \mathbb{R} | \, |f^{(k)}(x)| \leq 1, k = 0, \ldots, n \text{ and } x \in [0,1]\}.$$

We will prove the following

Theorem 1.45. *Let $n \in \mathbb{N}$. There exist $W = W(n), L = L(n)$ such that for every $\epsilon \in (0,1)$ and every $f \in \mathcal{F}_n$, there exists $\tilde{f} \in \mathcal{FC}(W, L(1 + (1/\epsilon)^{1/n} \log(1/\epsilon)), d_{in} = 1, d_{out} = 1)$ with*

$$|f(x) - \tilde{f}(x)| < \epsilon, \forall x \in [0,1]$$

Remark 1.46. Note that this result can be used, in a sense, for any n times continuously differentiable f: since the derivatives are continuous they are bounded on a bounded interval, and so there will be some M for which

$$|f^{(k)}(x)| \leq M, \forall k = 0, \ldots, n \text{ and } x \in [0,1].$$

We then have that $\frac{1}{M}f$ is in $\mathcal{F}_n$. If we use the theorem to obtain a neural network $\tilde{f}$ which approximates $\frac{1}{M}f$ to within an error of $\frac{\epsilon}{M}$, then $M\tilde{f}$ will approximate f to within an error of ϵ.

The idea of this proof is as following: we first consider approximation of f by a function space of 'piecewise polynomials', and then use what we showed so far to efficiently approximate these piecewise polynomials.

Let us first define our piecewise polynomials. Assume that for some given N, we are given an N-partition of unity: this term we will use to mean a collection of functions $\phi_0, \ldots, \phi_N : [0,1] \to \mathbb{R}$ which are non-negative and satisfy

$$\sum_{j=0}^{N} \phi_j(x) = 1, \forall x \in [0,1] \text{ and } \phi_j(x) = 0 \text{ if } |x - j/N| < 1/N$$

One example of such a partition which is useful for us are the CPwL functions ϕ_j defined by the requirement that they are linear on the intervals $[k/N, (k+1)/N]$ for all $k = 0, \ldots, N-1$, and satisfy

$$\phi_j(k/N) = \begin{cases} 1 & \text{if } k = j \\ 0 & \text{if } k \neq j \end{cases}$$

clearly these functions are non-negative, are zero where they should be, and since $\sum_{j=0}^{N} \phi_j(x) = 1$ for $x = 0, 1/N/2/N, \ldots, 1$, and is affine between these points, we see that this equality holds for all $x \in [0,1]$.

Proposition 1.47. *For any $n \in \mathbb{N}$, there exists some $C = C(n)$ such that for every $\epsilon > 0$ and every $f \in \mathcal{F}_n$ there exists $N \leq 1 + C(n)\epsilon^{-1/n}$ polynomials $p_0, p_1, \ldots, p_N$ of degree $\leq n$ such that*

$$|f(x) - \sum_{j=0}^{N} \phi_j(x) p_j(x)| < \epsilon, \quad \forall x \in [0,1]$$

Proof. Assume we are given n, given $f \in \mathcal{F}_n$ and given $\epsilon > 0$. Assume we pick some N. For given $j = 0, \ldots, N$, we define p_j to be the Taylor expansion of order $n-1$ of f around j/N. Then we know that for all $x \in [0,1]$ with $|x - j/N| \leq 1/N$, there exists some $c \in [0,1]$ such that

$$|f(x) - p_j(x)| \leq \left| \frac{1}{n!}(x - j/N)^n f^{(n)}(c) \right| \leq \frac{1}{n!}(1/N)^n$$

We want the expression on the right hand side to be smaller than ϵ, which equivalently means

$$N \geq \left(\frac{1}{\epsilon n!} \right)^{1/n}.$$

For such a value of N we get for all $x \in [0,1]$ that

$$\begin{aligned} |f(x) - \sum_{j=0}^{N} \phi_j(x) p_j(x)| &= |\sum_{j=0}^{N} \phi_j(x)(f(x) - p_j(x))| \\ &\leq \sum_{j=0}^{N} \phi_j(x) |f(x) - p_j(x)| \leq \epsilon \end{aligned}$$

□

We see that functions in F_n can be approximated to ϵ accuracy with $N \sim (1/\epsilon)^{1/n}$ polynomials of degree $\leq n-1$. All in all this function space is spanned by $nN \sim C(n)(1/\epsilon)^{1/n}$ parameters.

We now turn to prove Theorem 1.45, where we see that we can do almost as well with neural networks. The idea behind this claim is as follows: the construction we just saw is based on products of functions we can easily build with neural networks: they are a product of hat functions with polynomials, which themselves are linear combinations of products of linear functions. Since we known how to swiftly approximate products, we can successfully 'imitate' the approximation we just saw with neural networks.

Proof of Theorem 1.45. For given $\epsilon > 0$, let $N < C2^{1/n}(1/\epsilon)^{1/n}$ be such that the piecewise polynomials $P(x) := \sum_{j=0}^{N} \phi_j(x)p_j(x)$ defined in Proposition 1.47 are an $\epsilon/2$ approximation of f. We can rewrite P as

$$P(x) = \sum_{j=0}^{N}\sum_{k=0}^{n-1} a_{j,k} \underbrace{\phi_j(x)(x-j/N)^k}_{P_{j,k}(x)}$$

and note that since we took p_j to be the Taylor expansion of f at j/N and the derivatives of f are bounded by 1, we have that $|a_{j,k}| \le 1$. We will show that we can approximate each $P_{j,k}$ with a neural network $h_{j,k}$ up to accuracy of $\tilde{\epsilon} = \frac{\epsilon}{2n(N+1)}$. For fixed j,k, we know from Proposition 1.44 that there is some $m_{k+1} \in \mathcal{FC}(W_2, L_2(1+\log(1/\tilde{\epsilon})+\log(k+1)), d_{in} = k+1, 1)$ which can approximate the function $x_1x_2\ldots x_{k+1}$ to accuracy of $\tilde{\epsilon}$. We thus have that

$$|P_{j,k}(x) - \underbrace{m_{k+1}\left(\phi_j(x), x-\frac{j}{N}, x-\frac{j}{N}, \ldots, x-\frac{j}{N}\right)}_{h_{j,k}}| \le \tilde{\epsilon}$$

and $h_{j,k}$ can be written as

$$x \mapsto \begin{pmatrix} \phi_j(x) \\ (x-j/N) \\ \vdots \\ (x-j/N) \end{pmatrix} \mapsto m_{k+1}\left(\phi_j(x), x-\frac{j}{N}, x-\frac{j}{N}, \ldots, x-\frac{j}{N}\right) = h_{j,k}(x)$$

which is a neural network with width independent of $\tilde{\epsilon}$ and depth logarithmic in $1/\tilde{\epsilon}$ and hence in $1/\epsilon$. We can the approximate f by $h = \sum_{j=0}^{N}\sum_{k=0}^{n-1} a_{j,k}h_{j,k}(x)$ with error

$$\begin{aligned}|f(x)-h(x)| \le |f(x)-P(x)| + |P(x)-h(x)| &\le \epsilon/2 + \sum_{j=0}^{N}\sum_{k=0}^{n-1} |a_{j,k}||P_{j,k}(x)-h_{j,k}(x)| \\ &\le \epsilon/2 + \sum_{j=0}^{N}\sum_{k=0}^{n-1} \tilde{\epsilon} = \epsilon.\end{aligned}$$

The network h is a sum of $(N+1)n \sim (1/\epsilon)^{1/n}$ networks $h_{j,k}$ with fixed width W and depth proportional to $\log(1/\epsilon)$. In the homework you will show that this addition can be done with fixed width and depth which grows like $(N+1)n$. This concludes the claim. □

1.5.4 Additional results

Approximation of multivariate smooth functions The paper [Yarotsky, 2017] focuses on the more general case of multivariate smooth functions. The function class $\mathcal{F}_{d,n}$ it discusses is functions $f : [0,1]^d \to \mathbb{R}$ which have continuous partial derivatives of order $\le n$ all bounded by one. The construction used in this case the story is similar to the 1D case: the function f is approximated by expression $\sum_{j=1}^{N} \phi_j(x)p_j(x)$ where p_j are multivariate polynomials of degree $\le n-1$ and ϕ_j is a partition of unity supported on sets with small diameter δ. The number of parameters in this construction is $\sim (1/\epsilon)^{d/n}$, that is, it increases exponentially with the dimension and decreases exponentially with the smoothness, a known phenomena in approximation theory. Thus the 'curse of dimensionality' can be countered by smoothness. Neural network approximation in the multidimensional case again performs similarly to the piecewise polynomials up to a logarithmic factor $\sim (1/\epsilon)^{d/n}(1+\log(1/\epsilon))$.

'Super approximation' of univariate functions For the function space $\mathcal{F}_{1,1} = \mathcal{F}_1$ the theorem we saw guarantees ϵ approximation with $\sim (1/\epsilon)$ parameters. However [Yarotsky, 2017] showed that this can be improved to $\sim (1/(\epsilon\log(1/\epsilon))$ and this was later improved even further to $\sim 1/\epsilon^2$ (see [DeVore et al., 2021]. Theorem 8.11). More generally, in [Lu et al., 2021] it is shown that approximation of functions on a d-

dimensional cube $[0, 1]^d$ with s derivatives can be obtained with $\sim (1/\epsilon)^{\frac{d}{2s}}$ (up to logarithmic factors). We stress that unlike the previous results where neural networks achieved slightly worse rates than piecewise polynomials, these rates are actually better than what piecewise polynomials and other standard function bases can do...

Chapter 2

Invariant Learning and Approximation Theory

2.0.1 Preliminaries

Definition 2.1 (Group)**.** A pair $(G, \cdot)$, where G is a set and $\cdot : G \times G \to G$ is called a group if

1. Associativity For all $a, b, c \in G$,
$$(a \cdot b) \cdot c = a \cdot (b \cdot c)$$
.

2. Identity There exists an element $e \in G$ such that
$$a \cdot e = a = e \cdot a, \forall a \in G.$$

3. Inverse element For every $a \in G$ there exists some element $b \in G$ such that
$$a \cdot b = e = b \cdot a.$$
We denote b by a^{-1}.

Examples: The pair $(\mathbb{R}^{n\times n}, +)$ is a group whose identity element is the zero matrix. The pair $(\mathbb{R}^{n\times n}, \cdot)$ is not a group (where $\cdot$ denotes matrix multiplication). The pair $(GL(n)), \cdot)$ is a group whose identity element is the identity matrix I_n, where
$$GL(n) = \{A \in \mathbb{R}^{n\times n} | \, det(A) \neq 0\}$$
A subgroup of $(G, \cdot)$ is a group $(H \cdot)$ with the same operation $\cdot$, and where H is a subset of G. Subgroups of $GL(n)$ include

$$\begin{aligned} O(n) &= \{R \in \mathbb{R}^{n\times n} | \, RR^T = I_n\} \\ SO(n) &= \{R \in \mathbb{R}^{n\times n} | \, RR^T = I_n, \, det(R) = 1\} \\ SL(n) &= \{A \in \mathbb{R}^{n\times n} | \, det(A) = 1\} \end{aligned}$$

The set of bijections from $[n] := \{1, \ldots, n\}$ to $[n]$ is denoted by S_n. The bijections are called permutations. They are a group with respect to composition.

New Lesson

Remark 2.2. In the following we will use the convention that a vector $v \in \mathbb{R}^n$ is actually a degenerate matrix $v \in \mathbb{R}^{n\times 1}$ (a 'column vector'). This allows expressing all vector operations in terms of matrix multiplication. For example the inner product of two vectors $v, u \in \mathbb{R}^n$ is given by

$$\langle v, u\rangle = v^T u.$$

The matrix $v^T \in \mathbb{R}^{1\times n}$ is called a 'row vector'.

We will also use the notation

$$1_n = (1, \dots, 1)^T \text{ and } 1_{n\times n} = 1_n 1_n^T$$

for the n by 1 and n by n all-one matrices.

Definition 2.3 (Group Action)**.** Let $(G, \cdot)$ be a group with an identity element e, and let S be a set. We say that $\rho : G \times S \to S$ is a group action if

1. For all $s \in S$,
$$\rho(e, s) = s.$$
2. for all $g, h \in G$ and $s \in S$ we have
$$\rho(g, \rho(h, s)) = \rho(g \cdot h, s)$$

Remark 2.4. We will use the notation

$$g.s = \rho(g, s).$$

With this notation the requirements of a group action can be written more conveniently as

$$e.s = s \text{ and } g.(h.s) = (g \cdot h).s$$

Example: A natural attempt to define a group action of S_n on $\mathbb{R}^n$ would be

$$[\sigma.x]_i = x_{\sigma(i)}, \sigma \in S_n, x \in \mathbb{R}^n, i = 1, \dots, n.$$

However as it turns out this is not a group action in our definition because

$$[\tau.(\sigma.x)]_i = [\sigma.x]_{\tau(i)} = x_{\sigma\circ\tau(i)}$$

which generally may not be equal to

$$[(\tau \circ \sigma).x]_i = x_{\tau\circ\sigma(i)}$$

The correct way to define a group action of S_n on $\mathbb{R}^n$ is via the group action

$$[\sigma.x]_i = x_{\sigma^{-1}(i)}, \forall \sigma \in S_n, x \in \mathbb{R}^n, i = 1, \dots, n$$

Note that

$$[\sigma.(\tau.x)]_i = [\tau.x]_{\sigma^{-1}](i)} = x_{\tau^{-1}\circ\sigma^{-1}(i)} = x_{(\sigma\circ\tau)^{-1}(i)} = [(\sigma \circ \tau).x]_i$$

Example Similarly, we can define an action of S_n on $\mathbb{R}^{d\times n}$ by

$$[\sigma(X)]_{ij} = X_{i\sigma^{-1}(j)}$$

Definition 2.5. Let G be a group acting on a set S. The orbit of $s \in S$ is denoted by $[s]$ and is defined as the set of all element in S related to s by a group transformation, that is

$$[s] = \{g.s |\, g \in G\}.$$

We denote $s \sim s'$ if $[s] = [s']$. One can verify that this is an equivalence relation. The collection of all orbits is denoted by S/G.

Example 2.6. The group $G = \{-1, 1\}$ acts on $\mathbb{R}$ by multiplication. We have that $[x] = [y]$ if and only if $|x| = |y|$, so we can identify $\mathbb{R}/G$ with $[0, \infty)$.

Example 2.7. The group of permutations $G = S_n$ acts on $\mathbb{R}^{d\times n}$ as described above. We have that

$$X = (x_1, \ldots, x_n) \sim Y = (y_1, \ldots, y_n)$$

if and only if

$$(x_1, \ldots, x_n) = (y_{\sigma(1)}, \ldots, y_{\sigma(n)}). \tag{2.1}$$

This means that we have equality as sets:

$$\{x_1, \ldots, x_n\} = \{y_1, \ldots, y_n\} \tag{2.2}$$

The converse is not completely true. For example, in the case $d = 1$ if we choose

$$x_1 = 1, x_2 = 1, x_3 = 2 \text{ and } y_1 = 1, y_2 = 2, y_3 = 2$$

Then we will have equality in (2.2) but not in (2.1). This can be remedied by introducing the concept of 'multisets', where order does not matter (like with sets) but repetitions are allowed (unlike sets). For multisets (denoted by double curly brackets) we will have that

$$\{\{1, 1, 2\}\} \neq \{\{1, 2, 2\}\}$$

and in general we will have that (2.1) if and only if we have the corresponding multiset equality

$$\{\{x_1, \ldots, x_n\}\} = \{\{y_1, \ldots, y_n\}\}.$$

Definition 2.8. Let G be a group acting on a real vector space V. We say that the action $\sigma : G \times V \to V$ is linear if for all $g \in G$ the map $V \ni v \mapsto g.v \in V$ is linear

Example 2.9. The group $\mathbb{R}^d$ acts on $\mathbb{R}^{d\times n}$ via

$$g_.(x_1, x_2, \ldots, x_n) = (x_1 + g, x_2 + g, \ldots, x_n + g), \forall g \in \mathbb{R}^d, (x_1, \ldots, x_n) \in \mathbb{R}^{d\times n}.$$

For $g \neq 0$ the map $g : \mathbb{R}^{d\times n} \to \mathbb{R}^{d\times n}$ is not linear as

$$g_.(0 + 0) = g_.(0) \neq 2g = g_.(0) + g_.(0).$$

The group $O(d)$ acts on $\mathbb{R}^{d\times n}$ via matrix multiplication $R_.X = RX$. Clearly this action is linear.

Example 2.10 (Multisets)**.** The action of S_n on $\mathbb{R}^n$ is linear: for every $\sigma \in S_n$ and $x, y \in \mathbb{R}^n$ and $\lambda \in \mathbb{R}$ we have

$$(\sigma_.(\lambda x))_j = (\lambda x)_{\sigma^{-1}(j)} = \lambda(x)_{\sigma^{-1}(j)} = \lambda(\sigma_.(x))_j$$

and

$$(\sigma_.(x + y))_j = (x + y)_{\sigma^{-1}(j)} = x_{\sigma^{-1}(j)} + y_{\sigma^{-1}(j)} = (\sigma_.x)_j + (\sigma_.y)_j$$

Matrix notation Since the action of $\sigma \in S_n$ on vectors in $\mathbb{R}^n$ is linear, there is some matrix $P_\sigma \in \mathbb{R}^{n\times n}$ satisfying

$$\sigma(x) = P_\sigma x, \forall x \in \mathbb{R}^n.$$

We can find $P = P_\sigma$ explicitly by noting that for all i and j

$$P_{ij} = [Pe_j]_i = (\sigma(e_j))_i = (e_j)_{\sigma^{-1}(i)} = \begin{cases} 1 & \text{if } j = \sigma^{-1}(i) \\ 0 & \text{if } j \neq \sigma^{-1}(i) \end{cases} = \begin{cases} 1 & \text{if } \sigma(j) = i \\ 0 & \text{if } \sigma(j) \neq i \end{cases}$$

A matrix P induced by a permutation σ is called a *permutation* matrix. Note that $P \in \mathbb{R}^{n\times n}$ is a permutation matrix if and only if $P_{ij} \in \{0, 1\}$ for all $1 \leq i, j \leq n$ and each row and column of P has exactly one non-zero entry. In particular we have that the rows and columns of P sum to one, or in other words

$P1_n = 1_n$ and $1_n^T P = 1_n^T$. We also see that the columns of P form an orthonormal basis, and therefore P is a unitary matrix.

Since the action of permutations is a well-defined group action we deduce that $P_{\tau\circ\sigma} = P_\tau P_\sigma$ because

$$P_{\tau\circ\sigma} x = (\tau\circ\sigma)_. x = \tau_.(\sigma_.(x)) = P_\tau \sigma_.(x) = P_\tau P_\sigma x, \quad \forall x \in \mathbb{R}^n$$

and if $Ax = 0$ for all x then $A = 0$ (verify).

So far we have described the action of S_n on $\mathbb{R}^n$. The corresponding action on $\mathbb{R}^{d\times n}$ can be written as (for a permutation $\sigma \in S_n$ and $X \in \mathbb{R}^{d\times n}$, using X_{i*} to denote the i-th row of X)

$$\sigma_.(X) = \begin{bmatrix} \sigma_.(X_{1*}) \\ \sigma_.(X_{2*}) \\ \vdots \\ \sigma_.(X_{d*}) \end{bmatrix} = \begin{bmatrix} X_{1*}P_\sigma^T \\ X_{2*}P_\sigma^T \\ \vdots \\ X_{d*}P_\sigma^T \end{bmatrix} = XP_\sigma^T$$

Definition 2.11 (Invariant)**.** Let $(G,\cdot)$ be a group, let S, T be sets, and $\rho : G\times S \to S$ a group action . We say that $f : S \to T$ is invariant if

$$f(\rho(g,s)) = f(s), \forall g \in G, s \in S$$

Example: The function $\|x\|$ is invariant with respect to the action of both $O(n)$ and S_n on $\mathbb{R}^n$. The functions $x \mapsto \sum_i x_i$ is S_n invariant but not $O(n)$ invariant.

Problem 2.12. Is there a function $f : \mathbb{R}^{d\times n} \to \mathbb{R}$ which is $SO(d)$ invariant but not $O(d)$ invariant? Is there a function which is $O(d)$ invariant and not $SO(d)$ invariant?

Definition 2.13 (Intertwining)**.** Let $(G,\cdot)$ be a group, let S be a set, and $\rho : G\times S \to S$ a group action . We say that $f : S \to S$ is intertwining if

$$f(\rho(g,s)) = \rho(g, f(s)), \forall g \in G, s \in S$$

Example The function $f(x) = \|x\|x$ is $O(d)$ intertwining because

$$f(Rx) = \|Rx\|Rx = R\left(\|x\|x\right) = Rf(x)$$

Definition 2.14 (Equivariant)**.** Let $(G,\cdot)$ be a group, let S_1, S_2 be sets, and $\rho_i : G\times S_i \to S_i, i = 1,2$ be group actions. We say that $f : S_1 \to S_2$ is equivariant if

$$f(\rho_1(g,s)) = \rho_2(g, f(s)), \forall g \in G, s \in S$$

Examples If we take $S_1 = S_2$ and $\rho_1 = \rho_2$ then we get an intertwining map. If we take ρ_2 to be the trivial action $\rho_2(g,s) = s$ we get an invariant map.

Here is an example which is neither intertwining nor invariant. $O(d)$ acts on $\mathbb{R}^{d\times n}$ by $\rho_1(R,X) = Rx$ and on $\mathbb{R}^{d\times d}$ via $\rho_2(R,Y) = RYR^T$. The mapping $f(X) = XX^T$ is equivariant because

$$f(\rho_1(R,X)) = f(RX) = (RX)(RX)^T = RXX^TR^T = \rho_2(R, f(X)).$$

The following proposition is very useful for the construction of equivariant neural networks, since it means that we can construct equivariant networks by compositions of equivariant functions.

Proposition 2.15. *Let $(G,\cdot)$ be a group acting on S_1, S_2, S_3 through group actions ρ_1, ρ_2, ρ_3 respectively. If $f_1 : S_1 \to S_2$ and $f_2 : S_2 \to S_3$ are equivariant then $f_2 \circ f_1$ is equivariant.*

Proof. For every $g \in G, s \in S_1$ we have

$$f_2 \circ f_1(\rho_1(g,s)) = f_2(\rho_2(g, f_1(s))) = \rho_3(g, f_2\circ f_1(s))$$

□

2.0.2 Permutation equivariance for multisets

We will now focus for a while on construction of permutation equivariant networks for multisets and their theoretical properties. As described above, multisets can be identified with the orbit of a matrix element $X = (x_1, \ldots, x_n) \in \mathbb{R}^{d\times n}$ with respect to the action of the permutations S_n on $\mathbb{R}^{d\times n}$ as described above

$$(\sigma(X))_{ij} = X_{i\sigma^{-1}(j)}$$

In geometric applications $d = 3$ and X is a discretization of a 3D object. In this context it is often called a *point cloud*. Among the first papers to address such data was the famous PointNet paper [Qi et al., 2017]. The paper builds an architecture for point clouds which is permutation equivariant/invariant. It also gives a more ad-hoc solution for rigid motions which we will not discuss.

PointNet's strategy to construct permutation equivariant functions can be seen as a simple generalization of fully connected neural networks: recall that neural networks are functions of the form

$$h_{L+1} \circ \rho \ldots h_2 \circ \rho \circ h_1 \tag{2.3}$$

where ρ could be ReLU or any other elementwise activation function, and the h_i are affine functions. The point net architecture has the same structure, where

$$h_i : \mathbb{R}^{c_{i-1}\times n} \to \mathbb{R}^{c_i \times n}$$

are linear affine functions applied pointwise

$$h_i(x_1, \ldots, x_n) = (A(x_1) + b, \ldots, A(x_n) + b) = AX + b1_n^T$$

where $A = A^i$ is a matrix in $\mathbb{R}^{c_i\times c_{i-1}}$ and $b \in \mathbb{R}^{c_i}$. Clearly each h_i are equivariant with respect to the action of the permutation group: if P is a permutation matrix then using the fact that P^T is also a permutation matrix and that $1_n^T P = 1_n^T$ for every permutation matrix, we obtain

$$h_i(XP^T) = AXP^T + b1_n^T = (AX + b1_n^T)P^T = h_i(X)P^T$$

The activation function ρ could be any function $\rho : \mathbb{R} \to \mathbb{R}$ which is extended to a mapping $\rho : \mathbb{R}^{c_i\times n} \to \mathbb{R}^{c_i\times n}$ by elementwise application $(\rho(X))_{ij} = \rho(X_{ij})$. This map is clearly equivariant=intertwining because for every permutation τ we have that

$$(\tau \circ \rho(X))_{ij} = \rho(X)_{i\tau^{-1}(j)} = \rho(X_{i\tau^{-1}(j)}) \text{ and } (\rho \circ \tau(X))_{ij} = \rho(\tau X)_{ij} = \rho(X_{i\tau^{-1}(j)})$$

Thus by Proposition 2.15 functions obtained by composition of these pointwise applied functions as in (2.3) are permutation equivariant functions from $\mathbb{R}^{c_0\times n}$ to $\mathbb{R}^{c_{L+1}\times n}$.

To obtain a permutation invariant function PointNet uses the permutation invariant 'max pooling': a permutation invariant function on $\mathbb{R}^{d\times n}$ (in our case $d = c_{L+1}$) is given by applying the maximum function row-wise (we call this function $\max_{\text{row}}$)

$$\max_{\text{row}} \begin{pmatrix} X_{11} & X_{12} & \ldots & X_{1n} \\ X_{21} & X_{22} & \ldots & X_{2n} \\ \vdots & \vdots & & \\ X_{d1} & X_{d2} & \ldots & X_{dn} \end{pmatrix} = \begin{pmatrix} \max[X_{11}, X_{12}, \ldots, X_{1n}] \\ \max[X_{21}, X_{22}, \ldots, X_{2n}] \\ \vdots \\ \max[X_{d1}, X_{d2}, \ldots, X_{dn}] \end{pmatrix}.$$

A popular permutation invariant alternative is 'sum pooling' where the maximum over each row is replaced by summation over each row. After pooling was applied and an invariant function was achieved, the function is composed with a standard fully connected network. To summarize, the PointNet architecture includes functions of the form

$$\mathcal{H}_{pointnet} = \{f(\max_{\text{row}}(h(x_1), h(x_2), \ldots, h(x_n)))|h : \mathbb{R}^d \to \mathbb{R}^m, \text{ and } f : \mathbb{R}^m \to \mathbb{R}^{n_{out}} \text{are neural networks}\} \tag{2.4}$$

Specifying the hyper-parameters d, n, m, n_{out} determines a function space $\mathcal{H}_{pointnet}$. In implementation we take f and h to be neural networks with prescribed width and depth.

Deep sets and characterization of linear equivariant functions In Deep Sets [Zaheer et al., 2017] the authors suggested that, since we are looking for expressions of the form (**??**), where the h_i are linear (actually affine) equivariant functions, we might as well look for *all possible* linear equivariant functions. We now turn to characterize all linear permutation equivariant functions, first from $\mathbb{R}^n \to \mathbb{R}^n$ and then eventually from $\mathbb{R}^{c\times n}$ to $\mathbb{R}^{c'\times n}$.

Problem 2.16. Find linear mappings $L:\mathbb{R}^n \to \mathbb{R}^n$ which are intertwining with respect to the action of S_n.

Solution: The map $I_n(v)=v$ is intertwining because

$$I_n(\sigma_. v) = \sigma_.(v) = \sigma_.(I_n(v)), \forall v \in \mathbb{R}^n, \sigma \in S_n$$

Similarly the linear map

$$S(x) = 1_n 1_n^T x = \left(\sum_{j=1}^n x_j\right) 1_n$$

is intertwining because for every permutation σ and corresponding permutation matrix $P = P_\sigma$

$$\sigma_.(S(x)) = \sigma(1_n 1_n^T x) = P 1_n 1_n^T x = 1_n 1_n^T x = 1_n 1_n^T P x = S(\sigma_. x)$$

Finally, we note that the space of intertwining linear mappings

$$\mathcal{L} = \{A \in \mathbb{R}^{n\times n} |\, AP_\sigma = P_\sigma A \quad \forall \sigma \in S_n\}$$

is a linear subspace of $\mathbb{R}^{n\times n}$. It follows that linear combinations of I_n and $S = 1_{n\times n}$ are intertwining. We will now see that these are the only linear intertwining maps.

Proposition 2.17. $L:\mathbb{R}^n \to \mathbb{R}^n$ *is a linear mapping which is intertwining with respect to the action of* S_n*, if and only if it is of the form*

$$L = \alpha I_n + \beta 1_{n\times n}$$

Remark: Equivalently, linear equivariant mappings $L:\mathbb{R}^n \to \mathbb{R}^n$ are of the form

$$L(x) = \alpha x + \beta\left(\sum_{i=1}^n x_i\right) 1_n$$

Proof. In our previous discussion we saw that any linear combination of I_n and $1_{n\times n}$ is intertwining. We now prove the reverse:

Assume L is linear and intertwining, we need to show that $L_{ii} = L_{11}$ and $L_{ij} = L_{12}$ for all $1 \le i \le n, 1 \le j \le n$ with $i \neq j$.

For given j, let σ be the transposition of j and 1 (the permutation that swaps between j and 1 and fixes all other indices) and let $P = P_\sigma$. So $Pe_j = e_1$ and $Pe_1 = e_j$. Then

$$L_{jj} = \langle e_j, Le_j\rangle = \langle e_j, LPe_1\rangle = \langle e_j, PLe_1\rangle = \langle P^T e_j, Le_1\rangle = \langle P^T P e_1, Le_1\rangle = \langle e_1, Le_1\rangle = L_{11}.$$

Moreover, if $a \neq b$ we can define σ with $\sigma(1) = a, \sigma(2) = b$. Let P denote the corresponding permutation matrix. Then

$$P(e_a) = e_1, P(e_b) = e_2$$

and we get

$$L_{1,2} = \langle e_1, Le_2\rangle = \langle e_1, LPe_b\rangle = \langle e_1, PLe_b\rangle = \langle P^T e_1, Le_b\rangle = \langle P^T P e_a, Le_b\rangle = \langle e_a, Le_b\rangle = L_{a,b}$$

. $\square$

Now that we characterized all linear equivariant functions from $\mathbb{R}^n$ to $\mathbb{R}^n$, we discuss how to handle several copies of $\mathbb{R}^n$:

Proposition 2.18. *A function $L : \mathbb{R}^{a\times n} \to \mathbb{R}^n$ is linear and equivariant with respect to the action of S_n, in and only if it is of the form*

$$L(X) = \sum_{i=1}^{a} L^{(i)}(X_{i,*}) \tag{2.5}$$

where each $L^{(i)} : \mathbb{R}^n \to \mathbb{R}^n$ is linear and intertwining.

Remark: Note that for convenience of notation we allow L to be applied to row vectors instead of column vectors.

Remark: Equivalently, linear equivariant mappings $L : \mathbb{R}^{a\times n} \to \mathbb{R}^n$ are of the form

$$L(X) = \sum_{i=1}^{a} \alpha_i (X_{i*})^T + \sum_{i=1}^{a} \beta_i \left(\sum_{j=1}^{n} X_{ij} \right) 1_n$$

Proof. First assume that L is of the form (2.5). We need to show that for all $X \in \mathbb{R}^{a\times n}$ and permutation σ

$$L(\sigma_. X) = \sigma_.(LX)$$

Indeed

$$L(\sigma_. X) = \sum_{i=1}^{a} L^{(i)}((\sigma_. X)_{i,*}) = \sum_{i=1}^{a} L^{(i)}(\sigma_.(X_{i,*})) = \sum_{i=1}^{a} \sigma_. L^{(i)}(X_{i,*}) = \sigma_. \left(\sum_{i=1}^{a} L^{(i)}(X_{i,*}) \right) = \sigma_. L(X)$$

where we used the fact that permutation of L is equivalent to permuting each of the rows.

In the other direction, Assume $L : \mathbb{R}^{a\times n} \to \mathbb{R}^n$ is linear and equivariant. The linearity implies that L can be decomposed as

$$L(X) = \sum_{i=1}^{a} L^{(i)}(X_{i,*})$$

where each $L^{(i)}$ is linear. We need to show that each $L^{(i)}$ is also equivariant. Due to the equivariance of L, we have for all permutations σ, for all $x \in \mathbb{R}^n$ and for every X whose j-th row is x^T and all other rows are zero

$$\sigma_.(L^{(j)} x^T) = \sigma_.(LX) = L(\sigma_.(X)) = L^{(j)}(\sigma_.(x^T))$$

and so each $L^{(j)}$ is equivariant and we are done. □

New Lesson

Finally we have

Proposition 2.19. *$L : \mathbb{R}^{a\times n} \to \mathbb{R}^{b\times n}$ is a linear mapping which is equivariant with respect to the action of S_n, if and only if it is of the form*

$$L = \begin{pmatrix} L_{(1)} \\ L_{(2)} \\ \vdots \\ L_{(b)} \end{pmatrix}$$

where $L_{(i)} : \mathbb{R}^{a\times n} \to \mathbb{R}^n$ is linear and equivariant.

Proof. L is linear if and only if

$$L = \begin{pmatrix} L_{(1)} \\ L_{(2)} \\ \vdots \\ L_{(b)} \end{pmatrix}$$

where each $L_{(i)}$ are linear. Moreover by equivariance of L we have that

$$\begin{pmatrix} \sigma L_{(1)}(X) \\ \sigma L_{(2)}(X) \\ \vdots \\ \sigma L_{(b)}(X) \end{pmatrix} = \sigma L(X) = L(\sigma X) = \begin{pmatrix} L_{(1)}(\sigma X) \\ L_{(2)}(\sigma X) \\ \vdots \\ L_{(b)}(\sigma X) \end{pmatrix}$$

which shows that each $L_{(i)}$ is equivariant. □

Remark: Piecing all of this together, linear equivariant mappings $L : \mathbb{R}^{a\times n} \to \mathbb{R}^{b\times n}$ are of the form

$$L(X) = \begin{pmatrix} \sum_{i=1}^a \alpha_i^{(1)} X_{i*} + \sum_{i=1}^a \beta_i^{(1)} \left(\sum_{j=1}^n X_{ij}\right) 1_n^T \\ \vdots \\ \sum_{i=1}^a \alpha_i^{(b)} X_{i*} + \sum_{i=1}^a \beta_i^{(b)} \left(\sum_{j=1}^n X_{ij}\right) 1_n^T \end{pmatrix}$$

We denote the space of all such mappings by $\mathcal{L}(n, a, b)$. Note that this space is determined by $2a \cdot b$ parameters. In contrast, Pointnet considers only functions of the form $X = (x_1, \ldots, x_n) \mapsto (Ax_1, \ldots, Ax_n)$ where $A \in \mathbb{R}^{b\times a}$. This functions space has $a \cdot b$ parameters which correspond to choosing all $\beta_i^{(j)} = 0$.

The deep set architecture, like the point net architecture, consist of functions of the form

$$h_{L+1} \circ \rho \ldots h_2 \circ \rho \circ h_1 \tag{2.6}$$

where the linear part of $h_i : \mathbb{R}^{d_{i-1}\times n} \to \mathbb{R}^{d_i\times n}$ is in $\mathcal{L}(n, d_{i-1}, d_i)$. This ensures that all such functions are permutation equivariant.

2.0.3 Equivariant Universality

In Chapter 1 we discussed universality of fully connected neural networks and more refined approximation properties. We now discuss the notion of equivariant universality which is the analogue of universality for equivariant networks.

Assume that we have a function class $\mathcal{H}$ which consists of continuous functions $h : V_1 \to V_2$ which are all equivariant with respect to the actions ρ_1, ρ_2 of G on V_1 and V_2 (these are defined in order to learn a function f with the same equivariant structure). We say that $\mathcal{H}$ is *Equivariantly Universal* if for any $K \subseteq V_1$, any continuous equivariant $f : V_1 \to V_2$ can be approximated uniformly in K by functions $h \in \mathcal{H}$. Invariant universality is the special case where the action ρ_2 is trivial $\rho_2(g, x) = x$. In the course we will only discuss the invariant case.

We now begin by discussing a general methodology for proving invariant universality, and then move on to consider universality of the pointnet architecture defined above (with sum pooling though, and not max-pooling).

2.0.4 Separating Invariants and Universality

Definition 2.20. Let G be a group acting on a set V. We say that $\alpha : V \to \mathbb{R}^m$ is a separating invariant mapping if

1. **Invariance** α is invariant, that is if $w, v \in V$ and $w \sim v$ then
$$\alpha(v) = \alpha(w).$$
2. **Separation** If $\alpha(v) = \alpha(w)$ then $w \sim v$.

We will sometimes say that the functions $\alpha_1, \ldots, \alpha_m$ are invariant and separating, instead of saying that α is invariant and separating.

Example 2.21. Consider the action of $O(1)=\{-1,1\}$ on $\mathbb{R}$. The mapping $\alpha(x)=x^2$ is invariant and separating .

Similarly for the action of $O(d)$ on $\mathbb{R}^d$ the mapping $\alpha(x)=\|x\|$ is invariant and separating (see that you know why). Can you think of any other invariants for this group action?

For the action of $O(d)$ on $\mathbb{R}^{d\times n}$ is the mapping

$$\alpha(x_1,\ldots,x_n)=(\|x_1\|,\ldots,\|x_n\|)$$

invariant and separating ? What about the mapping $\alpha(X)=X$?(the first is not separating, the second is not invariant).

If α is invariant then $F\circ\alpha$ is invariant as well. The next theorem show that when α is separating these are in fact all possible invariants

Proposition 2.22. *Let G be a group acting on a set V and let $\alpha: V\to\mathbb{R}^m$ be a separating invariant mapping. A function $f: V\to\mathbb{R}$ is G-invariant if and only if there exists some $F:\mathbb{R}^m\to\mathbb{R}$ such that*

$$f(x)=F(\alpha(x)). \tag{2.7}$$

Proof. The invariance of α means that we can define a function $\tilde{\alpha}[v]=\alpha(v)$ on the quotient space V/G and this definition does not depend on the choice of the representative in the orbit $[v]$. Similarly, since f is invariant there exists $\tilde{f}: V/G\to\mathbb{R}$ with

$$\tilde{f}([v])=f(v).$$

The fact that α is separating implies that $\tilde{\alpha}$ is injective, so for all $v\in V$

$$f(v)=\tilde{f}[v]=\underbrace{\tilde{f}\circ\tilde{\alpha}^{-1}}_{F}\circ\tilde{\alpha}[v]=F\circ\alpha(v). \tag{2.8}$$

Note that $f\circ\tilde{\alpha}^{-1}$ is defined on $\alpha(V)$ and we think of F is some extension of this function to all of $\mathbb{R}^m$. $\square$

Typically our invariant functions f and α are *continuous*, and we would like to say that the function F in Proposition 2.22 is also continuous. This is indeed the case once we add some minor assumptions:

Proposition 2.23 (Without proof). *Let G be a group acting on a metric space V, and let $\alpha: V\to\mathbb{R}^m$ be a* continuous *separating invariant mapping. For every* continuous *G-invariant function $f: V\to\mathbb{R}$ and every compact $K\subseteq V$, there exists some* continuous *$F=F_K:\mathbb{R}^m\to\mathbb{R}$ such that*

$$f(x)=F(\alpha(x)),\forall x\in K. \tag{2.9}$$

Sketch of proof. The proof is simple but relies on some facts from topology which are not in the course's prerequisites, and can be found in [Munkres, 2000]. The general idea is that under the standard definition of the quotient topology for V/G, the maps $\tilde{\alpha},\tilde{f}$ defined as in the previous theorem will be continuous, as is the quotient map $q(v)=[v]$. We need to have the map $\tilde{\alpha}^{-1}$ continuous as well. The map $\tilde{\alpha}$ is continuous and a bijection. In general this does not imply continuity of the inverse but it does in our case where the domain $q(K)$ is compact. Thus the functions in (2.8) are continuous, and F is a continuous extension of $\tilde{f}\circ\tilde{\alpha}^{-1}$ from its compact domain to all of $\mathbb{R}^m$. $\square$

Proposition 2.23 can be used to construct universal invariant hypothesis classes. In particular functions classes of the form

$$\mathcal{H}_{inv}=\{h\circ\alpha(x)|h \text{ is fully connected. }\}$$

will be universal. We can also construction universal invariant function classes of the form

$$\mathcal{H}_{inv}=\{h\circ h_{inv}(x)|h \text{ is fully connected. } h_{inv} \text{ comes from a family of invariant functions }\}$$

where the collection of possible h_{inv} includes α or at least can approximate α. This is based on the following lemma:

Lemma 2.24. *[Don't need proof for exam] Let G be a group acting on a metric space V, and let $\alpha : V \to \mathbb{R}^m$ be a* continuous *separating invariant mapping. Let $K \subseteq V$ be a compact set and assume that $\alpha_n : V \to \mathbb{R}^m$ are continuous invariant functions which converge uniformly to α in $C(K, \mathbb{R}^m)$. Then for every continuous invariant $f : V \to \mathbb{R}$ there exists neural network functions $F_n : \mathbb{R}^m \to \mathbb{R}$ such that $F_n \circ \alpha_n$ converges to f uniformly on $C(K)$.*

Proof. Since $\alpha(K)$ is compact it is contained in some closed ball $\bar{B}_R \subseteq \mathbb{R}^m$. Since $\|\alpha_n - \alpha\|_{C(K,\mathbb{R}^m)} \to 0$ there exists some $r > 0$ such that $\|\alpha_n - \alpha\| \leq r$ for all n. It follows that $\alpha_n(K) \subseteq \bar{B}_{R+r}$ for all n. Let $F : \mathbb{R}^m \to \mathbb{R}$ be the function satisfying (2.9) as promised in Proposition 2.23. Let $F_n : \mathbb{R}^m \to \mathbb{R}$ be neural network functions which approximate F uniformly on $\bar{B}_{R+r}$.

Choose some $\epsilon > 0$, we need to show that there exists some N such that for all $n > N$

$$\max_{x \in K} |(F_n \circ \alpha_n)(x) - (F \circ \alpha)(x)| < \epsilon$$

Note that for every $n \in \mathbb{N}$ and $x \in K$,

$$\begin{aligned} |(F_n \circ \alpha_n)(x) - (F \circ \alpha)(x)| &= |(F_n \circ \alpha_n)(x) - (F \circ \alpha_n)(x)| + |(F \circ \alpha_n)(x) - (F \circ \alpha)(x)| \\ &\leq \|F_n - F\|_{\bar{B}_{R+r}} + |(F \circ \alpha_n)(x) - (F \circ \alpha)(x)| \end{aligned} \tag{2.10}$$

We choose N large enough such that

1. For all $n \geq N$ we have $\|F_n - F\|_{\bar{B}_{R+r}} \leq \epsilon/2$.

2. Since F is continuous it is uniformly continuous on $\bar{B}_{R+r}$. We choose some $\delta > 0$ such that $F(y_1) - F(y_2)| < \epsilon/2$ whenever $\|y_1 - y_2\| \leq \delta$ and then choose N large enough so that $\max_{x \in K} |\|\alpha_n(x) - \alpha(x)\| < \delta$. This implies that $|(F \circ \alpha_n)(x) - (F \circ \alpha)(x)| \leq \epsilon/2$ for all $x \in K$.

Due to (2.10) this concludes our argument. □

We will now consider separating invariants for the action of permutations on multisets and then use them to show universality of point-net/deep sets/ other models.

Continuous Separating Invariants for the action of S_n on $\mathbb{R}^n$ We consider three different types of separating invariants for the action of S_n on $\mathbb{R}^n$:

1. **Sorting** The sorting mapping $\mathbf{sort}(x)$ sorts a vector $x \in \mathbb{R}^n$ by size, e.g.,

$$\mathbf{sort}(1, 3, 2, 7, 4) = (1, 2, 3, 4, 7).$$

This mapping is clearly permutation invariant and separating. It is also continuous piecewise linear (you will see in homework).

2. **Elementary symmetric polynomials** The elementary symmetric polynomials are defined as

$$e_0(x) = 1, \quad e_1(x) = \sum_{i=1}^{n} x_i, \quad e_2(x) = \sum_{1 \leq i < j \leq n} x_i x_j,$$
$$e_3(x) = \sum_{1 \leq i < j < k \leq n} x_i x_j x_k, \quad \ldots e_n(x) = x_1 \cdot x_2 \cdot \ldots \cdot x_n$$

Clearly these polynomials are continuous and invariant to permutations. We prove that they are separating using the proof in [Zaheer et al., 2017]

Theorem 2.25. *The mapping $e : \mathbb{R}^n \to \mathbb{R}^n$ defined by*

$$e(x) = (e_1(x), \ldots, e_n(x))$$

is separating with respect to the action of S_n on $\mathbb{R}^n$.

Proof. Given $x, y \in \mathbb{R}^n$, we need to show that if $e(x) = e(y)$ then $[x] = [y]$.

We use x, y to define two polynomials

$$P_x(t) = \prod_{i=1}^{n}(t - x_i) \text{ and } P_y(t) = \prod_{i=1}^{n}(t - y_i)$$

the roots of P_x (respectively P_y) are exactly the entries of x (or y) and they are equal as polynomials if and only if x and y are equal up to permutation. We can rewrite

$$P_x(t) = a_0 + a_1 t + \ldots + a_n t^n$$
$$P_y(t) = b_0 + b_1 t + \ldots + b_n t^n$$

and note that

$$a_k = (-1)^{n-k} e_k(x), \quad k = 0, 1, \ldots, n$$
$$b_k = (-1)^{n-k} e_k(y), \quad k = 0, 1, \ldots, n$$

It follows that if $e(x) = e(y)$ then $a_k = b_k, k = 0, 1, \ldots, n$ and therefore $P_x = P_y$ which implies that x and y are equal up to permutation. □

3. **Power sum polynomials** The symmetric power sum polynomials are defined as

$$p_0(x) = 1, \quad p_1(x) = \sum_{i=1}^{n} x_i, \quad p_2(x) = \sum_{i=1}^{n} x_i^2, \quad p_3(x) = \sum_{i=1}^{n} x_i^3, \quad \ldots p_n(x) = \sum_{i=1}^{n} x_i^n.$$

It turns out that these polynomials hold the same information as the elementary symmetric polynomials. Indeed, note that

$$p_0 = e_0 \text{ and } p_1 = e_1.$$

Moreover

$$p_1(x)^2 = \left(\sum_{i=1}^{n} x_i\right)^2 = \sum_{i=1}^{n} x_i^2 + 2 \sum_{1 \le i < j \le n} x_i x_j = p_2(x) + 2e_2(x)$$

so

$$2s_2(x) = p_1(x)^2 - p_2(x).$$

In general Newton's identities (which we will not prove) state that

$$ke_k(x) = \sum_{i=1}^{k} (-1)^{k-i} e_{k-i}(x) p_i(x).$$

In particular if $p_k(x) = p_k(y)$ for all $k = 1, \ldots, n$ then $e_k(x) = e_k(y)$ for all $k = 1, \ldots, n$ which means that x and y are equivalent up to permutation according to the previous theorem. Thus we see that the power sum polynomials are also continuous separating invariants.

Remark 2.26. A related but stronger result which we will not need here is that the algebra generated by the elementary symmetric polynomials (equivalently, the algebra generated by the power sum polynomials) is in fact equal to the algebra of *all* S_n invariant polynomials. In other words, for every polynomial $p : \mathbb{R}^n \to \mathbb{R}$ which is S_n invariant there is some polynomial $q : \mathbb{R}^{n+1} \to \mathbb{R}$ such that

$$p(x) = q(e_0(x), e_1(x), \ldots, e_n(x))$$

Universal hypothesis classes for the action of S_n on $\mathbb{R}^n$ It follows immediately from our discussion above that taking $\alpha : \mathbb{R}^n \to \mathbb{R}^m$ to be a continuous separating invariant mapping such as the (i) sorting, (ii) elementary polynomials or (iii) power sum polynomials defined above, that functions in

$$\mathcal{H} = \{h \circ \alpha | h : \mathbb{R}^m \to \mathbb{R} \text{ is a neural network}\}$$

can approximate any continuous S_n invariant function uniformly on compact subsets of $\mathbb{R}^n$.

We now prove universality of the pointnet architecture with sum-pooling. This can also be used to show universality of the deep set architecture.

Recall that we denote elements in $\mathbb{R}^{d\times n}$ by $X = (x_1, \dots, x_n)$ where each x_i is a vector in $\mathbb{R}^d$. We prove

Theorem 2.27. *For every n, and every compact $K \subseteq \mathbb{R}^n$, every $f : \mathbb{R}^n \to \mathbb{R}$ which is continuous and S_n invariant can be approximated uniformly on K by functions in*

$$\mathcal{H}_{pointnet-sum} = \{F\left(\sum_{i=1}^{n} h(x_i)\right) |h : \mathbb{R} \to \mathbb{R}^n, \textit{ and } F : \mathbb{R}^n \to \mathbb{R} \textit{ are neural networks}\}$$

Proof. Consider the mapping $m : \mathbb{R} \to \mathbb{R}^n$ defined by

$$m(x) = (x, x^2, \dots, x^n).$$

and define $p(x)$ to be the mapping

$$p(x) := \sum_{i=1}^{n} m(x_i) = \left(\sum_{i=1}^{n} x_i, \sum_{i=1}^{n} x_i^2, \dots \sum_{i=1}^{n} x_i^n\right)$$

which is continuous invariant and separating. Note that universality of neural networks implies that $m : \mathbb{R} \to \mathbb{R}^n$ can be approximated on compact sets by a neural network $h : \mathbb{R} \to \mathbb{R}^n$, and as a result $p(x) = \sum_{i=1}^n m(x_i)$ can be approximated on compact sets by expression of the form $\sum_{i=1}^n h(x_i)$. The claim now follows from Lemma 2.24: □

Continuous Separating Invariants for the action of S_n on $\mathbb{R}^{d\times n}$ We now turn to consider separating invariants (and universality) for the action of S_n on $\mathbb{R}^{d\times n}$ where now $d > 1$. A natural approach to generalize the one-dimensional **sort** function is by using lexicographical sorting, where e.g., we sort according to the first coordinate from small to large, and break equalities by sorting according to the second coordinate from small to large, e.g.

$$\textbf{lexsort}\begin{pmatrix} 1 & 5 & 5 & 8 & 7 & 7 \\ 6 & 5 & 4 & 1 & 2 & 3 \end{pmatrix} = \begin{pmatrix} 1 & 5 & 5 & 7 & 7 & 8 \\ 6 & 4 & 5 & 2 & 3 & 1 \end{pmatrix}.$$

This type of mapping is indeed both permutation invariant and separating but it is not continuous. In contrast, we can get polynomial invariants which are continuous separating invariants.

We consider S_n polynomial invariants for $\mathbb{R}^{d\times n}$ which are obtained from polynomial invariants of $\mathbb{R}^n$ by a process called *polarization*. For given $k = 0, \dots, n$ and $w \in \mathbb{R}^d$ we define

$$q(X; w, k) = p_k(w^T X) \text{ where } p_k(t_1, \dots, t_n) = \sum_{i=1}^{n} t_i^k.$$

these are indeed invariant as for all permutation matrices P, since $p_k : \mathbb{R}^n \to \mathbb{R}$ are permutation invariant:

$$q(XP^T; w, k) = p_k(w^T XP^T) = p_k(w^T X).$$

Note that for each k, w, the function $q(X; w, k)$ is a multivariate polynomial from $\mathbb{R}^{d\times n}$ of degree $k \leq n$. We claim that this infinite family of polynomial is separating in the following sense:

Lemma 2.28. *If $X, Y \in \mathbb{R}^{d\times n}$ and*

$$q(X; k, w) = q(Y; k, w), \forall k = 0, \dots, n \textit{ and } w \in \mathbb{R}^d$$

then $X = \sigma_. Y$ for some $\sigma \in S_n$.

To prove this lemma we will need a sub-lemma

Lemma 2.29. *Let $A^{(i)} \in \mathbb{R}^{n\times d}, i = 1, \dots, m$ be non-zero matrices, and denote*

$$S_i = \{w \in \mathbb{R}^d | \quad A^{(i)} w = 0\}$$

Then the union $\cup_{i=1}^m S_i$ does not cover all of $\mathbb{R}^d$.

Proof of Lemma 2.29. This can be done using measure considerations, by showing that each S_i has Lebesgue measure zero and therefore so does there union.

Alternatively, we can do this inductively. Denote $T_0 = \mathbb{R}^d$ and

$$T_i = T_{i-1} \setminus S_i, i = 1, \dots, m.$$

Note that T_m is the complement of $\cup_{i=1}^m S_i$. We want to show T_m isn't empty. To do this we will recursively show that each T_i is open and not empty. For T_0 this is clear. Now assuming that T_{i-1} is open and non-empty we see that T_i is also open since it is the intersection of two open sets: T_{i-1} and the complement of S_i. Now let us show that T_i is not empty: if $T_i = T_{i-1}$ we are done. Otherwise, there exists some $w \in T_{i-1} \cap S_i$. Since $A^{(i)} \neq 0$, there exists some row of $A^{(i)}$ which we denote by a_j^T which is not zero. It follows that for every $\epsilon > 0$,

$$a_j^T(w + \epsilon a_j) = \epsilon \|a_j\|^2 \neq 0 \text{ and therefore } A(w + \epsilon a_j) \neq 0,$$

and so $w + \epsilon a_j$ will not be in S_i. Since T_{i-1} is open, for small enough ϵ the vector $w + \epsilon a_j$ will be in T_{i-1} and so will be in $T_i = T_{i-1} \setminus S_i$ which concludes the proof. ☐

Proof of Lemma 2.28. We assume that X, Y are such that $X \neq \sigma_. Y$ for all $\sigma \in S_n$ and we will prove that there exist k, w such that $p_k(w^T X) \neq p_k(w^T Y)$. Indeed for fixed σ let us consider the set

$$B_\sigma = \{w \in \mathbb{R}^d | w^T X = w^T \sigma_.(Y)\} = \{w \in \mathbb{R}^d | w^T (X - \sigma_.(Y)) = 0\}.$$

Note that by assumption for each σ we have that $X - \sigma_. Y \neq 0$ and so by Lemma 2.29 there exists $\bar{w}$ which is not in $\cup_{\sigma \in S_n} B_\sigma$. Since the power-sum polynomials are separating on $\mathbb{R}^n$ and $\bar{w}^T X \not\sim \bar{w}^T Y$, it follows that there exists some k such that $p_k(\bar{w}^T X) \neq p_k(\bar{w}^T Y)$. ☐

The next step is to find a finite number of separating invariants using this lemma. Recall that $\mathcal{P}(d, n)$, the space of multivariate polynomials $b : \mathbb{R}^d \to \mathbb{R}$ of degree $\leq n$ is a linear space of dimension $m = m(n, d) = \binom{n+d}{d}$. We will consider some basis $b_1, \dots, b_m$ for this space. They induce permutation invariant polynomials defined on $\mathbb{R}^{d\times n}$ by

$$B_k(X) = \sum_{i=1}^n b_k(x_i).$$

We prove

Theorem 2.30. *Let $b_1, \dots, b_m : \mathbb{R}^d \to \mathbb{R}$ be polynomials which span the space of polynomials, then the polynomial mappings*

$$B_\ell(X) = \sum_{i=1}^n b_\ell(x_i), \quad \ell = 1, \dots, m \tag{2.11}$$

are invariant and separating with respect to the action of S_n on $\mathbb{R}^{d\times n}$

Proof. Invariance is clear. To show separation, let $X, Y \in \mathbb{R}^{d\times n}$ be such that $B_\ell(X) = B_\ell(Y), \ell = 1, \dots, m$. We will show that this implies that $p_k(wX) = p_k(wY)$ for all k and w, which in turn implies that $X = \sigma_. Y$ for some $\sigma \in S_n$, according to the previous lemma.

We note that for every $w \in \mathbb{R}^d$ and $k = 0, \dots, n$ the polynomial

$$x \mapsto \left(w^T x\right)^k$$

is a polynomial on $\mathbb{R}^d$ of degree $k \leq n$, and therefore there exist coefficients $c_\ell^{(k,w)}, \ell = 1, \ldots, m$ such that

$$\left(w^T x\right)^k = \sum_{\ell=1}^{m} c_\ell^{(k,w)} b_\ell(x), \forall x \in \mathbb{R}^d$$

It follows that

$$p_k(w^T X) = \sum_{i=1}^{n} \left(w^T x_i\right)^k = \sum_{i=1}^{n} \sum_{\ell=1}^{m} c_\ell^{(k,w)} b_\ell(x_i) = \sum_{\ell=1}^{m} c_\ell^{(k,w)} B_\ell(X)$$

and

$$p_k(w^T Y) = \sum_{\ell=1}^{m} c_\ell^{(k,w)} B_\ell(Y).$$

Since by assumption $B_\ell(X) = B_\ell(Y)$ for all ℓ it follows that $p_k(w^T X) = p_k(w^T Y)$ for all k, w and therefore $X = \sigma_. Y$ for some $\sigma \in S_n$ as required. $\square$

Remark 2.31. It is possible to obtain $\tilde{m} = n(2nd + 1) << m(n, d)$ continuous separating invariants by taking random $w_{(1)}, \ldots, w_{(2nd+1)}$, and defining the invariants to be

$$X \mapsto q_k(w_{(j)}^T X), k = 1, \ldots, n, j = 1, \ldots, 2nd + 1.$$

For more details see [Dym and Gortler, 2022]. In this construction we can also replace the power sum polynomials with the sorting function.

New Lesson

An immediate consequence of Theorem 2.30 is the universality of point-net with sum-pooling for the case $d > 1$:

Theorem 2.32. *For every d, n, and every compact $K \subseteq \mathbb{R}^{d\times n}$, every $f : \mathbb{R}^{d\times n} \to \mathbb{R}$ which is continuous and S_n invariant can be approximated uniformly on K by functions in*

$$\mathcal{H}_{pointnet-sum} = \{F\left(\sum_{i=1}^{n} h(x_i)\right) | h : \mathbb{R}^d \to \mathbb{R}^m, \text{ and } F : \mathbb{R}^m \to \mathbb{R} \text{ are neural networks}\}$$

Sketch of proof. The proof is the same of the proof of Theorem 2.27: we use the fact that the polynomials $B_\ell(X) = \sum_{i=1}^{n} b_\ell(x_i)$ in (2.11) are separating, and the fact that the function $x \mapsto (b_\ell(x))_{\ell=1}^{m}$ can be approximated by neural networks of the form $h : \mathbb{R}^d \to \mathbb{R}^m$ due to universality of neural networks. □

2.0.5 Networks for sets and multi-sets

We've constructed functions $B : \mathbb{R}^{d\times n} \to \mathbb{R}^m$ which are invariant and separating with respect to the action of S_n. As we discussed, equivalence classes in $\mathbb{R}^{d\times n}$ can be identified with multi-sets with n elements in $\mathbb{R}^d$. Let us denote the space of all such multisets by $\mathcal{S}_n(\mathbb{R}^d)$. Note that since B is invariant, it induces a function $\hat{B}$ on $\mathcal{S}_n(\mathbb{R}^d)$: for a given multiset we can choose any ordering of its elements and define

$$\hat{B}(\{\!\{x_1, \ldots, x_n\}\!\}) = B(x_1, \ldots, x_n).$$

The permutation invariance implies that the definition of $\hat{B}$ does not depend on the order we chose.

Next, we note that the fact that B is separating implies that $\hat{B}$ is injective on $\mathcal{S}_n(\mathbb{R}^d)$.

We will often want to extend our functions to spaces of multisets of varying but bounded sizes. Let us denote by $\mathcal{S}_{\leq n}(\mathbb{R}^d)$ the collection of all multisets of at most n elements from $\mathbb{R}^d$. We would like to define a function $\tilde{B} : \mathcal{S}_{\leq n}(\mathbb{R}^d) \to \mathbb{R}^m$ which is injective.

This goal is achieved relatively easily using the power sum polynomials we considered previously: assume that $b_1, \ldots, b_m$ span the space of polynomials of degree at most n in $\mathbb{R}^d$. We have showed that the function $B = (B_1, \ldots, B_m)$ defined by

$$B_j(x_1, \ldots, x_n) = \sum_{i=1}^{n} b_j(x_i)$$

is invariant and injective on $\mathbb{R}^{d\times n}$. Note that if $n' < n$, this same function is also invariant and injective with respect to the action of $S_{n'}$ on $\mathbb{R}^{d\times n'}$. Accordingly, we can define $b_0(x) = 1$ and $\tilde{B} = (\tilde{B}_0, \ldots, \tilde{B}_m)$ to be

$$\tilde{B}_j\{\!\{x_1, \ldots, x_{n'}\}\!\} = \sum_{i=1}^{n'} b_j(x_i) \tag{2.12}$$

and we can then prove

Proposition 2.33. *The function $\tilde{B} : \mathcal{S}_{\leq n}(\mathbb{R}^d) \to \mathbb{R}^{m+1}$ is injective.*

Proof. Assume that $\tilde{B}\{\!\{x_1, \ldots, x_{n'}\}\!\} = \{\!\{x_1, \ldots, x_{n''}\}\!\})$ for some $n', n'' \leq n$. Then in particular

$$n' = \sum_{i=1}^{n'} B_0(x_i) = \sum_{i=1}^{n''} B_0(y_i) = n''$$

and then the fact that (2.12) holds for $j = 1, \ldots, m$ implies that $x_1, \ldots, x_{n'}$ and $y_1, \ldots, y_{n'}$ are related by a permutation and so the multisets containing these points are equal. □

Remark 2.34 (Dimension optimality)**.** In general, for any continuous injective function $B : \mathcal{S}_n(\mathbb{R}^d) \to \mathbb{R}^m$ or $B : \mathcal{S}_{\leq n}(\mathbb{R}^d) \to \mathbb{R}^m$ the dimension m must satisfy $m \geq nd$ (see [Wagstaff et al., 2022] for the $d = 1$ case and [Joshi et al., 2023, Amir et al., 2023] for the general case). When the set elements are assumed to come from a finite or countable set (we can think of this as $d = 0$) then we can actually get $m = 1$ as discussed in [Zaheer et al., 2017, Xu et al., 2018, Amir et al., 2023].

2.1 Graph Neural Networks and Graph Isomorphism test

Our presentation here mostly follows [Xu et al., 2018]. We consider learning tasks over graphs. A graph is defined as usual through a finite set of n nodes V which we denote by $\{1,\ldots,n\}$, and a set of unordered pairs of nodes $E \subseteq V \times V$ which we call edges. These can be encoded by the adjacency matrix A, where $A_{u,v} = 1$ if $(u,v) \in E$ and otherwise $A_{uv} = 0$. Additionally, we have a specified vector valued label per node $x_v \in \mathbb{R}^d$, which can be represented using a $d \times n$ matrix X. Thus our graph can be represented by two matrices $G = (A, X)$.

We would like to learn functions of the form

$$G \mapsto F(G) \in \mathbb{R}^k \tag{2.13}$$

or of the form

$$G \mapsto (f_v(G))_{v \in V} \tag{2.14}$$

For example, we can think of a graph representing a molecule, where the nodes V represent atoms, and the edges represent atoms which are linked by a chemical bond. The node features x_v in this case could represent the type of atom (hydrogen, oxygen, etc.). The function we would like to learn could be certain properties of the molecules e.g., is the molecule toxic? Is it effective for treatment of a given disease? Is it soluble in water? The answer to these three question could be encoded as a binary vector in $\mathbb{R}^3$, and a function answering these questions can be encoded by a function F as in (2.13) whose input is a graph and output is an $\mathbb{R}^3$ vector. For applications of graph neural networks for chemistry see e.g. [Reiser et al., 2022].

Another example could be a graph representing a social network such as LinkedIn, where nodes represent users, and edges between nodes represent users who are friends. The node features x_v could represent known data on users (e.g., age, number of degrees) and a node valued function $f_v(G)$ such as the one described in (2.14) could represent the likeliness that the user corresponding to node v will be a good job candidate.

2.1.1 Symmetries of graphs

We would like to design hypothesis classes for approximating F and f in (2.13) and (2.14) respectively, which respect the permutation symmetry of graphs. Note that relabeling the nodes of a graph $G = (A, X)$ using a permutation τ corresponds to the following action

$$(\tau_. (A, X)) = (\tau_. A, \tau_. X)$$

where

$$(\tau_. (A))_{ij} = A_{\tau^{-1}(i),\tau^{-1}(j)} \text{ and } (\tau . X)_{ij} = X_{i\tau^{-1}(j)}$$

We have

Lemma 2.35. *For every $A \in \mathbb{R}^{n\times n}, X \in \mathbb{R}^{d\times n}$ and permutation $\tau \in S_n$, we have that*

$$\tau_. A = P_\tau A P_\tau^T, \quad \tau . X = X P_\tau^T$$

and the action defined by τ is a well defined group action.

Proof. For every permutation σ and $A \in \mathbb{R}^{n\times n}$ we have that

$$(\sigma_. A)_{ij} = (P_\sigma A P_\sigma^T)_{ij} = (P_\sigma A)_{i\sigma^{-1}(j)} = (A^T P_\sigma^T)_{\sigma^{-1}(j),i} = A^T_{\sigma^{-1}(j),\sigma^{-1}(i)} = A_{\sigma^{-1}(i),\sigma^{-1}(j)}$$

We note that this is a group action because

$$\tau_. (\sigma_. A) = P_\tau (\sigma_. A) P_\tau^T = P_\tau P_\sigma A P_\sigma^T P_\tau^T = P_{\tau\circ\sigma} A P_{\tau\circ\sigma}^T = (\tau \circ \sigma)_. A$$

□

A function F defined in (2.13) will typically be permutation invariant

$$F(P_\sigma A P_\sigma^T, X P_\sigma^T) = F(\sigma_. (A, X)) = F(A, X)$$

and the function f defined in (2.14) will typically be permutation equivariant

$$f(P_\sigma A P_\sigma^T, X P_\sigma^T) = f(\sigma_. (A, X)) = \sigma_. (f(G)) = f(G) P_\sigma^T$$

Equivariant operations on graphs To construct hypothesis classes which consist only of graph-equivariant functions, we need to identify equivariant operations we can use. One simple example: we can extend any $\rho : \mathbb{R} \to \mathbb{R}$ to an elementwise function on X and A and this will be permutation equivariant.

Here is a more interesting example: the function $h : \mathbb{R}^{n\times n} \oplus \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$ defined by

$$h(A, X) = (A, XA)$$

is permutation equivariant because for every permutation τ and corresponding permutation matrix $P = P_\tau$.

$$h\left(\tau_.(A, X)\right) = h(PAP^T, XP^T) = XP^T PAP^T = XAP^T = \tau.(h(A, X)).$$

When A represents an adjacent matrix, the operation $(A, X) \mapsto Y = XA$ can be understood as a 'message passing' procedure, where Y_v is the sum of all features in X belonging to its neighbors. The set of neigbors of a node v is defined as

$$\mathcal{N}(v) = \{u \in V|(u, v) \in E\} = \{u \in V|A_{u,v} = 1\}.$$

Note that

$$(XA)_{i,v} = \sum_{u=1}^{n} X_{iu}A_{uv} = \sum_{u\in\mathcal{N}(v)} X_{iu} = S(\{\!\{X_{iu}|\, u \in \mathcal{N}(v)\}\!\})$$

where S denotes the summation function which is well defined on multisets (it does not depend on the order of the elements).

This 'message passing' interpretation can be generalized: let us denote the spaces of $n \times n$ adjacency matrices by

$$\mathcal{A}_n = \{A \in \{0, 1\}^{n\times n}, A_{ii} = 0 \text{ and } A_{ij} = A_{ji} \forall 1 \le i < j \le n\}$$

This is a subset of $\mathbb{R}^{n\times n}$ and for every $A \in \mathcal{A}_n$ and permutation $\tau \in S_n$ we have that $\tau_.A \in \mathcal{A}_n$ so that the action of the permutation group on this subset is well defined. We then have

Proposition 2.36. *Let $(A, X) \in \mathcal{A}_n \oplus \mathbb{R}^{d\times n}$ and let m be a function which maps finite multisets with elements in $\mathbb{R}^d$ into $\mathbb{R}^c$. Then the function $h : (A, X) \in \mathcal{A}_n \oplus \mathbb{R}^{d\times n} \to \mathbb{R}^{c\times n}$ defined by*

$$\left(h(A, X)\right)_v = m\{\!\{X_u|\, A_{u,v} = 1\}\!\} = m\{\!\{X_u|\, u \in \mathcal{N}(v)\}\!\}$$

is permutation equivariant.

Proof. For every permutation τ and node $v \in V$ we have that

$$\begin{aligned}
\left(h(\tau_.(A, X))\right)_v &= m\{\!\{(\tau_.X)_u|\, (\tau_.A)_{u,v} = 1\}\!\} \\
&= m\{\!\{X_{\tau^{-1}(u)}|\, A_{\tau^{-1}(u),\tau^{-1}(v)} = 1\}\!\} \\
&= m\{\!\{X_u|\, A_{u,\tau^{-1}(v)} = 1\}\!\} \\
&= \left(h(A, X)\right)_{\tau^{-1}(v)} = \left(\tau_.h(A, X)\right)_v
\end{aligned}$$

□

Proposition 2.36 gives us a general method to construct graph-equivariant functions using functions on multisets. We can take the function m to be the summation function as above, or take m to be more complicated multiset valued functions such as the injective multiset functions we discussed previously. Note that since different nodes generally have neighborhoods of different sizes, we will typically require functions which are will defined on multisets of different cardinalities.

Message Passing Neural Networks Message passing neural networks are a family of (arguably, the most) popular equivariant graph neural networks. They iteratively use the graph structure to redefine node labels. A message passing neural network is initialized via

$$X_v^{(0)} = X_v$$

and then we recursively define $X_v^{(k)}$ from $X_v^{(k-1)}$ via

$$\begin{aligned} a_v^{(k)} &= AGGREGATE^{(k)}\left(\{\!\{X_u^{(k-1)}, u \in \mathcal{N}_v\}\!\}\right) \\ X_v^{(k)} &= COMBINE^{(k)}\left(X_v^{(k-1)}, a_v^{(k)}\right). \end{aligned} \tag{2.15}$$

We can think of this process is applying an equivariant function to obtain.

$$(A, X^{(k)}) = h(A, X^{(k-1)})$$

We can apply this procedure recursively K times, which ultimately gives us a permutation equivariant function

$$f(A, X) = (A, X^{(K)}).$$

If our final goal is to achieve a permutation invariant function we add an additional readout function

$$F(A, X) = READOUT\left(\{\!\{X_v^{(K)} | \, v \in V\}\!\}\right)$$

and the obtained function F is invariant.

As our notation suggests, the functions $AGGREGATE$ and $READOUT$ are required to be well defined on finite multisets. Once the number of iterations K, and the functions $AGGREGATE^{(k)}, COMBINE^{(k)}$ and $READOUT$, are defined, the function F is a well defined, permutation invariant function.

Several well known GNN architectures can be cast in this framework. For example, the pooling variant of GraphSAGE [Hamilton et al., 2017] uses the aggregation function

$$a_v^{(k)} = \max\{\!\{ReLU\left(WX_u^{(k-1)}\right), u \in \mathcal{N}_v\}\!\}$$

where W is some given matrix (which is learned in practice). This is a well defined multiset function, as the same function

$$y_u = ReLU(WX_u^{(k-1)})$$

is applied to all neighbors of v, and the maximum, which is applied row-wise to the matrix whose columns are $y_u, u \in \mathcal{N}(v)$ is a permutation invariant operation.

The COMBINE function in GraphSAGE is taken to be

$$X_v^{(k)} = ReLU\left(B \cdot CONCAT\left(X_v^{(k-1)}, a_v^{(k)}\right)\right),$$

where B is a learnable matrix.

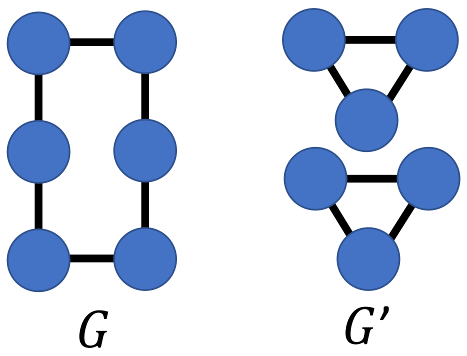

Figure 2.1: Two graphs which are not isomorphic (are not related by a permutation) but cannot be separated by any message passing neural network

Approximation Power It would be natural to inquire what the expressive power of message passing neural networks (MPNN) is, and to hope that like pointnet architecture for point clouds, it will be universal in the sense that it can approximate any continuous permutation invariant function on graphs. As we saw, this question is very strongly linked to the question of finding separating invariants for the group action (in our case, permutation and graphs). Thus one could inquire whether MPNN can separate any graphs up to permutation equivalence.

To be more precise, can we define $F(A, X)$ via a message passing procedure with some specified iteration number K, and some multiset valued functions $AGGREGATE^{(k)}, COMBINE^{(k)}, k = 1, \dots, K$ and $READOUT$, such that $F(A, X) = F(A', X')$ if and only if $A' = PAP^T, X' = XP^T$ for some appropriate permutation matrix P?

It turns out that the answer to this question is negative. In general, since F is permutation invariant by construction we always have that if $A' = PAP^T, X' = XP^T$ then $F(A, X) = F(A', X')$. The converse is not true: Figure 2.1 shows an example of two graphs G, G' which are not isomorphic. Here we can take A and A' to be the adjacency matrices of the two graphs and X and X' to just be a all one vector. We note that for every node v or v' in either graph we have the same initial coloring $X_v = 1$ or $X_{v'} = 1$, and the neighborhoods of all nodes contain exactly two nodes, so that the multisets encountered in the first step of the message passing procedure are just $\{\!\{1, 1\}\!\}$. We can see that the message passing procedure gets 'stuck', where no matter what we do all nodes in both graphs will assume the same value.

The fact that separation of graphs fails is not surprising. The problem of telling whether two graphs are related by a permutation is known as the *Graph Isomorphism Problem (GI)*. There is no known polynomial time algorithm to solve this problem, though it is also not known to be NP-hard. It is one of the examples of a problem which may be NP-intermediate.

In general, the separation power of MPNNs is strongly related to the Weisfeiler-Lehman (WL) graph isomorphism tests which we did not have time to define this year. In general, MPNNs separation power is bounded from above by the WL tests: they cannot separate graphs which the WL test cannot separate. An MPNN which uses injective multiset functions, and injective COMBINE functions, will be equivalent to the WL test in terms of graph separation abilities. For more on this see [Morris et al., 2019, Xu et al., 2018].

Bibliography

[Amir et al., 2023] Amir, T., Gortler, S. J., Avni, I., Ravina, R., and Dym, N. (2023). Neural injective functions for multisets, measures and graphs via a finite witness theorem.

[Daubechies et al., 2021] Daubechies, I., DeVore, R., Foucart, S., Hanin, B., and Petrova, G. (2021). Nonlinear approximation and (deep) relu networks. *Constructive Approximation*, pages 1–46.

[DeVore et al., 2021] DeVore, R., Hanin, B., and Petrova, G. (2021). Neural network approximation. *Acta Numerica*, 30:327–444.

[Dym and Gortler, 2022] Dym, N. and Gortler, S. J. (2022). Low dimensional invariant embeddings for universal geometric learning. *arXiv preprint arXiv:2205.02956*.

[Dym et al., 2020] Dym, N., Sober, B., and Daubechies, I. (2020). Expression of fractals through neural network functions. *IEEE Journal on Selected Areas in Information Theory*, 1(1):57–66.

[Eldan and Shamir, 2016] Eldan, R. and Shamir, O. (2016). The power of depth for feedforward neural networks. In *Conference on learning theory*, pages 907–940. PMLR.

[Hamilton et al., 2017] Hamilton, W., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30.

[Hanin and Rolnick, 2019] Hanin, B. and Rolnick, D. (2019). Deep relu networks have surprisingly few activation patterns. *Advances in neural information processing systems*, 32.

[Joshi et al., 2023] Joshi, C. K., Bodnar, C., Mathis, S. V., Cohen, T., and Liò, P. (2023). On the expressive power of geometric graph neural networks. *arXiv preprint arXiv:2301.09308*.

[Lu et al., 2021] Lu, J., Shen, Z., Yang, H., and Zhang, S. (2021). Deep network approximation for smooth functions. *SIAM Journal on Mathematical Analysis*, 53(5):5465–5506.

[Malach and Shalev-Shwartz, 2019] Malach, E. and Shalev-Shwartz, S. (2019). Is deeper better only when shallow is good? *Advances in Neural Information Processing Systems*, 32.

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

[Morris et al., 2019] Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. (2019). Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4602–4609.

[Munkres, 2000] Munkres, J. R. (2000). *Topology*, volume 2. Prentice Hall Upper Saddle River.

[Pinkus, 1999] Pinkus, A. (1999). Approximation theory of the mlp model in neural networks. *Acta numerica*, 8:143–195.

[Qi et al., 2017] Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017). Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660.

[Raghu et al., 2017] Raghu, M., Poole, B., Kleinberg, J., Ganguli, S., and Sohl-Dickstein, J. (2017). On the expressive power of deep neural networks. In *international conference on machine learning*, pages 2847–2854. PMLR.

[Reiser et al., 2022] Reiser, P., Neubert, M., Eberhard, A., Torresi, L., Zhou, C., Shao, C., Metni, H., van Hoesel, C., Schopmans, H., Sommer, T., et al. (2022). Graph neural networks for materials science and chemistry. *Communications Materials*, 3(1):93.

[Simmons, 1963] Simmons, G. F. (1963). *Introduction to topology and modern analysis*, volume 44. Tokyo.

[Telgarsky, 2016] Telgarsky, M. (2016). Benefits of depth in neural networks. In *Conference on learning theory*, pages 1517–1539. PMLR.

[Wagstaff et al., 2022] Wagstaff, E., Fuchs, F. B., Engelcke, M., Osborne, M. A., and Posner, I. (2022). Universal approximation of functions on sets. *Journal of Machine Learning Research*, 23(151):1–56.

[Xu et al., 2018] Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2018). How powerful are graph neural networks? In *International Conference on Learning Representations*.

[Yarotsky, 2017] Yarotsky, D. (2017). Error bounds for approximations with deep relu networks. *Neural Networks*, 94:103–114.

[Zaheer et al., 2017] Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. (2017). Deep sets. *Advances in neural information processing systems*, 30.

[Zaslavsky, 1975] Zaslavsky, T. (1975). *Facing up to arrangements: Face-count formulas for partitions of space by hyperplanes: Face-count formulas for partitions of space by hyperplanes*, volume 154. American Mathematical Soc.