MATH 6397 - Mathematics of Data Science

Instructor: Demetrio Labate

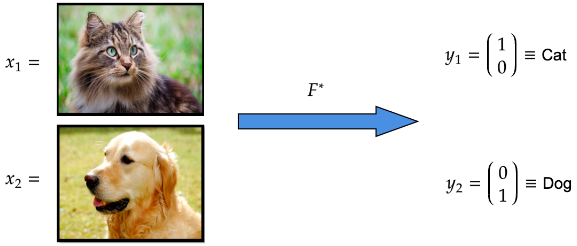
February 8, 2023

Part II

Mathematics of Data Science

Supervised Learning

In the classical **supervised learning** problem, the main objective is to write an algorithm that is able to learn a pattern from a series of examples, as a human being might learn it.


$x_1 =$
$x_2 =$
F^*
$y_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \equiv$ Cat
$y_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \equiv$ Dog

The specific goal is to acquire the ability to identify the class to which each element of a set belongs from a finite number of possibilities (**classification problem**), e.g., cats vs dog, or to assign each element to a value from a continuous range (**regression problem**).

Supervised Learning

More formally, the problem is to approximate an **unknown function** F^* that maps any element (input vector) x of an input space X either (classification task) to its corresponding label y from a finite dimensional output space Y or (regression) to a continuous range of values $y \in Y$.

For simplicity, I will focus now on the classification task.

In this situation, we have access to a set of points, the **training set**:

$$D = \{(x_i, y_i) : i = 1, \dots, N\}, \quad \text{where } y_i = F^*(x_i).$$

The goal is to obtain a good **approximation** of F^* that can accurately predict the unknown label of any new input point.

Supervised Learning

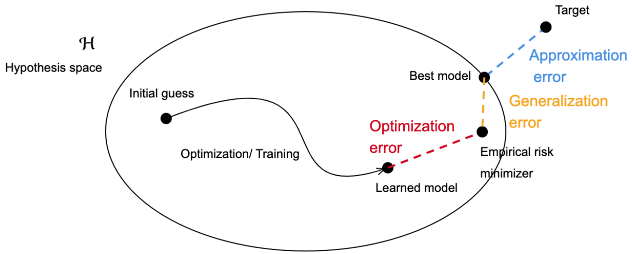
To process to construct a predictive model $\widehat{F}$ is a called the **learning process**.

Where shall we look for it?
We will limit the search for $\widehat{F}$ to a fixed parameterized class of functions called the **hypothesis space** H.

How can we select $\widehat{F}$ among all the functions in the hypothesis space?
Ideally, the best possible predictor in H_θ would be obtained through **population risk minimization**.

In practice, there is no prior knowledge of the population, so the best possible predictor is constructed through the so-called **empirical risk minimization.**

Supervised Learning



Approximation error: distance separating the true solution F^* from the hypothesis space H
Generalization error: due to the finiteness of the training set causing by the difference between population and empirical risk minimization.
Optimization error: distance from $\widehat{F}$ to the predictor that we obtain when we stop running the algorithm.

Supervised Learning and function spaces

Moral of the story:

- Supervised learning aims at learning a **function.** Usually a high-dimensional function.
- The choice of the hypothesis space is critical to ensure we can compute a satisfactory predictive (or regression) model.
- The algorithm we select to approximate functions in the hypothesis space must be numerically efficient.

Neural Networks

Neural Networks

A **neural network** is an algorithm for processing an input $x \in \mathbb{R}^D$ and returning an output in $\mathbb{R}^k$.

Unlike the classical transformations used in traditional signal processing which are defined by it linear superposition, neural networks are defined by **composition**.

The process alternates between two simple steps:

1. an **affine linear transformation**, that is, a map of the form

$$x \mapsto Ax + b$$

2. a **non-linear transformation** ρ applied coordinate-wise.

Neural Networks

A neural network processes information as follows:

1. Denote the **input** as
$$\hat{x}^0 = x$$
2. For $1 \leq \ell \leq L$, set
$$x^\ell = A^\ell \hat{x}^{\ell-1} + b^\ell, \quad \hat{x}^\ell = \rho(x_\ell)$$
3. The **output** is
$$\Phi(x) = x^L$$

The values x^ℓ, $1 \leq \ell \leq L-1$ which are not seen by a user are the **hidden layers** or **latent variables** of the neural network.

Remark

In practical applications, the weight matrices A^ℓ and the biases b^ℓ of the neural networks are **learned** during an appropriate training process. We do ignore the training process for now.

Neural Networks

A particularly simple case is a neural network with only one hidden layer. This is called a **shallow neural network**

In this case, we take a linear map of the input, apply a nonlinear transformation ρ once, and apply another linear map:

$$\Phi(x) = A^2(\rho(A^1 x + b^1)) + b^2$$

This can be written more compactly as

$$\Phi(x) = \sum_{i=1}^{n} a_i\, \rho(w_i^t x + b_i) + c$$

The bias c in the last layer is often omitted.

Neural Networks

More generally, a neural networks may have multiple layers, including several inner layers

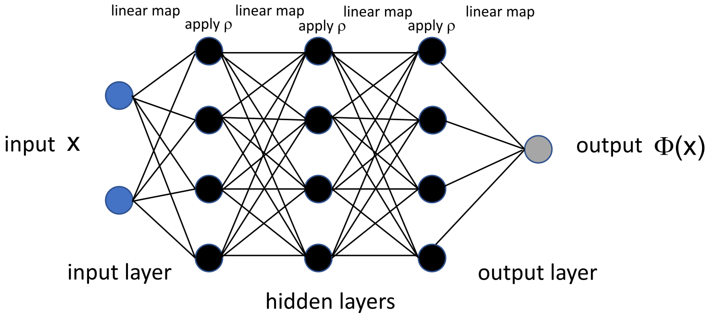
$$\Phi(x) = A^4\rho(A^3\rho(A^2\rho(A^1x + b^1) + b^2) + b^3) + b^4$$

Here we have a neural network with 4 layers.

Clearly, for the function to be well defined, the dimensions of $x \in \mathbb{R}^D$, the vectors b^i and the matrices A^i must be matched.

Function classes of Neural Networks

Graphical representation of a neural network with input $x \in \mathbb{R}^2$, output $\Phi(x) \in \mathbb{R}$ and 4 layers ($L = 4$).



Function classes of Neural Networks

Neural networks produce **structured parametric families of functions** of the form

$$\Phi(x) = W^L \circ \rho \circ W^{L-1} \circ \ldots \rho \circ W^1(x), \quad x \in \mathbb{R}^D$$

Function classes of Neural Networks

Neural networks produce **structured parametric families of functions** of the form

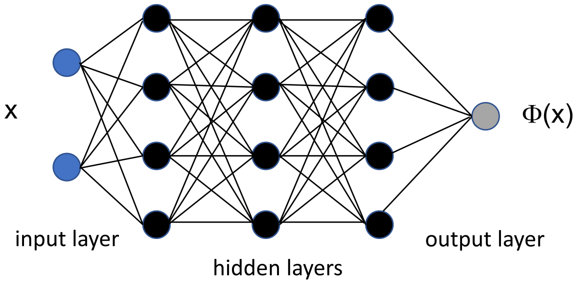
$$\Phi(x) = W^L \circ \rho \circ W^{L-1} \circ \ldots \rho \circ W^1(x), \quad x \in \mathbb{R}^D$$

where

- $W^\ell(x) = A^\ell x + b^\ell, \quad \ell = 1, \ldots, L$
- $A^\ell \in \mathbb{R}^{N_\ell \times N_{\ell-1}}$ are the **filters** and $b^\ell \in \mathbb{R}^{N_\ell}$ are the **biases**
- $\rho : \mathbb{R} \to \mathbb{R}$ is the **activation function**
- $L(\Phi)$ is the **number of layers** of Φ
- $N_\ell \in \mathbb{N}$, $i = \ell, \ldots, L$ is the **width** of the ℓ-th layer, $N_0 = D$, and $N(\Phi) = \sum_{i=0}^{L} N_i$ is the **number of neurons** of Φ
- $M(\Phi) = \sum_{\ell=1}^{L} ||A^\ell||_0 + ||b^\ell||_0$ is the **number of weights** (or **parameters**) of Φ

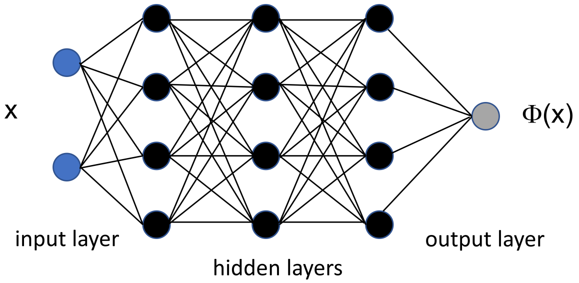
Function classes of Neural Networks

Graph representation



Function classes of Neural Networks

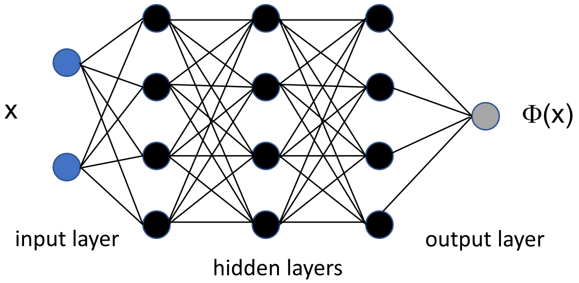
Graph representation



- Number of layers $L = 4$

Function classes of Neural Networks

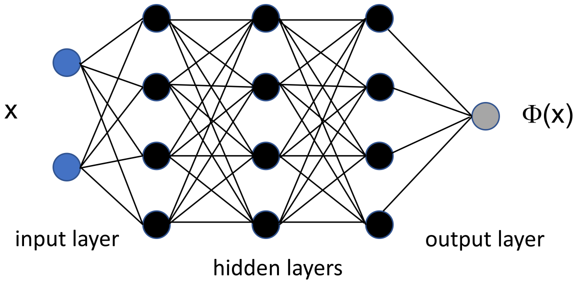
Graph representation



- Number of layers $L = 4$
- Number of neurons $N = 15$

Function classes of Neural Networks

Graph representation



- Number of layers $L = 4$
- Number of neurons $N = 15$
- Number of weights $M = \sum_{\ell=1}^{4} ||A^\ell||_0 + ||b^\ell||_0 = 44 + 12 = 56$

$$\Phi(x) = W^4(\rho W^3(\rho W^2(\rho W^1(x))))$$

Function classes of Neural Networks

Assumption: We do identify a neural network with the **function** implemented by the neural network

In fact, multiple graph representations may realize the same function. So, some authors distinguish a neural network from its realization.

Function classes of Neural Networks

Assumption: We do identify a neural network with the **function** implemented by the neural network

In fact, multiple graph representations may realize the same function. So, some authors distinguish a neural network from its realization.

Definition

For a tuple (M_0, L_0, B_0), where $M_0, L_0 \in \mathbb{N} \cup \{\infty\}$ and $B_0 > 0$, $\mathcal{F}(M_0, L_0, B_0)$ denotes the **function class** of neural networks with number of weights M_0, number of layers L_0 and with scale B_0:

$$\mathcal{F}(M_0, L_0, B_0)$$
$$= \left\{ \Phi \colon [0,1]^D \to \mathbb{R}^{N_L} \colon \; L(\Phi) \leq L_0, M(\Phi) \leq M_0, B(\Phi) \leq B_0 \right\}$$

where $B(\Phi) = \max_\ell \{||vec(A^\ell)||_\infty, ||b^\ell||_\infty\}$ is the **scale** of the weights of Φ

Function classes of Neural Networks

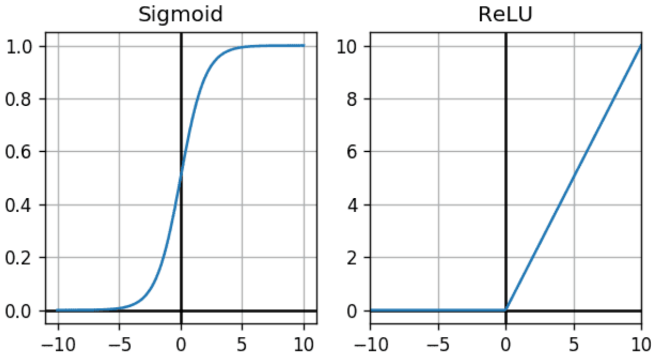
Examples of activation functions:

- **Sigmoid**: $\rho(x) = \frac{1}{1+e^{-x}}$
- **Rectified linear unit (ReLU)**: $\rho(x) = \max\{x, 0\}$

Function classes of Neural Networks

Examples of activation functions:

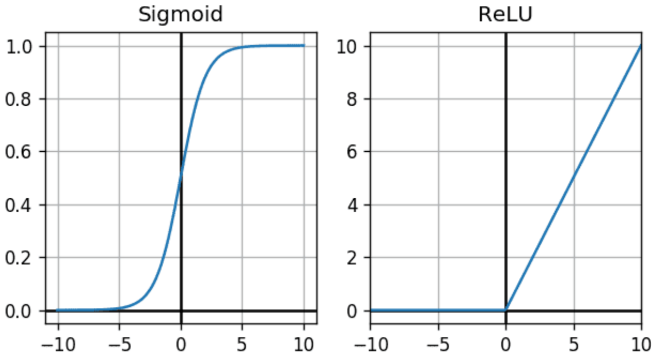
- **Sigmoid**: $\rho(x) = \frac{1}{1+e^{-x}}$
- **Rectified linear unit (ReLU)**: $\rho(x) = \max\{x, 0\}$



Function classes of Neural Networks

Examples of activation functions:

- **Sigmoid**: $\rho(x) = \frac{1}{1+e^{-x}}$
- **Rectified linear unit (ReLU)**: $\rho(x) = \max\{x, 0\}$



- ReLU is most commonly used in applications

Approximations using Neural Networks

Approximation theorems have been used to derive approximation estimates in neural networks and to quantify their expressive power.

Approximations using Neural Networks

Approximation theorems have been used to derive approximation estimates in neural networks and to quantify their expressive power.

We start with an **elementary construction** in the univariate case.

Approximations using Neural Networks

Approximation theorems have been used to derive approximation estimates in neural networks and to quantify their expressive power.

We start with an **elementary construction** in the univariate case.

Proposition

Assume $f : [0,1] \to \mathbb{R}$ is B-Lipschitz. For any $\epsilon > 0$ there is a 2-layer neural network Φ with threshold nonlinearities $z \to \chi_{[z \geq 0]}$ so that

$$\sup_{x \in [0,1]} |f(x) - \Phi(x)| < \epsilon.$$

This neural networks has $\lceil \frac{B}{\epsilon} \rceil$ weights.

Approximations using Neural Networks

Proof. Define $m = \lceil \frac{B}{\epsilon} \rceil$ and for $i = 0, \ldots, m-1$, set

$$a_0 = f(0), a_i = f(b_i) - f(b_{i-1}), \text{ with } b_i = i\epsilon/B$$

We next define

$$\Phi(x) = \sum_{i=0}^{m-1} a_i \, \chi_{[x_i \geq b_i](x)}$$

This is a neural network with $L = 2$ layers and m weights.

Approximations using Neural Networks

For any $x \in [0,1]$, letting k to be the largest index such that $b_k \leq x$, then f is constant on $[b_k, x]$.
Hence we have

$$\begin{aligned}
|f(x) - \Phi(x)| &\leq |f(x) - f(b_k)| + |f(b_k) - \Phi(b_k)| + |\Phi(b_k) - \Phi(x))| \\
&\leq B|x - b_k| + |f(b_k) - \sum_{i=0}^{k} a_i| + 0 \\
&\leq B\,\frac{\epsilon}{B} + |f(b_k) - a_0 - \sum_{i=1}^{k} a_i| \\
&= \epsilon + |f(b_k) - f(b_0) - \sum_{i=1}^{k}(f(b_i) - f(b_{i-1}))| \\
&= \epsilon
\end{aligned}$$

Approximations using Neural Networks

One can extend the constructive argument of the univariate case to the multivariate case. Even though it is more complicate now to localize the function properties, one can prove the following result.

Approximations using Neural Networks

One can extend the constructive argument of the univariate case to the multivariate case. Even though it is more complicate now to localize the function properties, one can prove the following result.

Proposition

Assume $f : [0,1]^D \to \mathbb{R}$ is B-Lipschitz. For any $\epsilon > 0$ there is a 3-layer neural network Φ with ReLU activation functions so that

$$\int_{x \in [0,1]^D} |f(x) - \Phi(x)| < \epsilon.$$

This neural networks has $O((\frac{B}{\epsilon})^D)$ weights.

Remark. This result suffers from the curse of dimension, as the number of weights scaled exponentially with D. In addition, the error is measured in the L^1 norm rather than uniformly.

Approximations using Neural Networks

One can extend the constructive argument of the univariate case to the multivariate case. Even though it is more complicate now to localize the function properties, one can prove the following result.

Proposition

Assume $f : [0,1]^D \to \mathbb{R}$ is B-Lipschitz. For any $\epsilon > 0$ there is a 3-layer neural network Φ with ReLU activation functions so that

$$\int_{x \in [0,1]^D} |f(x) - \Phi(x)| < \epsilon.$$

This neural networks has $O((\frac{B}{\epsilon})^D)$ weights.

Remark. This result suffers from the curse of dimension, as the number of weights scaled exponentially with D. In addition, the error is measured in the L^1 norm rather than uniformly.

Stronger results have been derived using non-constructive arguments.

Approximations using Neural Networks

Universal approximation theorems have historically been used as a justification of the expressive power of neural networks.

Definition.

A class of functions $\mathcal{F}$ is called a **universal approximator** over a compact set S if, for every continuous function g and target accuracy $\epsilon > 0$, there exists $f \in \mathcal{F}$ such that

$$\sup_{x \in S} |f(x) - g(x)| < \epsilon.$$

Remarks:

- Typically one chooses $S = [0, 1]^D$ since this set can be rescaled.
- Compactness is necessary. For instance, one can show that we need infinitely many ReLUs to approximate the "sin" function uniformly over $\mathbb{R}$.

Approximations using Neural Networks

Observation:

- The classical Weierstrass theorem (Weierstrass 1885) establishes that polynomials are universal approximators.
- The Stone-Weierstrass theorem extends the original Weierstrass theorem:

Theorem

Let $\mathcal{F}$ be a collection of functions such that

1. each $f \in \mathcal{F}$ is continuous;
2. for every x, there exist $f \in \mathcal{F}$ such that $f(x) \neq 0$;
3. for every $x \neq x'$, there exist $f \in \mathcal{F}$ such that $f(x) \neq f(x')$ ($\mathcal{F}$ separates points);
4. $\mathcal{F}$ is closed under multiplication and vector space operations ($\mathcal{F}$ is an algebra).

Then $\mathcal{F}$ is a universal approximator over $[0,1]^D$.

Note: The second and third conditions in the Stone-Weierstrass theorem are necessary.

Universal Approximation Theorems

Theorem [Cybenko, 1989]

Assume ρ is a sigmoidal activation function. Then the class of shallow neural networks $\mathcal{F}(M, L=2, B)$ is a universal approximator over $[0,1]^D$. That is, for every $f \in C([0,1]^D)$, given $\epsilon > 0$, there exists $\Phi \in \mathcal{F}(M, L=2, B)$, such that

$$|\Phi(x) - f(x)| < \epsilon, \qquad \text{for any } x \in [0,1]^D$$

Universal Approximation Theorems

Theorem [Cybenko, 1989]

Assume ρ is a sigmoidal activation function. Then the class of shallow neural networks $\mathcal{F}(M, L = 2, B)$ is a universal approximator over $[0,1]^D$. That is, for every $f \in C([0,1]^D)$, given $\epsilon > 0$, there exists $\Phi \in \mathcal{F}(M, L = 2, B)$, such that

$$|\Phi(x) - f(x)| < \epsilon, \qquad \text{for any } x \in [0,1]^D$$

Note: number of weights $M(\Phi)$ ($\Rightarrow$ number of neurons $N_2(\Phi)$) can become arbitrarily large.

Universal Approximation Theorems

Idea of the proof:

Universal Approximation Theorems

Idea of the proof:

- $\mathcal{F}(M, L = 2, B) \subset C([0, 1]^D)$ is a linear subspace

Universal Approximation Theorems

Idea of the proof:

- $\mathcal{F}(M, L = 2, B) \subset C([0,1]^D)$ is a linear subspace
- Arguing by contradiction, suppose $\mathcal{F}(M, L = 2, B)$ not dense

Universal Approximation Theorems

Idea of the proof:

- $\mathcal{F}(M, L = 2, B) \subset C([0,1]^D)$ is a linear subspace
- Arguing by contradiction, suppose $\mathcal{F}(M, L = 2, B)$ not dense
- By Hahn-Banach Theorem., there exists a signed Radon measure μ such that $\int_{[0,1]^D} \Phi(x)\, d\mu(x) = 0$ for all $\Phi \in \mathcal{F}(M, L = 2, B)$.

Universal Approximation Theorems

Idea of the proof:

- $\mathcal{F}(M, L = 2, B) \subset C([0,1]^D)$ is a linear subspace
- Arguing by contradiction, suppose $\mathcal{F}(M, L = 2, B)$ not dense
- By Hahn-Banach Theorem., there exists a signed Radon measure μ such that $\int_{[0,1]^D} \Phi(x)\, d\mu(x) = 0$ for all $\Phi \in \mathcal{F}(M, L = 2, B)$.
- The functions $\rho(ax + b)$ belong to $\Phi \in \mathcal{F}(M, L = 2, B)$ for any $a \in \mathbb{R}^D$, $b \in \mathbb{R}$.

Universal Approximation Theorems

Idea of the proof:

- $\mathcal{F}(M, L = 2, B) \subset C([0,1]^D)$ is a linear subspace
- Arguing by contradiction, suppose $\mathcal{F}(M, L = 2, B)$ not dense
- By Hahn-Banach Theorem., there exists a signed Radon measure μ such that $\int_{[0,1]^D} \Phi(x)\, d\mu(x) = 0$ for all $\Phi \in \mathcal{F}(M, L = 2, B)$.
- The functions $\rho(ax + b)$ belong to $\Phi \in \mathcal{F}(M, L = 2, B)$ for any $a \in \mathbb{R}^D$, $b \in \mathbb{R}$.
- Contradiction follows since ρ is **discriminatory**, i.e., the only Radom measure μ for which $\int_{[0,1]^D} \rho(ax + b)\, d\mu(x) = 0$, $a \in \mathbb{R}^D$, $b \in \mathbb{R}$, is the zero measure.

Universal Approximation Theorems

The universal approximation property extends to a much larger class of activation functions.

Theorem (Hornik, 1991)

Assume ρ is a C^∞ non-polynomial activation function. Then the class of shallow neural networks $\mathcal{F}(M, L = 2, B)$ is a universal approximator over $[0, 1]^D$.

Theorem (Pinkus, 1999)

Let ρ be a continuous activation function. The class of shallow neural networks $\mathcal{F}(M, L = 2, B)$ is a universal approximator over $[0, 1]^D$ if and only if ρ is non-polynomial.

Note. Also in this case, the number weights $M(\Phi)$ and number of neurons N_2 of the approximating network Φ can become arbitrarily large.

Universal Approximation Theorems

Sketch of the proof [Hornik, 1991]:

Universal Approximation Theorems

Sketch of the proof [Hornik, 1991]:

- Since ρ is a C^∞ non-polynomial activation function, for any $k \in \mathbb{N}_0 := \mathbb{N} \cup \{0\}$ there is x_k such that $\rho^{(k)}(x_k) \neq 0$.
- Recall that the derivative function can be expressed as the limit of difference quotients

  $$g^{(k)}(x) = \lim_{h \to 0} \frac{1}{h^k} \sum_{i=0}^{k} \binom{k}{i} (-1)^{k-i} g(x + ih)$$

  with uniform convergence.
- By the last observation, we can uniformly approximate the monomial

  $$x^k = \frac{1}{\rho^{(k)}(x_k)} \frac{d^k}{dt^k}\Big|_{t=0} \rho(x_k + tx)$$

  using a neural network $\Phi \in \mathcal{F}(M, L = 2, B)$.

Universal Approximation Theorems

Sketch of the proof (continued):

- Similarly, for any $v \in \mathbb{R}^D$, we can approximate the function $x \to \langle x, v \rangle^k$ using a neural network $\Phi \in \mathcal{F}(M, L = 2, B)$.
- It follows from the last two observations that, for any $(i_1, \ldots, i_D) \in \mathbb{N}_0^D$, we can uniformly approximate the monomial

$$x_1^{i_1} \ldots x_D^{i_D} = \frac{\partial^{i_1+\cdots+i_D}}{\partial t_1^{i_1} \ldots \partial t_D^{i_D}} (t_1 x_1 + \cdots + t_D x_D)^{i_1+\cdots+i_D}$$

  using a neural network $\Phi \in \mathcal{F}(M, L = 2, B)$.
- By linearity, we can approximate every polynomial using a neural network $\Phi \in \mathcal{F}(M, L = 2, B)$.
- The proof now follows by the Stone-Weierstrass Theorem.

Universal Approximation Theorems

The theorem we proved states that shallow neural networks (with non-polynomial activation function) can approximate exactly the same functions as polynomials.

We already know by of the Stone-Weierstrass theorem that polynomials are universal approximators.

The result is, in some sense, disappointing. If polynomials can approximate everything, why use neural networks? If shallow networks can approximate everything, why use deep networks?

We will argue further below that approximations by neural networks can be advantageous in terms of **computational complexity**.

Universal Approximation Theorems

As we observed in the constructive arguments, for a B-Lipschitz function f, we need a neural network with $M = \lceil \frac{B}{\epsilon} \rceil$ weights to approximate f with ϵ accuracy. Equivalently, there is a shallow network Φ such that, for some constant $C > 0$, we have

$$\max_{x \in [0,1]} |f(x) - \Phi(x)| \leq C\, M^{-1}$$

In dimension D, as suggested by the extension of the constructive approach and proved rigorously by [Mhaskar 1996], we can similarly find a shallow network Φ such that

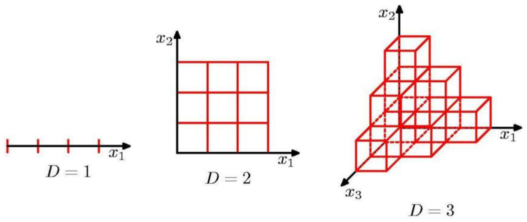
$$\max_{x \in [0,1]} |f(x) - \Phi(x)| \leq C\, M^{-1/D}.$$

Decay rate is increasingly slower with larger dimension D.

- Manifestation of the **curse of dimensionality.**
  cf. [Bellman R.E. *Adaptive Control Processes*. Princeton University Press, Princeton, NJ, 1961.]

Universal Approximation Theorems

The curse of dimensionality phenomenon appears in classical approximations



Number of cells needed to partition $[0, 1]^D$ grows exponentially with D.

Need ϵ^{-D} points to partition $[0, 1]^D$ with uniform ϵ-size grid. Equivalently, N points result in a grid with partition width $N^{-1/D}$

Barron spaces

Approximations with Neural Networks

In some cases, shallow networks can do better than classical approximation methods which are linear in the parameters.

Theorem [Maurey-Barron-Jones]

Let $\mathcal{H}$ be a pre-Hilbert space, $R, \epsilon > 0$, and $A \subset \mathcal{H}$ be such that $\|h\| \leq R$ for all $h \in A$. For any $m \in \mathbb{N}$ and any $f \in \overline{\text{conv}(A)}$, there exist m triples $(a_i, w_i, b_i) \in \mathbb{C} \times \mathbb{R}^D \times \mathbb{R}$ such that

$$\|f - \frac{1}{m}\sum_{i=1}^{m} a_i\, \rho(w_i^t x + b_i)\|_{L^2} \leq \frac{R+\epsilon}{\sqrt{m}} = (R+\epsilon)\, m^{-1/2}.$$

By contrast, if V_m is an m-dimensional linear function space

$$\sup\{\min_{v \in V_m} \|f - v\|_{L^2} : f \in \overline{\text{conv}(A)}\} \geq \frac{e}{4D} m^{-1/D}.$$

That is, the approximation rate of $f \in \overline{\text{conv}(A)}$ using a shallow network is independent of the dimension D.

Approximations with Neural Networks

The proof of first statement follows from the following lemma.

Lemma [Maurey-Barron-Jones]. Let $\mathcal{H}$ be a pre-Hilbert space, $R, \epsilon > 0$, and $A \subset \mathcal{H}$ be such that $\|h\| \leq R$ for all $h \in A$. For any $m \in \mathbb{N}$, if $g \in \overline{\text{conv}(A)}$ there exist $h_1, \dots, h_m \in A$ such that

$$\|g - \frac{1}{m}\sum_{i=1}^{m} h_i\| \leq \frac{R}{\sqrt{m}} + \epsilon.$$

Proof. By the hypothesis on g, given $\epsilon > 0$, there exists $M = M(\epsilon)$ points $h_1, \dots, h_M \in A$ and $\lambda_1, \dots, \lambda_M \in [0, 1]$ such that

$$\|h - \sum_{i=1}^{M} \lambda_i h_i\| \leq \epsilon, \quad \sum_{i=1}^{M} \lambda_i = 1.$$

Set $\tilde{h} := \sum_{i=1}^{M} \lambda_i h_i$ and choose m indices from $\{1, \dots, M\}$ independently according to the distribution for which $p(i) = \lambda_i$.

Approximations with Neural Networks

Then, using the observations that $\tilde{h} = E_\alpha h_{i(\alpha)}$ and that $i(\alpha)$ and $i(\beta)$ are independent for $\alpha \neq \beta$, we have that

$$\begin{aligned}
E_\alpha \|\tilde{h} - \tfrac{1}{m}\sum_{\alpha=1}^m h_{i(\alpha)}\|^2 &= E_{\alpha,\beta}\left\langle \tilde{h} - \tfrac{1}{m}\sum_{\alpha=1}^m h_{i(\alpha)}, \tilde{h} - \tfrac{1}{m}\sum_{\beta=1}^m h_{i(\beta)} \right\rangle \\
&= \tfrac{1}{m^2}\sum_{\alpha,\beta=1}^m E_{\alpha,\beta}[\langle \tilde{h} - h_{i(\alpha)}, \tilde{h} - h_{i(\beta)}\rangle] \\
&= \tfrac{1}{m^2}\sum_{\alpha=1}^m E_\alpha[\langle \tilde{h} - h_{i(\alpha)}, \tilde{h} - h_{i(\alpha)}\rangle] \\
&= \tfrac{1}{m} E_\alpha[\|\tilde{h} - h_{i(\alpha)}\|^2] \leq \tfrac{R^2}{m}.
\end{aligned}$$

In particular, there must be $i_1, \ldots, i_m \subset \{1, \ldots, m\}$ such that

$$\|h - \sum_{j=1}^m h_{i_j}\| \leq \|h - \tilde{h}\| + \|\tilde{h} - \sum_{j=1}^m h_{i_j}\| \leq \epsilon + \tfrac{R}{\sqrt{m}}. \quad \square$$

Barron spaces

Definition [Barron, 1993]

Consider functions $f : [0,1]^D \to \mathbb{R}$ of the form

$$f(x) = \int_\Omega a\,\rho(w^t x + b)\mu(da, dw, db) = E_\mu[a\,\rho(w^t x + b)], \quad (1)$$

where $x \in [0,1]^D$, ρ is the ReLU, $\Omega = \mathbb{R} \times \mathbb{R}^D \times \mathbb{R}$ is the space of parameters, and μ is a probability distribution on (Ω, Σ_Ω), being Σ_Ω a Borel σ-algebra on Ω. For a function that admits a representation (1), we define the **Barron norm**

$$\|f\|_{\mathcal{B}} = \inf_\mu \sqrt{\mathbb{E}_\mu[|a|^2(\|w\|_1 + |b|)^2]},$$

where the infimum is taken over all μ for which the representations of f holds and $x \in [0,1]^D$. The **Barron space** is defined as

$$\mathcal{B} = \{f \in C([0,1]^D) \colon f \text{ admits a representation (1) and } \|f\|_{\mathcal{B}} < \infty\}$$

Barron spaces

Barron spaces are constructed to mimic functions that are approximated by "well-behaved" two-layer neural networks.

In the integral representation (1), we can think of μ as an unknown probability distribution of the parameters that is the limit to which converges the sum of atomic measures defined by the specific finite set of parameters of a neural networks when its width (or, equivalently, the number of M parameters) tends to infinity.
With the representation (1), approximating a continuous function with a shallow neural network can be interpreted as a Montecarlo integration problem.

Recall: Using a function with m parameters f_m, we have

- (**Monte Carlo error rate**):

$$\inf_{f_m} \|f_m - f\|_{L^2} \leq C \frac{\|f\|_{W^r}}{m^r},$$

where r is the regularity index.

Barron spaces

Theorem [Barron, 1993]

For a function $f : [0,1]^D \to \mathbb{R}$, let $\hat{f}(\omega)$ be the Fourier transform of any extension of f to $\mathbb{R}^D$. If

$$\gamma(f) := \int_{\mathbb{R}^D} |\omega|^2 \, |\hat{f}(\omega)| \, d\omega < \infty$$

then, for any $m > 0$, there exists a shallow neural network $f_m(x) = \sum_{k=1}^m a_k \rho(w_k^T x + b_k)$, with ρ being the ReLU activation function, satisfying

$$\|f_m - f\|_{L^2(\Omega)} \leq \frac{3\,\gamma(f)^2}{m},$$

and $\|\theta\|_{\mathcal{P}} := \frac{1}{m} \sum_{k=1}^m |a_k|(\|w_k\|_1 + |b_k|) \leq \frac{2\,\gamma(f)}{m}$

The sum $\|\theta\|_{\mathcal{P}}$ is called the **path norm** of the network, where $\theta = (a_k, w_k, b_k)$ are the network parameters.

Barron spaces

Note that, in the estimate of the theorem

$$\|f_m - f\|_{L^2(\Omega)} \leq \frac{3\,\gamma(f)^2}{m},$$

the numerator $\gamma(f)^2$ is an integral over $\mathbb{R}^D$ and could therefore bring back to the bound an exponential dependence on the dimension D.
That is, the finiteness of $\gamma(f)$ in general is not sufficient to avoid the curse of dimensionality, even though, in several cases, $\gamma(f)$ has only a polynomial dependence on D

Barron spaces

Recall that the Fourier transform of $f : \mathbb{R}^D \to \mathbb{R}$ is formally defined as

$$\hat{f}(\omega) = \int_{\mathbb{R}^D} f(x)\, e^{2\pi i \omega \cdot x}\, dx$$

and it satisfies the property that

$$(Df)^{\wedge}(\omega) = 2\pi i \omega \hat{f}(\omega)$$

Thus, the condition $\gamma(f) < \infty$ in Barron's theorem is equivalent to

$$\|(D^2 f)^{\wedge}\|_{L^1} < \infty$$

and this imposes a regularity condition of f. In particular, it implies the boundedness of all the first order partial derivatives of f.

Barron spaces

- What classes of functions belong to the Barron space $\mathcal{B}$?

Theorem

Let $f \in C([0,1]^D)$, where $\gamma(f) < \infty$. Then

$$\|f\|_{\mathcal{B}} \leq 2\,\gamma(f) + 2\,|f(0)| + 2\|\nabla f(0)\|_{L^1}$$

It follows that Gaussian functions, positive definite functions, linear functions and radial functions belong to $\mathcal{B}$.

Barron spaces

Theorem of direct approximation [Barron, 1993]

For any $f \in \mathcal{B}$ and integer $m > 0$, there exists a shallow neural network $f_m(x) = \sum_{k=1}^{m} a_k \rho(w_k^T x + b_k)$ such that

$$\|f_m(\cdot;\theta) - f(\cdot)\| \leq \frac{3\|f\|_{\mathcal{B}}^2}{m}.$$

Furthermore, we have $\|\theta\|_{\mathcal{P}} \leq 2\|f\|_{\mathcal{B}}$, where $\|\theta\|_{\mathcal{P}} = \frac{1}{m}\sum_{i=1}^{m} |a_i|(\|w_i\|_1 + |b_i|)$.

Theorem of inverse approximation [Barron, 1993]

Let $f \in C([0,1]^D)$ and assume there a constant Q and a sequence $(f_n) \subset \{f_n(\cdot,\theta) : \|\theta\|_{\mathcal{P}} \leq Q\}$ such that

$$\lim_n f_n(x) = f(x) \quad \forall x \in [0,1]^D$$

Then $f \in \mathcal{B}$ and $\|f\|_{\mathcal{B}} \leq Q$.

Barron spaces

To summarize, the Barron space $\mathcal{B}$ is the space of all functions that can be approximated by shallow neural networks with bounded path norm.
In some cases, the bound of the approximation error in the Barron spaces overcomes the curse of dimensionality.

Some papers have investigated conditions for PDE solutions to lie on the Barron space with the goal to apply shallow neural networks to approximate the solutions of high-dimensional PDEs:

Cf. *On the representation of solutions to elliptic PDEs in Barron spaces*, by Z. Chen, J. Lu, and Y. Lu. In: Advances in neural information processing systems, Vol 34 (2021), pp. 6454–6465

The role of depth

Deep neural networks

Universal Approximation Theorems

There are *dual* versions of the approximation theorem above where the network has bounded width and arbitrarily large depth.

Universal Approximation Theorems

There are *dual* versions of the approximation theorem above where the network has bounded width and arbitrarily large depth.

Theorem [Kidger and Lyons, 2020]

Assume ρ is a nonaffine continuous function which is continuously differentiable at least one point, with nonzero derivative at that point.
For every $f \in C([0,1]^D)$, given $\epsilon > 0$, there exists a neural network $\Phi : \mathbb{R}^D \to \mathbb{R}^d$ where the number of neurons N_ℓ for each layer ℓ bounded by $D + d + 2$ such that

$$|\Phi(x) - f(x)| < \epsilon, \qquad \text{for any } x \in [0,1]^D$$

Universal Approximation Theorems

There are *dual* versions of the approximation theorem above where the network has bounded width and arbitrarily large depth.

Theorem [Kidger and Lyons, 2020]

Assume ρ is a nonaffine continuous function which is continuously differentiable at least one point, with nonzero derivative at that point.
For every $f \in C([0,1]^D)$, given $\epsilon > 0$, there exists a neural network $\Phi : \mathbb{R}^D \to \mathbb{R}^d$ where the number of neurons N_ℓ for each layer ℓ bounded by $D + d + 2$ such that

$$|\Phi(x) - f(x)| < \epsilon, \qquad \text{for any } x \in [0,1]^D$$

Note: Unlike shallow networks, the width can be bounded; however the number of layers $L(\Phi)$ can become arbitrarily large.

Deep Neural Networks

Several papers in the literature have addressed the **benefits of depth in neural networks**.

- Deep neural networks have shown remarkable success in applications.
- Many observations aimed to explain the benefit of depths are heuristic (e.g., Chui-Li-Mhaskar (1994) shows that deep networks can do localized approximation while shallow ones can not).
- A few rigorous results show that deep neural networks achieve properties that cannot be matched by shallow or shallower networks.
- Historical note: [Sipser, 1986; Hastad, 1987] have shown that deep circuits are more efficient in representing certain Boolean functions than shallow circuits.

Deep Neural Networks

- *Benefits of depth in neural networks*, Matus Telgarsky (2016).

Paper shows that: *For any $k > 0$, there exists a neural network with $\Theta(k^3)$ layers, $\Theta(1)$ nodes per layer and $\Theta(1)$ distinct parameters that cannot be approximated by networks with $O(k)$ layers unless they are exponentially large, that is, they have $\Omega(2^k)$ nodes.*

Notation:
$f(n) = O(g(n))$ means that $f(n)$ grows no faster than $g(n)$
$f(n) = \Theta(g(n))$ means that $f(n)$ grows as fast as $g(n)$
$f(n) = \Omega(g(n))$ means that $f(n)$ grows faster than $g(n)$

Equivalently

- $f(n) = O(g(n))$ iff $\exists k > 0, \exists n_0 : |f(n)| \leq kg(n), \forall n > n_0$
- $f(n) = \Theta(g(n))$ iff $\exists k_1, k_2 > 0, \exists n_0 : k_1 g(n) \leq f(n) \leq k_2 g(n), \forall n > n_0$
- $f(n) = \Omega(g(n))$ iff $\exists k > 0, \exists n_0 : f(n) \geq kg(n), \forall n > n_0$

Deep Neural Networks

The idea of Telgarski is that a deep network cannot be well approximated by a reasonably-sized shallow network.

Consider the function $\Delta(x) = \begin{cases} 2x & \text{if } x \in [0, 1/2) \\ 2 - 2x & \text{if } x \in [1/2, 1) \\ 0 & \text{otherwise.} \end{cases}$
which is implemented by a simple ReLU network.

By combining the function with itself L times creates 2^{L-1} copies of it self, uniformly shrunk down. The complexity of the function, in terms of discontinuities, has increased exponentially with the number of nodes and layers (both $O(L)$)

Deep Neural Networks

One can show

Theorem

Choose any $L \geq 2$. $f = \Delta^{L^2+1}$ is a ReLU neural network with $3L^2 + 6$ nodes and $2L^2 + 4$ layers but any ReLU network with at most 2^L nodes and L layer cannot approximate f with arbitrary accuracy, since

$$\int_{[0,1]} |f(x) - g(x)|\, dx \geq \frac{1}{32}.$$

This was the first rigorous proof showing that a deep network can not be approximated by a reasonably-sized shallow network.

Deep Neural Networks

- *The Power of Depth for Feedforward Neural Networks*, Ronen Eldan, Ohad Shamir (2016)

Paper shows that: *There is an approximately radial function on $\mathbb{R}^D$, expressible by a small 3-layer feedforward neural networks, which cannot be approximated by any 2-layer network, to more than a certain constant accuracy, unless its width is exponential in the dimension.*

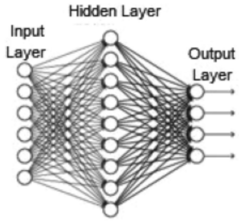
Result holds for general activation functions including ReLU, sigmoid and threshold.

Deep Neural Networks

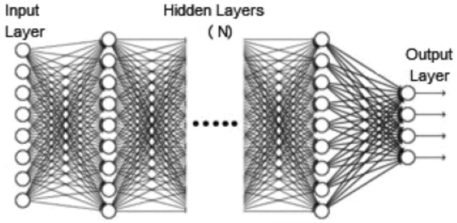
Idea of the proof:

- Approximating radial functions is rather straightforward using 3 layers:
  1. First, we can construct a linear combination of neurons expressing the univariate mapping $z \to z^2$ arbitrarily well in any bounded domain.
  2. Next, by adding the above combinations together, one for each coordinate, we can have our network first compute (approximately) the mapping $x \to \|x\|^2 = \sum_i x_i^2$ inside any bounded domain.
  3. Finally, we use the next layer to compute some univariate function of $\|x\|^2$, resulting in an approximately radial function.
- With only 2 layers, it is less clear how to approximate such radial functions. Indeed, approximating radial functions with 2 layers would require exponentially large width.

Deep Neural Networks



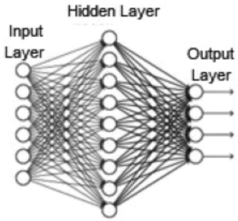
(a) A Shallow Network



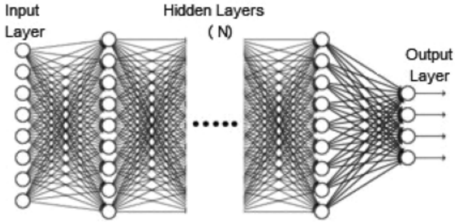
(b) A Deep Neural Network

- Modern network architectures are typically very deep

Deep Neural Networks



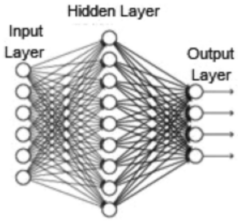
(a) A Shallow Network



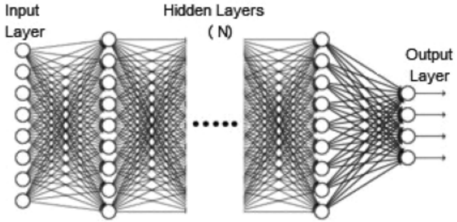
(b) A Deep Neural Network

- Modern network architectures are typically very deep
- **Deep** vs. **shallow networks**: Depth improves expressive power

Deep Neural Networks



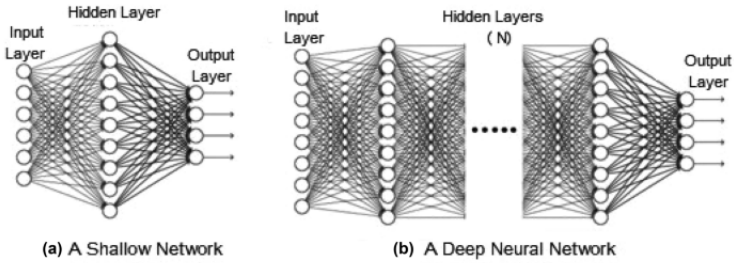
(a) A Shallow Network



(b) A Deep Neural Network

- Modern network architectures are typically very deep
- **Deep** vs. **shallow networks**: Depth improves expressive power
- With respect to shallow networks (and traditional function representations), deep neural networks can exploit **composition**

Deep Neural Networks



(a) A Shallow Network (b) A Deep Neural Network

- Modern network architectures are typically very deep
- **Deep** vs. **shallow networks**: Depth improves expressive power
- With respect to shallow networks (and traditional function representations), deep neural networks can exploit **composition**
  $\rightarrow$ **Blessing of compositionality**

Example: piecewise linear functions on $\mathbb{R}$

Triangle function:

$$T(x) = \begin{cases} 2x & \text{if } 0 \leq x < \frac{1}{2} \\ 2(1-x) & \text{if } \frac{1}{2} \leq x \leq 1 \end{cases} \qquad x \in \mathbb{R},$$

Example: piecewise linear functions on $\mathbb{R}$

Triangle function:

$$T(x) = \begin{cases} 2x & \text{if } 0 \leq x < \frac{1}{2} \\ 2(1-x) & \text{if } \frac{1}{2} \leq x \leq 1 \end{cases} \qquad x \in \mathbb{R},$$

can be expressed using $\rho(x) = \max\{x, 0\}$ as

$$\Phi(x) = \begin{bmatrix} 2 & -4 \end{bmatrix} \rho\left(\begin{bmatrix} 1 \\ 1 \end{bmatrix} x + \begin{bmatrix} 0 \\ -\frac{1}{2} \end{bmatrix} \right) = 2(x-0)_+ - 4(x - \tfrac{1}{2})_+$$

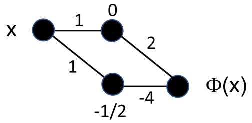
Example: piecewise linear functions on $\mathbb{R}$

Triangle function:

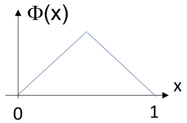
$$T(x) = \begin{cases} 2x & \text{if } 0 \leq x < \frac{1}{2} \\ 2(1-x) & \text{if } \frac{1}{2} \leq x \leq 1 \end{cases} \qquad x \in \mathbb{R},$$

can be expressed using $\rho(x) = \max\{x, 0\}$ as

$$\Phi(x) = \begin{bmatrix} 2 & -4 \end{bmatrix} \rho\left(\begin{bmatrix} 1 \\ 1 \end{bmatrix} x + \begin{bmatrix} 0 \\ -\frac{1}{2} \end{bmatrix} \right) = 2(x-0)_+ - 4(x - \tfrac{1}{2})_+$$



Neural Network



Output function

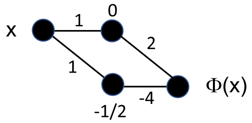
Example: piecewise linear functions on $\mathbb{R}$

Triangle function:

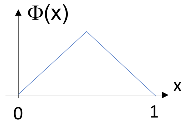
$$T(x) = \begin{cases} 2x & \text{if } 0 \leq x < \frac{1}{2} \\ 2(1-x) & \text{if } \frac{1}{2} \leq x \leq 1 \end{cases} \qquad x \in \mathbb{R},$$

can be expressed using $\rho(x) = \max\{x, 0\}$ as

$$\Phi(x) = \begin{bmatrix} 2 & -4 \end{bmatrix} \rho\left(\begin{bmatrix} 1 \\ 1 \end{bmatrix} x + \begin{bmatrix} 0 \\ -\frac{1}{2} \end{bmatrix} \right) = 2(x-0)_+ - 4(x - \tfrac{1}{2})_+$$



Neural Network



Output function

- $\Phi(x)$ is a neural network with $L = 2,\ N_2 = 2,\ M = 6$.

Example: piecewise linear functions on $\mathbb{R}$

- Neural Network with $L = 2$, $N_2 = 2$
  $\rightarrow$ piecewise linear function with at most 2 breakpoints

Example: piecewise linear functions on $\mathbb{R}$

- Neural Network with $L = 2$, $N_2 = 2$
  $\rightarrow$ piecewise linear function with at most 2 breakpoints
- Neural Networks with $L = 2$, $N - 2 = N$
  $\rightarrow$ piecewise linear function with at most N breakpoints

Example: piecewise linear functions on $\mathbb{R}$

- Neural Network with $L = 2$, $N_2 = 2$
  $\rightarrow$ piecewise linear function with at most 2 breakpoints
- Neural Networks with $L = 2$, $N - 2 = N$
  $\rightarrow$ piecewise linear function with at most N breakpoints
  - Same expressive power ($M = O(N^2)$) as continuous piecewise linear functions with the same number of parameters:
    N breakpoints $\leftrightarrow$ N^2 parameters

Example: piecewise linear functions on $\mathbb{R}$

- Neural Network with $L = 2$, $N_2 = 2$
  $\rightarrow$ piecewise linear function with at most 2 breakpoints
- Neural Networks with $L = 2$, $N - 2 = N$
  $\rightarrow$ piecewise linear function with at most N breakpoints
  - Same expressive power ($M = O(N^2)$) as continuous piecewise linear functions with the same number of parameters:
    N breakpoints $\leftrightarrow$ N^2 parameters
- **Composition** increases the number of breakpoints.
  Neural Networks with L layers, N neurons/layer
  (complexity $M = O(LN^2)$)
  $\rightarrow$ piecewise linear function with N^L breakpoints

Example: piecewise linear functions on $\mathbb{R}$

- Neural Network with $L = 2$, $N_2 = 2$
  $\rightarrow$ piecewise linear function with at most 2 breakpoints
- Neural Networks with $L = 2$, $N - 2 = N$
  $\rightarrow$ piecewise linear function with at most N breakpoints
  - Same expressive power ($M = O(N^2)$) as continuous piecewise linear functions with the same number of parameters:
    N breakpoints $\leftrightarrow$ N^2 parameters
- **Composition** increases the number of breakpoints.
  Neural Networks with L layers, N neurons/layer
  (complexity $M = O(LN^2)$)
  $\rightarrow$ piecewise linear function with N^L breakpoints
- Deep neural networks can **improve classical approximation methods** for several function classes
  [Daubechies, DeVore, Foucart, Hanin, Petrova, 2022]

Approximations using Neural Networks

In general, up to the observation we made about Barron spaces, shallow network are comparable to traditional approximation methods in terms of computational complexity.

Approximations using Neural Networks

In general, up to the observation we made about Barron spaces, shallow network are comparable to traditional approximation methods in terms of computational complexity.

Theorem - Shallow Network Approximation [Mhaskar 1996]

Consider $f \in C([0,1]^D)$. Then

$$\inf_{\Phi \in \mathcal{F}(M, L=2, B)} \|f - \Phi\|_\infty \leq c\, M^{-1/D}$$

That is, the complexity of a single-layer neural networks Φ that approximates f with accuracy at least ϵ is

$$M = O(\epsilon^{-D})$$

Approximations using Neural Networks

In general, up to the observation we made about Barron spaces, shallow network are comparable to traditional approximation methods in terms of computational complexity.

Theorem - Shallow Network Approximation [Mhaskar 1996]

Consider $f \in C([0,1]^D)$. Then

$$\inf_{\Phi \in \mathcal{F}(M,L=2,B)} \|f - \Phi\|_\infty \leq c\, M^{-1/D}$$

That is, the complexity of a single-layer neural networks Φ that approximates f with accuracy at least ϵ is

$$M = O(\epsilon^{-D})$$

- Same approximation rate as classical piecewise linear approximations

Approximations using Deep Neural Networks

Deep networks can exploit compositionality to **reduce the number of parameters** needed to approximate functions.

Approximations using Deep Neural Networks

Deep networks can exploit compositionality to **reduce the number of parameters** needed to approximate functions.

Theorem - Deep Network Approximation [Yarotsky, 2017,2018]

For $f \in C([0,1]^D)$

$$\inf_{\Phi \in \mathcal{F}(M,L,B)} \|f - \Phi\|_\infty \leq c\, M^{-2/D}$$

That is, the complexity of a deep networks Φ that approximates f with accuracy at least ϵ is

$$M = O(\epsilon^{-D/2})$$

Approximations using Deep Neural Networks

Deep networks can exploit compositionality to **reduce the number of parameters** needed to approximate functions.

Theorem - Deep Network Approximation [Yarotsky, 2017,2018]

For $f \in C([0,1]^D)$

$$\inf_{\Phi \in \mathcal{F}(M,L,B)} \|f - \Phi\|_\infty \leq c\, M^{-2/D}$$

That is, the complexity of a deep networks Φ that approximates f with accuracy at least ϵ is

$$M = O(\epsilon^{-D/2})$$

- It improves the approximation rate of shallow networks

Approximations using Deep Neural Networks

Deep networks can exploit compositionality to **reduce the number of parameters** needed to approximate functions.

Theorem - Deep Network Approximation [Yarotsky, 2017,2018]

For $f \in C([0,1]^D)$

$$\inf_{\Phi \in \mathcal{F}(M,L,B)} \|f - \Phi\|_\infty \leq c\, M^{-2/D}$$

That is, the complexity of a deep networks Φ that approximates f with accuracy at least ϵ is

$$M = O(\epsilon^{-D/2})$$

- It improves the approximation rate of shallow networks
- Optimal approximation rate achievable with a ReLU NN

Approximations using Deep Neural Networks

Deep networks can exploit compositionality to **reduce the number of parameters** needed to approximate functions.

Theorem - Deep Network Approximation [Yarotsky, 2017,2018]

For $f \in C([0,1]^D)$

$$\inf_{\Phi \in \mathcal{F}(M,L,B)} \|f - \Phi\|_\infty \leq c\, M^{-2/D}$$

That is, the complexity of a deep networks Φ that approximates f with accuracy at least ϵ is

$$M = O(\epsilon^{-D/2})$$

- It improves the approximation rate of shallow networks
- Optimal approximation rate achievable with a ReLU NN
- $L(\Phi)$ grown like $O(\ln(1/\epsilon))$

Approximations using Deep Neural Networks

Theorem - Deep Network Approximation [Petersen-Voigtländer, 2018]

Let f be piecewise $C^\beta([0,1]^D)$ with $\beta > 0$. Then

$$\inf_{\Phi \in \mathcal{F}(M,L,B)} \|f - \Phi\|_{L^2([0,1]^D)} \leq c\, M^{-\beta/(2(D-1))}$$

That is, the complexity of a deep networks Φ that approximates f with accuracy at least ϵ is

$$M = O(\epsilon^{-2(D-1)/\beta})$$

Approximations using Deep Neural Networks

Theorem - Deep Network Approximation [Petersen-Voigtländer, 2018]

Let f be piecewise $C^{\beta}([0,1]^D)$ with $\beta > 0$. Then

$$\inf_{\Phi \in \mathcal{F}(M,L,B)} \|f - \Phi\|_{L^2([0,1]^D)} \leq c\, M^{-\beta/(2(D-1))}$$

That is, the complexity of a deep networks Φ that approximates f with accuracy at least ϵ is

$$M = O(\epsilon^{-2(D-1)/\beta})$$

- Optimal approximation rate achievable with a ReLU NN

Approximations using Deep Neural Networks

Theorem - Deep Network Approximation [Petersen-Voigtländer, 2018]

Let f be piecewise $C^\beta([0,1]^D)$ with $\beta > 0$. Then

$$\inf_{\Phi \in \mathcal{F}(M,L,B)} \|f - \Phi\|_{L^2([0,1]^D)} \leq c\, M^{-\beta/(2(D-1))}$$

That is, the complexity of a deep networks Φ that approximates f with accuracy at least ϵ is

$$M = O(\epsilon^{-2(D-1)/\beta})$$

- Optimal approximation rate achievable with a ReLU NN
- The number of layers satisfies $L \leq c' \log(\beta + 2)(1 + \beta/D)$ where c' is an absolute constant

Approximations using Deep Neural Networks

Theorem - Deep Network Approximation [Petersen-Voigtländer, 2018]

Let f be piecewise $C^\beta([0,1]^D)$ with $\beta > 0$. Then

$$\inf_{\Phi \in \mathcal{F}(M,L,B)} \|f - \Phi\|_{L^2([0,1]^D)} \leq c\, M^{-\beta/(2(D-1))}$$

That is, the complexity of a deep networks Φ that approximates f with accuracy at least ϵ is

$$M = O(\epsilon^{-2(D-1)/\beta})$$

- Optimal approximation rate achievable with a ReLU NN
- The number of layers satisfies $L \leq c' \log(\beta + 2)(1 + \beta/D)$ where c' is an absolute constant
- The constant c depends on D, β but dependence is not explicit

Approximations using Deep Neural Networks

Most approximations results using neural network - including those presented above - are **non-quantitative** and **asymptotic**

$$\inf_{\Phi \in \mathcal{F}(M,L,B)} \|f - \Phi\| \leq c\, M^{-\beta/D}$$

Approximations using Deep Neural Networks

Most approximations results using neural network - including those presented above - are **non-quantitative** and **asymptotic**

$$\inf_{\Phi \in \mathcal{F}(M,L,B)} \|f - \Phi\| \leq c\, M^{-\beta/D}$$

- Approximation constant c depend on D non-explicitly

Approximations using Deep Neural Networks

Most approximations results using neural network - including those presented above - are **non-quantitative** and **asymptotic**

$$\inf_{\Phi \in \mathcal{F}(M,L,B)} \|f - \Phi\| \leq c\, M^{-\beta/D}$$

- Approximation constant c depend on D non-explicitly
- L need to be *sufficiently large* but no quantitative bound is shown

Approximations using Deep Neural Networks

Most approximations results using neural network - including those presented above - are **non-quantitative** and **asymptotic**

$$\inf_{\Phi \in \mathcal{F}(M,L,B)} \|f - \Phi\| \leq c\, M^{-\beta/D}$$

- Approximation constant c depend on D non-explicitly
- L need to be *sufficiently large* but no quantitative bound is shown

[Lu, Shen, Yang, Zhang, 2020], [Shen, Yang, Zhang, 2021] are the first papers to provide **non-asymptotic** and **quantitative** approximation results.

Approximations using Deep Neural Networks

Theorem [Shen, Yang, Zhang, 2021]

Let $f \in B_\lambda(C^\alpha([0,1]^D))$, the space of Hölder continuous function of order $\alpha \in [0,1)$ with Hölder constant λ, and let $\mathcal{F}(N, L, B)$ where N is the width and L is the depth of Φ
Then, for $p \in [1, \infty]$

$$\inf_{\Phi \in \mathcal{F}(N,L,B)} \|f - \Phi\|_{L^p([0,1]^D)} \leq 19\sqrt{D}\lambda\, N^{-2\alpha/D} L^{-2\alpha/D}$$

That is, the complexity of a deep networks Φ that approximates f with accuracy at least ϵ is about

$$M = O(\epsilon^{-D/\alpha}) \qquad \text{Note: } M = O(LN^2)$$

Approximations using Deep Neural Networks

Theorem [Shen, Yang, Zhang, 2021]

Let $f \in B_\lambda(C^\alpha([0,1]^D))$, the space of Hölder continuous function of order $\alpha \in [0,1)$ with Hölder constant λ, and let $\mathcal{F}(N, L, B)$ where N is the width and L is the depth of Φ
Then, for $p \in [1, \infty]$

$$\inf_{\Phi \in \mathcal{F}(N,L,B)} \|f - \Phi\|_{L^p([0,1]^D)} \leq 19\sqrt{D}\lambda\, N^{-2\alpha/D} L^{-2\alpha/D}$$

That is, the complexity of a deep networks Φ that approximates f with accuracy at least ϵ is about

$$M = O(\epsilon^{-D/\alpha}) \qquad \text{Note: } M = O(LN^2)$$

- This is the nearly optimal approximation rate

Approximations using Deep Neural Networks

Theorem [Shen, Yang, Zhang, 2021]

Let $f \in B_\lambda(C^\alpha([0,1]^D))$, the space of Hölder continuous function of order $\alpha \in [0,1)$ with Hölder constant λ, and let $\mathcal{F}(N,L,B)$ where N is the width and L is the depth of Φ
Then, for $p \in [1,\infty]$

$$\inf_{\Phi \in \mathcal{F}(N,L,B)} \|f - \Phi\|_{L^p([0,1]^D)} \leq 19\sqrt{D}\lambda\, N^{-2\alpha/D} L^{-2\alpha/D}$$

That is, the complexity of a deep networks Φ that approximates f with accuracy at least ϵ is about

$$M = O(\epsilon^{-D/\alpha}) \qquad \text{Note: } M = O(LN^2)$$

- This is the nearly optimal approximation rate
- Quantitative values for N and L are given

Dimensionality reduction using Deep Neural Networks

Curse of Dimensionality

- Success of deep neural networks in applications is not fully explained by their mere approximation properties.

Curse of Dimensionality

- Success of deep neural networks in applications is not fully explained by their mere approximation properties.
- Their performance on problems where input dimension is high often appears to overcome the **curse of dimensionality** (COD).

Curse of Dimensionality

- Success of deep neural networks in applications is not fully explained by their mere approximation properties.
- Their performance on problems where input dimension is high often appears to overcome the **curse of dimensionality** (COD).

[Bellman, 1952, Novak & Wozniakowsk, 2009]

Approximation methods deteriorate exponentially fast with increasing dimension D

Curse of Dimensionality

- Success of deep neural networks in applications is not fully explained by their mere approximation properties.
- Their performance on problems where input dimension is high often appears to overcome the **curse of dimensionality** (COD).

[Bellman, 1952, Novak & Wozniakowsk, 2009]

Approximation methods deteriorate exponentially fast with increasing dimension D

  - Computational cost of traditional numerical PDE solvers such as finite difference, finite element and spectral methods, scales with D

Curse of Dimensionality

- Success of deep neural networks in applications is not fully explained by their mere approximation properties.
- Their performance on problems where input dimension is high often appears to overcome the **curse of dimensionality** (COD).

[Bellman, 1952, Novak & Wozniakowsk, 2009]

Approximation methods deteriorate exponentially fast with increasing dimension D

- Computational cost of traditional numerical PDE solvers such as finite difference, finite element and spectral methods, scales with D
- Pointwise approximation of the solution with accuracy ϵ requires $M = O(\epsilon^{-D})$ parameters $\rightarrow$ practically impossible to achieve satisfactory accuracy for very large D

Curse of Dimensionality

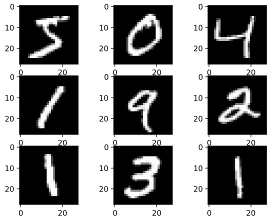
In many multi-dimensional problems, data is highly structured and the information of interest is low-dimensional

Curse of Dimensionality

In many multi-dimensional problems, data is highly structured and the information of interest is low-dimensional

Widely used image datasets:

- MNIST: $28 \times 28 = 784$ pixels per image $\rightarrow \mathbb{R}^{784}$
    - intrinsic dimension: between 8 and 13

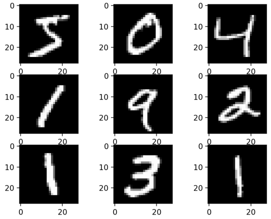


Curse of Dimensionality

In many multi-dimensional problems, data is highly structured and the information of interest is low-dimensional

Widely used image datasets:

- MNIST: $28 \times 28 = 784$ pixels per image $\rightarrow \mathbb{R}^{784}$
  - intrinsic dimension: between 8 and 13



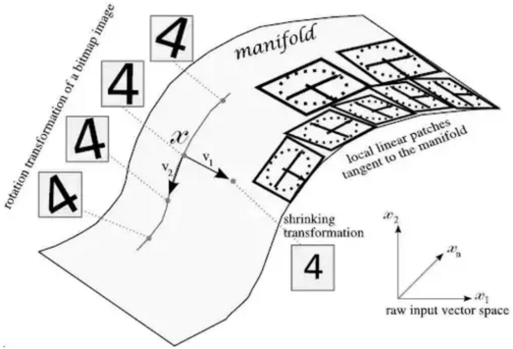
- ImageNet: $224 \times 224 \times 3 = 150528$ pixels per image $\rightarrow \mathbb{R}^{150528}$
  - intrinsic dimension: between 26 and 43

Manifold hypothesis

Many theoretical results explains this phenomenon either explicitly or implicitly using the manifold hypothesis.

Manifold hypothesis:

There is a d-dimensional manifold containing our D-dimensional data of interest where $d \ll D$


rotation transformation of a bitmap image
manifold
x
v_1
v_2
local linear patches tangent to the manifold
shrinking transformation
x_2
x_n
x_1
raw input vector space

Manifold hypothesis

We want to approximate a function

$$f : \mathbb{R}^D \supset S \mapsto \mathbb{R}$$

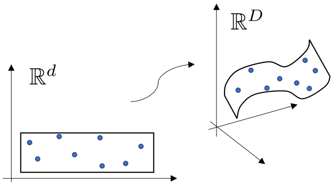
Under the manifold hypothesis, we are not seeking to approximate f with respect to a norm on $\mathbb{R}^D$.

Manifold hypothesis

We want to approximate a function

$$f : \mathbb{R}^D \supset S \mapsto \mathbb{R}$$

Under the manifold hypothesis, we are not seeking to approximate f with respect to a norm on $\mathbb{R}^D$.
Rather, we approximate f on a d-dimensional manifold $\mathcal{M}$, where $d \ll D$.



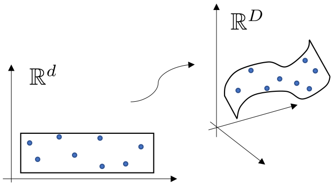
D : **ambient dimension** vs. d: **intrinsic dimension**

Manifold hypothesis

We want to approximate a function

$$f : \mathbb{R}^D \supset S \mapsto \mathbb{R}$$

Under the manifold hypothesis, we are not seeking to approximate f with respect to a norm on $\mathbb{R}^D$.
Rather, we approximate f on a d-dimensional manifold $\mathcal{M}$, where $d \ll D$.



D : **ambient dimension** vs. d: **intrinsic dimension**

Bonus: neural networks can learn local coordinate transformations

Manifold hypothesis

[Shaham,Cloninger,Coifman,2018; Schmidt-Hieber,2019; Nakada,Imaizumi,2020]

Manifold hypothesis

[Shaham,Cloninger,Coifman,2018; Schmidt-Hieber,2019; Nakada,Imaizumi,2020]

Theorem (informal version)

Let $\mathcal{M} \subset \mathbb{R}^D$ be a smooth d-dimensional manifold and let $f \in B_\lambda(C^\alpha([0,1]^D))$, space of Hölder continuous function of order $\alpha \in [0,1)$ with Hölder constant λ. Then

$$\inf_{\Phi \in \mathcal{F}(N,L,B)} \|f - \Phi\|_{L^\infty(\mathcal{M})} < c(\lambda, \alpha, D)\, M^{-\alpha/d}$$

The complexity of a deep neural network Φ that approximates f with accuracy at least ϵ satisfies

$$M \leq \tilde{c}(\lambda, \alpha, D)\, \epsilon^{-d/\alpha}$$

Manifold hypothesis

[Shaham,Cloninger,Coifman,2018; Schmidt-Hieber,2019; Nakada,Imaizumi,2020]

Theorem (informal version)

Let $\mathcal{M} \subset \mathbb{R}^D$ be a smooth d-dimensional manifold and let $f \in B_\lambda(C^\alpha([0,1]^D))$, space of Hölder continuous function of order $\alpha \in [0,1)$ with Hölder constant λ. Then

$$\inf_{\Phi \in \mathcal{F}(N,L,B)} \|f - \Phi\|_{L^\infty(\mathcal{M})} < c(\lambda, \alpha, D)\, M^{-\alpha/d}$$

The complexity of a deep neural network Φ that approximates f with accuracy at least ϵ satisfies

$$M \leq \tilde{c}(\lambda, \alpha, D)\, \epsilon^{-d/\alpha}$$

- Complexity grows like $\epsilon^{-d/\alpha}$ (d rather than D)

Manifold hypothesis

[Shaham,Cloninger,Coifman,2018; Schmidt-Hieber,2019; Nakada,Imaizumi,2020]

Theorem (informal version)

Let $\mathcal{M} \subset \mathbb{R}^D$ be a smooth d-dimensional manifold and let $f \in B_\lambda(C^\alpha([0,1]^D))$, space of Hölder continuous function of order $\alpha \in [0,1)$ with Hölder constant λ. Then

$$\inf_{\Phi \in \mathcal{F}(N,L,B)} \|f - \Phi\|_{L^\infty(\mathcal{M})} < c(\lambda, \alpha, D)\, M^{-\alpha/d}$$

The complexity of a deep neural network Φ that approximates f with accuracy at least ϵ satisfies

$$M \leq \tilde{c}(\lambda, \alpha, D)\, \epsilon^{-d/\alpha}$$

- Complexity grows like $\epsilon^{-d/\alpha}$ (d rather than D)
- $\tilde{c}$ **depends on** D with polynomial or logarithmic dependence

Manifold hypothesis

Idea of the proof [Shaham,Cloninger,Coifman,2018]

Manifold hypothesis

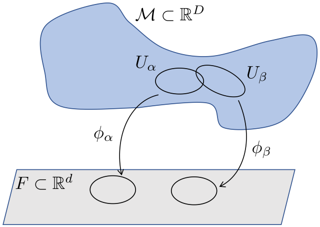
Idea of the proof [Shaham,Cloninger,Coifman,2018]

1. Construct a basis or a frame (e.g., a wavelet frame or a polynomial basis) of $C^\alpha([0,1]^D)$ whose elements are implemented as neural networks

Manifold hypothesis

Idea of the proof [Shaham,Cloninger,Coifman,2018]

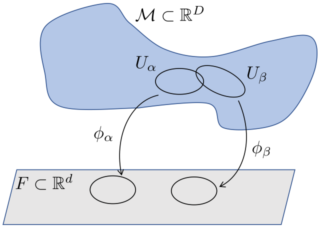
1. Construct a basis or a frame (e.g., a wavelet frame or a polynomial basis) of $C^\alpha([0,1]^D)$ whose elements are implemented as neural networks
2. Construct an atlas for $\mathcal{M} \subset \mathbb{R}^D$ by covering it with open balls



Manifold hypothesis

Idea of the proof [Shaham,Cloninger,Coifman,2018]

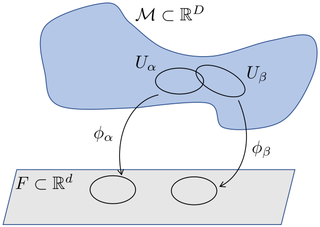
1. Construct a basis or a frame (e.g., a wavelet frame or a polynomial basis) of $C^\alpha([0,1]^D)$ whose elements are implemented as neural networks
2. Construct an atlas for $\mathcal{M} \subset \mathbb{R}^D$ by covering it with open balls
3. Use the open cover to obtain a partition of unity of $\mathcal{M}$ and expand any f on $\mathcal{M}$ using a basis or a frame on $\mathbb{R}^d$



Manifold hypothesis

Idea of the proof [Shaham,Cloninger,Coifman,2018]

1. Construct a basis or a frame (e.g., a wavelet frame or a polynomial basis) of $C^\alpha([0,1]^D)$ whose elements are implemented as neural networks
2. Construct an atlas for $\mathcal{M} \subset \mathbb{R}^D$ by covering it with open balls
3. Use the open cover to obtain a partition of unity of $\mathcal{M}$ and expand any f on $\mathcal{M}$ using a basis or a frame on $\mathbb{R}^d$



4. Extend the basis or frame terms from the original domain on $\mathbb{R}^d$ to $\mathbb{R}^D$ in a way that depends on the curvature of the manifold.

Network approximation - Manifold hypothesis

A recent result [Labate,Shi, 2022]

Network approximation - Manifold hypothesis

A recent result [Labate,Shi, 2022]

Theorem (informal version)

Let $\mathcal{M} \subset \mathbb{R}^D$ be a Riemannian d-dimensional manifold contained in $[0,1]^D$ having condition number $1/\tau$. Given $f \in B_\lambda(C^\alpha([0,1]^D))$, $\alpha \in (0,1)$, $\delta \in (0,1/3]$, for any $\varepsilon \in (0,1/2)$, there exists a neural network Φ^{f_0} with

$$\sup_{x \in \mathcal{M}} |\Phi^{f_0}(x) - f_0(x)| \leq \varepsilon.$$

The complexity of a deep networks Φ^{f_0} satisfies

$$M \leq c(\lambda, \alpha)\, \epsilon^{-d_\delta/\alpha},$$

where d_δ depends on d, δ, τ.

Network approximation - Manifold hypothesis

A recent result [Labate,Shi, 2022]

Theorem (informal version)

Let $\mathcal{M} \subset \mathbb{R}^D$ be a Riemannian d-dimensional manifold contained in $[0,1]^D$ having condition number $1/\tau$. Given $f \in B_\lambda(C^\alpha([0,1]^D))$, $\alpha \in (0,1)$, $\delta \in (0,1/3]$, for any $\varepsilon \in (0,1/2)$, there exists a neural network Φ^{f_0} with

$$\sup_{x \in \mathcal{M}} |\Phi^{f_0}(x) - f_0(x)| \leq \varepsilon.$$

The complexity of a deep networks Φ^{f_0} satisfies

$$M \leq c(\lambda, \alpha)\, \epsilon^{-d_\delta/\alpha},$$

where d_δ depends on d, δ, τ.

- Complexity grows like $\epsilon^{-d_\delta/\alpha}$ (d_δ rather than D)

Network approximation - Manifold hypothesis

A recent result [Labate,Shi, 2022]

Theorem (informal version)

Let $\mathcal{M} \subset \mathbb{R}^D$ be a Riemannian d-dimensional manifold contained in $[0,1]^D$ having condition number $1/\tau$. Given $f \in B_\lambda(C^\alpha([0,1]^D))$, $\alpha \in (0,1)$, $\delta \in (0,1/3]$, for any $\varepsilon \in (0,1/2)$, there exists a neural network Φ^{f_0} with

$$\sup_{x \in \mathcal{M}} |\Phi^{f_0}(x) - f_0(x)| \leq \varepsilon.$$

The complexity of a deep networks Φ^{f_0} satisfies

$$M \leq c(\lambda, \alpha)\, \epsilon^{-d_\delta/\alpha},$$

where d_δ depends on d, δ, τ.

- Complexity grows like $\epsilon^{-d_\delta/\alpha}$ (d_δ rather than D)
- c **does not depend on** D

Johnson-Lindenstrauss Lemma

One key tool to prove this result is an appropriate version of the **Johnson-Lindenstrauss lemma**, establishing low-distortion embeddings of points from high- into low-dimensional space.

Johnson-Lindenstrauss Lemma

One key tool to prove this result is an appropriate version of the **Johnson-Lindenstrauss lemma**, establishing low-distortion embeddings of points from high- into low-dimensional space.

Theorem [Johnson-Lindenstrauss, 1984]

Let $\delta \in (0,1)$ and $x_1, \dots, x_p \in \mathbb{R}^D$ be arbitrary points. Let $m = O(\delta^{-2} \log p)$. Then there is a Lipschitz map $f : \mathbb{R}^D \to \mathbb{R}^m$ such that

$$(1-\delta)|x_i - x_j|^2 \leq |f(x_i) - f(x_j)|^2 \leq (1+\delta)|x_i - x_j|^2, \quad \text{for all } i, j$$

Johnson-Lindenstrauss Lemma

One key tool to prove this result is an appropriate version of the **Johnson-Lindenstrauss lemma**, establishing low-distortion embeddings of points from high- into low-dimensional space.

Theorem [Johnson-Lindenstrauss, 1984]

Let $\delta \in (0,1)$ and $x_1, \dots, x_p \in \mathbb{R}^D$ be arbitrary points. Let $m = O(\delta^{-2} \log p)$. Then there is a Lipschitz map $f : \mathbb{R}^D \to \mathbb{R}^m$ such that

$$(1-\delta)|x_i - x_j|^2 \leq |f(x_i) - f(x_j)|^2 \leq (1+\delta)|x_i - x_j|^2, \quad \text{for all } i, j$$

- Low-distortion embeddings $\longleftrightarrow$ Restricted Isometry Property (RIP)

Johnson-Lindenstrauss Lemma

One key tool to prove this result is an appropriate version of the **Johnson-Lindenstrauss lemma**, establishing low-distortion embeddings of points from high- into low-dimensional space.

Theorem [Johnson-Lindenstrauss, 1984]

Let $\delta \in (0,1)$ and $x_1, \ldots, x_p \in \mathbb{R}^D$ be arbitrary points. Let $m = O(\delta^{-2} \log p)$. Then there is a Lipschitz map $f : \mathbb{R}^D \to \mathbb{R}^m$ such that

$$(1-\delta)|x_i - x_j|^2 \leq |f(x_i) - f(x_j)|^2 \leq (1+\delta)|x_i - x_j|^2, \quad \text{for all } i, j$$

- Low-distortion embeddings $\longleftrightarrow$ Restricted Isometry Property (RIP)
- Applications in compressed sensing, manifold learning, dimensionality reduction, ...

Johnson-Lindenstrauss Lemma

Manifold extension: preservation of ambient distances on a manifold under the action of random projections

Johnson-Lindenstrauss Lemma

Manifold extension: preservation of ambient distances on a manifold under the action of random projections

Theorem [Eftekhari and Wakin, 2015]

Let $\mathcal{M}$ be a compact d-dimensional Riemannian submanifold of $\mathbb{R}^D$ having condition number $1/\tau$ and volume V, satisfying $\frac{V}{\tau^d} \geq \left(\frac{21}{2\sqrt{d}}\right)^d$. Fix $\delta \in (0, 1/3]$, $\rho \in (0,1)$ and let A be a random $d_\delta \times D$ matrix populated with i.i.d. zero-mean Gaussian random variables with variance $1/d_\delta$ and

$$d_\delta \geq 18\delta^{-2} \max\left\{24d + 2d\log\left(\frac{\sqrt{d}}{\tau\delta^2}\right) + \log(2V^2), \log\left(\frac{8}{\rho}\right)\right\}.$$

Then, with probability at least $1-\rho$, for every pair of points $x_1, x_2 \in \mathcal{M}$,

$$(1-\delta)\|x_1 - x_2\|_2 \leq \|Ax_1 - Ax_2\|_2 \leq (1+\delta)\|x_1 - x_2\|_2.$$

Johnson-Lindenstrauss Lemma

Technical requirements on manifold $\mathcal{M} \subset \mathbb{R}^D$:

Johnson-Lindenstrauss Lemma

Technical requirements on manifold $\mathcal{M} \subset \mathbb{R}^D$:

- **Condition number** $1/\tau \rightarrow$ norm of the second fundamental form of $\mathcal{M}$ is bounded by $1/\tau$ in all directions.
  This implies:

Johnson-Lindenstrauss Lemma

Technical requirements on manifold $\mathcal{M} \subset \mathbb{R}^D$:

- **Condition number** $1/\tau \to$ norm of the second fundamental form of $\mathcal{M}$ is bounded by $1/\tau$ in all directions.
  This implies:
  - manifold cannot curve too much locally

Johnson-Lindenstrauss Lemma

Technical requirements on manifold $\mathcal{M} \subset \mathbb{R}^D$:

- **Condition number** $1/\tau \rightarrow$ norm of the second fundamental form of $\mathcal{M}$ is bounded by $1/\tau$ in all directions.
  This implies:
  - manifold cannot curve too much locally
  - angle between tangent spaces at nearby points cannot be too large

Johnson-Lindenstrauss Lemma

Technical requirements on manifold $\mathcal{M} \subset \mathbb{R}^D$:

- **Condition number** $1/\tau \rightarrow$ norm of the second fundamental form of $\mathcal{M}$ is bounded by $1/\tau$ in all directions.
  This implies:
  - manifold cannot curve too much locally
  - angle between tangent spaces at nearby points cannot be too large
  - geodesic and ambient distance cannot differ too much

Johnson-Lindenstrauss Lemma

Technical requirements on manifold $\mathcal{M} \subset \mathbb{R}^D$:

- **Condition number** $1/\tau \rightarrow$ norm of the second fundamental form of $\mathcal{M}$ is bounded by $1/\tau$ in all directions.
  This implies:
  - manifold cannot curve too much locally
  - angle between tangent spaces at nearby points cannot be too large
  - geodesic and ambient distance cannot differ too much
- **Random orthoprojection** $A : \mathbb{R}^D \rightarrow \mathbb{R}^{d_\delta}$

Johnson-Lindenstrauss Lemma

Technical requirements on manifold $\mathcal{M} \subset \mathbb{R}^D$:

- **Condition number** $1/\tau \rightarrow$ norm of the second fundamental form of $\mathcal{M}$ is bounded by $1/\tau$ in all directions.
  This implies:
  - manifold cannot curve too much locally
  - angle between tangent spaces at nearby points cannot be too large
  - geodesic and ambient distance cannot differ too much
- **Random orthoprojection** $A : \mathbb{R}^D \rightarrow \mathbb{R}^{d_\delta}$

  Here the parameter $\delta \in (0,1)$ controls the balance between isometry and dimension reduction

Johnson-Lindenstrauss Lemma

Technical requirements on manifold $\mathcal{M} \subset \mathbb{R}^D$:

- **Condition number** $1/\tau \to$ norm of the second fundamental form of $\mathcal{M}$ is bounded by $1/\tau$ in all directions.
  This implies:
  - manifold cannot curve too much locally
  - angle between tangent spaces at nearby points cannot be too large
  - geodesic and ambient distance cannot differ too much
- **Random orthoprojection** $A : \mathbb{R}^D \to \mathbb{R}^{d_\delta}$

  Here the parameter $\delta \in (0, 1)$ controls the balance between isometry and dimension reduction
  - δ is closer to 1 $\Rightarrow$ d_δ is closer to d $\Rightarrow$ weaker isometric property

Johnson-Lindenstrauss Lemma

Technical requirements on manifold $\mathcal{M} \subset \mathbb{R}^D$:

- **Condition number** $1/\tau \to$ norm of the second fundamental form of $\mathcal{M}$ is bounded by $1/\tau$ in all directions.
  This implies:
  - manifold cannot curve too much locally
  - angle between tangent spaces at nearby points cannot be too large
  - geodesic and ambient distance cannot differ too much
- **Random orthoprojection** $A : \mathbb{R}^D \to \mathbb{R}^{d_\delta}$

  Here the parameter $\delta \in (0, 1)$ controls the balance between isometry and dimension reduction
  - δ is closer to 1 $\Rightarrow$ d_δ is closer to d $\Rightarrow$ weaker isometric property
  - δ is closer to 0 $\Rightarrow$ d_δ farther from d $\Rightarrow$ better isometric property

Network approximation - Manifold hypothesis

Network approximation result - Idea of the proof

Network approximation - Manifold hypothesis

Network approximation result - Idea of the proof

1. Given $f_0 \in B_\lambda(C^\alpha([0,1]^D))$, $\alpha \in (0,1)$, define a lower dimensional function g_0 on $[0,1]^{d_\delta}$ by projecting f_0 via a random orthoprojection $A : \mathbb{R}^D \supset \mathcal{M} \to \mathbb{R}^{d_\delta}$ [Theorem by Eftekhari and Wakin, 2015]

Network approximation - Manifold hypothesis

Network approximation result - Idea of the proof

1. Given $f_0 \in B_\lambda(C^\alpha([0,1]^D))$, $\alpha \in (0,1)$, define a lower dimensional function g_0 on $[0,1]^{d_\delta}$ by projecting f_0 via a random orthoprojection $A : \mathbb{R}^D \supset \mathcal{M} \to \mathbb{R}^{d_\delta}$ [Theorem by Eftekhari and Wakin, 2015]
2. Extend g_0 continuously to a function $\tilde{g}_0$ on $B_\lambda(C^\alpha([0,1]^{d_\delta}))$

Network approximation - Manifold hypothesis

Network approximation result - Idea of the proof

1. Given $f_0 \in B_\lambda(C^\alpha([0,1]^D))$, $\alpha \in (0,1)$, define a lower dimensional function g_0 on $[0,1]^{d_\delta}$ by projecting f_0 via a random orthoprojection $A : \mathbb{R}^D \supset \mathcal{M} \to \mathbb{R}^{d_\delta}$ [Theorem by Eftekhari and Wakin, 2015]
2. Extend g_0 continuously to a function $\tilde{g}_0$ on $B_\lambda(C^\alpha([0,1]^{d_\delta}))$
3. Construct a neural network Φ^g to approximate functions $g \in B_\lambda(C^\alpha([0,1]^{d_\delta}))$, $\alpha \in (0,1)$

Network approximation - Manifold hypothesis

Network approximation result - Idea of the proof

1. Given $f_0 \in B_\lambda(C^\alpha([0,1]^D))$, $\alpha \in (0,1)$, define a lower dimensional function g_0 on $[0,1]^{d_\delta}$ by projecting f_0 via a random orthoprojection $A : \mathbb{R}^D \supset \mathcal{M} \to \mathbb{R}^{d_\delta}$ [Theorem by Eftekhari and Wakin, 2015]
2. Extend g_0 continuously to a function $\tilde{g}_0$ on $B_\lambda(C^\alpha([0,1]^{d_\delta}))$
3. Construct a neural network Φ^g to approximate functions $g \in B_\lambda(C^\alpha([0,1]^{d_\delta}))$, $\alpha \in (0,1)$
4. By applying the mapping A, derive from $\Phi^{\tilde{g}_0}$ a neural network Φ^{f_0} to approximate f_0 on $\mathcal{M}$

Challenges:

Network approximation - Manifold hypothesis

Network approximation result - Idea of the proof

1. Given $f_0 \in B_\lambda(C^\alpha([0,1]^D))$, $\alpha \in (0,1)$, define a lower dimensional function g_0 on $[0,1]^{d_\delta}$ by projecting f_0 via a random orthoprojection $A : \mathbb{R}^D \supset \mathcal{M} \to \mathbb{R}^{d_\delta}$ [Theorem by Eftekhari and Wakin, 2015]
2. Extend g_0 continuously to a function $\tilde{g}_0$ on $B_\lambda(C^\alpha([0,1]^{d_\delta}))$
3. Construct a neural network Φ^g to approximate functions $g \in B_\lambda(C^\alpha([0,1]^{d_\delta}))$, $\alpha \in (0,1)$
4. By applying the mapping A, derive from $\Phi^{\tilde{g}_0}$ a neural network Φ^{f_0} to approximate f_0 on $\mathcal{M}$

Challenges:

- To ensure the projected function g_0 is Hölder continuous

Network approximation - Manifold hypothesis

Network approximation result - Idea of the proof

1. Given $f_0 \in B_\lambda(C^\alpha([0,1]^D))$, $\alpha \in (0,1)$, define a lower dimensional function g_0 on $[0,1]^{d_\delta}$ by projecting f_0 via a random orthoprojection $A : \mathbb{R}^D \supset \mathcal{M} \to \mathbb{R}^{d_\delta}$ [Theorem by Eftekhari and Wakin, 2015]
2. Extend g_0 continuously to a function $\tilde{g}_0$ on $B_\lambda(C^\alpha([0,1]^{d_\delta}))$
3. Construct a neural network Φ^g to approximate functions $g \in B_\lambda(C^\alpha([0,1]^{d_\delta}))$, $\alpha \in (0,1)$
4. By applying the mapping A, derive from $\Phi^{\tilde{g}_0}$ a neural network Φ^{f_0} to approximate f_0 on $\mathcal{M}$

Challenges:

- To ensure the projected function g_0 is Hölder continuous
- To approximate $B_\lambda(C^\alpha([0,1]^D))$ using neural networks

Network approximation - Manifold hypothesis

Network approximation result - Idea of the proof

1. Given $f_0 \in B_\lambda(C^\alpha([0,1]^D))$, $\alpha \in (0,1)$, define a lower dimensional function g_0 on $[0,1]^{d_\delta}$ by projecting f_0 via a random orthoprojection $A : \mathbb{R}^D \supset \mathcal{M} \to \mathbb{R}^{d_\delta}$ [Theorem by Eftekhari and Wakin, 2015]
2. Extend g_0 continuously to a function $\tilde{g}_0$ on $B_\lambda(C^\alpha([0,1]^{d_\delta}))$
3. Construct a neural network Φ^g to approximate functions $g \in B_\lambda(C^\alpha([0,1]^{d_\delta}))$, $\alpha \in (0,1)$
4. By applying the mapping A, derive from $\Phi^{\tilde{g}_0}$ a neural network Φ^{f_0} to approximate f_0 on $\mathcal{M}$

Challenges:

- To ensure the projected function g_0 is Hölder continuous
- To approximate $B_\lambda(C^\alpha([0,1]^D))$ using neural networks
- To control number of parameters of Φ^{g_0} and Φ^{f_0}

Network approximation - Proof

Idea of the proof: 1. Construction of low-dim function

By the theorem of Eftekhari and Wakin, with probability at least $1-\rho$, we can pick a random matrix $A \in \mathbb{R}^{d_\delta \times D}$, with d_δ given above, such that for every pair of points $x_1, x_2 \in \mathcal{M}$, we have

$$(1-\delta)\|x_1 - x_2\|_2 \leq \|Ax_1 - Ax_2\|_2 \leq (1+\delta)\|x_1 - x_2\|_2$$

Network approximation - Proof

Idea of the proof: 1. Construction of low-dim function

By the theorem of Eftekhari and Wakin, with probability at least $1-\rho$, we can pick a random matrix $A \in \mathbb{R}^{d_\delta \times D}$, with d_δ given above, such that for every pair of points $x_1, x_2 \in \mathcal{M}$, we have

$$(1-\delta)\|x_1 - x_2\|_2 \leq \|Ax_1 - Ax_2\|_2 \leq (1+\delta)\|x_1 - x_2\|_2$$

Hence we construct a neural network $\Psi : \mathcal{M} \mapsto [0,1]^{d_\delta}$ as

$$\Psi(x) := \frac{1}{2\operatorname{diam}(\mathcal{M})} A x - \frac{1}{2\operatorname{diam}(\mathcal{M})} y_0,$$

where we choose y_0 appropriately so that

$$\frac{1-\delta}{2\operatorname{diam}(\mathcal{M})}\|x_1 - x_2\|_2 \leq \|\Psi(x_1) - \Psi(x_2)\|_2 \leq \frac{1+\delta}{2\operatorname{diam}(\mathcal{M})}\|x_1 - x_2\|_2$$

Network approximation - Proof

Idea of the proof: 1. Construction of low-dim function

We can now define a unique low-dimensional function g_0, with values on $[0,1]^{d_\delta}$, to represent the function f_0 defined on $\mathcal{M}$. For any $y \in \Psi(\mathcal{M}) \subseteq [0,1]^{d_\delta}$, we set

$$g_0(y) := f_0(x_y), \text{ where } x_y = \{x \in \mathcal{M} \,\big|\, \Psi(x) = y, \, y \in \Psi(\mathcal{M})\}.$$

Network approximation - Proof

Idea of the proof: 1. Construction of low-dim function

We can now define a unique low-dimensional function g_0, with values on $[0,1]^{d_\delta}$, to represent the function f_0 defined on $\mathcal{M}$. For any $y \in \Psi(\mathcal{M}) \subseteq [0,1]^{d_\delta}$, we set

$$g_0(y) := f_0(x_y), \text{ where } x_y = \{x \in \mathcal{M} \,\big|\, \Psi(x) = y,\, y \in \Psi(\mathcal{M})\}.$$

We next need to show that g_0 is Hölder continuous on $[0,1]^{d_\delta}$.

Network approximation - Proof

Idea of the proof: 1. Construction of low-dim function

We observe that, for $y_1, y_2 \in \Psi(\mathcal{M})$, there are $x_1, x_2 \in \mathcal{M}$ defined by $x_i = \{x \in \mathcal{M} \mid \Psi(x_i) = y_i\}$, $i = 1, 2$. Using the properties of f_0 and Ψ, it follows that

$$\begin{aligned}
& |g_0(y_1) - g_0(y_2)| \\
= \; & |f_0(x_1) - f_0(x_2)| \\
\leq \; & M\|x_1 - x_2\|_2^\alpha \\
\leq \; & M\left(\left(\tfrac{2\,\mathrm{diam}(\mathcal{M})}{1-\delta}\right)^\alpha \|\Psi(x_1) - \Psi(x_2)\|_2^\alpha\right) \\
\leq \; & \left(\tfrac{2\,\mathrm{diam}(\mathcal{M})}{1-\delta}\right)^\alpha M\|y_1 - y_2\|_2^\alpha.
\end{aligned}$$

We conclude that $g_0 \in B_\lambda(C^\alpha(\Psi(\mathcal{M})))$ with $\lambda = \left(\frac{2\,\mathrm{diam}(\mathcal{M})}{1-\delta}\right)^\alpha M$
That is, g_0 is Hölder continuous on $\Psi(\mathcal{M})$.

Network approximation - Proof

Idea of the proof: 2. Extension of low-dim function

We can extend the function g_0, originally defined on $\Psi(\mathcal{M})$, to a Hölder function $\widetilde{g_0}$ defined on $[0,1]^{d_\delta}$.

Extension Lemma

Let $g \in B_\lambda(C^\alpha(E))$, where $0 < \alpha \leq 1$, $\lambda > 0$ and $E \subseteq [0,1]^D$ is a closed set with $D \in \mathbb{N}$.
Then there exists a function $\tilde{g} \in B_{\lambda'}(C^\alpha([0,1^D]))$ with $\lambda' = 2D^{\alpha/2}\lambda$ such that $\tilde{g}(x) = g(x)$ for any $x \in E$.

Network approximation - Proof

Idea of the proof: 3. Approximation of $B_\lambda(C^\alpha([0,1]^{d_\delta}))$

Several contributions have shown how to build neural networks implementing functions efficiently.

Network approximation - Proof

Idea of the proof: 3. Approximation of $B_\lambda(C^\alpha([0,1]^{d_\delta}))$

Several contributions have shown how to build neural networks implementing functions efficiently.

Operations on neural networks [Petersen-Voigtlaender,2018]

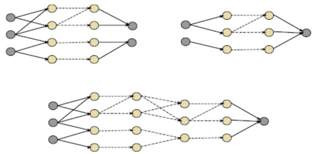
Network approximation - Proof

Idea of the proof: 3. Approximation of $B_\lambda(C^\alpha([0,1]^{d_\delta}))$

Several contributions have shown how to build neural networks implementing functions efficiently.

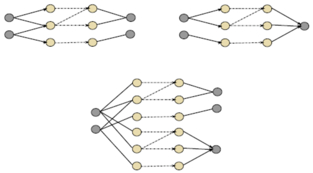
Operations on neural networks [Petersen-Voigtlaender,2018]

Concatenation. Given neural networks
$\Phi_1 \in \mathcal{F}(M_1, L_1, B_1)$ and $\Phi_2 \in \mathcal{F}(M_2, L_2, B_2)$,
the concatenation of Φ_1 and Φ_2
is another neural network
$\Phi_1 \circ \Phi_2 \in \mathcal{F}(2M_1 + 2M_2, L_1 + L_2 - 1, \max(B_1, B_2))$



Network approximation - Proof

Parallelization. Given neural networks $\Phi_1 \in \mathcal{F}(M_1, L, B_1)$ and $\Phi_2 \in \mathcal{F}(M_2, L, B_2)$, with the same input dimension, the parallelization of Φ_1 and Φ_2 is a neural network $P(\Phi_1, \Phi_2) \in \mathcal{F}(M_1 + M_2, L, \max(B_1, B_2))$



Network approximation - Proof

Idea of the proof: 3. Approximation of $B_\lambda(C^\alpha([0,1]^{d_\delta}))$

Proposition (Approximation of monomials). Fix $b > 0$ and $d_\delta \in \mathbb{N}$. For any $\epsilon > 0$ and $\nu \in \mathbb{N}_\delta^d$ with $|\nu| \le b$ there is a neural network $\Phi_\varepsilon^{\mathrm{mul}} \in \mathcal{F}(M, L, B)$ with

- $W \le 384\, 6^{d_\delta}\, b \left(119 + 36\lfloor 1/d_\delta \rfloor + (384)\, 4^{\lfloor 1/D \rfloor}\right) \epsilon^{-d_\delta}$,
- $L \le (1 + \lceil \log_2 \lfloor b \rfloor \rceil)(11 + 1/d_\delta)$,
- $B \le c(\epsilon, b, d_\delta)$

satisfying

$$\sup_{x \in [0,1]_\delta^d} \left| \Phi_\epsilon^{\mathrm{mul}}(x) - x^\nu \right| \le \epsilon.$$

Network approximation - Proof

Idea of the proof: 3. Approximation of $B_\lambda(C^\alpha([0,1]^{d_\delta}))$

Proposition (Approximation of monomials). Fix $b > 0$ and $d_\delta \in \mathbb{N}$. For any $\epsilon > 0$ and $\nu \in \mathbb{N}_\delta^d$ with $|\nu| \leq b$ there is a neural network $\Phi_\varepsilon^{\text{mul}} \in \mathcal{F}(M, L, B)$ with

- $W \leq 384\, 6^{d_\delta}\, b \left(119 + 36\lfloor 1/d_\delta \rfloor + (384)\, 4^{\lfloor 1/D \rfloor}\right) \epsilon^{-d_\delta}$,
- $L \leq (1 + \lceil \log_2 \lfloor b \rfloor \rceil)(11 + 1/d_\delta)$,
- $B \leq c(\epsilon, b, d_\delta)$

satisfying

$$\sup_{x \in [0,1]_\delta^d} \left| \Phi_\epsilon^{\text{mul}}(x) - x^\nu \right| \leq \epsilon.$$

It follows that, for $\tilde{g}_0 \in B_\lambda(C^\alpha([0, 1^D]))$, we can construct a neural network $\Phi^{\widetilde{g_0}}$ satisfying similar approximation properties.

Network approximation - Proof

Idea of the proof: 4. Conclusion

We can finally approximate f_0 using a neural network.

Using the $\Phi^{\widetilde{g}_0}$ and Ψ constructed above, we define the neural network

$$\Phi^{f_0} = \Phi^{\widetilde{g}_0} \odot \Psi.$$

For any $x \in \mathcal{M}$, given any $\epsilon > 0$, we have that

$$\begin{aligned} |f_0(x) - \Phi^{f_0}(x)| &= |g_0(\Psi(x)) - \Phi^{\widetilde{g}_0}(\Psi(x))| \\ &= |\widetilde{g}_0(\Psi(x)) - \Phi^{\widetilde{g}_0}(\Psi(x))| \\ &\leq \varepsilon \end{aligned}$$

Application: Generalization error

We next apply our approximation result to the problem of controlling the **generalization error** in a regression problem.

Application: Generalization error

We next apply our approximation result to the problem of controlling the **generalization error** in a regression problem.

Let us consider a **nonparametric regression problem** corresponding to n observations $\{(X_i, Y_i)\}_{i=1}^n \in [0,1]^D \times \mathbb{R}$ from the model

$$\boxed{Y_i = f_0(X_i) + \varepsilon_i, \quad i = 1, \cdots, n,}$$

Application: Generalization error

We next apply our approximation result to the problem of controlling the **generalization error** in a regression problem.

Let us consider a **nonparametric regression problem** corresponding to n observations $\{(X_i, Y_i)\}_{i=1}^n \in [0,1]^D \times \mathbb{R}$ from the model

$$\boxed{Y_i = f_0(X_i) + \varepsilon_i, \quad i = 1, \cdots, n,}$$

where

- $f_0 \in B_\lambda(C^\alpha([0,1]^D))$,
- the covariates X_i marginally follow a probability measure μ,
- the errors ε_i are i.i.d normally distributed with mean 0 and variance σ^2 and are independent of the X_i.

Application: Generalization error

The solution of the regression problem is an estimator $\widehat{f}$ approximating the unknown function $f_0 \in B_\lambda(C^\alpha([0,1]^D))$

Application: Generalization error

The solution of the regression problem is an estimator $\widehat{f}$ approximating the unknown function $f_0 \in B_\lambda(C^\alpha([0,1]^D))$

The performance of the estimator is assessed by the **generalization error**

$$\|\widehat{f} - f_0\|^2_{L^2(\mu)} = E_{X\sim\mu}\left[(\widehat{f}(X) - f_0(X))^2\right]$$

Application: Generalization error

The solution of the regression problem is an estimator $\widehat{f}$ approximating the unknown function $f_0 \in B_\lambda(C^\alpha([0,1]^D))$

The performance of the estimator is assessed by the **generalization error**

$$\|\widehat{f} - f_0\|^2_{L^2(\mu)} = E_{X\sim\mu}\left[(\widehat{f}(X) - f_0(X))^2\right]$$

We identify the estimator class with neural networks $\mathcal{F}(N, L, B)$

Application: Generalization error

The solution of the regression problem is an estimator $\widehat{f}$ approximating the unknown function $f_0 \in B_\lambda(C^\alpha([0,1]^D))$

The performance of the estimator is assessed by the **generalization error**

$$\|\widehat{f} - f_0\|^2_{L^2(\mu)} = E_{X\sim\mu}\left[(\widehat{f}(X) - f_0(X))^2\right]$$

We identify the estimator class with neural networks $\mathcal{F}(N, L, B)$

Fact: The generalization error using neural networks is on the order

$$O(n^{-2\alpha/(2\alpha+D)})$$

This rate is optimal in the minimax sense [Schmidt-Hieber, 2020]

Application: Generalization error

The solution of the regression problem is an estimator $\widehat{f}$ approximating the unknown function $f_0 \in B_\lambda(C^\alpha([0,1]^D))$

The performance of the estimator is assessed by the **generalization error**

$$\|\widehat{f} - f_0\|^2_{L^2(\mu)} = E_{X\sim\mu}\left[(\widehat{f}(X) - f_0(X))^2\right]$$

We identify the estimator class with neural networks $\mathcal{F}(N, L, B)$

Fact: The generalization error using neural networks is on the order

$$O(n^{-2\alpha/(2\alpha+D)})$$

This rate is optimal in the minimax sense [Schmidt-Hieber, 2020]

- Generalization error suffers from the curse of dimensionality

Application: Generalization error

The solution of the regression problem is an estimator $\widehat{f}$ approximating the unknown function $f_0 \in B_\lambda(C^\alpha([0,1]^D))$

The performance of the estimator is assessed by the **generalization error**

$$\|\widehat{f} - f_0\|^2_{L^2(\mu)} = E_{X\sim\mu}\left[(\widehat{f}(X) - f_0(X))^2\right]$$

We identify the estimator class with neural networks $\mathcal{F}(N, L, B)$

Fact: The generalization error using neural networks is on the order

$$O(n^{-2\alpha/(2\alpha+D)})$$

This rate is optimal in the minimax sense [Schmidt-Hieber, 2020]

- Generalization error suffers from the curse of dimensionality
- The network complexity also depends on D

Application: Generalization error

Our approach: **manifold assumption**

Application: Generalization error

Our approach: **manifold assumption**

We assume data lies on a d-dimensional manifold with $d \ll D$

Application: Generalization error

Our approach: **manifold assumption**

We assume data lies on a d-dimensional manifold with $d \ll D$

To estimate the regression function f_0, we compute the least square estimator $\widehat{\Phi} \in \mathcal{F}(M, L, B)$ of f_0 associated with the **empirical risk minimization**

$$\widehat{\Phi} = \underset{\substack{\Phi = g \circ A \\ g \in \mathcal{F}(M,L,B)}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} (Y_i - \Phi(X_i))^2 ,$$

Application: Generalization error

Our approach: **manifold assumption**

We assume data lies on a d-dimensional manifold with $d \ll D$

To estimate the regression function f_0, we compute the least square estimator $\widehat{\Phi} \in \mathcal{F}(M, L, B)$ of f_0 associated with the **empirical risk minimization**

$$\widehat{\Phi} = \underset{\substack{\Phi = g \circ A \\ g \in \mathcal{F}(M,L,B)}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} (Y_i - \Phi(X_i))^2 ,$$

with the estimator returning a neural network $\widehat{\Phi}$ of the form $g \circ A$ where $g \in \mathcal{F}(M, L, B)$ and A is a random orthoprojection

$$A : \mathbb{R}^D \to \mathbb{R}^{d_\delta} \quad d < d_\delta < D$$

Application: Generalization error

Our result [Labate,Shi, 2022]

Application: Generalization error

Our result [Labate,Shi, 2022]

Theorem (informal version)

Let $\mathcal{M} \subset \mathbb{R}^D$ be a Riemannian d-dimensional manifold (with some regularity) and let $f \in B_\lambda(C^\alpha([0,1]^D))$, $\alpha \in (0,1)$. Let $\widehat{\Phi}$ be the solution of the empirical risk minimization problem given above. Then there exists a constant $c = c(\sigma, \beta, d_\delta, \lambda)$ such that

$$\|\widehat{\Phi} - f_0\|^2_{L^2(\mu)} \leq c\, n^{-2\alpha/(2\alpha+d_\delta)}(1 + \log n)^2$$

holds with probability at least $1 - 2\exp(-n^{d_\delta/(2\alpha+d_\delta)})$ for n large.

Application: Generalization error

Our result [Labate,Shi, 2022]

Theorem (informal version)

Let $\mathcal{M} \subset \mathbb{R}^D$ be a Riemannian d-dimensional manifold (with some regularity) and let $f \in B_\lambda(C^\alpha([0,1]^D))$, $\alpha \in (0,1)$. Let $\widehat{\Phi}$ be the solution of the empirical risk minimization problem given above. Then there exists a constant $c = c(\sigma, \beta, d_\delta, \lambda)$ such that

$$\|\widehat{\Phi} - f_0\|^2_{L^2(\mu)} \leq c\, n^{-2\alpha/(2\alpha+d_\delta)}(1+\log n)^2$$

holds with probability at least $1 - 2\exp(-n^{d_\delta/(2\alpha+d_\delta)})$ for n large.

- Constant c does not depend on D, improving existing result [Nakada and Imaizumi, 2019]

Application: Generalization error

Our result [Labate,Shi, 2022]

Theorem (informal version)

Let $\mathcal{M} \subset \mathbb{R}^D$ be a Riemannian d-dimensional manifold (with some regularity) and let $f \in B_\lambda(C^\alpha([0,1]^D))$, $\alpha \in (0,1)$. Let $\widehat{\Phi}$ be the solution of the empirical risk minimization problem given above. Then there exists a constant $c = c(\sigma, \beta, d_\delta, \lambda)$ such that

$$\|\widehat{\Phi} - f_0\|^2_{L^2(\mu)} \leq c\, n^{-2\alpha/(2\alpha+d_\delta)}(1 + \log n)^2$$

holds with probability at least $1 - 2\exp(-n^{d_\delta/(2\alpha+d_\delta)})$ for n large.

- Constant c does not depend on D, improving existing result [Nakada and Imaizumi, 2019]
- Complexity of the network depends weakly on D. M depends linearly with D and L, B do not depend on D but only on d_δ.