

Mathematics of Deep Learning

Lecture 2: Curse of dimensionality & Approximation theory

Bruno Loureiro
Département d'Informatique, École Normale Supérieure - PSL & CNRS, France

17/01/2025

Typos, comments or suggestions? Get in touch at: bruno.loureiro@di.ens.fr

1 The curse of dimensionality

Some of the key challenges in machine learning theory arise from the fact that the data and the functions we want to learn live in high-dimensional spaces, such as \mathbb{R}^d with $d \gg 1$. Think for instance of one of the simplest classification problems, MNIST, given by $n = 60000$ images of black and white digits with 28×28 pixels. When vectorised, this gives a data matrix in $\mathbb{R}^{60000 \times 728}$!

The term *curse of dimensionality* was introduced by the [Richard E. Bellman](#), pioneer of dynamical programming, to refer to the hindrance of doing computer science in high-dimensional spaces. We now discuss some examples that illustrate this, and why high-dimensionality does not always make things harder, but sometimes also easier.

1.1 k-Nearest neighbours

Consider two vectors sampled uniformly from the hypercube $\mathbf{x}, \mathbf{x}' \sim \text{Unif}([0, 1]^d)$. What is their expected Euclidean distance?

$$\begin{aligned}\mathbb{E}[\|\mathbf{x} - \mathbf{x}'\|_2^2] &= \sum_{k=1}^d \mathbb{E}[(x_k - x'_k)^2] = d \mathbb{E}[(U - U')^2] \\ &= 2s \left(\underbrace{\mathbb{E}[U^2]}_{1/3} - \underbrace{\mathbb{E}[U]^2}_{1/2^2} \right) = \frac{d}{6}\end{aligned}\tag{1.1}$$

On the other hand, the variance of the distance is given by:

$$\text{Var}[\|\mathbf{x} - \mathbf{x}'\|_2^2] = d \text{Var}[(U - U')^2] = \frac{7d}{180}\tag{1.2}$$

Therefore, the ratio between the typical fluctuation of the distance (given by the standard deviation) and its expected value is given by:

$$\frac{\text{Std}[\|\mathbf{x} - \mathbf{x}'\|_2^2]}{\mathbb{E}[\|\mathbf{x} - \mathbf{x}'\|_2^2]} = O\left(\frac{1}{\sqrt{d}}\right)\tag{1.3}$$

This means that in the high-dimensional limit $d \rightarrow \infty$, the distance between two uniformly sampled points grows much faster than its fluctuations. Why is this a potential a problem?

Let's consider a supervised regression task with training data $\mathcal{D} = \{(\mathbf{x}_i, y_i) \in \mathbb{R}^{d+1} : i \in [n]\}$ drawn i.i.d. from a model:

$$y_i = f_\star(\mathbf{x}_i) + \varepsilon_i, \quad \mathbf{x}_i \sim \text{Unif}([0, 1]^d)\tag{1.4}$$

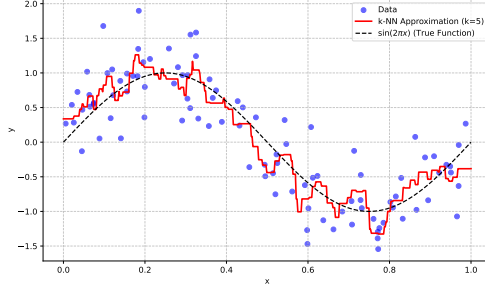


Figure 1: Approximating $f_*(x) = \cos(2\pi x)$ with the 5-NN algorithm. Here, we draw $n = 100$ points from $y_i = f_*(x) + \varepsilon$ with $x_i \sim \text{Unif}([0, 1])$ and $\varepsilon_i \sim \mathcal{N}(0, 0.2)$.

where $f_* : [0, 1]^d \rightarrow \mathbb{R}$ is a regular function (e.g. Lipschitz) and ε_i independent from \mathbf{x}_i and have zero mean. Consider one of the first algorithms we learn in the machine learning course: the k-Nearest Neighbours (kNN) algorithm, consisting of estimating the label of a point \mathbf{x} by taking an average over the labels of all its k closest neighbours in Euclidean norm:

$$f(\mathbf{x}) = \frac{1}{k} \sum_{i: k \text{ smallest } \|\mathbf{x} - \mathbf{x}_i\|_2^2} y_i \quad (1.5)$$

For regular target functions f_* in $d = 1$, this works very well, see fig. 3 for an example. What about for general $d > 1$? In order for eq. (1.5) to be meaningful, for every $\mathbf{x} \in [0, 1]^d$ we need to have at least one training sample \mathbf{x}_i nearby. However, in high-dimensions this is not simple: as we have seen in eq. (1.3) the distance between uniformly sampled points grows in d . How many training points n we then need in order to have at least one \mathbf{x}_i at distance 1 to an arbitrary $\mathbf{x} \in [0, 1]^d$?

Let any $\mathbf{x}_0 \in [0, 1]^d$ consider the set of points at distance at least $r > 0$ from \mathbf{x}_0 :

$$B_r(\mathbf{x}_0, r) := \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x} - \mathbf{x}_0\|_2 \leq r\} \quad (1.6)$$

Asking for having at least one training point \mathbf{x}_i distance 1 from any point in the hypercube $\mathbf{x} \in [0, 1]^d$ is equivalent:

$$[0, 1]^d \subset \bigcup_{i=1}^n B_d(\mathbf{x}_i, 1) \quad (1.7)$$

Taking the volume on both sides, we must have:

$$\text{Vol}([0, 1]^d) \leq \text{Vol}\left(\bigcup_{i=1}^n B_d(\mathbf{x}_i, 1)\right) \leq n \text{Vol}(B_d(0, 1)) \quad (1.8)$$

However, since:

$$\text{Vol}(B_d(0, r)) = \frac{\pi^{d/2}}{\Gamma(d/2 + 1)} r^d \asymp \left(\frac{2\pi e r^2}{d}\right)^{d/2} (d\pi)^{-1/2} = O(d^{-d/2 - \frac{1}{2}}), \text{ as } d \rightarrow \infty \quad (1.9)$$

while $\text{Vol}([0, 1]^d) = 1$, we have:

$$1 \leq n \frac{\pi^{d/2}}{\Gamma(d/2 + 1)} r^d \Leftrightarrow n \geq O(d^{d/2 + 1/2}) \text{ as } d \rightarrow \infty \quad (1.10)$$

In other words: to get a meaningful estimation with the kNN estimator, we need exponential data in d !

1.2 Grids in high-dimensions

As we will see in section 2, many of the classical approximation results in analysis involve approximating a function uniformly on a compact subset of $[0, 1]^d \subset \mathbb{R}^d$. A very common proof scheme consists of partitioning $[0, 1]^d$ in a uniform grid of constant size and approximating the target function by a piecewise constant function at each element of the grid. In dimension $d = 1$, it is easy to see we need $N = \lceil 1/\delta \rceil$ points to do it. In dimension 2, we need $N = \lceil 1/\delta \rceil^2$. More generally, in dimension d we will need $N = \lceil 1/\delta \rceil^d$ points. Therefore, the finer we want the partition to be, the more points we need, scaling exponentially in the dimension.

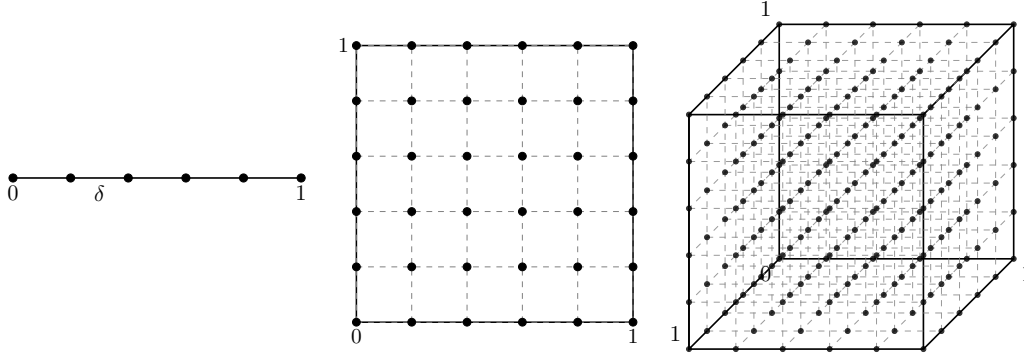


Figure 2: Partition of $[0, 1]^d$ in a uniform grid of size $\delta = 0.2$ for $d \in \{1, 2, 3\}$

1.3 The empirical covariance is not reliable

The curse of dimensionality also plagues many of the statistical procedures we are used. For instance, consider the problem of estimating the covariance of data $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^d$ samples i.i.d. from a distribution. The maximum likelihood estimator in this case is given by the empirical covariance matrix:

$$\hat{\Sigma}_n := \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i \mathbf{x}_i^\top \quad (1.11)$$

where we assumed for simplicity data is centred. If $d = O(1)$ is fixed, by the law of large numbers we have:

$$\hat{\Sigma}_n \xrightarrow{a.s.} \Sigma \text{ as } n \rightarrow \infty \quad (1.12)$$

where $\Sigma = \mathbb{E}[\mathbf{x}\mathbf{x}^\top]$ is the population covariance of the data. But what happens when d is also large? For instance, when $n = \Theta(d)$? As we will see in Lecture 4, in this limit the matrix $\hat{\Sigma}_n$ can have a very different behaviour. Let's consider a concrete example: take $\mathbf{x}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ i.i.d. and consider the question: how far are the eigenvalues of $\hat{\Sigma}_n$ from 1 (the eigenvalues of $\Sigma = \mathbf{I}_d$)? In eq. (3.3) we show a histogram of the eigenvalues of $\hat{\Sigma}_n$ when $d = 500$ and $n = 1000$. Even though the expected eigenvalue is one, most of the eigenvalues are closer to zero. This has important consequences for learning. For instance, if we consider the PCA problem where the goal is to estimate the directions of the data with largest variance, a naive look at the spectrum of $\hat{\Sigma}_n$ will suggest that most of directions have small variance, which can lead to the misleading conclusion that data is effectively low-dimensional — while the true data distribution is actually isotropic in \mathbb{R}^d .

1.4 Blessings of dimensionality

While learning in high-dimensional spaces certainly come with its challenges, it also comes with benefits, known as the *blessings of dimensionality* — a terminology coined by statistician David

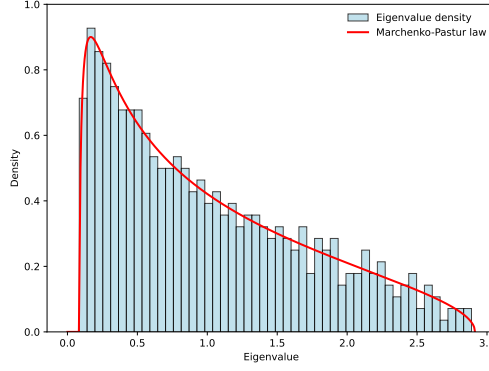


Figure 3: Histogram of eigenvalues of the empirical covariance matrix $\hat{\Sigma}_n$ of i.i.d. covariates $\mathbf{x}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ with $d = 500$ and $n = 1000$.

Donoho in his AMS math challenges lecture (Donoho et al., 2000). The most notable one is the phenomenon of *concentration of measure*, where the statistical fluctuations of some random quantities of interest get suppressed in high-dimensions.

The most well known example of this is the *thin-shell phenomena* for random Gaussian vector: the property that in dimension \mathbb{R}^d , random Gaussian points tightly cluster around the hypersphere.

Proposition 1 (Thin-shell). Let $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ denote a random Gaussian vector. Then, for every $\epsilon > 0$, there exists $C > 0$ such that:

$$\mathbb{P}\left((1 - \epsilon)\sqrt{d} \leq \|\mathbf{x}\|_2 \leq (1 + \epsilon)\sqrt{d}\right) \geq 1 - e^{-C(\epsilon)d} \quad (1.13)$$

Or in words: with high-probability \mathbf{x} is close to $\mathbb{S}^{d-1}(\sqrt{d})$.

The proof of this result follows from Bernstein's inequality applied to $\|\mathbf{x}\|_2 - d$, a sum of zero mean sub-exponential random variables. See fig. 4 for an illustration. This can have important consequences for computer science.

Theorem 1 (Johnson-Lenderstrauss lemma, 1984). Let $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^d$. Then, for any $i, j \in [n]$ and any desired precision $\epsilon > 0$, there exists a linear map $T : \mathbb{R}^d \rightarrow \mathbb{R}^k$ with $k \leq C \log n$ such that:

$$(1 - \epsilon)\|\mathbf{x}_i - \mathbf{x}_j\|_2 \leq \|T(\mathbf{x}_i) - T(\mathbf{x}_j)\|_2 \leq (1 + \epsilon)\|\mathbf{x}_i - \mathbf{x}_j\|_2 \quad (1.14)$$

In other words: T is an approximate isometry.

Although we will not discuss it here, the proof of this theorem is given by taking a Gaussian random mapping and using Gaussian concentration of the norm to show the isotropy property. We refer the interested reader to (Vershynin, 2018) for a proof. This result is striking, since it tell us that any learning algorithm with $d \gg n$ that depends only on the pairwise distances of the data (a.k.a. data *Gram matrix*) can be substantially reduced in computational cost. For instance, if $n = 10^3$ and $d = 10^4$, we have $k \sim 10$!

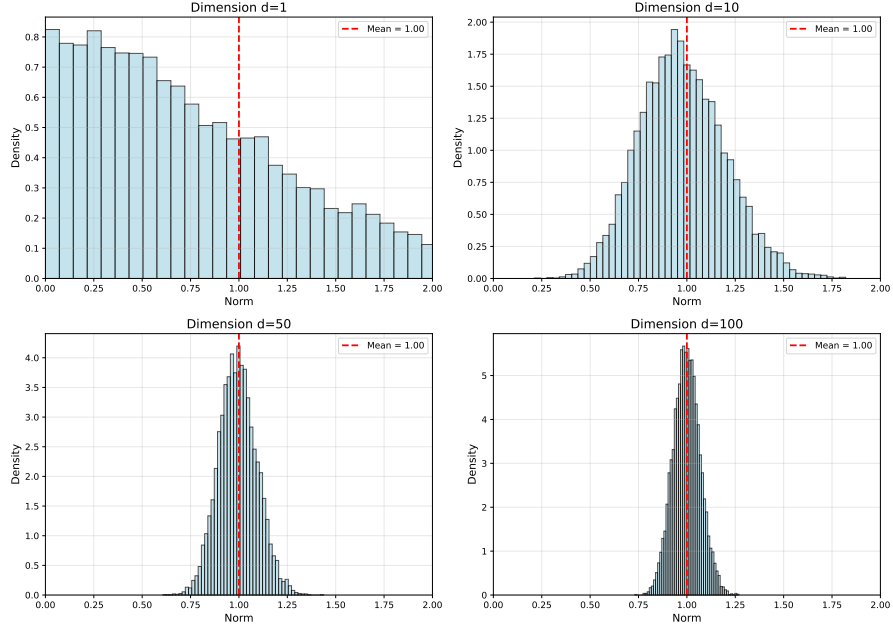


Figure 4: Histogram of the norm $\|\mathbf{x}_i\|_2^2$ of $i \in [10^3]$ Gaussian vectors $\mathbf{x}_i \sim \mathcal{N}(\mathbf{0}, 1/d \mathbf{I}_d)$ for increasing $d \in \{1, 10, 50, 100\}$.

2 Universal approximation of neural networks

2.1 Motivation & Introduction

Last lecture we have seen that we can decompose the excess risk in two terms:

$$R(\hat{\theta}) - R_{\star} = \underbrace{\left\{ R(\hat{\theta}) - \inf_{\theta \in \Theta} R(\theta) \right\}}_{\text{estimation}} + \underbrace{\left\{ \inf_{\theta \in \Theta} R(\theta) - R_{\star} \right\}}_{\text{approximation}}. \quad (2.1)$$

Our focus in today's lecture is about the second term, the *approximation error*, which quantifies the capacity of the hypothesis $\mathcal{H} = \{f_{\theta} : \mathbb{R}^d \rightarrow \mathcal{Y} : \theta \in \Theta\}$ in expressing the Bayes predictor. This term is only a function of the choice of architecture and the underlying data distribution, and is independent of the training data. It can also be thought as the irreducible error in the best case scenario where we have infinite amount of data and are able to perfectly optimise the risk.

2.2 Introduction to approximation theory

An approximation problem is composed of two ingredients:

1. A target class of functions one wishes to approximate.
2. A metric that defines how good the approximation is.

In eq. (2.1), we wish to approximate the Bayes predictor f_{\star} . However, the Bayes predictor is implicitly a function of the data distribution and the loss function, and its properties will be problem dependent. Instead of focusing of a particular case, classical approximation theory take as a gold standard the class of continuous functions \mathcal{C} , which is often rich enough for machine learning.

The second point depends on the loss function. For instance, a regression problem with the squared-loss will have low excess risk if:

$$\|f_{\star} - f_{\theta}\|_{L^2(\mu)} := \mathbb{E}_{\mathbf{x}}[(f_{\star}(\mathbf{x}) - f_{\theta}(\mathbf{x}))^2] \quad (2.2)$$


is small, where we denoted by μ is the marginal distribution over the covariates $\mathbf{x} \in \mathbb{R}^d$. Therefore, the $L^2(\mu)$ error a good enough measure when one wants to benchmark the generalisation error of a network trained by ERM with the square loss, for example. In other cases, other norms might be more adapted. Consider for instance the problem of binary classification with a L-Lipschitz margin-based loss function $\ell(yf_\theta(x))$ (e.g. logistic or hinge). Then:

$$\mathbb{E}[\ell(yf_\theta(\mathbf{x})) - \ell(yf_\star(\mathbf{x}))] \leq L \mathbb{E}[|f_\theta(x) - f_\star(x)|] := \|f_\theta - f_\star\|_{L_1(\mu)} \quad (2.3)$$

which is the $L_1(\mu)$ norm. This is a weaker notion than before, since $L^2(\mu) \subset L^1(\mu)$. More generally, we have that $\|\cdot\|_p$ is an increasing function of $p \in [1, \infty]$ (Exercise 1), and therefore the norm giving the strongest guarantees is $L^\infty(\mu)$, also known in this context as the *uniform norm*. More frequently, results are proven in terms of the uniform norm over a compact set $K \subset \mathbb{R}^d$:

$$\|f\|_{L^\infty(K)} = \sup_{\mathbf{x} \in K} |f(\mathbf{x})| \quad (2.4)$$

for example, $K = [0, 1]^d$ — and this will be the focus of our discussion in this lecture.

 When $\text{supp}(\mu) \subset K$, we have $\|f\|_{L^\infty(\mu)} \geq \|f\|_{L^\infty(K)}$, but in general these are different notions (Exercise 2). Overall, focusing on $\|\cdot\|_{L^\infty(K)}$ allow to have results which are independent of the covariate distribution, but which are not adapted to a problem of interest. Indeed, as we will going to see later, these guarantees tend to be pessimistic.

Example 1 (Uniform vs. non-uniform). To get an intuition for uniform vs. non-uniform guarantees, let's consider a simple but instructive example on $[0, 1] \subset \mathbb{R}$. Let $g(x) = 0$ for all $x \in [0, 1]$ and define:

$$f_n(x) = \begin{cases} 1 & x \in [1/n, 1] \\ 0 & \text{otherwise} \end{cases} \quad (2.5)$$

Then, we have:

$$\|g - f_n\|_{L^1([0,1])} = \int_0^1 |f_n(x)| dx = \frac{1}{n} \quad (2.6)$$

and hence, for any $\epsilon > 0$, taking $n > \lceil 1/\epsilon \rceil$ we can approximate g in L^1 to precision ϵ . More generally, for any $p \geq 1$ we have:

$$\|g - f_n\|_{L^p([0,1])} = \left(\int_0^1 |f_n(x)|^p dx \right)^{1/p} = \frac{1}{n^{1/p}} \quad (2.7)$$

and therefore getting a L^p guarantee to precision ϵ requires larger $n > O(\epsilon^{-p})$. Indeed, in uniform norm:

$$\|g - f_n\|_{L^\infty([0,1])} = 1 \quad (2.8)$$

for all $n \in \mathbb{N}$.

This motivates the following definition:

Definition 1 (Universal approximator). Let $K \subset \mathbb{R}^d$ denote a compact subset. We say a class of functions \mathcal{H} is a *universal approximator* over K if for any continuous function $g \in \mathcal{C}(K)$ and any $\epsilon > 0$, there exists $f \in \mathcal{H}$ such that:

$$|f(\mathbf{x}) - g(\mathbf{x})| \leq \epsilon, \quad \forall \mathbf{x} \in K \quad (2.9)$$

Remark 1. Two comments are in order:

- Equation (2.9) can be equivalently rewritten to to:

$$\sup_{g \in \mathcal{C}(K)} \inf_{f \in \mathcal{H}} \|f - g\|_{L^\infty(K)} < \epsilon \quad (2.10)$$

- Most of the proofs that follow will consider $K = [0, 1]^d$. This is without loss of generality since we can always cover a compact by a cube and then apply a rescaling.
- Restricting to a compact set K is crucial though. See exercise 4 for one example.

You might remember from your analysis course the following result:

Theorem 2 (Weierstrass theorem, 1885). Polynomials with unbounded degree are universal approximators, i.e. for every continuous real-valued function $f : [a, b] \rightarrow \mathbb{R}$ and for every $\epsilon > 0$, there exists a polynomial p such that:

$$\sup_{x \in [a, b]} |f(x) - p(x)| \leq \epsilon \quad (2.11)$$

Remark 2. Sometimes, this is also stated as “polynomials are dense in $\mathcal{C}(K)$ ”.

Actually, polynomials are just a particular case of one of the central results in approximation theory:

Theorem 3 (Stone-Weierstrass). Let $K \subset \mathbb{R}^d$ denote a compact subset and \mathcal{H} a class of functions $f : K \rightarrow \mathbb{R}$ satisfying the following properties:

- **Continuity:** Every $f \in \mathcal{H}$ is continuous, i.e. $\mathcal{H} \subset \mathcal{C}(K)$.
- **Non-zero element:** For all $x \in K$, there exists $f \in \mathcal{H}$ such that $f(x) \neq 0$. This is sometimes also stated as \mathcal{H} containing the constant functions.
- **Separability:** For all $x, x' \in K$, there exists a $f \in \mathcal{H}$ such that $f(x) \neq f(x')$.
- **Closure:** \mathcal{H} is a sub-algebra of $\mathcal{C}(K)$.¹

Then, \mathcal{H} is a universal approximator over K .

Remark 3 (Curse of dimensionality). We will not prove this result in the lectures. However, it is worth highlighting that a constructive proof of this result, due to Sergei Bernstein in 1912, involves approximating g by a suitable choice of polynomials (known as **Bernstein polynomials**) on a grid. Therefore, as a consequence of the discussion in Section 1.2, results derived as a consequence of the Stone-Weierstrass theorem 3 typically suffer from the curse of dimensionality, i.e. the number of grid points needed to cover $[0, 1]^d$ with intervals of size ϵ scale with $O(\epsilon^{-d})$.

It is easy to derive theorem 2 from the Stone-Weierstrass theorem by checking that the algebra generated by the coordinates x_1, \dots, x_d plus constants separates points. More generally, this provides a powerful way of proving that a given class of functions are universal approximators.

3 Neural networks

We now move to the main subject of this lecture: neural networks. A L -layer fully connected neural network is a parametric function of the type:

$$\mathbf{f}_\theta(\mathbf{x}) = \sigma_L(\mathbf{W}_L \sigma_{L-1}(\dots \mathbf{W}_2 \sigma_1(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2 \dots) + \mathbf{b}_L) \quad (3.1)$$

where:

¹In other words, it is closed under vector space operations and point-wise multiplication.

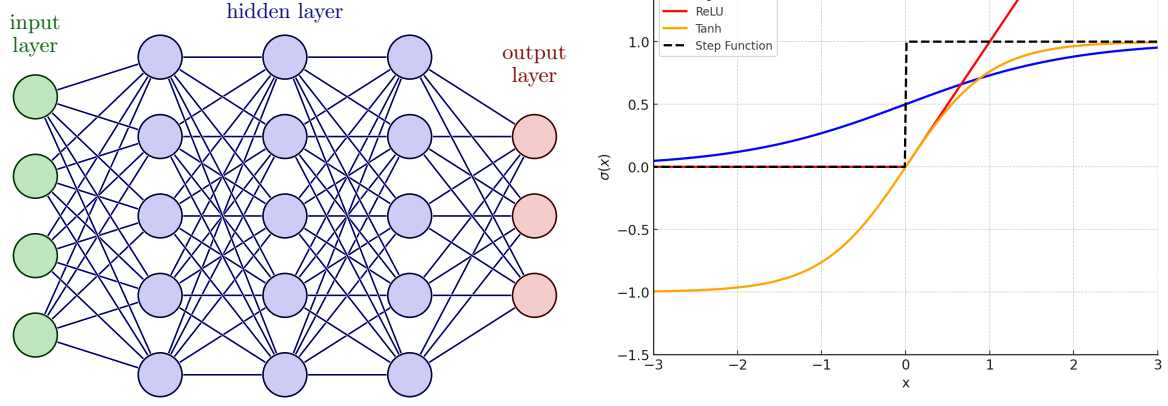


Figure 5: **(Left)** Fully connected neural network of depth $L = 4$, hidden-layer widths $p_\ell = 5$ for $\ell \in [4]$ and $p_5 = 4$ in $d = 4$. **(Right)** Popular activation functions used in neural networks.

- The parameters $\mathbf{W}_\ell \in \mathbb{R}^{p_{\ell+1} \times p_\ell}$, $\ell \in [L]$ are known as the *weight matrices* and $\mathbf{b}_\ell \in \mathbb{R}^{p_\ell}$ as the *biases*,² and are the trainable parameters of the model $\theta = (\mathbf{W}_\ell, \mathbf{b}_\ell)_{\ell \in [L]}$.
- The non-linear functions $\sigma_\ell : \mathbb{R} \rightarrow \mathbb{R}$ are known as the activation functions. Some of the most common examples, which act component-wise on vectors, are illustrated in fig. 5 (right). More general examples with are vector valued are the softmax and the max-pooling activations.
- L is known as the *depth* of the network, and $p_{\ell+1}$ as the *width* of the layer $\ell \in [L]$.
- The first and last layers are known as the *input* and *output* layers, while the middle layers are known as the *hidden layers*.

See fig. 5 (left) for an illustration.

⚠ Fully connected networks are one of many classes of neural network architectures, such as convolutional networks, U -networks, transformers, etc. In this lecture, we focus on the fully connected case for simplicity.

A particular case of interest in these lectures will be *two-layer neural networks* ($L = 2$) with real-valued outputs:

$$f_\theta(\mathbf{x}) = \langle \mathbf{a}, \sigma(\mathbf{W}\mathbf{x}) \rangle = \sum_{j=1}^p a_j \sigma(\langle \mathbf{w}_j, \mathbf{x} + \mathbf{b}_j \rangle). \quad (3.2)$$

where for convenience we relabelled $\mathbf{a} := \mathbf{W}_2 \in \mathbb{R}^p$ to stress that the last-layer is a vector. Since it will play an important role in what follows, we will denote the class of two-layer neural networks over \mathbb{R}^d with activation function σ as:

$$\mathcal{F}_{\sigma,d,p} := \left\{ f_\theta : \mathbb{R}^d \rightarrow \mathbb{R} : f_\theta(\mathbf{x}) = \sum_{j=1}^p a_j \sigma(\langle \mathbf{w}_j, \mathbf{x} \rangle + b_j) \right\} \quad (3.3)$$

$$\mathcal{F}_{\sigma,d} := \bigcup_{p=0}^{\infty} \mathcal{F}_{\sigma,d,p} \quad (3.4)$$

Our goal in what follows is to study the approximation properties of fully-connected networks. On a high-level, we will show that two-layer neural networks with unbounded width are universal approximators in the sense of definition 1. Ideally, we would like to show not only that it is possible to approximate a given class of functions with a neural networks, but also to estimate how many neurons are needed. However, such *quantitative* results are harder to show.

²The name “bias” is quite misleading, and should in no way be mistaken for the standard notion of bias in statistics.

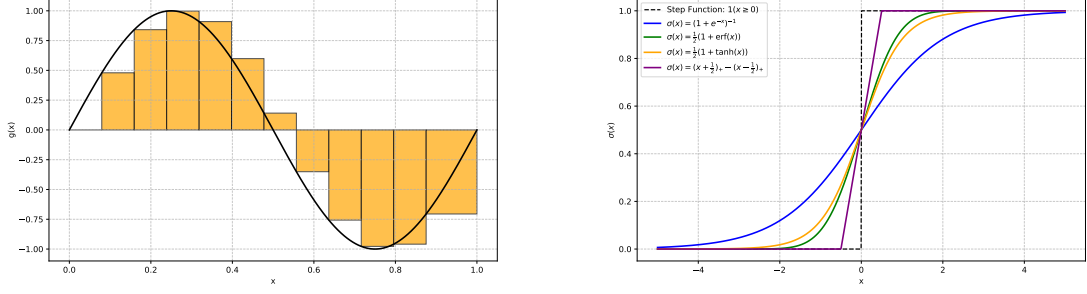


Figure 6: **(Left)** Approximating $g(x) = \sin(2\pi x)$ with a two-layer neural network with step-function activation to precision $\epsilon = 0.5$. The function g is 2π -Lipschitz in $[0, 1]$, and hence we have $p = 13$. **(Right)** Different activation functions which are close to the step-function $\mathbf{1}(x \geq 0)$.

3.1 Uniform results on \mathbb{R}

We start our discussion by considering $d = 1$.

Proposition 2. Any L -Lipschitz function on $[0, 1]$ can be ϵ -approximated by a two-layer neural network of width $\lceil L/\epsilon \rceil$ and step-function activation $\sigma(x) = \mathbf{1}_{x \geq 0}(x)$.

Proof. First, note that we can create a top-hat function of height 1 in an interval $[a, b]$ by combining two step-functions:

$$\mathbf{1}(x \geq a) - \mathbf{1}(x \geq b) = \mathbf{1}(x \in [a, b]) \quad (3.5)$$

Pictorially, it is easy to see that we can approximate any regular g with top-hat functions, see fig. 6 (left). The proof essentially follows this intuition. The idea is to partition $[0, 1]$ in p intervals of equal size δx and approximate g in each of these intervals by a top-hat function with height given by how much the function varies. The only tricky part is figuring out the good interval size to obtain the desired precision, which should depend on the regularity of the function.

Consider a L -Lipschitz function $g : [0, 1] \rightarrow \mathbb{R}$. Given a precision $\epsilon > 0$, let $p = \lceil L/\epsilon \rceil$, and define the two-layer neural network of width p :

$$f_{\theta}(x) = \sum_{j=0}^{p-1} a_j \mathbf{1}(x - b_j \geq 0) \quad (3.6)$$

where:

$$b_j = \frac{j\epsilon}{L}, \quad j \in \{0, 1, \dots, p-1\} \quad (3.7)$$

$$a_j = \begin{cases} g(0) & j = 1 \\ g(b_j) - g(b_{j-1}) & j > 1. \end{cases} \quad (3.8)$$

From eq. (3.5), this is equivalent to:

$$f_{\theta}(x) = \sum_{j=0}^{p-1} g(b_j) \mathbf{1}(x \in [b_j, b_{j+1}]) \quad (3.9)$$

It is easy to check that this does the job. Indeed, for any $x \in [0, 1]$, define:

$$k = \max\{j \in \{0, 1, \dots, p-1\} : b_j \leq x\} \quad (3.10)$$

such that $x \in [b_k, b_{k+1}]$ and $f(x) = f(b_k)$ is constant in this interval. Therefore:

$$\begin{aligned}
|g(x) - f(x)| &= |g(x) - g(b_k) + g(b_k) - f_\theta(b_k)| \\
&\leq |g(x) - g(b_k)| + |g(b_k) - f_\theta(b_k)| + |f_\theta(b_k) - f_\theta(x)| \\
&\leq L|x - b_k| + \left| g(b_k) - \sum_{j=0}^k a_j \right| + 0 \\
&\leq L \frac{\epsilon}{L} + \left| g(b_k) - g(b_0) - \sum_{j=1}^k (g(b_j) - g(b_{j-1})) \right| \\
&\leq \epsilon
\end{aligned} \tag{3.11}$$

Since the above was for arbitrary $x \in [0, 1]$, it implies the uniform bound. \square

Remark 4. The construction in the proof of proposition 2 is suboptimal, since the grid size is adapted only to the global regularity of g in $[0, 1]$, not to its local regularity. Indeed, we pay a high price for very regular regions of g , such as flat regions. See exercise 3 for a more detailed discussion of this point.

Since sigmoid-like activation functions such as $\sigma(x) = (1 + e^{-x})^{-1}$, $\sigma(x) = 1/2(1 + \operatorname{erf}(x))$, $\sigma(x) = 1/2(1 + \tanh(x))$ or even the sum of relu activations $\sigma(x) = \operatorname{relu}(x + \delta) - \operatorname{relu}(x - \delta)$ (see fig. 6 (right) for an illustration), it is intuitive that eq. (3.5) can be generalised to those cases.

Proposition 3. Every L -Lipschitz function $g : \mathbb{R} \rightarrow \mathbb{R}$ can be ϵ -approximated by a two layer neural network of width $p = O(L/\epsilon)$ with sigmoid-like activation σ :

$$\lim_{x \rightarrow -\infty} \sigma(x) = 0, \quad \lim_{x \rightarrow \infty} \sigma(x) = 1 \tag{3.12}$$

Sketch of the proof. Note that to show this result, from eq. (3.5) it suffices to show that we can approximate the step function $\mathbf{1}(x \geq t)$ with sigmoid-like functions in L^∞ over an interval. The tricky part is that a single sigmoid will not do the job. Indeed, away from the discontinuity $x \neq t$ we can approximate $\mathbf{1}(x \geq t)$ by rescaling $\sigma_\alpha(x) = \sigma(\alpha(t - x))$ with sufficiently large α . However, at the discontinuity $x = t$, the sup norm will be constant. To approximate the jump itself, we have to take two sigmoids with a small shift $\delta > 0$:

$$\phi_{\alpha, \delta}(x) = \sigma(\alpha(t - (x - \delta/2))) - \sigma(\alpha(x - (t + \delta/2))) \tag{3.13}$$

For $x \notin [t - \delta/2, t + \delta/2]$ and large α , the two terms are equal so the difference is zero, while for $x \in [t - \delta/2, t + \delta/2]$ the difference is close to 1 at large α . This is an approximation to the box function in an interval, and allow us to provide localised increments, see fig. 7 for an illustration. \square

3.2 Uniform results on \mathbb{R}^d

We now move to \mathbb{R}^d with $d > 1$. Note that the proof of proposition 2 can be decomposed in two steps: first, we showed that two-layer neural networks with step-function activation can express the top-hat function in an interval, i.e. eq. (3.5). Then, we showed that we can approximate any continuous function by a combination of piece-wise constant functions, which is precisely a linear combination of top-hats. The second part of this construction is easy to generalise to $d > 1$.

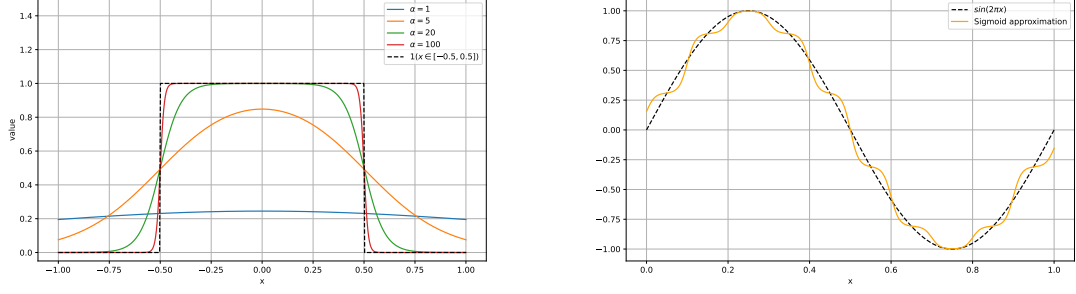


Figure 7: **(Left)** Approximating the box function $1(x \in [-1/2, 1/2])$ with the difference of two scaled sigmoids. **(Right)** Approximating the sign function $\sin(2\pi x)$ with a sum of $p = 20$ sigmoid functions.

Proposition 4. Let $g : \mathbb{R}^d \rightarrow \mathbb{R}$ denote a continuous function and $\epsilon > 0$ a desired precision. Then, there exists a partition $\mathcal{P} = (R_1, \dots, R_N)$ of $[0, 1]^d$ into $N = \lceil \delta^{-d} \rceil$ rectangles with side length $\delta > 0$ and real numbers $a_j \in \mathbb{R}$, $j \in [N]$ such that:

$$\sup_{\mathbf{x} \in [0, 1]^d} \left| g(\mathbf{x}) - \sum_{j=1}^N a_j \mathbf{1}(\mathbf{x} \in R_j) \right| < \epsilon \quad (3.14)$$

In other words: linear combinations of indicator functions are universal approximators over $[0, 1]^d$.

Proof. Let $\mathcal{P} = (R_1, \dots, R_N)$ of $[0, 1]^d$ denote a partition of $[0, 1]^d$ into N rectangles with side length smaller than $\delta > 0$. Note that since each rectangle has volume δ^d , this requires $N = \lceil \delta^{-d} \rceil$ rectangles. What we need to show is that we can choose δ and a_j such that eq. (3.14) holds. For every rectangle R_j , take an arbitrary point $\mathbf{x}_j \in R_j$ and let:

$$a_j = g(\mathbf{x}_j) \quad (3.15)$$

be the corresponding height of the top-hat function. Then:

$$\sup_{\mathbf{x} \in [0, 1]^d} |g(\mathbf{x}) - f(\mathbf{x})| = \sup_{j \in [N]} \sup_{\mathbf{x} \in R_j} |f(\mathbf{x}) - g(\mathbf{x})| \quad (3.16)$$

$$\leq \sup_{j \in [N]} \sup_{\mathbf{x} \in R_j} \left\{ |g(\mathbf{x}) - g(\mathbf{x}_j)| + \underbrace{|g(\mathbf{x}_j) - f(\mathbf{x})|}_{=0} \right\} \quad (3.17)$$

$$= \sup_{j \in [N]} \sup_{\mathbf{x} \in R_j} |g(\mathbf{x}) - g(\mathbf{x}_j)| \quad (3.18)$$

Since g is continuous on \mathbb{R}^d , it is uniformly continuous on the compact set $[0, 1]^d$, and therefore for every $\epsilon > 0$, there exists $\delta > 0$ such that:

$$\|\mathbf{x} - \mathbf{x}'\|_\infty \leq \delta \quad \Rightarrow \quad |g(\mathbf{x}) - g(\mathbf{x}')| \leq \epsilon \quad (3.19)$$

Therefore, choosing the size of our rectangle to be such δ , we have:

$$\sup_{\mathbf{x} \in [0, 1]^d} |g(\mathbf{x}) - f(\mathbf{x})| \leq \epsilon. \quad (3.20)$$

□

Remark 5 (Curse of dimensionality, again). As we had already seen in Section 1.2, the number of indicator function required to approximate g scale exponentially in the dimension $N \sim \delta^{-d}$, which is an instance of the curse of dimensionality.

To establish a similar result to proposition 2, we now need to approximate the indicator function over the rectangles $\mathbf{1}(R_j)$ with a neural network. The first proof of this result in the two-layer case is due to Cybenko (1989) with sigmoid functions and a slightly different argument. Here, we will present the proof by Hornik et al. (1989) which is based on the Stone-Weierstrass theorem 3.

Theorem 4 (Hornik, Stinchcombe, White 1989). The class of two-layer neural networks with unbounded width and sigmoid-like activation function σ :

- σ is increasing.
- $\lim_{x \rightarrow -\infty} \sigma(x) = 0$ and $\lim_{x \rightarrow \infty} \sigma(x) = 1$.

Are universal approximators over $[0, 1]^d$. In other words, for any continuous function $g : \mathbb{R}^d \rightarrow \mathbb{R}$, there exists a two-layer neural network $f \in \mathcal{F}_{\sigma, d}$ such that for any $\epsilon > 0$:

$$\sup_{\mathbf{x} \in [0, 1]^d} |g(\mathbf{x}) - f(\mathbf{x})| \leq \epsilon \quad (3.21)$$

The proof of theorem 4 proceeds in two steps: first, we will use the Stone-Weierstrass theorem to prove that two-layer networks with $\sigma(x) = \cos(x)$ are universal approximators. Then, we will prove that we can approximate the cosine with any activation satisfying the properties above.

Lemma 1. Two layer neural networks of unbounded width with cosine activation are universal approximators.

Proof. In order to apply the Stone-Weierstrass theorem, we need to check that the class of two-layer networks:

$$\mathcal{F}_{\cos, d} = \bigcup_{p \geq 1} \left\{ f_{\theta}(x) = \sum_{j=1}^p a_j \cos(\langle \mathbf{w}_j, \mathbf{x} \rangle + b_j) \right\} \quad (3.22)$$

satisfy the conditions of theorem 3:

- **Contituity:** Elements of $\mathcal{F}_{\cos, d, p}$ are continuous since they are linear combinations of continuous functions ($\cos(x)$).
- **Non-zero element:** For all $\mathbf{x} \in [0, 1]^d$, $1 \in \mathcal{F}_{\cos, d}$ since $\cos(\langle \mathbf{0}, \mathbf{x} \rangle) = 1$.
- **Separability:** For any $\mathbf{x}, \mathbf{x}' \in [0, 1]^d$, there exists a $\mathbf{w} \in \mathbb{R}^d$ such that $\cos(\langle \mathbf{w}, \mathbf{x} \rangle) \neq \cos(\langle \mathbf{w}, \mathbf{x}' \rangle)$. Take for instance:

$$f(\mathbf{z}) = \cos \left(\frac{\langle \mathbf{z} - \mathbf{x}', (\mathbf{x} - \mathbf{x}') \rangle}{\|\mathbf{x} - \mathbf{x}'\|_2^2} \right) \quad (3.23)$$

which satisfies $f(\mathbf{x}) = 1$ and $f(\mathbf{x}') = 0$.

- **Closure:** $\mathcal{F}_{\cos, d}$ is clearly closed under addition and multiplication by a real. It is also closed under multiplication since:

$$\cos(x) \cos(y) = \frac{1}{2}(\cos(x + y) + \cos(x - y)) \quad (3.24)$$

Therefore, for any $f, h \in \mathcal{F}_{\cos, d}$:

$$\begin{aligned}
f(\mathbf{x})h(\mathbf{x}) &= \left(\sum_{j=1}^p a_j \cos(\langle \mathbf{w}_j, \mathbf{x} \rangle + b_j) \right) \left(\sum_{k=1}^{p'} c_k \cos(\langle \mathbf{u}_k, \mathbf{x} \rangle + d_k) \right) \\
&= \sum_{j=1}^p \sum_{k=1}^{p'} a_j c_k \cos(\langle \mathbf{w}_j, \mathbf{x} \rangle + b_j) \cos(\langle \mathbf{u}_k, \mathbf{x} \rangle + d_k) \\
&= \frac{1}{2} \sum_{j=1}^p \sum_{k=1}^{p'} a_j c_k [\cos(\langle \mathbf{w}_j + \mathbf{u}_k, \mathbf{x} \rangle + b_j + d_k) + \cos(\langle \mathbf{w}_j - \mathbf{u}_k, \mathbf{x} \rangle + b_j - d_k)] \quad (3.25)
\end{aligned}$$

which implies $fh \in \mathcal{F}_{\cos, d}$.

Therefore, by the Stone-Weierstrass theorem 3, two-layer neural networks with cosine activation are universal approximators over $[0, 1]^d$. \square

Remark 6. As you will show in ??, the activation $\sigma(x) = \exp(x)$ also satisfy the conditions of the Stone-Weierstrass theorem.

We now move to the proof of the general case.

Sketch of the proof. Thanks to theorem 4, for any continuous function $g : \mathbb{R}^d \rightarrow \mathbb{R}$, there exists a two-layer neural network of width p with cosine activation:

$$f(\mathbf{x}) = \sum_{j=1}^p \tilde{a}_j \cos(\langle \tilde{\mathbf{w}}_j, \mathbf{x} \rangle + \tilde{b}_j) \quad (3.26)$$

such that:

$$\sup_{\mathbf{x} \in [0, 1]^d} |g(\mathbf{x}) - f(\mathbf{x})| \leq \frac{\epsilon}{2} \quad (3.27)$$

Therefore, our goal is to approximate $f(\mathbf{x})$ by a two-layer neural network with sigmoid-like activation. First, note that for any $x \in [0, 1]^d$, the pre-activations are bounded:

$$t_j = \langle \tilde{\mathbf{w}}_j, \mathbf{x} \rangle + \tilde{b}_j \in [|\tilde{b}_j|, |\langle \tilde{\mathbf{w}}_j, \mathbf{1} \rangle| + |\tilde{b}_j|] \quad (3.28)$$

Therefore, we can focus on the univariate problem of approximating $\cos(x)$ on a compact interval with a sigmoid two-layer neural network, which we saw can be done with proposition 3. \square

Remark 7. The most general universal approximation result for two-layer neural networks, proven by Leshno et al. (1993), shows that $\mathcal{F}_{\sigma, d}$ with σ a locally bounded piece-wise continuous activation is a universal approximator if and only if σ is not a polynomial.

4 Exercises

Exercise 1. Let μ denote a probability measure over \mathbb{R}^d . Show that the $L^p(\mu)$ norm:

$$\|f\|_{L^p(\mu)} := \left(\int \mu(d\mathbf{x}) |f(\mathbf{x})|^p \right)^{1/p} \quad (4.1)$$

is an increasing function of $p \in [1, \infty]$:

$$\|f\|_{L^\infty(\mu)} \geq \dots \geq \|f\|_{L^2(\mu)} \geq \|f\|_{L^1(\mu)} \quad (4.2)$$

where we recall $\|f\|_{L^\infty(\mu)} = \sup_{\mathbf{x} \in \text{supp}(\mu)} |f(\mathbf{x})|$. Conclude that we have the inclusion:

$$L^\infty(\mu) \subset \dots \subset L^2(\mu) \subset L^1(\mu) \quad (4.3)$$

Exercise 2. Give an example of a probability measure μ on \mathbb{R}^d and compact set $K \subset \mathbb{R}^d$ such that:

$$\|f\|_{L^\infty(\mu)} \leq \|f\|_{L^\infty(K)} \quad (4.4)$$

Exercise 3. Consider the following continuous function on $[0, 1]$:

$$g(x) = \begin{cases} 0 & x < 1/2 \\ 2x & x \in [1/2, 1] \\ 1 & x > 1 \end{cases} \quad (4.5)$$

How many neurons p are needed to approximate g within a precision $\epsilon > 0$ using the two-layer neural network with step-size activation from proposition 2? Show that we could do as well by using less neurons if we adapt the partition to the function.

Exercise 4. Show that:

$$\inf_{f_\theta \in \mathcal{F}_{\text{relu},1}} \sup_{x \in \mathbb{R}} |f_\theta(x) - \sin(x)| \geq 1 \quad (4.6)$$

where $\mathcal{F}_{\text{relu},1}$, the class of two-layer neural networks over \mathbb{R} with relu activation and unbounded width, is was defined in eq. (3.3). Conclude that the compactness assumption in definition 1 is crucial to define meaningful approximation results.

References

- George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.
- David L Donoho et al. High-dimensional data analysis: The curses and blessings of dimensionality. *AMS math challenges lecture*, 1(2000):32, 2000.
- Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.
- Moshe Leshno, Vladimir Ya Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867, 1993.
- Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.