Analysis of Neural Network

Jinchao Xu 许进超

Penn State University

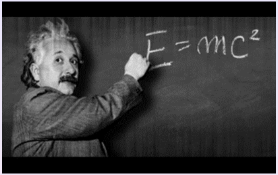
xu@math.psu.edu http://www.math.psu.edu/xu/

CUHK, June 2021

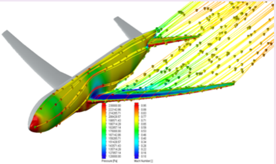
Acknowledgement:
NSF: DMS-1819157

Four major methods of scientific research

Experimental Science



Theoretical Science

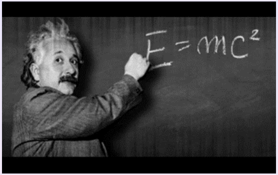


Computational Science

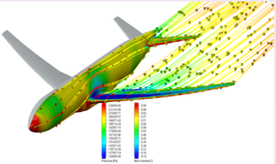
Data Science

Four major methods of scientific research

Experimental Science



Theoretical Science



Computational Science

Data Science

Question:

Is "Data Science" really a science?

A basic AI problem: classification

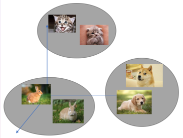
- Can a machine (function) tell the difference ?

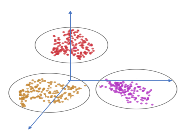
Supervised learning

- Function interpolation (data fitting)
  - Each image = a big vector of pixel values
    - $d = 1280 \times 720 \times 3$ (width$\times$ height $\times$ RGB channel) $\approx$ 3M.

Supervised learning

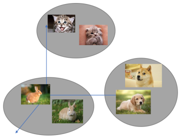
- Function interpolation (data fitting)
  - Each image = a big vector of pixel values
    - ★ $d = 1280 \times 720 \times 3$ (width$\times$ height $\times$ RGB channel) $\approx$ 3M.
  - 3 different sets of points in $\mathbb{R}^d$, are they separable?

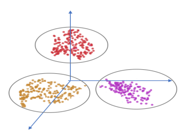




Supervised learning

- Function interpolation (data fitting)
  - Each image = a big vector of pixel values
    - $d = 1280 \times 720 \times 3$ (width$\times$ height $\times$ RGB channel) $\approx$ 3M.
  - 3 different sets of points in $\mathbb{R}^d$, are they separable?

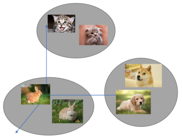


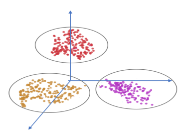


  - Mathematical problem: Find $f(\cdot;\Theta) : \mathbb{R}^d \to \mathbb{R}^3$ such that:

Supervised learning

- Function interpolation (data fitting)
  - Each image = a big vector of pixel values
    - ★ $d = 1280 \times 720 \times 3$ (width$\times$ height $\times$ RGB channel) $\approx$ 3M.
  - 3 different sets of points in $\mathbb{R}^d$, are they separable?



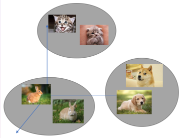


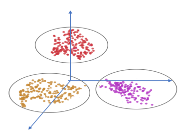
  - Mathematical problem: Find $f(\cdot;\Theta) : \mathbb{R}^d \to \mathbb{R}^3$ such that:

$f($$;\Theta) \approx \begin{pmatrix}1\\0\\0\end{pmatrix}$

Supervised learning

- Function interpolation (data fitting)
  - Each image = a big vector of pixel values
    - ★ $d = 1280 \times 720 \times 3$ (width$\times$ height $\times$ RGB channel) $\approx$ 3M.
  - 3 different sets of points in $\mathbb{R}^d$, are they separable?



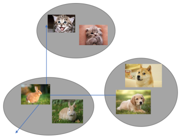


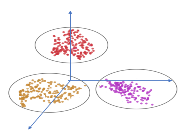
  - Mathematical problem: Find $f(\cdot;\Theta) : \mathbb{R}^d \to \mathbb{R}^3$ such that:

$f($$;\Theta) \approx \begin{pmatrix}1\\0\\0\end{pmatrix}$ $\quad$ $f($$;\Theta) \approx \begin{pmatrix}0\\1\\0\end{pmatrix}$

Supervised learning

- Function interpolation (data fitting)
  - Each image = a big vector of pixel values
    - ★ $d = 1280 \times 720 \times 3$ (width$\times$ height $\times$ RGB channel) $\approx$ 3M.
  - 3 different sets of points in $\mathbb{R}^d$, are they separable?



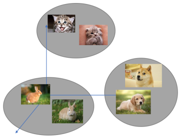
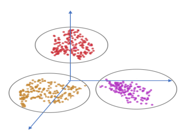


  - Mathematical problem: Find $f(\cdot;\Theta) : \mathbb{R}^d \to \mathbb{R}^3$ such that:

$f($  $;\Theta) \approx \begin{pmatrix}1\\0\\0\end{pmatrix}$ $f($  $;\Theta) \approx \begin{pmatrix}0\\1\\0\end{pmatrix}$ $f($  $;\Theta) \approx \begin{pmatrix}0\\0\\1\end{pmatrix}$

Supervised learning

- Function interpolation (data fitting)
  - Each image = a big vector of pixel values
    - ★ $d = 1280 \times 720 \times 3$ (width$\times$ height $\times$ RGB channel) $\approx$ 3M.
  - 3 different sets of points in $\mathbb{R}^d$, are they separable?

  - Mathematical problem: Find $f(\cdot;\Theta) : \mathbb{R}^d \to \mathbb{R}^3$ such that:

$f($$;\Theta) \approx \begin{pmatrix}1\\0\\0\end{pmatrix}$ $f($$;\Theta) \approx \begin{pmatrix}0\\1\\0\end{pmatrix}$ $f($$;\Theta) \approx \begin{pmatrix}0\\0\\1\end{pmatrix}$

How to formulate "learning"?

How to formulate “learning”?

- Data: $\{x_i, y_i\}_{i=1}^{m}$

How to formulate "learning"?

- Data: $\{x_i, y_i\}_{i=1}^m$
- Find f^* in some function class such that $f^*(x_i) \approx y_i$.

How to formulate “learning”?

- Data: $\{x_i, y_i\}_{i=1}^m$
- Find f^* in some function class such that $f^*(x_i) \approx y_i$.
- Mathematically: solve the optimization problem by parameterizing the abstract function class

$$\min_{\Theta} L(x, y, \Theta) \tag{1}$$

where

$$L(x, y, \Theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[L(f(x; \Theta), y)] \approx \frac{1}{N}\sum_{i=1}^{N} \|f(x_i; \Theta) - y_i\|^2$$

  - Or combine the feature map f with the logistic regression model to obtain cross-entropy loss function.

How to formulate “learning”?

- Data: $\{x_i, y_i\}_{i=1}^m$
- Find f^* in some function class such that $f^*(x_i) \approx y_i$.
- Mathematically: solve the optimization problem by parameterizing the abstract function class

$$\min_{\Theta} L(x, y, \Theta) \tag{1}$$

  where

$$L(x, y, \Theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[L(f(x;\Theta), y)] \approx \frac{1}{N}\sum_{i=1}^{N} \|f(x_i;\Theta) - y_i\|^2$$

  - Or combine the feature map f with the logistic regression model to obtain cross-entropy loss function.
- Application: image classification:

How to formulate "learning"?

- Data: $\{x_i, y_i\}_{i=1}^m$
- Find f^* in some function class such that $f^*(x_i) \approx y_i$.
- Mathematically: solve the optimization problem by parameterizing the abstract function class

$$\min_{\Theta} L(x, y, \Theta) \tag{1}$$

where

$$L(x, y, \Theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[L(f(x;\Theta), y)] \approx \frac{1}{N}\sum_{i=1}^{N} \|f(x_i;\Theta) - y_i\|^2$$

  - Or combine the feature map f with the logistic regression model to obtain cross-entropy loss function.
- Application: image classification:

$$f(\boxed{\ ?\ }; \Theta) = \begin{pmatrix} 0.7 \\ 0.2 \\ 0.1 \end{pmatrix}$$

How to formulate "learning"?

- Data: $\{x_i, y_i\}_{i=1}^m$
- Find f^* in some function class such that $f^*(x_i) \approx y_i$.
- Mathematically: solve the optimization problem by parameterizing the abstract function class

$$\min_{\Theta} L(x, y, \Theta) \tag{1}$$

where

$$L(x, y, \Theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[L(f(x; \Theta), y)] \approx \frac{1}{N}\sum_{i=1}^{N} \|f(x_i; \Theta) - y_i\|^2$$

  - Or combine the feature map f with the logistic regression model to obtain cross-entropy loss function.
- Application: image classification:

$$f(\boxed{?}; \Theta) = \begin{pmatrix} 0.7 \\ 0.2 \\ 0.1 \end{pmatrix} \implies \boxed{?} =$$

How to formulate "learning"?

- Data: $\{x_i, y_i\}_{i=1}^m$
- Find f^* in some function class such that $f^*(x_i) \approx y_i$.
- Mathematically: solve the optimization problem by parameterizing the abstract function class

$$\min_{\Theta} L(x, y, \Theta) \tag{1}$$

where

$$L(x, y, \Theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[L(f(x;\Theta), y)] \approx \frac{1}{N}\sum_{i=1}^{N} \|f(x_i;\Theta) - y_i\|^2$$

  - Or combine the feature map f with the logistic regression model to obtain cross-entropy loss function.

- Application: image classification:

$$f(\boxed{?};\Theta) = \begin{pmatrix} 0.7 \\ 0.2 \\ 0.1 \end{pmatrix} \implies \boxed{?} = \text{cat}$$

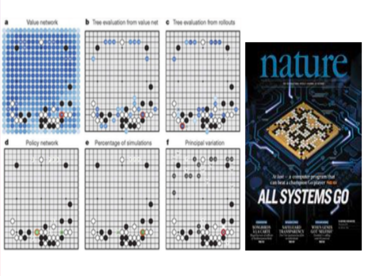
Deep Learning

Machine learning using a special function class:
deep neural networks!

Deep learning and its great successes

CNN has been successfully used in:

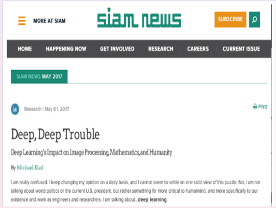
- Computer vision
  - Classification, detection, segmentation...
  - Medical image processing,
  - Face recognition,
- Reinforcement learning
  - AlphaGo,
  - Automated driving,
- Natural language processing
  - Speech recognition,
  - Machine translation,
- ...



Deep Concern

Quotes(Researchers from U Wash, Princeton, MIT, . . .)

- “Deep learning is killing image processing, natural language processing...”
- “Graduate students only work on deep learning”
- “One time extinction event – graduate students won’t know the fundamental tools”



Source: D. Donoho/ H. Monajemi/ V. Papyan “Stats 38” at Stanford and B. Dong at PKU

Mathematical understanding and Analysis?

Question:

Why and how do these neural network machine learning models and relevant algorithms work?

Mathematical understanding and Analysis?

Question:

Why and how do these neural network machine learning models and relevant algorithms work?

This series of talks:

1. Finite Element and Deep Neural Network
   - ReLU neural networks = linear finite elements
   - Largest function class that a stable neural network can approximate
   - Optimal approximation rates for popular neural networks
2. Multigrid and Image Classification
   - Linear separable sets and logistic regression
   - A model for feature extractions
   - Image classification by multigrid method
3. Neural Network and Numerical PDEs
   - Error analysis of neural network for numerical PDEs
   - Numerical quadrature and Rademacher complexity analysis
   - Training algorithm that achieves the best asymptotic convergence rate

Recall: supervised learning

- Mathematical problem: Find $f(\cdot;\Theta) : \mathbb{R}^d \to \mathbb{R}^3$ such that:

Recall: supervised learning

- Mathematical problem: Find $f(\cdot;\Theta) : \mathbb{R}^d \to \mathbb{R}^3$ such that:

$f($  $;\Theta) \approx \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$

Recall: supervised learning

- Mathematical problem: Find $f(\cdot;\Theta):\mathbb{R}^d \to \mathbb{R}^3$ such that:

$f($  $;\Theta) \approx \begin{pmatrix}1\\0\\0\end{pmatrix}$ $f($  $;\Theta) \approx \begin{pmatrix}0\\1\\0\end{pmatrix}$

Recall: supervised learning

- Mathematical problem: Find $f(\cdot;\Theta) : \mathbb{R}^d \to \mathbb{R}^3$ such that:

$f($  $;\Theta) \approx \begin{pmatrix}1\\0\\0\end{pmatrix}$ $f($  $;\Theta) \approx \begin{pmatrix}0\\1\\0\end{pmatrix}$ $f($  $;\Theta) \approx \begin{pmatrix}0\\0\\1\end{pmatrix}$

Recall: supervised learning

- Mathematical problem: Find $f(\cdot;\Theta) : \mathbb{R}^d \to \mathbb{R}^3$ such that:

$f($  $;\Theta) \approx \begin{pmatrix}1\\0\\0\end{pmatrix}$ $f($  $;\Theta) \approx \begin{pmatrix}0\\1\\0\end{pmatrix}$ $f($  $;\Theta) \approx \begin{pmatrix}0\\0\\1\end{pmatrix}$

- Image classification:

$$f(\boxed{\ ?\ };\Theta) = \begin{pmatrix}0.7\\0.2\\0.1\end{pmatrix}$$

Recall: supervised learning

- Mathematical problem: Find $f(\cdot;\Theta) : \mathbb{R}^d \to \mathbb{R}^3$ such that:

$f($$;\Theta) \approx \begin{pmatrix}1\\0\\0\end{pmatrix}$ $f($$;\Theta) \approx \begin{pmatrix}0\\1\\0\end{pmatrix}$ $f($$;\Theta) \approx \begin{pmatrix}0\\0\\1\end{pmatrix}$

- Image classification:

$$f(\boxed{?};\Theta) = \begin{pmatrix}0.7\\0.2\\0.1\end{pmatrix} \Longrightarrow \boxed{?} = \text{cat!}$$

Recall: supervised learning

- Mathematical problem: Find $f(\cdot;\Theta) : \mathbb{R}^d \to \mathbb{R}^3$ such that:

$f($  $;\Theta) \approx \begin{pmatrix}1\\0\\0\end{pmatrix}$ $f($  $;\Theta) \approx \begin{pmatrix}0\\1\\0\end{pmatrix}$ $f($  $;\Theta) \approx \begin{pmatrix}0\\0\\1\end{pmatrix}$

- Image classification:

$$f(\boxed{?};\Theta) = \begin{pmatrix}0.7\\0.2\\0.1\end{pmatrix} \Longrightarrow \boxed{?} = \text{cat!}$$

Question:

What is a good function class for f ?

What is a good function class for f?

Most favorable function class:

What is a good function class for f?

Most favorable function class:

- Polynomials!

$$\sum_{\alpha_1+\alpha_2+\ldots+\alpha_d\leq n} a_\alpha x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_d^{\alpha_d}$$

What is a good function class for f?

Most favorable function class:

- Polynomials!

$$\sum_{\alpha_1+\alpha_2+\ldots+\alpha_d \le n} a_\alpha x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_d^{\alpha_d}$$

Important property:

What is a good function class for f?

Most favorable function class:

- Polynomials!

$$\sum_{\alpha_1+\alpha_2+\ldots+\alpha_d \le n} a_\alpha x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_d^{\alpha_d}$$

Important property: polynomials can approximate any reasonable function!

What is a good function class for f?

Most favorable function class:

- Polynomials!

$$\sum_{\alpha_1+\alpha_2+\ldots+\alpha_d\le n} a_\alpha x_1^{\alpha_1} x_2^{\alpha_2}\cdots x_d^{\alpha_d}$$

Important property: polynomials can approximate any reasonable function!

- dense in $C(\Omega)$ [Weierstrass theorem]

What is a good function class for f?

Most favorable function class:

- Polynomials!

$$\sum_{\alpha_1+\alpha_2+\ldots+\alpha_d\le n} a_\alpha x_1^{\alpha_1} x_2^{\alpha_2}\cdots x_d^{\alpha_d}$$

Important property: polynomials can approximate any reasonable function!

- dense in $C(\Omega)$ [Weierstrass theorem]
- dense in all Sobolev spaces: $L^2(\Omega)$, $W^{m,p}(\Omega), \ldots$

What is a good function class for f?

Most favorable function class:

- Polynomials!

$$\sum_{\alpha_1+\alpha_2+\ldots+\alpha_d\le n} a_\alpha x_1^{\alpha_1} x_2^{\alpha_2}\cdots x_d^{\alpha_d}$$

Important property: polynomials can approximate any reasonable function!

- dense in $C(\Omega)$ [Weierstrass theorem]
- dense in all Sobolev spaces: $L^2(\Omega)$, $W^{m,p}(\Omega), \ldots$

Curse of dimensionality: Number of coefficients for polynomials of degrees n in $\mathbb{R}^d$:

$$N = \binom{d+n}{n} = \frac{(n+d)!}{d!n!}.$$

What is a good function class for f?

Most favorable function class:

- Polynomials!

$$\sum_{\alpha_1+\alpha_2+\ldots+\alpha_d\le n} a_\alpha x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_d^{\alpha_d}$$

Important property: polynomials can approximate any reasonable function!

- dense in $C(\Omega)$ [Weierstrass theorem]
- dense in all Sobolev spaces: $L^2(\Omega)$, $W^{m,p}(\Omega), \ldots$

Curse of dimensionality: Number of coefficients for polynomials of degrees n in $\mathbb{R}^d$:

$$N = \binom{d+n}{n} = \frac{(n+d)!}{d!n!}.$$

What is a good function class for f?

Most favorable function class:

- Polynomials!

$$\sum_{\alpha_1+\alpha_2+\ldots+\alpha_d\le n} a_\alpha x_1^{\alpha_1} x_2^{\alpha_2}\cdots x_d^{\alpha_d}$$

Important property: polynomials can approximate any reasonable function!

- dense in $C(\Omega)$ [Weierstrass theorem]
- dense in all Sobolev spaces: $L^2(\Omega)$, $W^{m,p}(\Omega)$, ...

Curse of dimensionality: Number of coefficients for polynomials of degrees n in $\mathbb{R}^d$:

$$N = \binom{d+n}{n} = \frac{(n+d)!}{d!n!}.$$

For example $n = 100$:

$d =$	2	4	8
$N =$	5×10^3	4.6×10^6	3.5×10^{11}

σ-DNN: Linears, activation and composition

σ-DNN: Linears, activation and composition

1. Start from a linear function

$$W^0 x + b^0$$

σ-DNN: Linears, activation and composition

1. Start from a linear function
$$W^0x + b^0$$

2. Compose with the activation function:
$$x^{(1)} = \sigma(W^0x + b^0)$$

σ-DNN: Linears, activation and composition

1. Start from a linear function
$$W^0x + b^0$$
2. Compose with the activation function:
$$x^{(1)} = \sigma(W^0x + b^0)$$
3. Compose with another linear function:
$$W^1x^{(1)} + b^1$$

σ-DNN: Linears, activation and composition

1. Start from a linear function
$$W^0 x + b^0$$
2. Compose with the activation function:
$$x^{(1)} = \sigma(W^0 x + b^0)$$
3. Compose with another linear function:
$$W^1 x^{(1)} + b^1$$
4. Compose with the activation function:
$$x^{(2)} = \sigma(W^1 x^{(1)} + b^1)$$

σ-DNN: Linears, activation and composition

1. Start from a linear function
$$W^0x + b^0$$
2. Compose with the activation function:
$$x^{(1)} = \sigma(W^0x + b^0)$$
3. Compose with another linear function:
$$W^1x^{(1)} + b^1$$
4. Compose with the activation function:
$$x^{(2)} = \sigma(W^1x^{(1)} + b^1)$$
5. Compose with another linear function
$$f(x;\Theta) = W^2x^{(2)} + b^2$$

σ-DNN: Linears, activation and composition

1. Start from a linear function
$$W^0 x + b^0$$
2. Compose with the activation function:
$$x^{(1)} = \sigma(W^0 x + b^0)$$
3. Compose with another linear function:
$$W^1 x^{(1)} + b^1$$
4. Compose with the activation function:
$$x^{(2)} = \sigma(W^1 x^{(1)} + b^1)$$
5. Compose with another linear function
$$f(x; \Theta) = W^2 x^{(2)} + b^2$$
6. ...

σ-DNN: Linears, activation and composition

1. Start from a linear function
$$W^0x + b^0$$
2. Compose with the activation function:
$$x^{(1)} = \sigma(W^0x + b^0)$$
3. Compose with another linear function:
$$W^1x^{(1)} + b^1$$
4. Compose with the activation function:
$$x^{(2)} = \sigma(W^1x^{(1)} + b^1)$$
5. Compose with another linear function
$$f(x;\Theta) = W^2x^{(2)} + b^2$$
6. ...

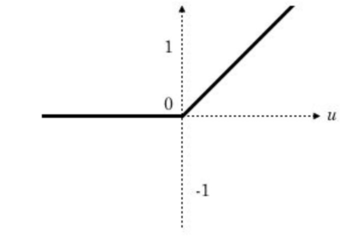
Deep neural network functions with ℓ-hidden layers

$$\Sigma^{\sigma}_{n_{1:\ell}} = \{W^{\ell}x^{(\ell)} + b^{\ell},\ W^i \in \mathbb{R}^{n_i},\ b_i \in \mathbb{R}\}$$

A popular activation function

Ramp or ReLU function

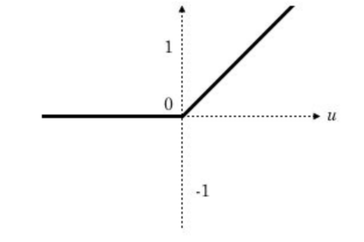
$$\sigma = \mathrm{ReLU}(x) = \max(0, x)$$



A popular activation function

Ramp or ReLU function

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



Notation:

$$\Sigma^{k}_{n_{1:\ell}} = \Sigma^{\mathrm{ReLU}^k}_{n_{1:\ell}}.$$

What does a function in $\Sigma^1_{n_{1:\ell}}$ look like?

Obviously:

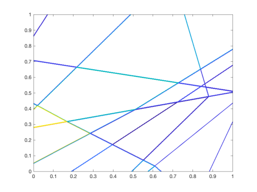
$$\Sigma^1_{n_{1:\ell}} = \text{a space of continuous piecewise linear functions!}^{[1]}$$

[1] Juncai He et al. "ReLU Deep Neural Networks and Linear Finite Elements". In: *Journal of Computational Mathematics* 38.3 (2020), pp. 502–527.

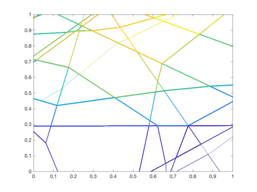
What does a function in $\Sigma^1_{n_{1:\ell}}$ look like?

Obviously:

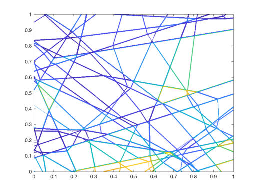
$$\Sigma^1_{n_{1:\ell}} = \text{a space of continuous piecewise linear functions!}^{[1]}$$



$(10, 10)$



$(20, 20)$



$(40, 40)$

[1] Juncai He et al. “ReLU Deep Neural Networks and Linear Finite Elements”. In: *Journal of Computational Mathematics* 38.3 (2020), pp. 502–527.

How is $\Sigma^1_{n_{1:\ell}}$ compared with (adaptive) linear FEM?

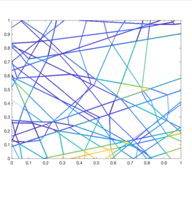


Figure: $(40, 40)$

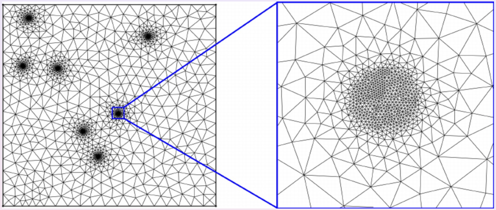


Figure: Adaptive Grid

1. Introduction

2. Finite Element and Neural Networks
   - Neural Network Functions
   - Connection of ReLU DNN and linear FEM

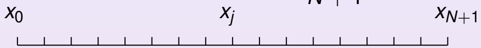
3. Neural Network Approximation Class
   - Density of Shallow Neural Network
   - Approximation Properties
   - Approximation Spaces

Finite element: piecewise linear functions

Finite element: piecewise linear functions

- Uniform grid $\mathcal{T}_h$

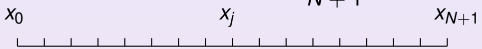
$$0 = x_0 < x_1 < \cdots < x_{N+1} = 1, \quad x_j = \frac{j}{N+1} \; (j = 0 : N+1).$$



Finite element: piecewise linear functions

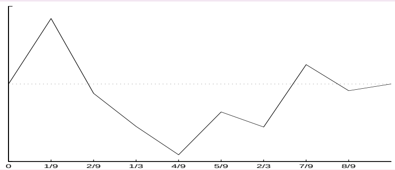
- Uniform grid $\mathcal{T}_h$

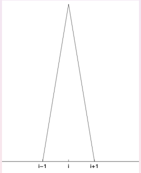
$$0 = x_0 < x_1 < \cdots < x_{N+1} = 1, \quad x_j = \frac{j}{N+1} \; (j = 0 : N+1).$$



- Linear finite element space

$$V_h = \{v : \; v \text{ is continuous and piecewise linear w.r.t. } \mathcal{T}_h\}.$$

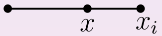


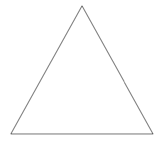


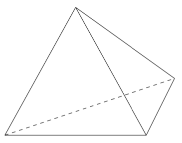
$$v_h(x) = \sum_{i=1}^{n} v_h(x_i)\phi_i(x).$$

Linear finite element in multi-dimensions

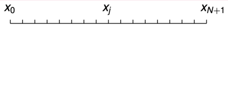
$w_1 x + b$ $\qquad$ $w_1 x_1 + w_2 x_2 + b$ $\qquad$ $w_1 x_1 + w_2 x_2 + w_3 x_3 + b$ $\qquad \cdots$

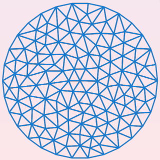


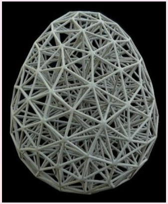




$\cdots$





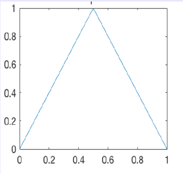


$\cdots$

FEM basis function in 1D

- Denote the basis function in $\mathcal{T}_1$

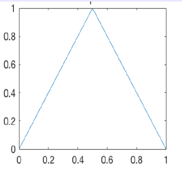
$$\varphi(x) = \begin{cases} 2x & x \in [0, \frac{1}{2}] \\ 2(1-x) & x \in [\frac{1}{2}, 1] \\ 0, & \text{others} \end{cases}. \tag{2}$$

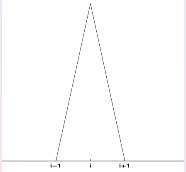


FEM basis function in 1D

- Denote the basis function in $\mathcal{T}_1$

$$\varphi(x) = \begin{cases} 2x & x \in [0, \frac{1}{2}] \\ 2(1-x) & x \in [\frac{1}{2}, 1] \\ 0, & \text{others} \end{cases}. \qquad (2)$$





- All basis functions φ_i can be written as

$$\varphi_i = \varphi(\frac{x - x_{i-1}}{2h}) = \varphi(w_h x + b_i). \qquad (3)$$

with $w_h = \frac{1}{2h}$, $\quad b_i = \frac{-(i-1)}{2}$.

FEM basis function in 1D

- Denote the basis function in $\mathcal{T}_1$

$$\varphi(x) = \begin{cases} 2x & x \in [0, \frac{1}{2}] \\ 2(1-x) & x \in [\frac{1}{2}, 1] \\ 0, & \text{others} \end{cases}. \qquad (2)$$

- All basis functions φ_i can be written as

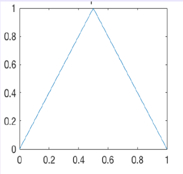
$$\varphi_i = \varphi(\frac{x - x_{i-1}}{2h}) = \varphi(w_h x + b_i). \qquad (3)$$

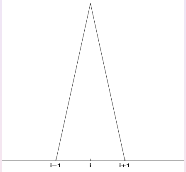
with $w_h = \frac{1}{2h}$, $\quad b_i = \frac{-(i-1)}{2}$.

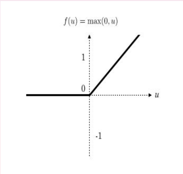
- Let $x_+ = \max(0, x) = \mathrm{ReLU}(x)$,

$$\varphi(x) = 2x_+ - 4(x - 1/2)_+ + 2(x-1)_+.$$

- $\varphi_i \in \mathrm{span}\left\{(wx+b)_+, w, b \in \mathbb{R}^1\right\}$







FEM $\Longrightarrow \Sigma_n^1$ $(d = 1)$

- FEM $\Longrightarrow \Sigma_n^1$ (make w_h and b_i arbitrary)

FEM $\Longrightarrow \Sigma_n^1$ $(d = 1)$

- FEM $\Longrightarrow \Sigma_n^1$ (make w_h and b_i arbitrary)

$$\text{FEM} \subset \left\{ \sum_{i=1}^{n} a_i(\omega_i x + b_i)_+, a_i, \omega_i, b_i \in \mathbb{R}^1 \right\} = \Sigma_n^1.$$

FEM $\Longrightarrow \Sigma_n^1$ $(d = 1)$

- FEM $\Longrightarrow \Sigma_n^1$ (make w_h and b_i arbitrary)

$$\text{FEM} \subset \left\{ \sum_{i=1}^{n} a_i(\omega_i x + b_i)_+, a_i, \omega_i, b_i \in \mathbb{R}^1 \right\} = \Sigma_n^1.$$

FEM $\Longrightarrow \Sigma_n^1\,(d=1)$

- FEM $\Longrightarrow \Sigma_n^1$ (make w_h and b_i arbitrary)

$$\text{FEM} \subset \left\{ \sum_{i=1}^{n} a_i(\omega_i x + b_i)_+, a_i, \omega_i, b_i \in \mathbb{R}^1 \right\} = \Sigma_n^1.$$

Σ_n^1 is one hidden layer "shallow" neural network with activation function ReLU, n neurons.

Generalization to multi-dimension:

Higher dimension $d \geq 1$

$$\Sigma_n^1 = \left\{ \sum_{i=1}^{n} a_i (w_i x + b_i)_+ : w_i \in \mathbb{R}^{1\times d}, b_i \in \mathbb{R} \right\} \tag{4}$$

where $w_i x = \sum_{j=1}^{d} w_{ij} x_j$.

Connection of ReLU-DNN and Linear FEM

1. $d = 1$,

$$\mathrm{FE} \subset \Sigma_n^1.$$

[2] Juncai He et al. “ReLU Deep Neural Networks and Linear Finite Elements”. In: *Journal of Computational Mathematics* 38.3 (2020), pp. 502–527.

[3] Raman Arora et al. “Understanding deep neural networks with rectified linear units”. In: *arXiv preprint arXiv:1611.01491* (2016).

Connection of ReLU-DNN and Linear FEM

1. $d = 1$,

$$\mathrm{FE} \subset \Sigma_n^1.$$

2. $d \geq 2$[2],

$$\mathrm{FE} \not\subseteq \Sigma_n^1, \quad \forall n \geq 1.$$

[2] Juncai He et al. "ReLU Deep Neural Networks and Linear Finite Elements". In: *Journal of Computational Mathematics* 38.3 (2020), pp. 502–527.

[3] Raman Arora et al. "Understanding deep neural networks with rectified linear units". In: *arXiv preprint arXiv:1611.01491* (2016).

Connection of ReLU-DNN and Linear FEM

1. $d = 1$,
$$\mathrm{FE} \subset \Sigma_n^1.$$

2. $d \geq 2$[2],
$$\mathrm{FE} \not\subset \Sigma_n^1, \quad \forall n \geq 1.$$

3. $d \geq 2$[3],
$$\mathrm{FE} \subset \Sigma_{n_{1:\ell}}^1 \quad \text{for some } \ell > 1,$$

   where $\Sigma_{n_{1:\ell}}^1$ is ReLU-DNN with ℓ layers.

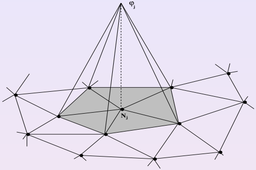
---

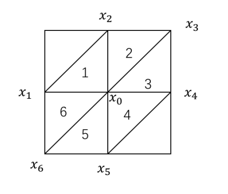
[2] Juncai He et al. "ReLU Deep Neural Networks and Linear Finite Elements". In: *Journal of Computational Mathematics* 38.3 (2020), pp. 502–527.

[3] Raman Arora et al. "Understanding deep neural networks with rectified linear units". In: *arXiv preprint arXiv:1611.01491* (2016).

A 2D example: FE basis function

Consider a 2D FE basis function, $\phi(x)$:



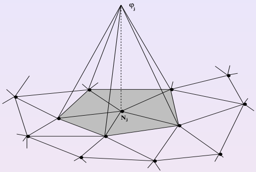


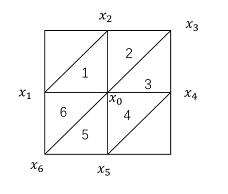
[4] Juncai He et al. "ReLU Deep Neural Networks and Linear Finite Elements". In: *Journal of Computational Mathematics* 38.3 (2020), pp. 502–527.

[5] Raman Arora et al. "Understanding deep neural networks with rectified linear units". In: *arXiv preprint arXiv:1611.01491* (2016).

A 2D example: FE basis function

Consider a 2D FE basis function, $\phi(x)$:





Here g_i is linear in Domain i, and $x_7 = x_1$, satisfying

$$g_i(x_0) = 1 \qquad g_i(x_i) = 0 \qquad g_i(x_{i+1}) = 0$$

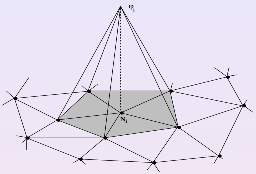
$$\phi(x) = \begin{cases} g_i(x), & x \in \text{Domain } i \\ 0. & x \in \mathbb{R}^2 - \overline{x_1x_2x_3x_4x_5x_6} \end{cases} \tag{5}$$

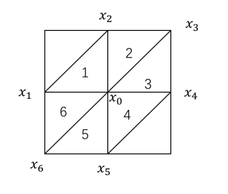
[4] Juncai He et al. “ReLU Deep Neural Networks and Linear Finite Elements”. In: *Journal of Computational Mathematics* 38.3 (2020), pp. 502–527.

[5] Raman Arora et al. “Understanding deep neural networks with rectified linear units”. In: *arXiv preprint arXiv:1611.01491* (2016).

A 2D example: FE basis function

Consider a 2D FE basis function, $\phi(x)$:





Here g_i is linear in Domain i, and $x_7 = x_1$, satisfying

$$g_i(x_0) = 1 \qquad g_i(x_i) = 0 \qquad g_i(x_{i+1}) = 0$$

$$\phi(x) = \begin{cases} g_i(x), & x \in \text{Domain } i \\ 0. & x \in \mathbb{R}^2 - \overline{x_1 x_2 x_3 x_4 x_5 x_6} \end{cases} \tag{5}$$

We have[4][5]

$$\phi \in \Sigma^1_{n_{1:4}}. \tag{6}$$

---

[4] Juncai He et al. "ReLU Deep Neural Networks and Linear Finite Elements". In: *Journal of Computational Mathematics* 38.3 (2020), pp. 502–527.

[5] Raman Arora et al. "Understanding deep neural networks with rectified linear units". In: *arXiv preprint arXiv:1611.01491* (2016).

Properties of ReLU

We can write the basis function $\phi(x)$ defined on hexagon $\overline{x_1x_2x_3x_4x_5x_6}$ which is satisfied that important identities:

Properties of ReLU

We can write the basis function $\phi(x)$ defined on hexagon $\overline{x_1 x_2 x_3 x_4 x_5 x_6}$ which is satisfied that important identities:

$$x = \mathrm{ReLU}(x) - \mathrm{ReLU}(-x), |x| = \mathrm{ReLU}(x) + \mathrm{ReLU}(-x)$$

Properties of ReLU

We can write the basis function $\phi(x)$ defined on hexagon $\overline{x_1x_2x_3x_4x_5x_6}$ which is satisfied that important identities:

$$x = \mathrm{ReLU}(x) - \mathrm{ReLU}(-x), |x| = \mathrm{ReLU}(x) + \mathrm{ReLU}(-x)$$

and

$$\min(a, b) = \frac{a+b}{2} - \frac{|a-b|}{2} = v \cdot \mathrm{ReLU}(W \cdot [a, b]^T) \tag{7}$$

Properties of ReLU

We can write the basis function $\phi(x)$ defined on hexagon $\overline{x_1x_2x_3x_4x_5x_6}$ which is satisfied that important identities:

$$x = \mathrm{ReLU}(x) - \mathrm{ReLU}(-x), |x| = \mathrm{ReLU}(x) + \mathrm{ReLU}(-x)$$

and

$$\min(a,b) = \frac{a+b}{2} - \frac{|a-b|}{2} = v \cdot \mathrm{ReLU}(W \cdot [a,b]^T) \tag{7}$$

where

$$v = \frac{1}{2}[1,-1,-1,-1] \qquad W = \begin{bmatrix} 1 & 1 \\ -1 & -1 \\ 1 & -1 \\ -1 & 1 \end{bmatrix}$$

Example of DNN and FEM: 2D-FEM basis function

$$\min(a, b, c) = \min(\min(a, b), c)$$

Example of DNN and FEM: 2D-FEM basis function

$$
\begin{aligned}
\min(a, b, c) &= \min(\min(a, b), c) \\
&= v \cdot \mathrm{ReLU}\left(W \cdot \begin{pmatrix} \min(a, b) \\ c \end{pmatrix}\right)
\end{aligned}
$$

Example of DNN and FEM: 2D-FEM basis function

$$
\begin{aligned}
\min(a,b,c) &= \min(\min(a,b),c) \\
&= v \cdot \mathrm{ReLU}\left(W \cdot \begin{pmatrix} \min(a,b) \\ c \end{pmatrix}\right) \\
&= v \cdot \mathrm{ReLU}\left(W \cdot \begin{pmatrix} v \cdot \mathrm{ReLU}(W \cdot [a,b]^T) \\ \mathrm{ReLU}(c) - \mathrm{ReLU}(-c) \end{pmatrix}\right)
\end{aligned}
$$

Example of DNN and FEM: 2D-FEM basis function

$$
\begin{aligned}
\min(a,b,c) &= \min(\min(a,b),c) \\
&= v\cdot \mathrm{ReLU}\left(W\cdot \begin{pmatrix}\min(a,b)\\ c\end{pmatrix}\right) \\
&= v\cdot \mathrm{ReLU}\left(W\cdot \begin{pmatrix} v\cdot \mathrm{ReLU}(W\cdot [a,b]^T)\\ \mathrm{ReLU}(c)-\mathrm{ReLU}(-c)\end{pmatrix}\right) \\
&= v\cdot \mathrm{ReLU}\left(W\cdot \begin{pmatrix} v\cdot \mathrm{ReLU}(W\cdot [a,b]^T)\\ [1,-1]\cdot \mathrm{ReLU}([1,-1]^T c)\end{pmatrix}\right)
\end{aligned}
$$

Example of DNN and FEM: 2D-FEM basis function

$$
\begin{aligned}
\min(a,b,c) &= \min(\min(a,b),c) \\
&= v \cdot \mathrm{ReLU}\left(W \cdot \begin{pmatrix} \min(a,b) \\ c \end{pmatrix}\right) \\
&= v \cdot \mathrm{ReLU}\left(W \cdot \begin{pmatrix} v \cdot \mathrm{ReLU}(W \cdot [a,b]^T) \\ \mathrm{ReLU}(c) - \mathrm{ReLU}(-c) \end{pmatrix}\right) \\
&= v \cdot \mathrm{ReLU}\left(W \cdot \begin{pmatrix} v \cdot \mathrm{ReLU}(W \cdot [a,b]^T) \\ [1,-1] \cdot \mathrm{ReLU}([1,-1]^T c) \end{pmatrix}\right) \\
&= v \cdot \mathrm{ReLU}\left(W_2 \cdot \mathrm{ReLU}(W_1 \cdot [a,b,c]^T)\right)
\end{aligned}
$$

Example of DNN and FEM: 2D-FEM basis function

$$
\begin{aligned}
\min(a,b,c) &= \min(\min(a,b),c) \\
&= v\cdot \mathrm{ReLU}\left(W\cdot\begin{pmatrix}\min(a,b)\\ c\end{pmatrix}\right) \\
&= v\cdot \mathrm{ReLU}\left(W\cdot\begin{pmatrix}v\cdot\mathrm{ReLU}(W\cdot[a,b]^T)\\ \mathrm{ReLU}(c)-\mathrm{ReLU}(-c)\end{pmatrix}\right) \\
&= v\cdot \mathrm{ReLU}\left(W\cdot\begin{pmatrix}v\cdot\mathrm{ReLU}(W\cdot[a,b]^T)\\ [1,-1]\cdot\mathrm{ReLU}([1,-1]^Tc)\end{pmatrix}\right) \\
&= v\cdot \mathrm{ReLU}\left(W_2\cdot\mathrm{ReLU}(W_1\cdot[a,b,c]^T)\right)
\end{aligned}
$$

where

$$
W_2=\begin{bmatrix}\frac12 & -\frac12 & -\frac12 & -\frac12 & 1 & -1\\ -\frac12 & \frac12 & \frac12 & \frac12 & -1 & 1\\ \frac12 & -\frac12 & -\frac12 & -\frac12 & -1 & 1\\ -\frac12 & \frac12 & \frac12 & \frac12 & 1 & -1\end{bmatrix}. \qquad W_1=\begin{bmatrix}1 & 1 & 0\\ -1 & -1 & 0\\ 1 & -1 & 0\\ -1 & 1 & 0\\ 0 & 0 & 1\\ 0 & 0 & -1\end{bmatrix}.
$$

Example of DNN and FEM: 2D-FEM basis function

It follows that

$$
\begin{aligned}
g \equiv & \min(g_1, g_2, g_3, g_4, g_5, g_6) = \min(\min(g_1, g_2, g_3), \min(g_4, g_5, g_6)) \\
= & v \cdot \mathrm{ReLU}(W \cdot \begin{bmatrix} \min(g_1, g_2, g_3) \\ \min(g_4, g_5, g_6) \end{bmatrix}) \\
= & v \cdot \mathrm{ReLU}(W \cdot \begin{bmatrix} v \cdot \mathrm{ReLU}(W_2 \cdot \mathrm{ReLU}(W_1 \cdot [g_1, g_2, g_3]^T)) \\ v \cdot \mathrm{ReLU}(W_2 \cdot \mathrm{ReLU}(W_1 \cdot [g_4, g_5, g_6]^T)) \end{bmatrix})
\end{aligned}
$$

Example of DNN and FEM: 2D-FEM basis function

It follows that

$$
\begin{aligned}
g \equiv & \min(g_1, g_2, g_3, g_4, g_5, g_6) = \min(\min(g_1, g_2, g_3), \min(g_4, g_5, g_6)) \\
= & v \cdot \mathrm{ReLU}(W \cdot \begin{bmatrix} \min(g_1, g_2, g_3) \\ \min(g_4, g_5, g_6) \end{bmatrix}) \\
= & v \cdot \mathrm{ReLU}(W \cdot \begin{bmatrix} v \cdot \mathrm{ReLU}(W_2 \cdot \mathrm{ReLU}(W_1 \cdot [g_1, g_2, g_3]^T) \\ v \cdot \mathrm{ReLU}(W_2 \cdot \mathrm{ReLU}(W_1 \cdot [g_4, g_5, g_6]^T) \end{bmatrix})
\end{aligned}
$$

Thus

$$
\phi = \mathrm{ReLU}\left(v \cdot \mathrm{ReLU}\left(W \cdot \begin{bmatrix} v \cdot \mathrm{ReLU}(W_2 \cdot \mathrm{ReLU}(W_1 \cdot [g_1, g_2, g_3]^T) \\ v \cdot \mathrm{ReLU}(W_2 \cdot \mathrm{ReLU}(W_1 \cdot [g_4, g_5, g_6]^T) \end{bmatrix}\right)\right)
$$

Example of DNN and FEM: 2D-FEM basis function

It follows that

$$
\begin{aligned}
g \equiv & \min(g_1, g_2, g_3, g_4, g_5, g_6) = \min(\min(g_1, g_2, g_3), \min(g_4, g_5, g_6)) \\
= & v \cdot \mathrm{ReLU}(W \cdot \begin{bmatrix} \min(g_1, g_2, g_3) \\ \min(g_4, g_5, g_6) \end{bmatrix}) \\
= & v \cdot \mathrm{ReLU}(W \cdot \begin{bmatrix} v \cdot \mathrm{ReLU}(W_2 \cdot \mathrm{ReLU}(W_1 \cdot [g_1, g_2, g_3]^T)) \\ v \cdot \mathrm{ReLU}(W_2 \cdot \mathrm{ReLU}(W_1 \cdot [g_4, g_5, g_6]^T)) \end{bmatrix})
\end{aligned}
$$

Thus

$$
\phi = \mathrm{ReLU}\left(v \cdot \mathrm{ReLU}\left(W \cdot \begin{bmatrix} v \cdot \mathrm{ReLU}(W_2 \cdot \mathrm{ReLU}(W_1 \cdot [g_1, g_2, g_3]^T)) \\ v \cdot \mathrm{ReLU}(W_2 \cdot \mathrm{ReLU}(W_1 \cdot [g_4, g_5, g_6]^T)) \end{bmatrix} \right)\right) \in \Sigma^1_{n_{1:4}}.
$$

ReLU-DNN and Linear FEM for H^1

$$\text{ReLU-DNN} = \Sigma^1_{n_{1:\ell}}$$

ReLU-DNN and Linear FEM for H^1

$$\textit{ReLU-DNN} = \Sigma^1_{n_{1:\ell}} = \textit{Linear FEM} \subset H^1(\Omega)$$

Shallow Neural Networks

- Superpositions of *ridge functions*

$$\Sigma_n^\sigma := \left\{ \sum_{i=1}^{n} a_i \sigma(\omega_i \cdot x + b_i),\ a_i \in \mathbb{R},\ \omega_i \in \mathbb{R}^d,\ b_i \in \mathbb{R} \right\}$$

Shallow Neural Networks

- Superpositions of *ridge functions*

$$\Sigma_n^\sigma := \left\{ \sum_{i=1}^{n} a_i \sigma(\omega_i \cdot x + b_i),\ a_i \in \mathbb{R},\ \omega_i \in \mathbb{R}^d,\ b_i \in \mathbb{R} \right\}$$

- Shallow networks means one hidden layer, i.e.

$$\Sigma_{[1]}^\sigma = \bigcup_{n=1}^{\infty} \Sigma_n^\sigma \tag{8}$$

Shallow Neural Networks

- Superpositions of *ridge functions*

$$\Sigma_n^\sigma := \left\{ \sum_{i=1}^{n} a_i \sigma(\omega_i \cdot x + b_i),\ a_i \in \mathbb{R},\ \omega_i \in \mathbb{R}^d,\ b_i \in \mathbb{R} \right\}$$

- Shallow networks means one hidden layer, i.e.

$$\Sigma_{[1]}^\sigma = \bigcup_{n=1}^{\infty} \Sigma_n^\sigma \tag{8}$$

- Let $\Omega \subset \mathbb{R}^d$ be a bounded domain

Shallow Neural Networks

- Superpositions of *ridge functions*

$$\Sigma_n^\sigma := \left\{ \sum_{i=1}^{n} a_i \sigma(\omega_i \cdot x + b_i),\ a_i \in \mathbb{R},\ \omega_i \in \mathbb{R}^d,\ b_i \in \mathbb{R} \right\}$$

- Shallow networks means one hidden layer, i.e.

$$\Sigma_{[1]}^\sigma = \bigcup_{n=1}^{\infty} \Sigma_n^\sigma \tag{8}$$

- Let $\Omega \subset \mathbb{R}^d$ be a bounded domain
- Want to use shallow networks to approximate functions f on Ω

Shallow Neural Networks

- Superpositions of *ridge functions*

$$\Sigma_n^\sigma := \left\{ \sum_{i=1}^{n} a_i \sigma(\omega_i \cdot x + b_i),\ a_i \in \mathbb{R},\ \omega_i \in \mathbb{R}^d,\ b_i \in \mathbb{R} \right\}$$

- Shallow networks means one hidden layer, i.e.

$$\Sigma_{[1]}^\sigma = \bigcup_{n=1}^{\infty} \Sigma_n^\sigma \tag{8}$$

- Let $\Omega \subset \mathbb{R}^d$ be a bounded domain
- Want to use shallow networks to approximate functions f on Ω
- Consider two problems:
  - Density of $\Sigma_{[1]}^\sigma$ in $L^2(\Omega)$
  - Approximation rates in $L^2(\Omega)$

Are Shallow Networks Dense?

Can shallow networks approximate arbitrary functions?

- Is $\Sigma^{\sigma}_{[1]}$ dense in $L^2(\Omega)$?

[6] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators.". In: *Neural networks* 2.5 (1989), pp. 359–366, Kurt Hornik. "Approximation capabilities of multilayer feedforward networks". In: *Neural networks* 4.2 (1991), pp. 251–257.

Are Shallow Networks Dense?

Can shallow networks approximate arbitrary functions?

- Is $\Sigma^{\sigma}_{[1]}$ dense in $L^2(\Omega)$?
- It depends upon σ
- If σ is a polynomial,

[6] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators.". In: *Neural networks* 2.5 (1989), pp. 359–366, Kurt Hornik. "Approximation capabilities of multilayer feedforward networks". In: *Neural networks* 4.2 (1991), pp. 251–257.

Are Shallow Networks Dense?

Can shallow networks approximate arbitrary functions?

- Is $\Sigma_{[1]}^{\sigma}$ dense in $L^2(\Omega)$?
- It depends upon σ
- If σ is a polynomial, No!

[6] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators.". In: *Neural networks* 2.5 (1989), pp. 359–366, Kurt Hornik. "Approximation capabilities of multilayer feedforward networks". In: *Neural networks* 4.2 (1991), pp. 251–257.

Are Shallow Networks Dense?

Can shallow networks approximate arbitrary functions?

- Is $\Sigma_{[1]}^{\sigma}$ dense in $L^2(\Omega)$?
- It depends upon σ
- If σ is a polynomial, No!
- Yes! As long as $\sigma \in C^k$ is not a polynomial[6].

[6] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators.". In: *Neural networks* 2.5 (1989), pp. 359–366, Kurt Hornik. "Approximation capabilities of multilayer feedforward networks". In: *Neural networks* 4.2 (1991), pp. 251–257.

Non-polynomial activation function $\Leftrightarrow$ approximation

Lemma

$\Sigma_{[1]}^{\sigma}$ is dense in $C^0(\Omega) \Longleftrightarrow \sigma$ is not a polynomial.

Non-polynomial activation function $\Leftrightarrow$ approximation

Lemma

$\Sigma^{\sigma}_{[1]}$ is dense in $C^0(\Omega) \Longleftrightarrow \sigma$ is not a polynomial.

Proof.

- $[\sigma((\omega + he_j) \cdot x + b) - \sigma(\omega \cdot x + b)]/h \in \Sigma^{\sigma}_{[1]}$,

Non-polynomial activation function $\Leftrightarrow$ approximation

Lemma

$\Sigma^{\sigma}_{[1]}$ is dense in $C^0(\Omega) \Longleftrightarrow \sigma$ is not a polynomial.

Proof.

- $[\sigma((\omega + he_j) \cdot x + b) - \sigma(\omega \cdot x + b)]/h \in \Sigma^{\sigma}_{[1]}$,
- $\frac{\partial}{\partial \omega_j} \sigma(\omega \cdot x + b)|_{\omega=0} = x_j \sigma'(b) \in \overline{\Sigma^{\sigma}_{[1]}}$

Non-polynomial activation function $\Leftrightarrow$ approximation

Lemma

$\Sigma^{\sigma}_{[1]}$ is dense in $C^0(\Omega) \Longleftrightarrow \sigma$ is not a polynomial.

Proof.

- $[\sigma((\omega + he_j) \cdot x + b) - \sigma(\omega \cdot x + b)]/h \in \Sigma^{\sigma}_{[1]}$,
- $\frac{\partial}{\partial \omega_j} \sigma(\omega \cdot x + b)|_{\omega=0} = x_j \sigma'(b) \in \overline{\Sigma^{\sigma}_{[1]}}$
  $\Rightarrow x_j \in \overline{\Sigma^{\sigma}_{[1]}}$ if $\sigma'(b) \neq 0$ for some b.

Non-polynomial activation function $\Leftrightarrow$ approximation

Lemma

$\Sigma^{\sigma}_{[1]}$ is dense in $C^0(\Omega) \Longleftrightarrow \sigma$ is not a polynomial.

Proof.

- $[\sigma((\omega + he_j)\cdot x + b) - \sigma(\omega \cdot x + b)]/h \in \Sigma^{\sigma}_{[1]}$,
- $\frac{\partial}{\partial \omega_j}\sigma(\omega \cdot x + b)|_{\omega=0} = x_j\sigma'(b) \in \overline{\Sigma^{\sigma}_{[1]}}$
  $\Rightarrow x_j \in \overline{\Sigma^{\sigma}_{[1]}}$ if $\sigma'(b) \neq 0$ for some b.
- Similarly $x^{\alpha} = x_1^{\alpha_1}\cdots x_d^{\alpha_d} \in \overline{\Sigma^{\sigma}_{[1]}}$ if $\sigma^{(|\alpha|)}(b) \neq 0$ for some b.

Non-polynomial activation function $\Leftrightarrow$ approximation

Lemma

$\Sigma^{\sigma}_{[1]}$ is dense in $C^0(\Omega) \Longleftrightarrow \sigma$ is not a polynomial.

Proof.

- $[\sigma((\omega + he_j)\cdot x + b) - \sigma(\omega \cdot x + b)]/h \in \Sigma^{\sigma}_{[1]}$,
- $\frac{\partial}{\partial \omega_j}\sigma(\omega \cdot x + b)|_{\omega=0} = x_j \sigma'(b) \in \overline{\Sigma^{\sigma}_{[1]}}$
  $\Rightarrow x_j \in \overline{\Sigma^{\sigma}_{[1]}}$ if $\sigma'(b) \neq 0$ for some b.
- Similarly $x^{\alpha} = x_1^{\alpha_1} \cdots x_d^{\alpha_d} \in \overline{\Sigma^{\sigma}_{[1]}}$ if $\sigma^{(|\alpha|)}(b) \neq 0$ for some b.
- $\overline{\Sigma^{\sigma}_{[1]}}$ contains all polynomials.

$\square$

Rich function class generated by activation function

We note that, for any w_i, b_i

$$\left\{ w_i x + b_i \right\}_{i=1}^{m} \text{ are linearly dependent if } m > d + 1!$$

Rich function class generated by activation function

We note that, for any w_i, b_i

$$\left\{w_i x + b_i\right\}_{i=1}^{m} \text{ are linearly dependent if } m > d+1!$$

Question:

$$\text{Given } w_i \text{ and } b_i, \text{ when are } \left\{\sigma(w_i x + b_i)\right\}_{i=1}^{n} \text{ linearly independent?}$$

Rich function class generated by activation function

We note that, for any w_i, b_i

$$\left\{w_i x + b_i\right\}_{i=1}^{m} \text{ are linearly dependent if } m > d + 1!$$

Question:

Given w_i and b_i, when are $\left\{\sigma(w_i x + b_i)\right\}_{i=1}^{n}$ linearly independent?

But

Theorem (He, Lin, Xu and Zheng 2018, Siegel and Xu 2019)

Assume that σ is NOT a polynomial, then

$$\left\{\sigma(w_i x + b_i)\right\}_{i=1}^{m} \text{ are linearly independent for any } m \geq 1$$

if

Rich function class generated by activation function

We note that, for any w_i, b_i

$$\Big\{w_i x + b_i\Big\}_{i=1}^{m} \text{ are linearly dependent if } m > d+1!$$

Question:

$$\text{Given } w_i \text{ and } b_i\text{, when are } \Big\{\sigma(w_i x + b_i)\Big\}_{i=1}^{n} \text{ linearly independent?}$$

But

Theorem (He, Lin, Xu and Zheng 2018, Siegel and Xu 2019)

Assume that σ is NOT a polynomial, then

$$\Big\{\sigma(w_i x + b_i)\Big\}_{i=1}^{m} \text{ are linearly independent for any } m \geq 1$$

if $\{w_i\}$ are pairwise linearly independent.

A keyword in deep learning

A keyword in deep learning

- nonlinearity

A keyword in deep learning

- nonlinearity $\Rightarrow$ Non-polynomial

Implications

- Shallow networks are universal function approximators!

Implications

- Shallow networks are universal function approximators!
- What about deeper networks?

Implications

- Shallow networks are universal function approximators!
- What about deeper networks?
  - Each layer can approximate the identity to arbitrary accuracy

Implications

- Shallow networks are universal function approximators!
- What about deeper networks?
  - Each layer can approximate the identity to arbitrary accuracy
  - So single hidden layer result extends to deeper networks as well

Implications

- Shallow networks are universal function approximators!
- What about deeper networks?
  - Each layer can approximate the identity to arbitrary accuracy
  - So single hidden layer result extends to deeper networks as well
  - One way of writing this is
    $$\Sigma^{\sigma}_{[1:k]} \subset \overline{\Sigma^{\sigma}_{[1:k+1]}} \tag{9}$$
    if σ is not a polynomial!
  - Note that we don't always have $\Sigma^{\sigma}_{[1:k]} \subset \Sigma^{\sigma}_{[1:k+1]}$

Implications

- Shallow networks are universal function approximators!
- What about deeper networks?
  - Each layer can approximate the identity to arbitrary accuracy
  - So single hidden layer result extends to deeper networks as well
  - One way of writing this is
    $$\Sigma^{\sigma}_{[1:k]} \subset \overline{\Sigma^{\sigma}_{[1:k+1]}} \tag{9}$$
    if σ is not a polynomial!
  - Note that we don't always have $\Sigma^{\sigma}_{[1:k]} \subset \Sigma^{\sigma}_{[1:k+1]}$
- How efficiently can functions be approximated?
  - No control of the size of the network or parameters
  - Want approximation rates for networks with controlled weights

Sampling Argument

Sampling Argument

Define

$$\mathbb{E}g := \int_G g(\omega, x)\lambda(\omega)d\omega. \tag{10}$$

Sampling Argument

Define

$$\mathbb{E}g := \int_G g(\omega, x)\lambda(\omega)d\omega. \tag{10}$$

$$\mathbb{E}_n g := \int_{G^n} g(\omega_1, \cdots, \omega_n)\lambda(\omega_1)\lambda(\omega_2)\ldots\lambda(\omega_n)d\omega_1 d\omega_2 \ldots d\omega_n. \tag{11}$$

Sampling Argument

Define

$$\mathbb{E}g := \int_G g(\omega, x)\lambda(\omega)d\omega. \tag{10}$$

$$\mathbb{E}_n g := \int_{G^n} g(\omega_1, \cdots, \omega_n)\lambda(\omega_1)\lambda(\omega_2)\ldots\lambda(\omega_n)d\omega_1 d\omega_2 \ldots d\omega_n. \tag{11}$$

Lemma

For any $g \in L^\infty(G)$, we have

$$\mathbb{E}_n\Big(\mathbb{E}g - \frac{1}{n}\sum_{i=1}^n g(\omega_i)\Big)^2 = \frac{1}{n}\Big(\mathbb{E}(g^2) - \big(\mathbb{E}(g)\big)^2\Big) \leq \frac{1}{n}\mathbb{E}(g^2). \tag{12}$$

Sampling Argument

1. Let $u = \int_G g(\omega, x)\lambda(x)dx$ and the sampling $u_n = \frac{1}{n}\sum_{i=1}^{n} g(\omega_i, x)$

$$u(x) - u_n(x) = \mathbb{E}g(x) - \frac{1}{n}\sum_{i=1}^{n} g(\omega_i, x); \tag{13}$$

Sampling Argument

1. Let $u = \int_G g(\omega, x)\lambda(x)dx$ and the sampling $u_n = \frac{1}{n}\sum_{i=1}^{n} g(\omega_i, x)$

$$u(x) - u_n(x) = \mathbb{E}g(x) - \frac{1}{n}\sum_{i=1}^{n} g(\omega_i, x); \tag{13}$$

2. It holds that

$$\mathbb{E}_n\left(\left\|\mathbb{E}g - \frac{1}{n}\sum_{i=1}^{n} g(\omega_i)\right\|^2\right) \leq \frac{1}{n}\mathbb{E}(\|g\|^2) \leq \frac{1}{n}; \tag{14}$$

Sampling Argument

1. Let $u = \int_G g(\omega, x)\lambda(x)dx$ and the sampling $u_n = \frac{1}{n}\sum_{i=1}^{n} g(\omega_i, x)$

$$u(x) - u_n(x) = \mathbb{E}g(x) - \frac{1}{n}\sum_{i=1}^{n} g(\omega_i, x); \qquad (13)$$

2. It holds that

$$\mathbb{E}_n\left(\left\|\mathbb{E}g - \frac{1}{n}\sum_{i=1}^{n} g(\omega_i)\right\|^2\right) \le \frac{1}{n}\mathbb{E}(\|g\|^2) \le \frac{1}{n}; \qquad (14)$$

3. There exist $\{\omega_i^*\}_{i=1}^n$ such that $u_n = \frac{1}{n}\sum_{i=1}^{n} g(\omega_i^*, x)$ satisfies

$$\|u - u_n\|_0 \le n^{-\frac{1}{2}} \quad \text{or} \quad \|\mathbb{E}g - \frac{1}{n}\sum_{i=1}^{n} g(\omega_i, x)\|_0 \le n^{-\frac{1}{2}}. \qquad (15)$$

Approximation Rates for Cosine Networks

Approximation Rates for Cosine Networks

- Cosine networks

$$\Sigma_n^{\cos} = \left\{ u_n : u_n = \sum_{i=1}^{n} a_i \cos(w_i x + b_i), \quad \forall a_i, w_i, b_i \right\}$$

- Integral representation of u in terms of cosine functions

Approximation Rates for Cosine Networks

- Cosine networks

$$\Sigma_n^{\cos} = \left\{ u_n : u_n = \sum_{i=1}^{n} a_i \cos(w_i x + b_i), \quad \forall a_i, w_i, b_i \right\}$$

- Integral representation of u in terms of cosine functions

$$u(x) \quad = \quad Re \int_{\mathbb{R}^d} e^{2\pi i \omega \cdot x} \hat{u}(\omega) d\omega$$

Approximation Rates for Cosine Networks

- Cosine networks

$$\Sigma_n^{\cos} = \left\{ u_n : u_n = \sum_{i=1}^{n} a_i \cos(w_i x + b_i), \quad \forall a_i, w_i, b_i \right\}$$

- Integral representation of u in terms of cosine functions

$$\begin{aligned} u(x) &= Re \int_{\mathbb{R}^d} e^{2\pi i \omega \cdot x} \hat{u}(\omega) d\omega \\ &= \int_{\mathbb{R}^d} \cos(2\pi\omega \cdot x + b(\omega)) |\hat{u}(\omega)| d\omega \qquad (\text{by } \hat{u}(\omega) = |\hat{u}(\omega)| e^{ib(\omega)}) \end{aligned}$$

Approximation Rates for Cosine Networks

- Cosine networks

$$\Sigma_n^{\cos} = \left\{ u_n : u_n = \sum_{i=1}^n a_i \cos(w_i x + b_i), \quad \forall a_i, w_i, b_i \right\}$$

- Integral representation of u in terms of cosine functions

$$\begin{aligned} u(x) &= Re \int_{\mathbb{R}^d} e^{2\pi i \omega \cdot x} \hat{u}(\omega) d\omega \\ &= \int_{\mathbb{R}^d} \cos(2\pi\omega \cdot x + b(\omega)) |\hat{u}(\omega)| d\omega \qquad (\text{by } \hat{u}(\omega) = |\hat{u}(\omega)| e^{ib(\omega)}) \\ &= \|\hat{u}\|_{L^1} \int_{\mathbb{R}^d} \cos(2\pi\omega \cdot x + b(\omega)) \lambda(\omega) d\omega \qquad (\lambda(\omega) = \frac{|\hat{u}(\omega)|}{\|\hat{u}\|_{L^1}}) \end{aligned}$$

Approximation Rates for Cosine Networks

- Cosine networks

$$\Sigma_n^{\cos} = \left\{ u_n : u_n = \sum_{i=1}^{n} a_i \cos(w_i x + b_i), \quad \forall a_i, w_i, b_i \right\}$$

- Integral representation of u in terms of cosine functions

$$\begin{aligned}
u(x) &= Re \int_{\mathbb{R}^d} e^{2\pi i \omega \cdot x} \hat{u}(\omega) d\omega \\
&= \int_{\mathbb{R}^d} \cos(2\pi\omega \cdot x + b(\omega)) |\hat{u}(\omega)| d\omega \qquad (\text{by } \hat{u}(\omega) = |\hat{u}(\omega)| e^{ib(\omega)}) \\
&= \|\hat{u}\|_{L^1} \int_{\mathbb{R}^d} \cos(2\pi\omega \cdot x + b(\omega)) \lambda(\omega) d\omega \qquad (\lambda(\omega) = \tfrac{|\hat{u}(\omega)|}{\|\hat{u}\|_{L^1}}) \\
&= \|\hat{u}\|_{L^1} \mathbb{E}\left(g(\omega, x)\right) \qquad (g(\omega, x) = \cos(2\pi\omega \cdot x + b(\omega)))
\end{aligned}$$

Approximation Rates for Cosine Networks

- Cosine networks

$$\Sigma_n^{\cos} = \left\{ u_n : u_n = \sum_{i=1}^{n} a_i \cos(w_i x + b_i), \quad \forall a_i, w_i, b_i \right\}$$

- Integral representation of u in terms of cosine functions

$$\begin{aligned}
u(x) &= Re \int_{\mathbb{R}^d} e^{2\pi i \omega \cdot x} \hat{u}(\omega) d\omega \\
&= \int_{\mathbb{R}^d} \cos(2\pi\omega \cdot x + b(\omega)) |\hat{u}(\omega)| d\omega \qquad (\text{by } \hat{u}(\omega) = |\hat{u}(\omega)| e^{ib(\omega)}) \\
&= \|\hat{u}\|_{L^1} \int_{\mathbb{R}^d} \cos(2\pi\omega \cdot x + b(\omega)) \lambda(\omega) d\omega \qquad (\lambda(\omega) = \tfrac{|\hat{u}(\omega)|}{\|\hat{u}\|_{L^1}}) \\
&= \|\hat{u}\|_{L^1} \mathbb{E}\left(g(\omega, x)\right) \qquad (g(\omega, x) = \cos(2\pi\omega \cdot x + b(\omega)))
\end{aligned}$$

Approximation Rates for Cosine Networks[7]

[7] Lee K Jones. “A simple lemma on greedy approximation in Hilbert space and convergence rates for projection pursuit regression and neural network training”. In: *The annals of Statistics* 20.1 (1992), pp. 608–613.

Approximation Rates for Cosine Networks[7]

1. The preceeding *sampling argument* gives the *approximation rate*:

Theorem

There exists $u_n \in \Sigma^{\cos}_{n,M} = \left\{ \sum_{i=1}^{n} a_i \cos(\omega_i \cdot x + b_i), \sum_{i=1}^{n} |a_i| \le M \right\}$ *such that*

$$\|u - u_n\| \lesssim n^{-\frac{1}{2}} \|\hat{u}\|_{L^1(\mathbb{R}^d)}.$$

[7] Lee K Jones. "A simple lemma on greedy approximation in Hilbert space and convergence rates for projection pursuit regression and neural network training". In: *The annals of Statistics* 20.1 (1992), pp. 608–613.

Spectral Barron Norm

Generalization:

$$\inf_{u_n \in \Sigma^{\cos}_{n,M}} \|u - u_n\|_{H^m(\Omega)} \lesssim n^{-\frac{1}{2}} \|u\|_{B^m(\Omega)}$$

where $B^m(\Omega)$ is the spectral Barron space.

$$\|u\|_{B^m(\Omega)} = \inf_{u_e|_\Omega = u} \int_{\mathbb{R}^d} (1 + |\omega|)^m |\hat{u}_e(\omega)| d\omega$$

Approximation Rates for the ReLU^k Network

Approximation Rates for the ReLU^k Network

Note that

$$u(x) - \sum_{|\alpha|\leq k} \frac{1}{\alpha!} D^\alpha u(0) x^\alpha = \|\rho\|_{L^1(G)} \int_G g(x,\theta)\lambda(\theta) d\theta = \|\rho\|_{L^1(G)} \mathbb{E}(g)$$

with $\theta = (z, t, \omega) \in G = \{-1, 1\} \times [0, T] \times \mathbb{R}^d$,

$$\rho(\theta) = \frac{1}{(2\pi)^d} |s(zt, \omega)| |\hat{u}(\omega)| \|\omega\|^{k+1}$$

$$s(zt, \omega) = \begin{cases} (-1)^{\frac{k+1}{2}} \cos(z\|\omega\| t + b(\omega)) & k \text{ is odd}, \\ (-1)^{\frac{k+2}{2}} \sin(z\|\omega\| t + b(\omega)) & k \text{ is even}. \end{cases} \tag{16}$$

$$g(x,\theta) = (z\bar{\omega} \cdot x - t)_+^k \mathrm{sgn} s(zt, \omega), \qquad \lambda(\theta) = \frac{\rho(\theta)}{\|\rho\|_{L^1(G)}}. \tag{17}$$

Approximation Rates for the ReLU^{k}[8][9]

Lemma (Sampling Analysis)

There exist ω_i, t_i such that

$$\|u - u_n\|_{H^m} \lesssim n^{-\frac{1}{2}} \|u\|_{B^{m+k+1}} \tag{18}$$

with $u_n = \sum_{|\alpha|\le k} \frac{1}{\alpha!} D^\alpha u(0) x^\alpha + \frac{c}{k!n} \sum_{i=1}^{n} \beta_i (\omega_i \cdot x - t_i)_+^k$.

[8] Jason M Klusowski and Andrew R Barron. "Risk bounds for high-dimensional ridge function combinations including neural networks". In: *arXiv preprint arXiv:1607.01434* (2016).

[9] Jinchao Xu. "The finite neuron method and convergence analysis". In: *arXiv preprint arXiv:2010.01458* (2020).

Approximation Rates for the ReLU^k[8][9]

Lemma (Sampling Analysis)

There exist ω_i, t_i such that

$$\|u-u_n\|_{H^m} \lesssim n^{-\frac{1}{2}}\|u\|_{B^{m+k+1}} \tag{18}$$

with $u_n = \sum_{|\alpha|\le k} \frac{1}{\alpha!}D^\alpha u(0)x^\alpha + \frac{c}{k!n}\sum_{i=1}^n \beta_i(\omega_i\cdot x - t_i)_+^k.$

Lemma (More Refined Analysis: Stratified Method)

There exist ω_i, t_i such that

$$\|u-u_n\|_{H^m} \lesssim n^{-\frac{1}{2}-\frac{1}{d}}\|u\|_{B^{m+k+1}} \tag{19}$$

with $\bar\omega_i = \frac{\omega_i}{\|\omega_i\|}$ *and* $u_n = \sum_{|\alpha|\le k} \frac{1}{\alpha!}D^\alpha u(0)x^\alpha + \frac{c}{k!n}\sum_{i=1}^n \beta_i(\bar\omega_i\cdot x - t_i)_+^k.$

[8] Jason M Klusowski and Andrew R Barron. "Risk bounds for high-dimensional ridge function combinations including neural networks". In: *arXiv preprint arXiv:1607.01434* (2016).

[9] Jinchao Xu. "The finite neuron method and convergence analysis". In: *arXiv preprint arXiv:2010.01458* (2020).

More General Activation Functions

- Using similar techniques, we can extend these rates to more general activation functions as well[10]

Theorem

Suppose that $\sigma \in L^\infty$ *and satisfies the decay condition*

$$|\sigma(t)| \lesssim (1 + |t|)^p \tag{20}$$

for some $p > 1$. *Let* $u \in L^2(\Omega)$. *Then we have*

$$\inf_{u_n \in \Sigma_n^\sigma} \|u - u_n\|_{L^2(\Omega)} \lesssim \|u\|_{B^1(\Omega)} n^{-\frac{1}{2}}. \tag{21}$$

[10] Jonathan W Siegel and Jinchao Xu. "Approximation rates for neural networks with general activation functions". In: *Neural Networks* (2020).

More General Activation Functions (cont.)

- At the cost of a worse rate, we can even drop almost all assumptions on[11] σ

Theorem

Suppose that $\sigma \in L^\infty$ and $\hat{\sigma}$ is continuous and non-zero at a single point. Let $u \in L^2(\Omega)$. Then we have

$$\inf_{u_n \in \Sigma_n^\sigma} \|u - u_n\|_{L^2(\Omega)} \lesssim \|u\|_{B^1(\Omega)} n^{-\frac{1}{4}}. \tag{22}$$

- In particular:
  - Holds for all $\sigma \in L^1 \cap L^\infty$
  - Holds for all $\sigma \in BV$

[11] Jonathan W Siegel and Jinchao Xu. "Approximation rates for neural networks with general activation functions". In: *Neural Networks* (2020).

Stable Neural Network Approximation

- Recall: approximation of cosine neural network from the function class (by sampling argument):

$$\Sigma_{n,M}^{\cos} = \left\{ \sum_{i=1}^{n} a_i \cos(\omega_i \cdot x + b_i), \sum_{i=1}^{n} |a_i| \leq M, \quad M = \|\hat{u}\|_{L^1} \right\}$$

Stable Neural Network Approximation

- Recall: approximation of cosine neural network from the function class (by sampling argument):

$$\Sigma_{n,M}^{\cos} = \left\{ \sum_{i=1}^{n} a_i \cos(\omega_i \cdot x + b_i), \sum_{i=1}^{n} |a_i| \le M, \quad M = \|\hat{u}\|_{L^1} \right\}$$

- How far can the sampling argument go?

Stable Neural Network Approximation

- Recall: approximation of cosine neural network from the function class (by sampling argument):

$$\Sigma_{n,M}^{\cos} = \left\{ \sum_{i=1}^{n} a_i \cos(\omega_i \cdot x + b_i), \sum_{i=1}^{n} |a_i| \le M, \quad M = \|\hat{u}\|_{L^1} \right\}$$

- How far can the sampling argument go?
- Consider approximation from the class

$$\Sigma_{n,M}^{\sigma} := \left\{ \sum_{i=1}^{n} a_i \sigma(\omega_i \cdot x + b_i),\ \omega_i \in \mathbb{R}^d,\ b_i \in \mathbb{R}, \sum_{i=1}^{n} |a_i| \le M \right\} \tag{23}$$

of neural networks with ℓ^1-bounded outer coefficients.

Stable Neural Network Approximation

- Recall: approximation of cosine neural network from the function class (by sampling argument):

$$\Sigma^{\cos}_{n,M} = \left\{ \sum_{i=1}^{n} a_i \cos(\omega_i \cdot x + b_i), \sum_{i=1}^{n} |a_i| \le M, \quad M = \|\hat{u}\|_{L^1} \right\}$$

- How far can the sampling argument go?
- Consider approximation from the class

$$\Sigma^{\sigma}_{n,M} := \left\{ \sum_{i=1}^{n} a_i \sigma(\omega_i \cdot x + b_i),\ \omega_i \in \mathbb{R}^d,\ b_i \in \mathbb{R}, \sum_{i=1}^{n} |a_i| \le M \right\} \tag{23}$$

  of neural networks with ℓ^1-bounded outer coefficients.
- More generally for a dictionary $\mathbb{D} \subset H = L^2(\Omega)$, consider

$$\Sigma_{n,M}(\mathbb{D}) = \left\{ \sum_{i=1}^{n} a_i h_i,\ h_i \in \mathbb{D},\ \sum_{i=1}^{n} |a_i| \le M \right\} \tag{24}$$

Stable Neural Network Approximation

- Recall: approximation of cosine neural network from the function class (by sampling argument):

$$\Sigma_{n,M}^{\cos} = \left\{ \sum_{i=1}^{n} a_i \cos(\omega_i \cdot x + b_i), \sum_{i=1}^{n} |a_i| \le M, \quad M = \|\hat{u}\|_{L^1} \right\}$$

- How far can the sampling argument go?
- Consider approximation from the class

$$\Sigma_{n,M}^{\sigma} := \left\{ \sum_{i=1}^{n} a_i \sigma(\omega_i \cdot x + b_i),\ \omega_i \in \mathbb{R}^d,\ b_i \in \mathbb{R}, \sum_{i=1}^{n} |a_i| \le M \right\} \tag{23}$$

of neural networks with ℓ^1-bounded outer coefficients.
- More generally for a dictionary $\mathbb{D} \subset H = L^2(\Omega)$, consider

$$\Sigma_{n,M}(\mathbb{D}) = \left\{ \sum_{i=1}^{n} a_i h_i,\ h_i \in \mathbb{D},\ \sum_{i=1}^{n} |a_i| \le M \right\} \tag{24}$$

- Let $M < \infty$ be fixed and consider approximation as $n \to \infty$.

Stable Dictionary Approximation Space

Siegel & Xu, 2021[12]:

- Define a closed convex hull of $\pm\mathbb{D}$:

$$B_1(\mathbb{D}) = \overline{\cup_{n=1}^{\infty}\Sigma_{n,1}^{\sigma}}, \tag{25}$$

[12] Jonathan W. Siegel and Jinchao Xu. *Optimal Approximation Rates and Metric Entropy of ReLU^k and Cosine Networks*. 2021.

Stable Dictionary Approximation Space

Siegel & Xu, 2021[12]:

- Define a closed convex hull of $\pm\mathbb{D}$:

$$B_1(\mathbb{D}) = \overline{\cup_{n=1}^{\infty} \Sigma_{n,1}^{\sigma}}, \tag{25}$$

- Define a norm

$$\|f\|_{\mathcal{K}_1(\mathbb{D})} = \inf\{r > 0 : f \in rB_1(\mathbb{D})\}, \tag{26}$$

  as the guage of the set $B_1(\mathbb{D})$.

[12] Jonathan W. Siegel and Jinchao Xu. *Optimal Approximation Rates and Metric Entropy of ReLU^k and Cosine Networks*. 2021.

Stable Dictionary Approximation Space

Siegel & Xu, 2021[12]:

- Define a closed convex hull of $\pm\mathbb{D}$:

$$B_1(\mathbb{D}) = \overline{\cup_{n=1}^{\infty}\Sigma_{n,1}^{\sigma}}, \tag{25}$$

- Define a norm

$$\|f\|_{\mathcal{K}_1(\mathbb{D})} = \inf\{r > 0 : f \in rB_1(\mathbb{D})\}, \tag{26}$$

  as the guage of the set $B_1(\mathbb{D})$.
- The unit ball is

$$\{f \in H : \|f\|_{\mathcal{K}_1(\mathbb{D})} \leq 1\} = B_1(\mathbb{D}). \tag{27}$$

[12] Jonathan W. Siegel and Jinchao Xu. *Optimal Approximation Rates and Metric Entropy of ReLU^k and Cosine Networks*. 2021.

Stable Dictionary Approximation Space

Siegel & Xu, 2021[12]:

- Define a closed convex hull of $\pm\mathbb{D}$:

$$B_1(\mathbb{D}) = \overline{\cup_{n=1}^{\infty}\Sigma_{n,1}^{\sigma}}, \tag{25}$$

- Define a norm

$$\|f\|_{\mathcal{K}_1(\mathbb{D})} = \inf\{r > 0 : f \in rB_1(\mathbb{D})\}, \tag{26}$$

as the guage of the set $B_1(\mathbb{D})$.

- The unit ball is

$$\{f \in H : \|f\|_{\mathcal{K}_1(\mathbb{D})} \leq 1\} = B_1(\mathbb{D}). \tag{27}$$

- We have

$$\{f \in H : \|f\|_{\mathcal{K}_1(\mathbb{D})} < \infty\} = \cup_{M>0}\overline{\cup_{n=1}^{\infty}\Sigma_{n,M}^{\sigma}} \tag{28}$$

is a Banach space.

[12] Jonathan W. Siegel and Jinchao Xu. *Optimal Approximation Rates and Metric Entropy of ReLU^k and Cosine Networks*. 2021.

Example: $H = \ell^2$

- Let $H = \ell^2$, $\mathbb{D} = \{e_1, e_2, ...\}$.
- What is $B_1(\mathbb{D})$?

Example: $H = \ell^2$

- Let $H = \ell^2$, $\mathbb{D} = \{e_1, e_2, ...\}$.
- What is $B_1(\mathbb{D})$?
- The convex hull of $\pm\mathbb{D}$ is

$$B_1(\mathbb{D}) = \{(a_1, a_2, ...) \in \ell^2 : \sum_{i=1}^{\infty} |a_i| \leq 1\} \tag{29}$$

Example: $H = \ell^2$

- Let $H = \ell^2$, $\mathbb{D} = \{e_1, e_2, ...\}$.
- What is $B_1(\mathbb{D})$?
- The convex hull of $\pm\mathbb{D}$ is

$$B_1(\mathbb{D}) = \{(a_1, a_2, ...) \in \ell^2 : \sum_{i=1}^{\infty} |a_i| \leq 1\} \tag{29}$$

- Thus the norm is given by

$$\mathcal{K}_1(\mathbb{D}) = \ell^1 \subset \ell^2. \tag{30}$$

Stable Dictionary Approximation Space

Theorem (Siegel & Xu 2021)

A function $f \in H = L^2(\Omega)$ can be approximated at all, i.e.

$$\lim_{n\to\infty} \inf_{f_n \in \Sigma_{n,M}(\mathbb{D})} \|f - f_n\|_H = 0, \tag{31}$$

for a fixed $M < \infty$ if and only if

$$f \in MB_1(\mathbb{D}) \subset \mathcal{K}_1(\mathbb{D}).$$

Stable Dictionary Approximation Space

Theorem (Siegel & Xu 2021)

A function $f \in H = L^2(\Omega)$ can be approximated at all, i.e.

$$\lim_{n\to\infty} \inf_{f_n \in \Sigma_{n,M}(\mathbb{D})} \|f - f_n\|_H = 0, \tag{31}$$

for a fixed $M < \infty$ if and only if

$$f \in MB_1(\mathbb{D}) \subset \mathcal{K}_1(\mathbb{D}).$$

Furthermore, if

$$\|\mathbb{D}\| \equiv \sup_{h\in\mathbb{D}} \|h\|_H < \infty$$

we have

$$\inf_{f_n \in \Sigma_{n,M}(\mathbb{D})} \|f - f_n\|_H \leq n^{-\frac{1}{2}} \|\mathbb{D}\| \|f\|_{\mathcal{K}_1(\mathbb{D})}. \tag{32}$$

The Spectral Barron Space

- Let $f \in B_1(\mathbb{D})$, $H = L^2(\Omega)$, $\Omega = B_1^d = \{x \in \mathbb{R}^d : |x| \leq 1\}$, and

$$\mathbb{D} = \mathbb{F}_s^d := \{(1 + |\omega|)^{-s} e^{2\pi i \omega \cdot x} : \omega \in \mathbb{R}^d\} \tag{33}$$

[13] Jonathan W. Siegel and Jinchao Xu. *Optimal Approximation Rates and Metric Entropy of ReLU^k and Cosine Networks*. 2021.

The Spectral Barron Space

- Let $f \in B_1(\mathbb{D})$, $H = L^2(\Omega)$, $\Omega = B_1^d = \{x \in \mathbb{R}^d : |x| \leq 1\}$, and

$$\mathbb{D} = \mathbb{F}_s^d := \{(1+|\omega|)^{-s} e^{2\pi i\omega\cdot x} : \omega \in \mathbb{R}^d\} \tag{33}$$

- In this case the norm is characterized by[13]

$$\|f\|_{\mathcal{K}_1(\mathbb{F}_s^d)} = \inf_{f_e|_{B_1^d}=f} \int_{\mathbb{R}^d} (1+|\xi|)^s |\hat{f}_e(\xi)| d\xi, \tag{34}$$

where the infimum is taken over all extensions $f_e \in L^1(\mathbb{R}^d)$.

[13] Jonathan W. Siegel and Jinchao Xu. *Optimal Approximation Rates and Metric Entropy of ReLU^k and Cosine Networks*. 2021.

The Spectral Barron Space

- Let $f \in B_1(\mathbb{D})$, $H = L^2(\Omega)$, $\Omega = B_1^d = \{x \in \mathbb{R}^d : |x| \leq 1\}$, and

$$\mathbb{D} = \mathbb{F}_s^d := \{(1 + |\omega|)^{-s} e^{2\pi i \omega \cdot x} \ : \ \omega \in \mathbb{R}^d\} \tag{33}$$

- In this case the norm is characterized by[13]

$$\|f\|_{\mathcal{K}_1(\mathbb{F}_s^d)} = \inf_{f_e|_{B_1^d} = f} \int_{\mathbb{R}^d} (1 + |\xi|)^s |\hat{f}_e(\xi)| d\xi, \tag{34}$$

  where the infimum is taken over all extensions $f_e \in L^1(\mathbb{R}^d)$.

- Property:

$$H^{s+\frac{d}{2}+\varepsilon}(\Omega) \hookrightarrow B^s(\Omega) \hookrightarrow W^{s,\infty}(\Omega). \tag{35}$$

[13] Jonathan W. Siegel and Jinchao Xu. *Optimal Approximation Rates and Metric Entropy of ReLU^k and Cosine Networks*. 2021.

The Barron Space

The results are proved in Siegel and Xu 2021[14]

- Let $H = L^2(\Omega)$, $\Omega = B_1^d = \{x \in \mathbb{R}^d : |x| \leq 1\}$, and

$$\mathbb{D} = \mathbb{P}_k^d := \{\sigma_k(\omega \cdot x + b) : \ \omega \in S^{d-1}, \ b \in [-2, 2]\}, \tag{36}$$

where $\sigma_k = [\max(0, x)]^k$.

[14] Jonathan W. Siegel and Jinchao Xu. *Optimal Approximation Rates and Metric Entropy of ReLU^k and Cosine Networks*. 2021.

[15] W. E, Chao Ma, and Lei Wu. "Barron spaces and the compositional function spaces for neural network models". In: *arXiv preprint arXiv:1906.08039* (2019).

[16] Andrew R Barron. "Universal approximation bounds for superpositions of a sigmoidal function". In: *IEEE Transactions on Information theory* 39.3 (1993), pp. 930–945.

The Barron Space

The results are proved in Siegel and Xu 2021[14]

- Let $H = L^2(\Omega)$, $\Omega = B_1^d = \{x \in \mathbb{R}^d : |x| \le 1\}$, and

$$\mathbb{D} = \mathbb{P}_k^d := \{\sigma_k(\omega \cdot x + b) : \ \omega \in S^{d-1}, \ b \in [-2, 2]\}, \tag{36}$$

  where $\sigma_k = [\max(0, x)]^k$.
- When $k = 1$, $\mathcal{K}_1(\mathbb{P}_k^d)$ is equivalentto the Barron space (introduced in[15]).

---

[14] Jonathan W. Siegel and Jinchao Xu. *Optimal Approximation Rates and Metric Entropy of ReLU^k and Cosine Networks*. 2021.

[15] W. E, Chao Ma, and Lei Wu. "Barron spaces and the compositional function spaces for neural network models". In: *arXiv preprint arXiv:1906.08039* (2019).

[16] Andrew R Barron. "Universal approximation bounds for superpositions of a sigmoidal function". In: *IEEE Transactions on Information theory* 39.3 (1993), pp. 930–945.

The Barron Space

The results are proved in Siegel and Xu 2021[14]

- Let $H = L^2(\Omega)$, $\Omega = B_1^d = \{x \in \mathbb{R}^d : |x| \leq 1\}$, and

$$\mathbb{D} = \mathbb{P}_k^d := \{\sigma_k(\omega \cdot x + b) : \ \omega \in S^{d-1}, \ b \in [-2, 2]\}, \tag{36}$$

  where $\sigma_k = [\max(0, x)]^k$.
- When $k = 1$, $\mathcal{K}_1(\mathbb{P}_k^d)$ is equivalentto the Barron space (introduced in[15]).
- When $k = 0$, $d = 1$, $\mathcal{K}_1(\mathbb{P}_k^d) = BV([-1, 1])$.

[14] Jonathan W. Siegel and Jinchao Xu. *Optimal Approximation Rates and Metric Entropy of ReLU^k and Cosine Networks*. 2021.

[15] W. E, Chao Ma, and Lei Wu. "Barron spaces and the compositional function spaces for neural network models". In: *arXiv preprint arXiv:1906.08039* (2019).

[16] Andrew R Barron. "Universal approximation bounds for superpositions of a sigmoidal function". In: *IEEE Transactions on Information theory* 39.3 (1993), pp. 930–945.

The Barron Space

The results are proved in Siegel and Xu 2021[14]

- Let $H = L^2(\Omega)$, $\Omega = B_1^d = \{x \in \mathbb{R}^d : |x| \leq 1\}$, and

$$\mathbb{D} = \mathbb{P}_k^d := \{\sigma_k(\omega \cdot x + b) : \ \omega \in S^{d-1}, \ b \in [-2, 2]\}, \tag{36}$$

  where $\sigma_k = [\max(0, x)]^k$.
- When $k = 1$, $\mathcal{K}_1(\mathbb{P}_k^d)$ is equivalentto the Barron space (introduced in[15]).
- When $k = 0$, $d = 1$, $\mathcal{K}_1(\mathbb{P}_k^d) = BV([-1, 1])$.
- We have$\mathcal{K}_1(\mathbb{P}_k^d) \supset \mathcal{K}_1(\mathbb{F}_{k+1}^d)$ (for $k = 0$, Barron 1993[16])

[14] Jonathan W. Siegel and Jinchao Xu. *Optimal Approximation Rates and Metric Entropy of ReLU^k and Cosine Networks*. 2021.

[15] W. E, Chao Ma, and Lei Wu. “Barron spaces and the compositional function spaces for neural network models”. In: *arXiv preprint arXiv:1906.08039* (2019).

[16] Andrew R Barron. “Universal approximation bounds for superpositions of a sigmoidal function”. In: *IEEE Transactions on Information theory* 39.3 (1993), pp. 930–945.

Previous Best Results

For some dictionaries $\mathbb{D}$, the $n^{-\frac{1}{2}}$ approximation rate can be improved!

- For $\mathbb{D} = \mathbb{P}_0^d$, we have[17]

$$\sup_{f\in B_1(\mathbb{D})} \inf_{f_n\in\Sigma_{n,M}} \|f - f_n\|_{L^2(B_1^d)} \lesssim n^{-\frac{1}{2}-\frac{1}{2d}}. \tag{37}$$

- For $\mathbb{D} = \mathbb{P}_k^d$ for $k \geq 1$, we have[18],[19], if f is in some spectral Barron space:

$$\inf_{f_n\in\Sigma_{n,M}} \|f - f_n\|_{L^2(B_1^d)} \lesssim n^{-\frac{1}{2}-\frac{1}{d}}. \tag{38}$$

[17] Yuly Makovoz. "Random approximants and neural networks". In: *Journal of Approximation Theory* 85.1 (1996), pp. 98–109.

[18] Jason M Klusowski and Andrew R Barron. "Approximation by Combinations of ReLU and Squared ReLU Ridge Functions With ℓ^1 and ℓ^0 Controls". In: *IEEE Transactions on Information Theory* 64.12 (2018), pp. 7649–7656.

[19] Jinchao Xu. "Finite Neuron Method and Convergence Analysis". In: *Communications in Computational Physics* 28.5 (2020), pp. 1707–1745.

Previous Best Results

For some dictionaries $\mathbb{D}$, the $n^{-\frac{1}{2}}$ approximation rate can be improved!

- For $\mathbb{D} = \mathbb{P}_0^d$, we have[17]

$$\sup_{f \in B_1(\mathbb{D})} \inf_{f_n \in \Sigma_{n,M}} \|f - f_n\|_{L^2(B_1^d)} \lesssim n^{-\frac{1}{2} - \frac{1}{2d}}. \tag{37}$$

- For $\mathbb{D} = \mathbb{P}_k^d$ for $k \geq 1$, we have[18],[19], if f is in some spectral Barron space:

$$\inf_{f_n \in \Sigma_{n,M}} \|f - f_n\|_{L^2(B_1^d)} \lesssim n^{-\frac{1}{2} - \frac{1}{d}}. \tag{38}$$

- What are the optimal approximation rates?

[17] Yuly Makovoz. "Random approximants and neural networks". In: *Journal of Approximation Theory* 85.1 (1996), pp. 98–109.

[18] Jason M Klusowski and Andrew R Barron. "Approximation by Combinations of ReLU and Squared ReLU Ridge Functions With ℓ^1 and ℓ^0 Controls". In: *IEEE Transactions on Information Theory* 64.12 (2018), pp. 7649–7656.

[19] Jinchao Xu. "Finite Neuron Method and Convergence Analysis". In: *Communications in Computational Physics* 28.5 (2020), pp. 1707–1745.

New Optimal Bounds

We have the optimal approximation rates[20]

Theorem

For $\mathbb{D} = \mathbb{P}_k^d$ for $k \geq 1$, we have

$$n^{-\frac{1}{2}-\frac{2k+1}{2d}} \lesssim \sup_{f \in B_1(\mathbb{D})} \inf_{f_n \in \Sigma_{n,M}} \|f - f_n\|_{L^2(\Omega)} \lesssim n^{-\frac{1}{2}-\frac{2k+1}{2d}} \tag{39}$$

[20] Jonathan W. Siegel and Jinchao Xu. *Optimal Approximation Rates and Metric Entropy of ReLU^k and Cosine Networks*. 2021.

[21] Qun Lin, Hehu Xie, and Jinchao Xu. “Lower bounds of the discretization error for piecewise polynomials”. In: *Mathematics of Computation* 83.285 (2014), pp. 1–13.

New Optimal Bounds

We have the optimal approximation rates[20]

Theorem

For $\mathbb{D} = \mathbb{P}_k^d$ for $k \geq 1$, we have

$$n^{-\frac{1}{2}-\frac{2k+1}{2d}} \lesssim \sup_{f \in B_1(\mathbb{D})} \inf_{f_n \in \Sigma_{n,M}} \|f - f_n\|_{L^2(\Omega)} \lesssim n^{-\frac{1}{2}-\frac{2k+1}{2d}} \tag{39}$$

In comparison: optimal bound for finite elements[21]

Theorem

Assume that V_h^k is a finite element of degree k on quasi-uniform mesh $\{\mathcal{T}_h\}$ of $\mathcal{O}(N)$ elements. Assume u is sufficiently smooth and not piecewise polynomials, then we have

$$c(u)n^{-\frac{k}{d}} \leq \inf_{v_h \in V_h^k} \|u - v_h\|_{L^2(\Omega)} \leq C(u)n^{-\frac{k}{d}} = \mathcal{O}(h^k). \tag{40}$$

[20] Jonathan W. Siegel and Jinchao Xu. *Optimal Approximation Rates and Metric Entropy of ReLU^k and Cosine Networks*. 2021.

[21] Qun Lin, Hehu Xie, and Jinchao Xu. "Lower bounds of the discretization error for piecewise polynomials". In: *Mathematics of Computation* 83.285 (2014), pp. 1–13.

Removing the constraint that $\sum_{i=1}^{n} |a_i| \leq M$

Define

$$\Sigma_n^k := \left\{ \sum_{i=1}^{n} a_i \sigma_k(\omega_i \cdot x + b_i),\ \omega_i \in \mathbb{R}^d,\ b_i \in \mathbb{R}, a_i \in \mathbb{R} \right\}. \tag{41}$$

Then Σ_n^k has the following approximation property[22]

Theorem (Siegel and Xu)

$$\inf_{f_n \in \Sigma_n^k} \|f - f_n\| \lesssim \begin{cases} n^{-\frac{1}{2}} & \|f\|_{\mathcal{K}_1(\mathbb{F}_s^d)} \quad \textit{if} \quad s = \frac{1}{2} \\ n^{-(k+1)} \log n & \|f\|_{\mathcal{K}_1(\mathbb{F}_s^d)} \quad \textit{for some} \quad s > 1 \end{cases} \tag{42}$$

- Improves result of Barron[23] by relaxing condition on f

[22] Jonathan W Siegel and Jinchao Xu. "High-Order Approximation Rates for Neural Networks with ReLU^k Activation Functions". In: *arXiv preprint arXiv:2012.07205* (2020).

[23] Andrew R Barron. "Universal approximation bounds for superpositions of a sigmoidal function". In: *IEEE Transactions on Information theory* 39.3 (1993), pp. 930–945.

Removing the constraint that $\sum_{i=1}^{n} |a_i| \le M$

Define

$$\Sigma_n^k := \left\{ \sum_{i=1}^{n} a_i \sigma_k(\omega_i \cdot x + b_i),\ \omega_i \in \mathbb{R}^d,\ b_i \in \mathbb{R}, a_i \in \mathbb{R} \right\}. \tag{41}$$

Then Σ_n^k has the following approximation property[22]

Theorem (Siegel and Xu)

$$\inf_{f_n \in \Sigma_n^k} \|f - f_n\| \lesssim \begin{cases} n^{-\frac{1}{2}} & \|f\|_{\mathcal{K}_1(\mathbb{F}_s^d)} \quad \textit{if} \quad s = \frac{1}{2} \\ n^{-(k+1)} \log n & \|f\|_{\mathcal{K}_1(\mathbb{F}_s^d)} \quad \textit{for some} \quad s > 1 \end{cases} \tag{42}$$

- Improves result of Barron[23] by relaxing condition on f
- Shows that very high order approximation rates can be attained with sufficient smoothness

[22] Jonathan W Siegel and Jinchao Xu. "High-Order Approximation Rates for Neural Networks with ReLU^k Activation Functions". In: *arXiv preprint arXiv:2012.07205* (2020).

[23] Andrew R Barron. "Universal approximation bounds for superpositions of a sigmoidal function". In: *IEEE Transactions on Information theory* 39.3 (1993), pp. 930–945.

Removing the constraint that $\sum_{i=1}^n |a_i| \leq M$

Define

$$\Sigma_n^k := \left\{\sum_{i=1}^n a_i\sigma_k(\omega_i \cdot x + b_i),\ \omega_i \in \mathbb{R}^d,\ b_i \in \mathbb{R}, a_i \in \mathbb{R}\right\}. \tag{41}$$

Then Σ_n^k has the following approximation property[22]

Theorem (Siegel and Xu)

$$\inf_{f_n \in \Sigma_n^k} \|f - f_n\| \lesssim \begin{cases} n^{-\frac{1}{2}} & \|f\|_{\mathcal{K}_1(\mathbb{F}_s^d)} & \text{if} \quad s = \frac{1}{2} \\ n^{-(k+1)} \log n & \|f\|_{\mathcal{K}_1(\mathbb{F}_s^d)} & \text{for some} \quad s > 1 \end{cases} \tag{42}$$

- Improves result of Barron[23] by relaxing condition on f
- Shows that very high order approximation rates can be attained with sufficient smoothness
- Comparison with FEM:

$$\inf_{w \in \Sigma_n^k} \|u - w\| \approx \left\{\inf_{v \in V_n^k} \|u - v\|\right\}^d.$$

[22] Jonathan W Siegel and Jinchao Xu. "High-Order Approximation Rates for Neural Networks with ReLU^k Activation Functions". In: *arXiv preprint arXiv:2012.07205* (2020).

[23] Andrew R Barron. "Universal approximation bounds for superpositions of a sigmoidal function". In: *IEEE Transactions on Information theory* 39.3 (1993), pp. 930–945.

References

- J. He, L, Li, J. Xu and C. Zheng, ReLU Deep Neural Networks and Linear Finite Elements. Journal of Computational Mathematics 38.3 (2020), pp. 502527.
- J. Xu, The Finite Neuron Method and Convergence Analysis, Commun. Comput. Phys., 28, pp. 1707-1745, (2020).
- J. W. Siegel and J. Xu, "High-Order Approximation Rates for Neural Networks with ReLU^k Activation Functions." arXiv preprint arXiv:2012.07205 (2020).
- J. W. Siegel and J. Xu, "Approximation Rates for Neural Networks with General Activation Functions." Neural Networks (2020).
- J. W. Siegel and J. Xu, "Optimal Approximation Rates and Entropy Bounds for ReLU^k Networks." arXiv preprint arXiv:2101.12365 (2021)

Thank you!