Provable approximations on smooth manifolds using deep neural networks

Demetrio Labate
(University of Houston)

11th International Conference on Harmonic Analysis and Partial Differential Equations
June 6-10, 2022

Outline

Outline...

Function classes of Neural Networks

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

$$\Phi(x) = W_L \circ \rho \circ W_{L-1} \circ \dots \rho \circ W_1(x), \quad x \in \mathbb{R}^D$$

Function classes of Neural Networks

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

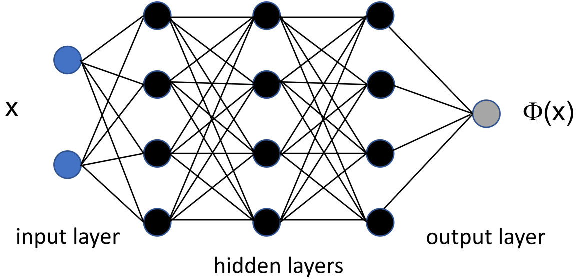
$$\Phi(x) = W_L \circ \rho \circ W_{L-1} \circ \ldots \rho \circ W_1(x), \quad x \in \mathbb{R}^D$$

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

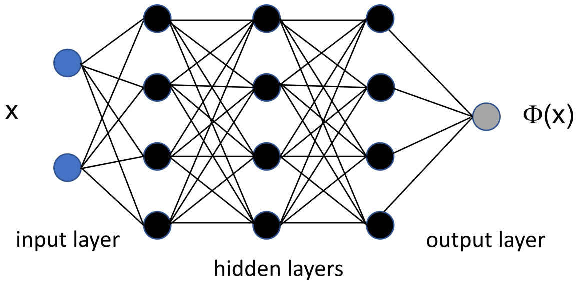
Function classes of Neural Networks

Graph representation



Function classes of Neural Networks

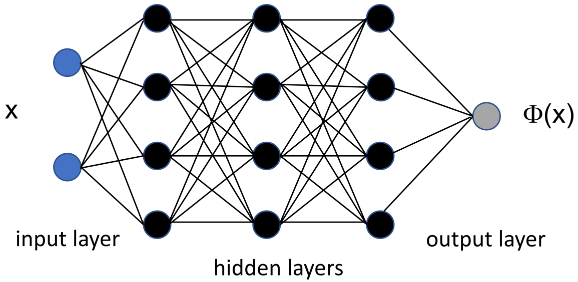
Graph representation



- Number of layers $L = 4$

Function classes of Neural Networks

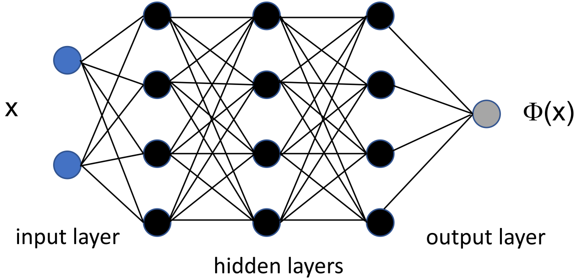
Graph representation



- 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 $M = \sum_{\ell=1}^{4} ||A_\ell||_0 + ||b_\ell||_0 = 44 + 12 = 56$

Function classes of Neural Networks

Assumption: I wil identify a neural network with the **function** implemented by the neural network

(In practice, multiple graph representations may realize the same function)

Function classes of Neural Networks

Assumption: I wil identify a neural network with the **function** implemented by the neural network

(In practice, multiple graph representations may realize the same function)

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:

$$\mathcal{F}(M_0, L_0, B_0) = \left\{ \Phi \colon [0,1]^D \to \mathbb{R}^{N_L} \,\middle|\, L(\Phi) \leq L_0, W(\Phi) \leq W_0, B(\Phi) \leq B_0 \right\}$$

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

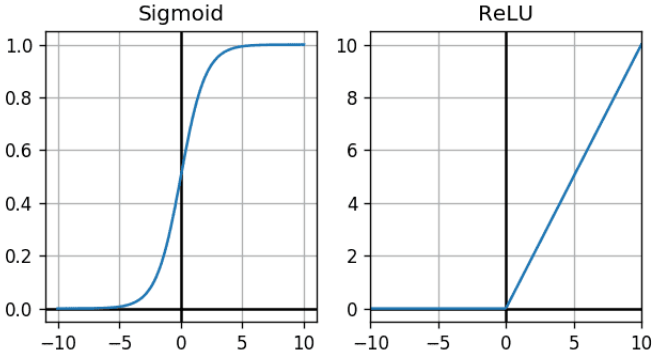
Examples of activation functions:

- **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:

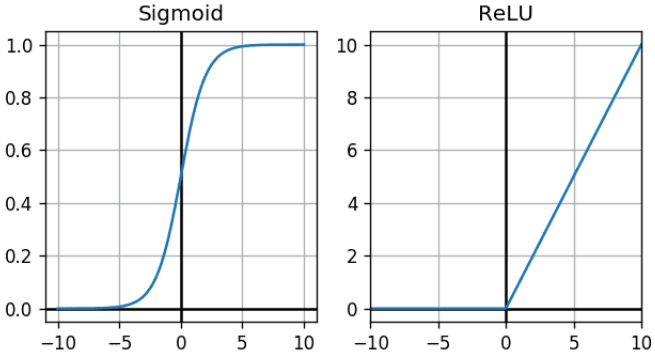
- **Sigmoid**: $\rho(x) = \frac{1}{1+e^{-x}}$
- **Rectified linear unit (ReLU)**: $\rho(x) = \max\{x, 0\}$



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

Outline...

Approximations using Neural Networks

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

Approximations using Neural Networks

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

They state that: *continuous functions on compact domains can be approximated arbitrarily well by neural networks.*

Approximations using Neural Networks

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

They state that: *continuous functions on compact domains can be approximated arbitrarily well by neural networks.*

Theorem [Cybenko, 1989]

Assume ρ is a sigmoidal activation function.
For every $f \in C([0,1]^D)$, given $\varepsilon > 0$, there exists $\Phi \in \mathcal{F}(M, L = 2, B)$, such that

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

Approximations using Neural Networks

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

They state that: *continuous functions on compact domains can be approximated arbitrarily well by neural networks.*

Theorem [Cybenko, 1989]

Assume ρ is a sigmoidal activation function.
For every $f \in C([0,1]^D)$, given $\varepsilon > 0$, there exists $\Phi \in \mathcal{F}(M, L = 2, B)$, such that

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

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

Approximations using Neural Networks

- Hornik (1991): universal approximation holds with continuous, bounded, non-constant activation functions and $L = 2$

Approximations using Neural Networks

- Hornik (1991): universal approximation holds with continuous, bounded, non-constant activation functions and $L = 2$
- Number weights $M(\Phi)$ and number of neurons N_2 of Φ can become arbitrarily large.

Approximations using Neural Networks

- Hornik (1991): universal approximation holds with continuous, bounded, non-constant activation functions and $L = 2$
- Number weights $M(\Phi)$ and number of neurons N_2 of Φ can become arbitrarily large.
- Idea of the proof:
  - $\mathcal{F}(M, L = 2, B) \subset C([0, 1]^D)$ is a linear subspace

Approximations using Neural Networks

- Hornik (1991): universal approximation holds with continuous, bounded, non-constant activation functions and $L = 2$
- Number weights $M(\Phi)$ and number of neurons N_2 of Φ can become arbitrarily large.
- 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

Approximations using Neural Networks

- Hornik (1991): universal approximation holds with continuous, bounded, non-constant activation functions and $L = 2$
- Number weights $M(\Phi)$ and number of neurons N_2 of Φ can become arbitrarily large.
- 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 Thm., there exists a signed Radon measure μ such that $\int_{[0,1]^D} \Phi(x)\, d\mu(x) = 0$ for all $\Phi \in \mathcal{F}(M, L = 2, B)$.

Approximations using Neural Networks

- Hornik (1991): universal approximation holds with continuous, bounded, non-constant activation functions and $L = 2$
- Number weights $M(\Phi)$ and number of neurons N_2 of Φ can become arbitrarily large.
- 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 Thm., 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 $a \in \mathbb{R}^D$, $b \in \mathbb{R}$.

Approximations using Neural Networks

- Hornik (1991): universal approximation holds with continuous, bounded, non-constant activation functions and $L = 2$
- Number weights $M(\Phi)$ and number of neurons N_2 of Φ can become arbitrarily large.
- 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 Thm., 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.

Approximations using Neural Networks

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

Approximations using Neural Networks

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 $\varepsilon > 0$, 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

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

Approximations using Neural Networks

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

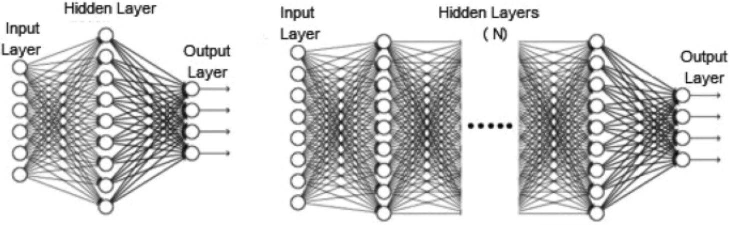
Theorem [Kidger and Lyons, 2020]

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 $\varepsilon > 0$, 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

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

Note: number of layers $L(\Phi)$ can become arbitrarily large.

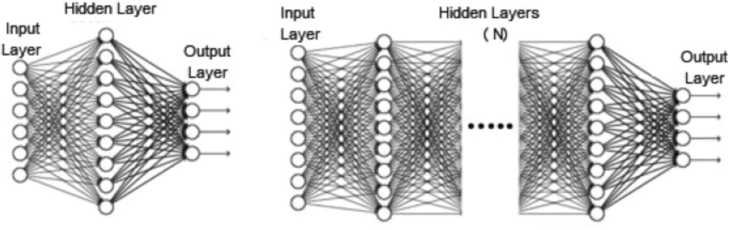
Deep Neural Networks



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

- Modern network architectures are typically very deep

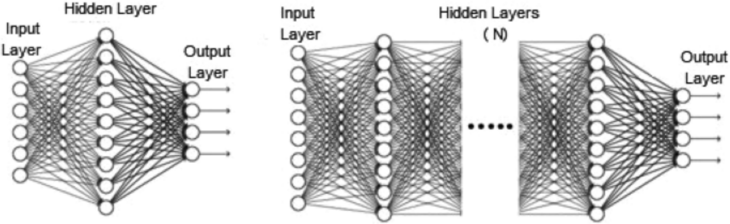
Deep Neural Networks



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

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

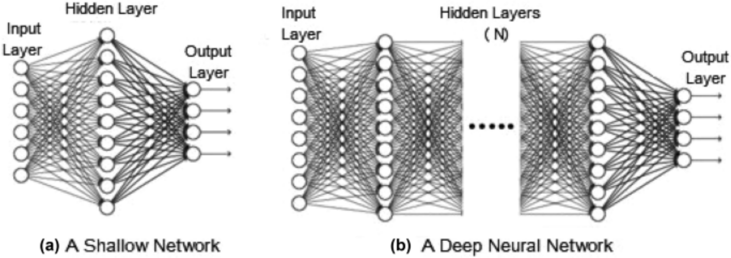
Deep Neural Networks



(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



- 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** → **Blessing of compositionality**

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

Triangle function:

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

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

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

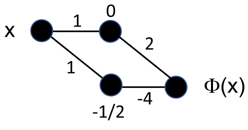
Example: piecewise linear functions on $\mathbb{R}$

Triangle function:

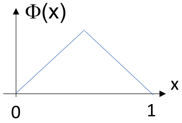
$$T(x) = \begin{cases} 2x & \text{if } 0 \le x < \frac{1}{2} \\ 2(1-x) & \text{if } \frac{1}{2} \le x \le 1 \end{cases} \qquad x \in \mathbb{R},$$

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

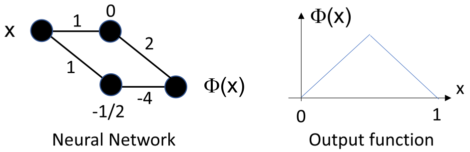
Example: piecewise linear functions on $\mathbb{R}$

Triangle function:

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

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

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



- $\Phi(x)$ 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

Shallow network are comparable to traditional approximation methods in terms of computational complexity.

Approximations using Neural Networks

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

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

Approximations using Neural Networks

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

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

- 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

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

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

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

- 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

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

- 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

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

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

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

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

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

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

- 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

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

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

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

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

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

- 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

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

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

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

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

- 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

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

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

Outline...

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 $M = O(\varepsilon^{-D})$ parameters $\rightarrow$ practically impossible to achieve satisfactory accuracy for very large D

Curse of Dimensionality

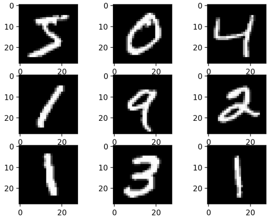
In many multi-dimensional problems, data is highly structured and the information of interest is low-dimensional

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

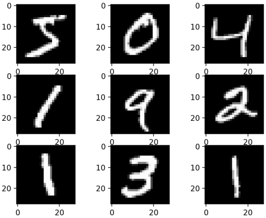


Curse of Dimensionality

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 $\to \mathbb{R}^{784}$
  - intrinsic dimension: between 8 and 13



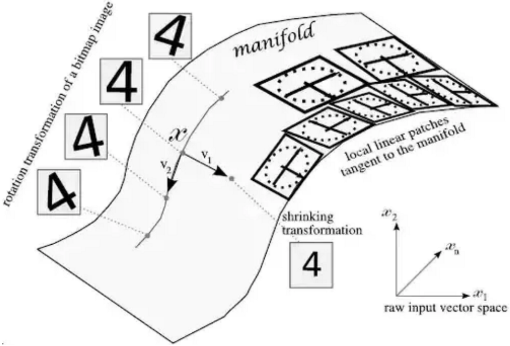
- ImageNet: $224 \times 224 \times 3 = 150528$ pixels per image $\to \mathbb{R}^{150528}$
  - intrinsic dimension: between 26 and 43

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 $d \ll D$



Manifold hypothesis

We want to approximate a function

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

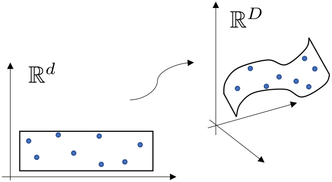
Under the manifold hypothesis, we are not seeking to approximate f with respect to a norm on $\mathbb{R}^D$.

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 $d \ll D$.



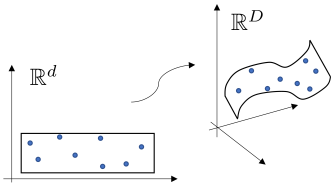
D : **ambient dimension** vs. d: **intrinsic dimension**

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

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

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

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

- Complexity grows like $\varepsilon^{-d/\alpha}$ (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

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

- Complexity grows like $\varepsilon^{-d/\alpha}$ (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

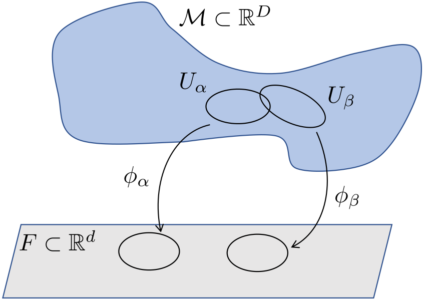
Idea of the proof [Shaham,Cloninger,Coifman,2018]

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]

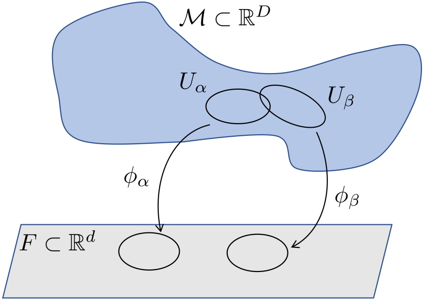
1. Construct a basis or a frame (e.g., a wavelet frame or a polynomial basis) of $C^\alpha([0,1]^D)$ whose elements are implemented as neural networks
2. Construct an atlas for $\mathcal{M} \subset \mathbb{R}^D$ by covering it with open balls



Manifold hypothesis

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

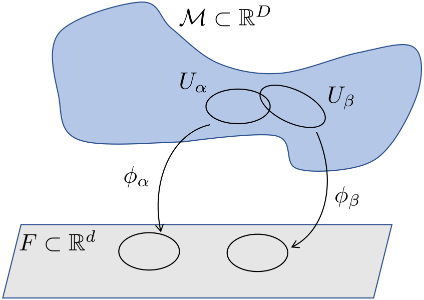
1. Construct a basis or a frame (e.g., a wavelet frame or a polynomial basis) of $C^\alpha([0,1]^D)$ whose elements are implemented as neural networks
2. Construct an atlas for $\mathcal{M} \subset \mathbb{R}^D$ by covering it with open balls
3. Use the open cover to obtain a partition of unity of $\mathcal{M}$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

Our result [Labate,Shi, 2022]

Network approximation - Manifold hypothesis

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 $f \in B_\lambda(C^\alpha([0,1]^D))$, $\alpha \in (0,1)$. For $\delta \in (0,1)$,

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

where $d < d_\delta < D$. The complexity of a deep networks Φ that approximates f with accuracy at least ε satisfies

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

Network approximation - Manifold hypothesis

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 $f \in B_\lambda(C^\alpha([0,1]^D))$, $\alpha \in (0,1)$. For $\delta \in (0,1)$,

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

where $d < d_\delta < D$. The complexity of a deep networks Φ that approximates f with accuracy at least ε satisfies

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

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

Network approximation - Manifold hypothesis

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 $f \in B_\lambda(C^\alpha([0,1]^D))$, $\alpha \in (0,1)$. For $\delta \in (0,1)$,

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

where $d < d_\delta < D$. The complexity of a deep networks Φ that approximates f with accuracy at least ε satisfies

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

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

Johnson-Lindenstrauss Lemma

One key tool in our approach is an appropriate version of the **Johnson-Lindenstrauss lemma**, establishing low-distortion embeddings of points from high-dimensional into low-dimensional space.

Johnson-Lindenstrauss Lemma

One key tool in our approach is an appropriate version of the **Johnson-Lindenstrauss lemma**, establishing low-distortion embeddings of points from high-dimensional 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

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

Johnson-Lindenstrauss Lemma

One key tool in our approach is an appropriate version of the **Johnson-Lindenstrauss lemma**, establishing low-distortion embeddings of points from high-dimensional 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

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

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

Johnson-Lindenstrauss Lemma

One key tool in our approach is an appropriate version of the **Johnson-Lindenstrauss lemma**, establishing low-distortion embeddings of points from high-dimensional 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

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

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

Johnson-Lindenstrauss Lemma

One key tool in our approach is an appropriate version of the **Johnson-Lindenstrauss lemma**, establishing low-distortion embeddings of points from high-dimensional 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, ...
- Recent application in the context of approximations using neural networks: [Cai, Li, Sun, Wang, 2019], [Shen, Yang, Zhang, 2021]

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 [Baraniuk and Wakin, 2009]

Let $\mathcal{M} \subset \mathbb{R}^D$ be a compact d-dimensional Riemannian submanifold having condition number $1/\tau$, volume V, and geodesic covering regularity $\mathcal{R}$. Fix $\delta, \gamma \in (0,1)$ and let A be a random orthoprojector from $\mathbb{R}^D$ to $\mathbb{R}^{d_\delta}$ with

$$d_\delta = \mathcal{O}\left(\frac{d \log(DV\mathcal{R}\tau^{-1}\delta^{-1})}{\delta^2}\right) = \mathcal{O}\left(d\frac{\log(D\delta^{-1})}{\delta^2}\right)$$

Then, for every pair of points $x_1, x_2 \in \mathcal{M}$

$$(1-\delta)\sqrt{\frac{d_\delta}{D}}\,|x_1 - x_2| \le |Ax_1 - Ax_2| \le (1+\delta)\sqrt{\frac{d_\delta}{D}}\,|x_1 - x_2|$$

holds with probability at least $1-\gamma$.

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** $1/\tau \to$ norm of the second fundamental form of $\mathcal{M}$ is bounded by $1/\tau$ in all directions:

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:
  - manifold cannot curve too much locally

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:
  - 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** $1/\tau \to$ norm of the second fundamental form of $\mathcal{M}$ is bounded by $1/\tau$ in all directions:
    - 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** $1/\tau \to$ norm of the second fundamental form of $\mathcal{M}$ is bounded by $1/\tau$ in all directions:
  - 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
- **Geodesic covering regularity** of the manifold: closely related to the more traditional notion of distance covering number

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:
  - 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
- **Geodesic covering regularity** of the manifold: closely related to the more traditional notion of distance covering number
  - it ensures that any ball centered on the manifold captures a certain amount of its volume

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:
  - 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
- **Geodesic covering regularity** of the manifold: closely related to the more traditional notion of distance covering number
  - it ensures that any ball centered on the manifold captures a certain amount of its volume

Random orthoprojection $A : \mathbb{R}^D \to \mathbb{R}^{d_\delta}$

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:
  - 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
- **Geodesic covering regularity** of the manifold: closely related to the more traditional notion of distance covering number
  - it ensures that any ball centered on the manifold captures a certain amount of its volume

Random orthoprojection $A : \mathbb{R}^D \rightarrow \mathbb{R}^{d_\delta}$

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

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:
  - 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
- **Geodesic covering regularity** of the manifold: closely related to the more traditional notion of distance covering number
  - it ensures that any ball centered on the manifold captures a certain amount of its volume

Random orthoprojection $A : \mathbb{R}^D \rightarrow \mathbb{R}^{d_\delta}$

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

- δ is closer to $1 \Rightarrow d_\delta$ 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:
  - 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
- **Geodesic covering regularity** of the manifold: closely related to the more traditional notion of distance covering number
  - it ensures that any ball centered on the manifold captures a certain amount of its volume

Random orthoprojection $A : \mathbb{R}^D \to \mathbb{R}^{d_\delta}$

$\delta \in (0, 1)$ controls 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 result - Idea of the proof

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 Baraniuk and Wakin, 2009]

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 Baraniuk and Wakin, 2009]
2. Extend g_0 continuously to a function $\tilde{g}_0$ on $B_\lambda(C^\alpha([0,1]^{d_\delta}))$

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 Baraniuk and Wakin, 2009]
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 Baraniuk and Wakin, 2009]
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 Baraniuk and Wakin, 2009]
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 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 Baraniuk and Wakin, 2009]
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 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 - 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 Baraniuk and Wakin, 2009]
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 approximate $B_\lambda(C^\alpha([0,1]^D))$ using neural networks
- To control number of parameters of Φ^{g_0} and Φ^{f_0}
- To ensure the projected function g_0 is Hölder continuous

Network approximation - Manifold hypothesis

Idea of the proof: 1-2. Approximation of $B_\lambda(C^\alpha([0,1]^D))$

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

Network approximation - Manifold hypothesis

Idea of the proof: 1-2. Approximation of $B_\lambda(C^\alpha([0,1]^D))$
Several contributions have shown how to build neural networks implementing functions efficiently.

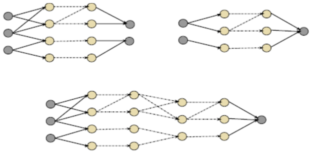
Operations on neural networks [Petersen-Voigtlaender,2018]

Network approximation - Manifold hypothesis

Idea of the proof: 1-2. Approximation of $B_\lambda(C^\alpha([0,1]^D))$
Several contributions have shown how to build neural networks implementing functions efficiently.

Operations on neural networks [Petersen-Voigtlaender,2018]

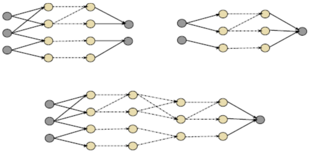
Concatenation. Given neural networks $\Phi_1 \in \mathcal{F}(M_1, L_1, B_1)$ and $\Phi_2 \in \mathcal{F}(M_2, L_2, B_2)$, the concatenation of Φ_1 and Φ_2 is another neural network $\Phi_1 \circ \Phi_2 \in \mathcal{F}(2M_1 + 2M_2, L_1 + L_2 - 1, \max(B_1, B_2))$



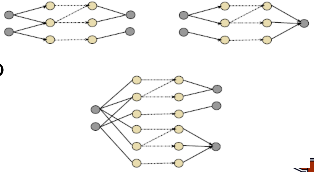
Network approximation - Manifold hypothesis

Idea of the proof: 1-2. Approximation of $B_\lambda(C^\alpha([0,1]^D))$
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
$\Phi_1 \circ \Phi_2 \in \mathcal{F}(2M_1 + 2M_2, L_1 + L_2 - 1, \max(B_1, B_2))$



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 parallelizatio
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 - Manifold hypothesis

Idea of the proof: 1-2. Approximation of $B_\lambda(C^\alpha([0,1]^D))$

Proposition (Approximation of monomials). Fix $b > 0$ and $D \in \mathbb{N}$. For any $\varepsilon > 0$ and $\nu \in \mathbb{N}^D$ with $|\nu| \leq b$ there is a neural network $\Phi_\varepsilon^{\mathrm{mul}} \in \mathcal{F}(M, L, B)$ with D-dimensional input and 1-dimensional output satisfying

$$\sup_{x \in [0,1]^D} \left| \Phi_\varepsilon^{\mathrm{mul}}(x) - x^\nu \right| \leq \varepsilon.$$

Idea of the proof: 1-2. Approximation of $B_\lambda(C^\alpha([0,1]^D))$

Proposition (Approximation of monomials). Fix $b > 0$ and $D \in \mathbb{N}$. For any $\varepsilon > 0$ and $\nu \in \mathbb{N}^D$ with $|\nu| \le b$ there is a neural network $\Phi_\varepsilon^{\mathrm{mul}} \in \mathcal{F}(M, L, B)$ with D-dimensional input and 1-dimensional output satisfying

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

We can choose $M = N(\Phi_\varepsilon^{\mathrm{mul}})$, $L = L(\Phi_\varepsilon^{\mathrm{mul}})$, $B = B(\Phi_\varepsilon^{\mathrm{mul}})$ such that

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

Network approximation - Manifold hypothesis

Idea of the proof: 1-2. Approximation of $B_\lambda(C^\alpha([0,1]^D))$

Proposition (Approximation of monomials). Fix $b > 0$ and $D \in \mathbb{N}$. For any $\varepsilon > 0$ and $\nu \in \mathbb{N}^D$ with $|\nu| \le b$ there is a neural network $\Phi_\varepsilon^{\text{mul}} \in \mathcal{F}(M, L, B)$ with D-dimensional input and 1-dimensional output satisfying

$$\sup_{x \in [0,1]^D} \left| \Phi_\varepsilon^{\text{mul}}(x) - x^\nu \right| \le \varepsilon.$$

We can choose $M = N(\Phi_\varepsilon^{\text{mul}})$, $L = L(\Phi_\varepsilon^{\text{mul}})$, $B = B(\Phi_\varepsilon^{\text{mul}})$ such that

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

Proof. Refinement of result in [Petersen and Voigtlaender, 2018].

Network approximation - Manifold hypothesis

Idea of the proof: 3. Construction of low-dimensional function

We apply orthoprojection $A : \mathbb{R}^D \supset \mathcal{M} \to \mathbb{R}^{d_\delta}$ to define a function g on $A(\mathcal{M})$

Idea of the proof: 3. Construction of low-dimensional function
We apply orthoprojection $A : \mathbb{R}^D \supset \mathcal{M} \to \mathbb{R}^{d_\delta}$ to define a function g on $A(\mathcal{M})$

g is not guaranteed to be Hölder continuous but can be approximated uniformly with small error by a function $\tilde{g} \in B_{\tilde{\lambda}}(C^\alpha([0,1]^{d_\delta}))$

Idea of the proof: 3. Construction of low-dimensional function
We apply orthoprojection $A : \mathbb{R}^D \supset \mathcal{M} \to \mathbb{R}^{d_\delta}$ to define a function g on $A(\mathcal{M})$

g is not guaranteed to be Hölder continuous but can be approximated uniformly with small error by a function $\tilde{g} \in B_{\tilde{\lambda}}(C^\alpha([0,1]^{d_\delta}))$
By concatenation we can define an approximating neural network, $\Phi^f = \Phi^{\widetilde{g}} \odot ((A, 0))$ giving

$$|g(y) - \Phi^f(x)| \leq |g(y) - \tilde{g}(y)| + |\tilde{g}(Ax) - \Phi^{\tilde{g}}(Ax)|$$

Application: Generalization error

We show an application of the approximation result to the problem of controlling the **generalization error** in a regression problem.

Application: Generalization error

We show an application of the 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 show an application of the 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**

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

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**

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

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.

Outline...

Conclusion

- Application of Johnson-Lindenstrauss lemma is useful to manage COD using neural networks without the need to define an explicit atlas from the ambient space $\mathbb{R}^D$ into the lower dimensional space $\mathbb{R}^d$ and improved control over the dependence of the parametrization of the network on D.

Conclusion

- Application of Johnson-Lindenstrauss lemma is useful to manage COD using neural networks without the need to define an explicit atlas from the ambient space $\mathbb{R}^D$ into the lower dimensional space $\mathbb{R}^d$ and improved control over the dependence of the parametrization of the network on D.
- The drawback is that this approach offers limited control over the regularity of f

Conclusion

- Application of Johnson-Lindenstrauss lemma is useful to manage COD using neural networks without the need to define an explicit atlas from the ambient space $\mathbb{R}^D$ into the lower dimensional space $\mathbb{R}^d$ and improved control over the dependence of the parametrization of the network on D.
- The drawback is that this approach offers limited control over the regularity of f
- How to adapt this approach to smooth and piecewise smooth functions such as Hölder function with smoothness $\alpha > 1$ and Besov spaces?

Questions?

References + codes at: `www.math.uh.edu\~dlabate`