Solving Mathematical Problems by Deep Learning: Partial Differential Equations

Gitta Kutyniok

(Technische Universität Berlin and University of Tromsø)

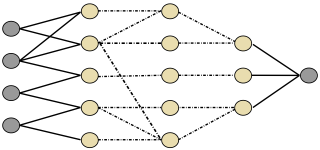
2019 Woudschoten Conference
Zeist, The Netherlands, October 9–11, 2019

Mathematics of Deep Neural Networks

The Mathematics of Deep Neural Networks

Definition:

Assume the following notions:



- $d \in \mathbb{N}$: Dimension of input layer.
- L: Number of layers.
- N: Number of neurons.
- $\rho : \mathbb{R} \to \mathbb{R}$: (Non-linear) function called *activation function*.
- $T_\ell : \mathbb{R}^{N_{\ell-1}} \to \mathbb{R}^{N_\ell}$, $\ell = 1, \ldots, L$: Affine linear maps.

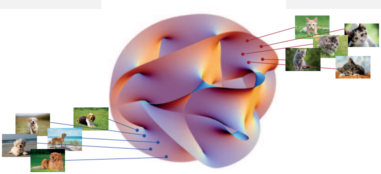
Then $\Phi : \mathbb{R}^d \to \mathbb{R}^{N_L}$ given by

$$\Phi(x) = T_L \rho(T_{L-1} \rho(\ldots \rho(T_1(x)))), \quad x \in \mathbb{R}^d,$$

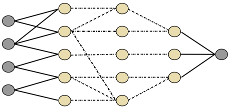
is called *(deep) neural network (DNN)*.

Training of Deep Neural Networks

High-Level Set Up:



- Samples $(x_i, f(x_i))_{i=1}^m$ of a function such as $f : \mathcal{M} \to \{1, 2, \ldots, K\}$.



- Select an architecture of a deep neural network, i.e., a choice of d, L, $(N_\ell)_{\ell=1}^L$, and ρ.
  Sometimes selected entries of the matrices $(A_\ell)_{\ell=1}^L$, i.e., weights, are set to zero at this point.
- Learn the affine-linear functions $(T_\ell)_{\ell=1}^L = (A_\ell \cdot + b_\ell)_{\ell=1}^L$ by

$$\min_{(A_\ell, b_\ell)_\ell} \sum_{i=1}^m \mathcal{L}(\Phi_{(A_\ell, b_\ell)_\ell}(x_i), f(x_i)) + \lambda \mathcal{R}((A_\ell, b_\ell)_\ell)$$

yielding the network $\Phi_{(A_\ell, b_\ell)_\ell} : \mathbb{R}^d \to \mathbb{R}^{N_L}$,

$$\Phi_{(A_\ell, b_\ell)_\ell}(x) = T_L \rho(T_{L-1} \rho(\ldots \rho(T_1(x)))).$$

This is often done by stochastic gradient descent.

Goal: $\Phi_{(A_\ell, b_\ell)_\ell} \approx f$

Fundamental Questions concerning Deep Neural Networks

- *Expressivity:*
  - How powerful is the network architecture?
  - Can it indeed represent the correct functions?

$\rightsquigarrow$ *Applied Harmonic Analysis, Approximation Theory, ...*

- *Learning:*
  - Why does the current learning algorithm produce anything reasonable?
  - What are good starting values?

$\rightsquigarrow$ *Differential Geometry, Optimal Control, Optimization, ...*

- *Generalization:*
  - Why do deep neural networks perform that well on data sets, which do not belong to the input-output pairs from a training set?
  - What impact has the depth of the network?

$\rightsquigarrow$ *Learning Theory, Optimization, Statistics, ...*

- *Interpretability:*
  - Why did a trained deep neural network reach a certain decision?
  - Which components of the input do contribute most?

$\rightsquigarrow$ *Information Theory, Uncertainty Quantification, ...*

What is Interpretability?

Main Questions: Given a trained deep neural network...

- Which input features contribute most to the decision?
- How can the outcome be explained?

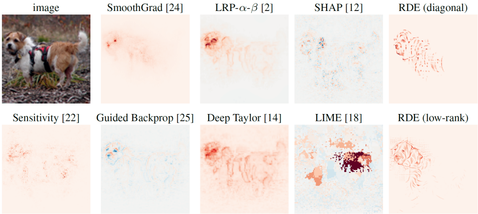
What is Interpretability?

Main Questions: Given a trained deep neural network...

- Which input features contribute most to the decision?
- How can the outcome be explained?

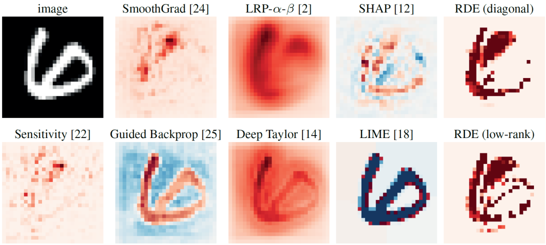
Some Recent Work:

- Sensitivity Analysis (Simonyan, Vedaldi, Zisserman; 2013)
- Layer-wise Relevance Propagation (Bach, Müller, Samek at al.; 2015)
- Deep Taylor Decompositions (Montavon, Samek, Müller; 2018)
- Rate Distortion Explanation (Waeldchen, Macdonald, Hauch, K; 2019)

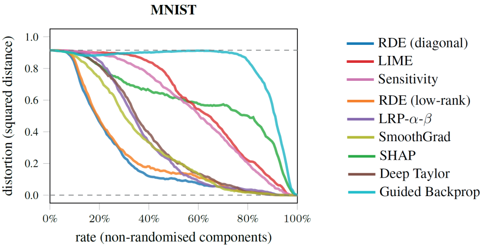


Quality Measure of Interpretability

Classification of the Digit 6:



Quality Measure:



Fundamental Questions concerning Deep Neural Networks

- *Expressivity:*
  - How powerful is the network architecture?
  - Can it indeed represent the correct functions?

$\rightsquigarrow$ *Applied Harmonic Analysis, Approximation Theory, ...*

- *Learning:*
  - Why does the current learning algorithm produce anything reasonable?
  - What are good starting values?

$\rightsquigarrow$ *Differential Geometry, Optimal Control, Optimization, ...*

- *Generalization:*
  - Why do deep neural networks perform that well on data sets, which do not belong to the input-output pairs from a training set?
  - What impact has the depth of the network?

$\rightsquigarrow$ *Learning Theory, Optimization, Statistics, ...*

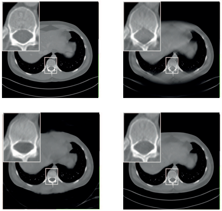
- *Interpretability:*
  - Why did a trained deep neural network reach a certain decision?
  - Which components of the input do contribute most?

$\rightsquigarrow$ *Information Theory, Uncertainty Quantification, ...*

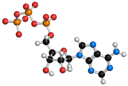
Impact of Deep Learning on Mathematics

Some Examples:

- Inverse Problems
  - ⇝ *Image denoising (Burger, Schuler, Harmeling; 2012)*
  - ⇝ *Superresolution (Klatzer, Soukup, Kobler, Hammernik, Pock; 2017)*
  - ⇝ *Limited-angle tomography (Bubba, K, Lassas, März, Samek, Siltanen, Srinivan; 2018)*
  - ⇝ *Edge detection (Andrade-Loarca, K, Öktem, Petersen; 2019)*



- Numerical Analysis of Partial Differential Equations
  - ⇝ *Schrödinger equation (Rupp, Tkatchenko, Müller, von Lilienfeld; 2012 –)*
  - ⇝ *Black-Scholes PDEs (Grohs, Hornung, Jentzen,von Wurstemberger; 2018)*
  - ⇝ *Parametric PDEs (Schwab, Zech; 2018)*
  - ⇝ *Parametric PDEs (K, Petersen, Raslan, Schneider; 2019)*



- Modelling
  - ⇝ *Learning equations from data (Sahoo, Lampert, Martius; 2018)*

Let's Now Enter the World of Parametric PDEs

Why Parametric PDEs?

Parameter dependent families of PDEs arise in basically any branch of science and engineering.

Some Exemplary Problem Classes:

- Complex design problems
- Inverse problems
- Optimization tasks
- Uncertainty quantification
- ...

The number of parameters can be

- finite (physical properties such as domain geometry, ...)
- infinite (modeling of random stochastic diffusion field, ...)

Parametric Map:

$$\mathcal{Y} \ni y \mapsto u_y \in \mathcal{H} \quad \text{such that} \quad \mathcal{L}(u_y, y) = f_y.$$

Parametric Partial Differential Equations

Our Setting: We will consider parameter-dependent equations of the form

$$b_y\left(u_y, v\right) = f_y(v), \qquad \text{for all } y \in \mathcal{Y},\ v \in \mathcal{H},$$

where

(i) $\mathcal{Y} \subseteq \mathbb{R}^p$ (p large) is the *compact parameter set*,

(ii) $\mathcal{H}$ is a Hilbert space,

(ii) $b_y : \mathcal{H} \times \mathcal{H} \to \mathbb{R}$ is a *symmetric, uniformally coercive, and uniformally continuous bilinear form*,

(iv) $f_y \in \mathcal{H}^*$ is the *uniformly bounded, parameter-dependent right-hand side*,

(v) $u_y \in \mathcal{H}$ is the *solution*.

We also assume the *solution manifold*

$$S(\mathcal{Y}) := \{u_y : y \in \mathcal{Y}\}$$

to be compact in $\mathcal{H}$.

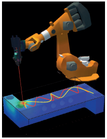
Multi-Query Situation

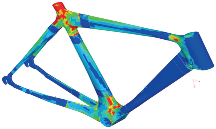
Many applications require solving the parametric PDE multiple times for different parameters:

$$\mathbb{R}^p \supset \mathcal{Y} \ni y = (y_1, \ldots, y_p) \quad \mapsto \quad u_y \in \mathcal{H}$$

Examples:

- Design optimization
- Optimal control
- Routine analysis
- Uncertainty quantification
- Inverse problems





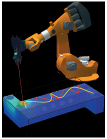
Multi-Query Situation

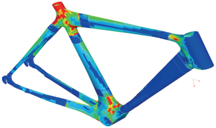
Many applications require solving the parametric PDE multiple times for different parameters:

$$\mathbb{R}^p \supset \mathcal{Y} \ni y = (y_1, \dots, y_p) \quad \mapsto \quad u_y \in \mathcal{H}$$

Examples:

- Design optimization
- Optimal control
- Routine analysis
- Uncertainty quantification
- Inverse problems





Curse of Dimensionality:

Computational cost often much too high!

High-Fidelity Approximations

Galerkin Approach: Instead of $b_y(u_y, v) = f_y(v)$, we solve

$$b_y\left(u_y^h, v\right) = f_y(v) \qquad \text{for all } v \in U^h,$$

where $U^h \subset \mathcal{H}$ with $D := \dim\left(U^h\right) < \infty$ is the *high-fidelity discretization* and $u_y^h \in U^h$ is the solution.

Cea's Lemma: u_y^h is (up to a constant) a best approximation of u_y by elements in U^h.

High-Fidelity Approximations

Galerkin Approach: Instead of $b_y(u_y, v) = f_y(v)$, we solve

$$b_y\left(u_y^h, v\right) = f_y(v) \qquad \text{for all } v \in U^h,$$

where $U^h \subset \mathcal{H}$ with $D := \dim\left(U^h\right) < \infty$ is the *high-fidelity discretization* and $u_y^h \in U^h$ is the solution.

Cea's Lemma: u_y^h is (up to a constant) a best approximation of u_y by elements in U^h.

Galerkin Solution: Let $(\varphi_i)_{i=1}^D$ be a basis for U^h. Then u_y^h satisfies

$$u_y^h = \sum_{i=1}^{D} (\mathbf{u}_y^h)_i \varphi_i \qquad \text{with} \qquad \mathbf{u}_y^h := \left(\mathbf{B}_y^h\right)^{-1} \mathbf{f}_y^h \in \mathbb{R}^D,$$

where $\mathbf{B}_y^h := (b_y(\varphi_j, \varphi_i))_{i,j=1}^D$ and $\mathbf{f}_y^h := (f_y(\varphi_i))_{i=1}^D$.

What about Deep Neural Networks?

Parametric Map:

$$\mathcal{Y} \ni y \mapsto \mathbf{u}_y^{\mathrm{h}} \in \mathbb{R}^D \quad \text{such that} \quad b_y\left(u_y^h, v\right) = f_y(v) \; \forall v \in U^h.$$

Can a Neural Network Approximate the Parametric Map?

What about Deep Neural Networks?

Parametric Map:

$$\mathcal{Y} \ni y \;\mapsto\; \mathbf{u}_y^{\mathrm{h}} \in \mathbb{R}^D \quad \text{such that} \quad b_y\left(u_y^h, v\right) = f_y(v)\; \forall v \in U^h.$$

Can a Neural Network Approximate the Parametric Map?

Advantages:

- After training, extremely rapid computation of the map.
- Flexible, universal approach.

Questions: Let $\varepsilon > 0$.

(1) Does there exist a neural network Φ such that

$$\|\Phi - \mathbf{u}_y^{\mathrm{h}}\| \le \varepsilon \qquad \text{for all } y \in \mathcal{Y}?$$

(2) How does the complexity of Φ depend on p and D?

Deep Learning Approaches to PDEs

Common Approach to Solve PDEs with Neural Networks:
Approximate the solution u of a PDE $\mathcal{L}(u) = f$ by a neural network Φ, i.e., solve

$$\mathcal{L}(\Phi) = f.$$

Key Idea: The size of the neural network does not depend exponentially on the underlying dimension.

Incomplete List:

- Lagaris, Likas, Fotiadis; 1998
- E, Yu; 2017
- Sirignano, Spiliopoulos; 2017
- Han, Jentzen, E; 2017
- Berner, Grohs, Jentzen; 2018
- Eigel, Schneider, Trunschke, Wolf; 2018
- Reisinger, Zhang; 2019
- ...

Solving Parametric PDEs

List of Deep Learning Approaches:

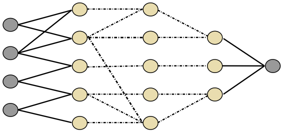
- K. Lee, K. Carlberg; 2018:
  Learn a parametrisation of $S(\mathcal{Y})$ represented by neural networks.
- J.S. Hesthaven, S. Ubbiali; 2018:
  Find reduced basis and then train neural networks to predict coeffcients of solution in that basis.
- Schwab, Zech; 2018:
  Assume that there is a reduced basis of polynomial chaos functions. These and the coefficients can be efficiently represented by neural networks.

Expressivity of Deep Neural Networks

Complexity of a Deep Neural Network

Complexity of a Deep Neural Network

Recall:



- $d \in \mathbb{N}$: Dimension of input layer.
- L: Number of layers.
- N: Number of neurons.
- $\rho : \mathbb{R} \to \mathbb{R}$: (Non-linear) function called *activation function*.
- $T_\ell : \mathbb{R}^{N_{\ell-1}} \to \mathbb{R}^{N_\ell}$, $\ell = 1, \ldots, L$: Affine linear maps $x \mapsto A_\ell x + b_\ell$.

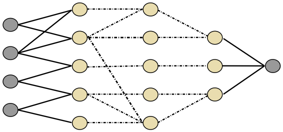
Then $\Phi : \mathbb{R}^d \to \mathbb{R}^{N_L}$ given by

$$\Phi(x) = T_L \rho(T_{L-1} \rho(\ldots \rho(T_1(x))), \quad x \in \mathbb{R}^d,$$

is called *(deep) neural network (DNN)*.

Complexity of a Deep Neural Network

Recall:



- $d \in \mathbb{N}$: Dimension of input layer.
- L: Number of layers.
- N: Number of neurons.
- $\rho : \mathbb{R} \to \mathbb{R}$: (Non-linear) function called *activation function*.
- $T_\ell : \mathbb{R}^{N_{\ell-1}} \to \mathbb{R}^{N_\ell}$, $\ell = 1, \ldots, L$: Affine linear maps $x \mapsto A_\ell x + b_\ell$.

Then $\Phi : \mathbb{R}^d \to \mathbb{R}^{N_L}$ given by

$$\Phi(x) = T_L \rho(T_{L-1} \rho(\ldots \rho(T_1(x))), \quad x \in \mathbb{R}^d,$$

is called *(deep) neural network (DNN)*.

Measure for Complexity: The *number of weights $W(\Phi)$* is defined by

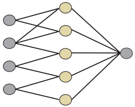
$$W(\Phi) := \sum_{\ell=1}^{L} \left(\|A_\ell\|_0 + \|b_\ell\|_0 \right).$$

We write $\Phi \in \mathcal{NN}_{L,W(\Phi),d,\rho}$.

One Size Fits All?

Universal Approximation Theorem (Cybenko, 1989)(Hornik, 1991):
Let $d \in \mathbb{N}$, $K \subset \mathbb{R}^d$ compact, $f : K \to \mathbb{R}$ continuous, $\rho : \mathbb{R} \to \mathbb{R}$ continuous and not a polynomial. Then, for each $\varepsilon > 0$, there exist $N \in \mathbb{N}$, $a_k, b_k \in \mathbb{R}, w_k \in \mathbb{R}^d$ such that

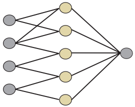
$$\|f - \sum_{k=1}^{N} a_k \rho(\langle w_k, \cdot \rangle - b_k)\|_\infty \le \varepsilon.$$



One Size Fits All?

Universal Approximation Theorem (Cybenko, 1989)(Hornik, 1991):
Let $d \in \mathbb{N}$, $K \subset \mathbb{R}^d$ compact, $f : K \to \mathbb{R}$ continuous, $\rho : \mathbb{R} \to \mathbb{R}$ continuous and not a polynomial. Then, for each $\varepsilon > 0$, there exist $N \in \mathbb{N}$, $a_k, b_k \in \mathbb{R}, w_k \in \mathbb{R}^d$ such that

$$\|f - \sum_{k=1}^{N} a_k \rho(\langle w_k, \cdot \rangle - b_k)\|_\infty \leq \varepsilon.$$

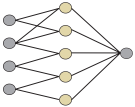


The complexity can be arbitrarily large!

One Size Fits All?

Universal Approximation Theorem (Cybenko, 1989)(Hornik, 1991):
Let $d \in \mathbb{N}$, $K \subset \mathbb{R}^d$ compact, $f : K \to \mathbb{R}$ continuous, $\rho : \mathbb{R} \to \mathbb{R}$ continuous and not a polynomial. Then, for each $\varepsilon > 0$, there exist $N \in \mathbb{N}$, $a_k, b_k \in \mathbb{R}, w_k \in \mathbb{R}^d$ such that

$$\|f - \sum_{k=1}^{N} a_k \rho(\langle w_k, \cdot \rangle - b_k)\|_\infty \leq \varepsilon.$$



The complexity can be arbitrarily large!

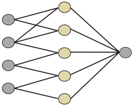
Theorem (Yarotsky; 2017): For all $f \in \mathcal{C} = C^s([0,1]^d)$ and ρ the *ReLU (Rectifiable Linear Unit $\rho(x) = \max\{0, x\}$)*, there exist neural networks $(\Phi_n)_{n \in \mathbb{N}}$ with $L(\Phi_n) \approx \log(n)$ such that

$$\|f - \Phi_n\|_\infty \lesssim W(\Phi_n)^{-\frac{s}{d}} \to 0 \quad \text{as } n \to \infty.$$

One Size Fits All?

Universal Approximation Theorem (Cybenko, 1989)(Hornik, 1991):
Let $d \in \mathbb{N}$, $K \subset \mathbb{R}^d$ compact, $f : K \to \mathbb{R}$ continuous, $\rho : \mathbb{R} \to \mathbb{R}$ continuous and not a polynomial. Then, for each $\varepsilon > 0$, there exist $N \in \mathbb{N}$, $a_k, b_k \in \mathbb{R}, w_k \in \mathbb{R}^d$ such that

$$\|f - \sum_{k=1}^{N} a_k \rho(\langle w_k, \cdot \rangle - b_k)\|_\infty \leq \varepsilon.$$



The complexity can be arbitrarily large!

Theorem (Yarotsky; 2017): For all $f \in \mathcal{C} = C^s([0,1]^d)$ and ρ the *ReLU (Rectifiable Linear Unit $\rho(x) = \max\{0, x\}$)*, there exist neural networks $(\Phi_n)_{n \in \mathbb{N}}$ with $L(\Phi_n) \approx \log(n)$ such that

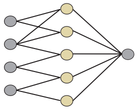
$$\|f - \Phi_n\|_\infty \lesssim W(\Phi_n)^{-\frac{s}{d}} \to 0 \quad \text{as } n \to \infty.$$

This result is not optimal!

One Size Fits All?

Universal Approximation Theorem (Cybenko, 1989)(Hornik, 1991):
Let $d \in \mathbb{N}$, $K \subset \mathbb{R}^d$ compact, $f : K \to \mathbb{R}$ continuous, $\rho : \mathbb{R} \to \mathbb{R}$ continuous and not a polynomial. Then, for each $\varepsilon > 0$, there exist $N \in \mathbb{N}$, $a_k, b_k \in \mathbb{R}, w_k \in \mathbb{R}^d$ such that

$$\|f - \sum_{k=1}^{N} a_k \rho(\langle w_k, \cdot\rangle - b_k)\|_\infty \leq \varepsilon.$$



The complexity can be arbitrarily large!

Theorem (Yarotsky; 2017): For all $f \in \mathcal{C} = C^s([0,1]^d)$ and ρ the *ReLU (Rectifiable Linear Unit $\rho(x) = \max\{0, x\}$)*, there exist neural networks $(\Phi_n)_{n \in \mathbb{N}}$ with $L(\Phi_n) \approx \log(n)$ such that

$$\|f - \Phi_n\|_\infty \lesssim W(\Phi_n)^{-\frac{s}{d}} \to 0 \quad \text{as } n \to \infty.$$

This result is not optimal!

Correct Function Spaces? (Gribonval, K, Nielsen, Voigtlaender; 2019)

A Fundamental Lower Bound

Key Ingredient from Information Theory:

Given $\mathcal{C} \subseteq L^2(\mathbb{R}^d)$. With $E : L^2(\mathbb{R}^d) \to \{0,1\}^\ell$, $D : \{0,1\}^\ell \to L^2(\mathbb{R}^d)$, set

$$L(\varepsilon, \mathcal{C}) := \min\{\ell \in \mathbb{N} : \exists (E, D) \in \mathfrak{E}^\ell \times \mathfrak{D}^\ell : \sup_{f \in \mathcal{C}} \|D(E(f)) - f\|_{L^2(\mathbb{R}^d)} \leq \varepsilon\}.$$

Then the optimal exponent $\gamma^*(\mathcal{C})$ is $\gamma^*(\mathcal{C}) := \inf\{\gamma \in \mathbb{R} : L(\varepsilon, \mathcal{C}) = O(\varepsilon^{-\gamma})\}$.

A Fundamental Lower Bound

Key Ingredient from Information Theory:
Given $\mathcal{C} \subseteq L^2(\mathbb{R}^d)$. With $E : L^2(\mathbb{R}^d) \to \{0,1\}^\ell$, $D : \{0,1\}^\ell \to L^2(\mathbb{R}^d)$, set

$$L(\varepsilon, \mathcal{C}) := \min\{\ell \in \mathbb{N} : \exists (E, D) \in \mathfrak{E}^\ell \times \mathfrak{D}^\ell : \sup_{f \in \mathcal{C}} \|D(E(f)) - f\|_{L^2(\mathbb{R}^d)} \leq \varepsilon\}.$$

Then the optimal exponent $\gamma^*(\mathcal{C})$ is $\gamma^*(\mathcal{C}) := \inf\{\gamma \in \mathbb{R} : L(\varepsilon, \mathcal{C}) = O(\varepsilon^{-\gamma})\}$.

Theorem (Bölcskei, Grohs, K, and Petersen; 2017):
Let $d \in \mathbb{N}$, $\rho : \mathbb{R} \to \mathbb{R}$, and let $\mathcal{C} \subset L^2(\mathbb{R}^d)$. Assume that

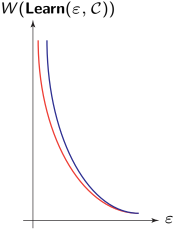
$$\mathbf{Learn} : (0,1) \times \mathcal{C} \to \mathcal{NN}_{\infty,\infty,d,\rho}$$

satisfies that, for each $f \in \mathcal{C}$ and $0 < \varepsilon < 1$

$$\sup_{f \in \mathcal{C}} \|f - \mathbf{Learn}(\varepsilon, f)\|_{L^2(\mathbb{R}^d)} \leq \varepsilon.$$

Then, for all $\gamma < \gamma^*(\mathcal{C})$, there is no $C > 0$ with

$$\sup_{f \in \mathcal{C}} W(\mathbf{Learn}(\varepsilon, f)) \leq C\varepsilon^{-\gamma} \quad \text{for all } \varepsilon > 0$$



A Fundamental Lower Bound

Key Ingredient from Information Theory:
Given $\mathcal{C} \subseteq L^2(\mathbb{R}^d)$. With $E : L^2(\mathbb{R}^d) \to \{0,1\}^\ell$, $D : \{0,1\}^\ell \to L^2(\mathbb{R}^d)$, set

$$L(\varepsilon, \mathcal{C}) := \min\{\ell \in \mathbb{N} : \exists (E,D) \in \mathfrak{E}^\ell \times \mathfrak{D}^\ell : \sup_{f \in \mathcal{C}} \|D(E(f)) - f\|_{L^2(\mathbb{R}^d)} \leq \varepsilon\}.$$

Then the optimal exponent $\gamma^*(\mathcal{C})$ is $\gamma^*(\mathcal{C}) := \inf\{\gamma \in \mathbb{R} : L(\varepsilon, \mathcal{C}) = O(\varepsilon^{-\gamma})\}$.

Theorem (Bölcskei, Grohs, K, and Petersen; 2017):
Let $d \in \mathbb{N}$, $\rho : \mathbb{R} \to \mathbb{R}$, and let $\mathcal{C} \subset L^2(\mathbb{R}^d)$. Assume that

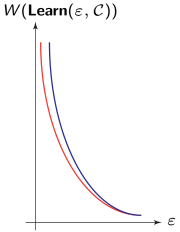
$$\mathbf{Learn} : (0,1) \times \mathcal{C} \to \mathcal{NN}_{\infty,\infty,d,\rho}$$

satisfies that, for each $f \in \mathcal{C}$ and $0 < \varepsilon < 1$

$$\sup_{f \in \mathcal{C}} \|f - \mathbf{Learn}(\varepsilon, f)\|_{L^2(\mathbb{R}^d)} \leq \varepsilon.$$

Then, for all $\gamma < \gamma^*(\mathcal{C})$, there is no $C > 0$ with

$$\sup_{f \in \mathcal{C}} W(\mathbf{Learn}(\varepsilon, f)) \leq C\varepsilon^{-\gamma} \quad \text{for all } \varepsilon > 0$$



What happens for $\gamma = \gamma^*(\mathcal{C})$?

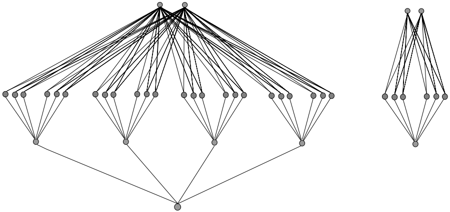
DNNs and Representation Systems, I

Observation: Assume a system $(\varphi_i)_{i \in I} \subset L^2(\mathbb{R}^d)$ satisfies:

- For each $i \in I$, there exists a neural network Φ_i with at most $C > 0$ edges such that $\varphi_i = \Phi_i$.

Then we can construct a network Φ with $O(M)$ edges with

$$\Phi = \sum_{i \in I_M} c_i \varphi_i, \quad \text{if } |I_M| = M.$$



DNNs and Representation Systems, II

Observation: Assume a system $(\varphi_i)_{i\in I} \subset L^2(\mathbb{R}^d)$ satisfies:

- For each $i \in I$, there exists a neural network Φ_i with at most $C > 0$ edges such that $\varphi_i = \Phi_i$.
- There exists $\tilde{C} > 0$ such that, for all $f \in \mathcal{C} \subset L^2(\mathbb{R}^d)$, there exists $I_M \subset I$ with
$$\|f - \sum_{i\in I_M} c_i\varphi_i\| \leq \tilde{C} M^{-1/\gamma^*(\mathcal{C})}.$$

Then every $f \in \mathcal{C}$ can be approximated up to an error of ε by a neural network with only $O(\varepsilon^{-\gamma^*(\mathcal{C})})$ edges.

DNNs and Representation Systems, II

Observation: Assume a system $(\varphi_i)_{i \in I} \subset L^2(\mathbb{R}^d)$ satisfies:

- For each $i \in I$, there exists a neural network Φ_i with at most $C > 0$ edges such that $\varphi_i = \Phi_i$.
- There exists $\tilde{C} > 0$ such that, for all $f \in \mathcal{C} \subset L^2(\mathbb{R}^d)$, there exists $I_M \subset I$ with

$$\|f - \sum_{i \in I_M} c_i \varphi_i\| \leq \tilde{C} M^{-1/\gamma^*(\mathcal{C})}.$$

Then every $f \in \mathcal{C}$ can be approximated up to an error of ε by a neural network with only $O(\varepsilon^{-\gamma^*(\mathcal{C})})$ edges.

Recall: Then, for all $\gamma < \gamma^*(\mathcal{C})$, there is no $C > 0$ with

$$\sup_{f \in \mathcal{C}} W(\textbf{Learn}(\varepsilon, f)) \leq C\varepsilon^{-\gamma} \qquad \text{for all } \varepsilon > 0.$$

Road Map

General Approach:

(1) Determine a class of functions $\mathcal{C} \subseteq L^2(\mathbb{R}^2)$.

(2) Determine an associated representation system with the following properties:

- The elements of this system can be realized by a neural network with controlled number of edges.
- This system provides optimally sparse approximations for $\mathcal{C}$.

Road Map

General Approach:

(1) Determine a class of functions $\mathcal{C} \subseteq L^2(\mathbb{R}^2)$.
$\rightsquigarrow$ *Cartoon-like functions!*

(2) Determine an associated representation system with the following properties:

- The elements of this system can be realized by a neural network with controlled number of edges.
- This system provides optimally sparse approximations for $\mathcal{C}$.

Road Map

General Approach:

(1) Determine a class of functions $\mathcal{C} \subseteq L^2(\mathbb{R}^2)$.
$\rightsquigarrow$ *Cartoon-like functions!*

(2) Determine an associated representation system with the following properties:
$\rightsquigarrow$ *Shearlets!*

- The elements of this system can be realized by a neural network with controlled number of edges.
- This system provides optimally sparse approximations for $\mathcal{C}$.

Road Map

General Approach:

(1) Determine a class of functions $\mathcal{C} \subseteq L^2(\mathbb{R}^2)$.
$\rightsquigarrow$ *Cartoon-like functions!*

(2) Determine an associated representation system with the following properties:
$\rightsquigarrow$ *Shearlets!*

- The elements of this system can be realized by a neural network with controlled number of edges.

- This system provides optimally sparse approximations for $\mathcal{C}$.
$\rightsquigarrow$ *This has been proven!*

Road Map

General Approach:

(1) Determine a class of functions $\mathcal{C} \subseteq L^2(\mathbb{R}^2)$.
$\rightsquigarrow$ *Cartoon-like functions!*

(2) Determine an associated representation system with the following properties:
$\rightsquigarrow$ *Shearlets!*

- The elements of this system can be realized by a neural network with controlled number of edges.
  $\rightsquigarrow$ *Still to be analyzed!*
- This system provides optimally sparse approximations for $\mathcal{C}$.
  $\rightsquigarrow$ *This has been proven!*

Affine Transforms

Building Principle:

Many systems from applied harmonic analysis such as

- wavelets,
- ridgelets,
- shearlets,

constitute affine systems:

$$\{|\det A|^{d/2}\psi(A\,\cdot\,-t) : A \in G \subseteq GL(d),\ t \in \mathbb{Z}^d\}, \quad \psi \in L^2(\mathbb{R}^d).$$

Affine Transforms

Building Principle:

Many systems from applied harmonic analysis such as

- wavelets,
- ridgelets,
- shearlets,

constitute affine systems:

$$\{|\det A|^{d/2}\psi(A\,\cdot\,-t) : A \in G \subseteq GL(d),\ t \in \mathbb{Z}^d\}, \quad \psi \in L^2(\mathbb{R}^d).$$

Realization by Neural Networks:

The following conditions are equivalent:

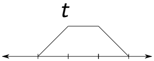
(i) $|\det A|^{d/2}\psi(A\,\cdot\,-t)$ can be realized by a neural network Φ_1.

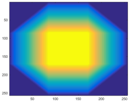
(ii) ψ can be realized by a neural network Φ_2.

Also, Φ_1 and Φ_2 have the same number of edges up to a constant factor.

Construction of Generators

Wavelet generators (LeCun; 1987), (Shaham, Cloninger, Coifman; 2017):

- Assume activation function $\rho(x) = \max\{x, 0\}$ (ReLUs).
- Define
$$t(x) := \rho(x) - \rho(x-1) - \rho(x-2) + \rho(x-3).$$


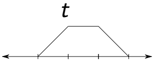
  $\rightsquigarrow$ *t can be constructed with a 2 layer network.*
- Observe that
$$\phi(x_1, x_2) := \rho(t(x_1) + t(x_2) - 1)$$


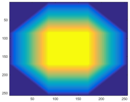
  yields a 2D bump function.
- Summing up shifted versions of ϕ yields a function ψ with vanishing moments.

  $\rightsquigarrow$ *ψ can be realized by a 3 layer neural network.*

Construction of Generators

Wavelet generators (LeCun; 1987), (Shaham, Cloninger, Coifman; 2017):

- Assume activation function $\rho(x) = \max\{x, 0\}$ (ReLUs).
- Define
$$t(x) := \rho(x) - \rho(x-1) - \rho(x-2) + \rho(x-3).$$


  $\rightsquigarrow$ *t can be constructed with a 2 layer network.*
- Observe that
$$\phi(x_1, x_2) := \rho(t(x_1) + t(x_2) - 1)$$


  yields a 2D bump function.
- Summing up shifted versions of ϕ yields a function ψ with vanishing moments.

  $\rightsquigarrow$ *ψ can be realized by a 3 layer neural network.*

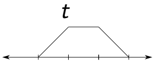
This cannot yield differentiable functions ψ!

Construction of Generators

Wavelet generators (LeCun; 1987), (Shaham, Cloninger, Coifman; 2017):

- Assume activation function $\rho(x) = \max\{x, 0\}$ (ReLUs).
- Define

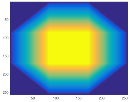
$$t(x) := \rho(x) - \rho(x-1) - \rho(x-2) + \rho(x-3).$$



$\rightsquigarrow$ *t can be constructed with a 2 layer network.*

- Observe that

$$\phi(x_1, x_2) := \rho(t(x_1) + t(x_2) - 1)$$



yields a 2D bump function.

- Summing up shifted versions of ϕ yields a function ψ with vanishing moments.

$\rightsquigarrow$ *ψ can be realized by a 3 layer neural network.*

Our Construction: Use a smoothed version of a ReLU.

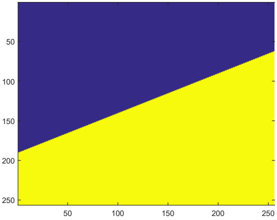
$\rightsquigarrow$ *Leads to appropriate shearlet generators!*

Optimal Approximation

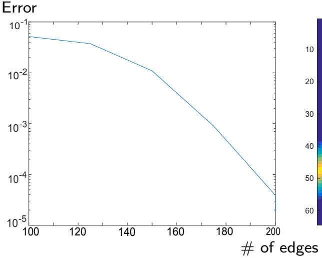
Theorem (Bölcskei, Grohs, K, and Petersen; 2017): Let ρ be an admissible smooth rectifier, and let $\varepsilon > 0$. Then there exist $C_\varepsilon > 0$ such that, for all cartoon-like functions f and $N \in \mathbb{N}$, we can construct a neural network $\Phi \in \mathcal{NN}_{3,O(N),2,\rho}$ satisfying

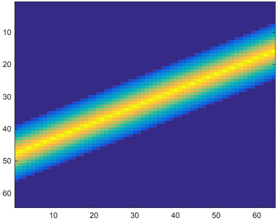
$$\|f - \Phi\|_{L^2(\mathbb{R}^2)} \leq C_\varepsilon N^{-1+\varepsilon}.$$

Function classes which are optimal representable by affine systems are also optimally approximated by sparsely connected neural networks!



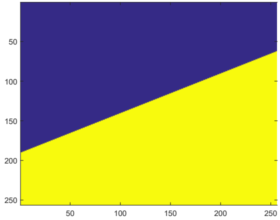
Linear Singularity



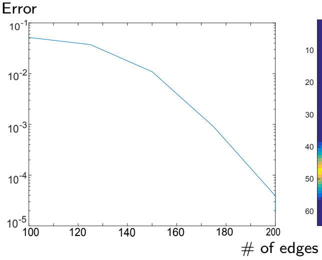


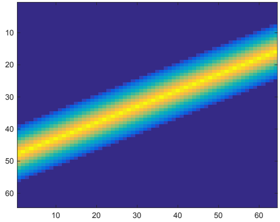
Subnetworks: Ridgelets!

Numerical Experiments (with ReLUs & Backpropagation)

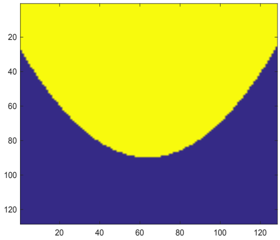


Linear Singularity

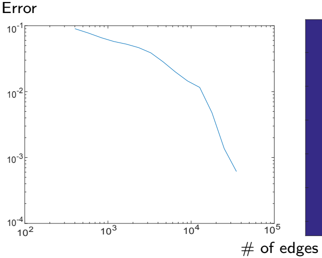


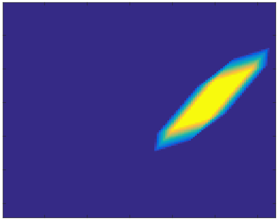


Subnetworks: Ridgelets!



Curvilinear Singularity





Subnetworks: $\approx$ Shearlets!

Deep Learning for Parametric PDEs

or

How to Beat the Curse of Dimensionality

Parametric Partial Differential Equations

Our Setting: We will consider parameter-dependent equations of the form

$$b_y(u_y, v) = f_y(v), \quad \text{for all } y \in \mathcal{Y},\ v \in \mathcal{H},$$

where

(i) $\mathcal{Y} \subseteq \mathbb{R}^p$ (p large) is the *compact parameter set*,

(ii) $\mathcal{H}$ is a Hilbert space,

(ii) $b_y : \mathcal{H} \times \mathcal{H} \to \mathbb{R}$ is a *symmetric, uniformally coercive, and uniformally continuous bilinear form*,

(iv) $f_y \in \mathcal{H}^*$ is the *uniformly bounded, parameter-dependent right-hand side*,

(v) $u_y \in \mathcal{H}$ is the *solution*.

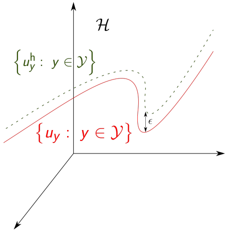
We also assume the *solution manifold*

$$S(\mathcal{Y}) := \{u_y : y \in \mathcal{Y}\}$$

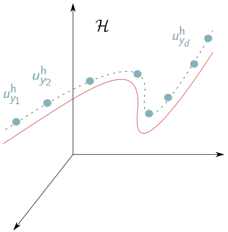
to be compact in $\mathcal{H}$.

Reduced Basis Method: Key Idea

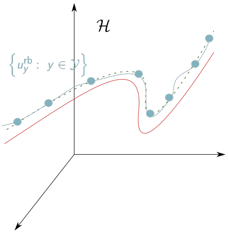
High-Fidelity Discretization:



Key Idea:



Offline (slow):
Compute snap shots



Online (fast):
Compute solutions for new parameters

Reduced Basis Method: Details

Assumption: For all $\varepsilon > \varepsilon_0$, there exists $U^{\mathrm{rb}} \subset \mathcal{H}$, $d(\varepsilon) \coloneqq \dim\left(U^{\mathrm{rb}}\right) \ll D$ such that

$$\sup_{y\in\mathcal{Y}} \inf_{w\in U^{\mathrm{rb}}} \|u_y - w\|_{\mathcal{H}} \leq \varepsilon.$$

$\rightsquigarrow$ Optimality through *Kolmogorov N-width*!

Reduced Basis Method: Details

Assumption: For all $\varepsilon > \varepsilon_0$, there exists $U^{\mathrm{rb}} \subset \mathcal{H}$, $d(\varepsilon) := \dim\left(U^{\mathrm{rb}}\right) \ll D$ such that

$$\sup_{y \in \mathcal{Y}} \inf_{w \in U^{\mathrm{rb}}} \|u_y - w\|_{\mathcal{H}} \leq \varepsilon.$$

$\rightsquigarrow$ Optimality through *Kolmogorov N-width*!

Transfer to Reduced Basis:

- Let $U^{\mathrm{rb}} := \mathrm{span}\,(\psi_i)_{i=1}^{d(\varepsilon)}$ with $(\psi_i)_{i=1}^{d(\varepsilon)} = \left(\sum_{j=1}^{D} \mathbf{V}_{j,i}\varphi_j\right)_{i=1}^{d(\varepsilon)}$.

Reduced Basis Method: Details

Assumption: For all $\varepsilon > \varepsilon_0$, there exists $U^{\mathrm{rb}} \subset \mathcal{H}$, $d(\varepsilon) \coloneqq \dim\left(U^{\mathrm{rb}}\right) \ll D$ such that

$$\sup_{y \in \mathcal{Y}} \inf_{w \in U^{\mathrm{rb}}} \|u_y - w\|_{\mathcal{H}} \leq \varepsilon.$$

$\rightsquigarrow$ Optimality through *Kolmogorov N-width*!

Transfer to Reduced Basis:

- Let $U^{\mathrm{rb}} := \mathrm{span}\,(\psi_i)_{i=1}^{d(\varepsilon)}$ with $(\psi_i)_{i=1}^{d(\varepsilon)} = \left(\sum_{j=1}^{D} \mathbf{V}_{j,i}\varphi_j\right)_{i=1}^{d(\varepsilon)}$.
- Set $\mathbf{B}_y^{\mathrm{rb}} := (b_y(\psi_j, \psi_i))_{i,j=1}^{d(\varepsilon)} = \mathbf{V}^T \mathbf{B}_y^{\mathrm{h}} \mathbf{V} \in \mathbb{R}^{d(\varepsilon) \times d(\varepsilon)}$.
- Set $\mathbf{f}_y^{\mathrm{rb}} := (f_y(\psi_i))_{i=1}^{d(\varepsilon)} = \mathbf{V}^T \mathbf{f}_y^{\mathrm{h}} \in \mathbb{R}^{d(\varepsilon)}$.

Reduced Basis Method: Details

Assumption: For all $\varepsilon > \varepsilon_0$, there exists $U^{\mathrm{rb}} \subset \mathcal{H}$, $d(\varepsilon) \coloneqq \dim\left(U^{\mathrm{rb}}\right) \ll D$ such that

$$\sup_{y \in \mathcal{Y}} \inf_{w \in U^{\mathrm{rb}}} \|u_y - w\|_{\mathcal{H}} \leq \varepsilon.$$

$\rightsquigarrow$ Optimality through *Kolmogorov N-width*!

Transfer to Reduced Basis:

- Let $U^{\mathrm{rb}} := \mathrm{span}\,(\psi_i)_{i=1}^{d(\varepsilon)}$ with $(\psi_i)_{i=1}^{d(\varepsilon)} = \left(\sum_{j=1}^{D} \mathbf{V}_{j,i}\varphi_j\right)_{i=1}^{d(\varepsilon)}$.
- Set $\mathbf{B}_y^{\mathrm{rb}} := (b_y(\psi_j, \psi_i))_{i,j=1}^{d(\varepsilon)} = \mathbf{V}^T \mathbf{B}_y^{\mathrm{h}} \mathbf{V} \in \mathbb{R}^{d(\varepsilon) \times d(\varepsilon)}$.
- Set $\mathbf{f}_y^{\mathrm{rb}} := (f_y(\psi_i))_{i=1}^{d(\varepsilon)} = \mathbf{V}^T \mathbf{f}_y^{\mathrm{h}} \in \mathbb{R}^{d(\varepsilon)}$.

Galerkin Solution: $\left(\sup_{y \in \mathcal{Y}} \|u_y - u_y^{\mathrm{rb}}\|_{\mathcal{H}} \leq C\varepsilon\right)$

$$u_y^{\mathrm{rb}} =$$

Reduced Basis Method: Details

Assumption: For all $\varepsilon > \varepsilon_0$, there exists $U^{\mathrm{rb}} \subset \mathcal{H}$, $d(\varepsilon) := \dim\left(U^{\mathrm{rb}}\right) \ll D$ such that

$$\sup_{y \in \mathcal{Y}} \inf_{w \in U^{\mathrm{rb}}} \|u_y - w\|_{\mathcal{H}} \leq \varepsilon.$$

$\rightsquigarrow$ Optimality through *Kolmogorov N-width*!

Transfer to Reduced Basis:

- Let $U^{\mathrm{rb}} := \operatorname{span}\left(\psi_i\right)_{i=1}^{d(\varepsilon)}$ with $\left(\psi_i\right)_{i=1}^{d(\varepsilon)} = \left(\sum_{j=1}^{D} \mathbf{V}_{j,i}\varphi_j\right)_{i=1}^{d(\varepsilon)}$.
- Set $\mathbf{B}_y^{\mathrm{rb}} := \left(b_y(\psi_j, \psi_i)\right)_{i,j=1}^{d(\varepsilon)} = \mathbf{V}^T \mathbf{B}_y^{\mathrm{h}} \mathbf{V} \in \mathbb{R}^{d(\varepsilon) \times d(\varepsilon)}$.
- Set $\mathbf{f}_y^{\mathrm{rb}} := \left(f_y(\psi_i)\right)_{i=1}^{d(\varepsilon)} = \mathbf{V}^T \mathbf{f}_y^{\mathrm{h}} \in \mathbb{R}^{d(\varepsilon)}$.

Galerkin Solution: $\left(\sup_{y \in \mathcal{Y}} \|u_y - u_y^{\mathrm{rb}}\|_{\mathcal{H}} \leq C\varepsilon\right)$

$$u_y^{\mathrm{rb}} = \sum_{i=1}^{d(\varepsilon)} \left(\mathbf{u}_y^{\mathrm{rb}}\right)_i \psi_i =$$

Reduced Basis Method: Details

Assumption: For all $\varepsilon > \varepsilon_0$, there exists $U^{\mathrm{rb}} \subset \mathcal{H}$, $d(\varepsilon) := \dim\left(U^{\mathrm{rb}}\right) \ll D$ such that

$$\sup_{y \in \mathcal{Y}} \inf_{w \in U^{\mathrm{rb}}} \|u_y - w\|_{\mathcal{H}} \leq \varepsilon.$$

$\rightsquigarrow$ Optimality through *Kolmogorov N-width*!

Transfer to Reduced Basis:

- Let $U^{\mathrm{rb}} := \mathrm{span}\,(\psi_i)_{i=1}^{d(\varepsilon)}$ with $(\psi_i)_{i=1}^{d(\varepsilon)} = \left(\sum_{j=1}^{D} \mathbf{V}_{j,i}\varphi_j\right)_{i=1}^{d(\varepsilon)}$.
- Set $\mathbf{B}_y^{\mathrm{rb}} := (b_y(\psi_j, \psi_i))_{i,j=1}^{d(\varepsilon)} = \mathbf{V}^T \mathbf{B}_y^{\mathrm{h}} \mathbf{V} \in \mathbb{R}^{d(\varepsilon) \times d(\varepsilon)}$.
- Set $\mathbf{f}_y^{\mathrm{rb}} := (f_y(\psi_i))_{i=1}^{d(\varepsilon)} = \mathbf{V}^T \mathbf{f}_y^{\mathrm{h}} \in \mathbb{R}^{d(\varepsilon)}$.

Galerkin Solution: $\left(\sup_{y \in \mathcal{Y}} \|u_y - u_y^{\mathrm{rb}}\|_{\mathcal{H}} \leq C\varepsilon\right)$

$$u_y^{\mathrm{rb}} = \sum_{i=1}^{d(\varepsilon)} \left(\mathbf{u}_y^{\mathrm{rb}}\right)_i \psi_i = \sum_{j=1}^{D} \left(\mathbf{V}\mathbf{u}_y^{\mathrm{rb}}\right)_j \varphi_j =$$

Reduced Basis Method: Details

Assumption: For all $\varepsilon > \varepsilon_0$, there exists $U^{\mathrm{rb}} \subset \mathcal{H}$, $d(\varepsilon) := \dim\left(U^{\mathrm{rb}}\right) \ll D$ such that

$$\sup_{y\in\mathcal{Y}} \inf_{w\in U^{\mathrm{rb}}} \|u_y - w\|_{\mathcal{H}} \leq \varepsilon.$$

$\rightsquigarrow$ Optimality through *Kolmogorov N-width*!

Transfer to Reduced Basis:

- Let $U^{\mathrm{rb}} := \mathrm{span}\,(\psi_i)_{i=1}^{d(\varepsilon)}$ with $(\psi_i)_{i=1}^{d(\varepsilon)} = \left(\sum_{j=1}^{D} \mathbf{V}_{j,i}\varphi_j\right)_{i=1}^{d(\varepsilon)}$.
- Set $\mathbf{B}_y^{\mathrm{rb}} := (b_y(\psi_j, \psi_i))_{i,j=1}^{d(\varepsilon)} = \mathbf{V}^T\mathbf{B}_y^{\mathrm{h}}\mathbf{V} \in \mathbb{R}^{d(\varepsilon)\times d(\varepsilon)}$.
- Set $\mathbf{f}_y^{\mathrm{rb}} := (f_y(\psi_i))_{i=1}^{d(\varepsilon)} = \mathbf{V}^T\mathbf{f}_y^{\mathrm{h}} \in \mathbb{R}^{d(\varepsilon)}$.

Galerkin Solution: $\left(\sup_{y\in\mathcal{Y}} \|u_y - u_y^{\mathrm{rb}}\|_{\mathcal{H}} \leq C\varepsilon\right)$

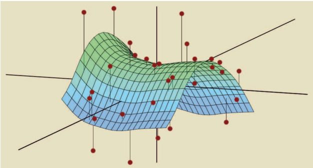
$$u_y^{\mathrm{rb}} = \sum_{i=1}^{d(\varepsilon)} \left(\mathbf{u}_y^{\mathrm{rb}}\right)_i \psi_i = \sum_{j=1}^{D} \left(\mathbf{V}\mathbf{u}_y^{\mathrm{rb}}\right)_j \varphi_j = \sum_{j=1}^{D} \left(\mathbf{V}(\mathbf{B}_y^{\mathrm{rb}})^{-1}\mathbf{V}^T\mathbf{f}_y^{\mathrm{h}}\right)_j \varphi_j.$$

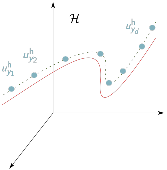
Our Analysis

Statistical Learning Problem = Parametric Problem?

Comparison/Similarities:

Statistical Learning Problem	Parametric Problem
Learn $f : X \to Y$	
Distribution on $X \times Y$	
Loss function $\mathcal{L}\colon Y \times Y \to \mathbb{R}^+$	
Training data $(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^N$	
Training phase $\sum_{i=1}^N \mathcal{L}(f(\mathbf{x}_i), \mathbf{y}_i)$	

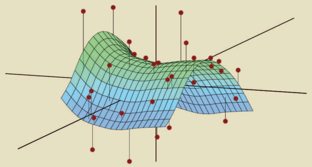


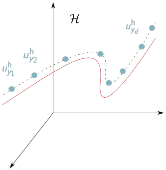


Statistical Learning Problem = Parametric Problem?

Comparison/Similarities:

Statistical Learning Problem	Parametric Problem
Learn $f : X \to Y$	Learn $\mathcal{Y} \ni y \mapsto u_y \in \mathcal{H}$
Distribution on $X \times Y$	
Loss function $\mathcal{L}$: $Y \times Y \to \mathbb{R}^+$	
Training data $(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^N$	
Training phase $\sum_{i=1}^N \mathcal{L}(f(\mathbf{x}_i), \mathbf{y}_i)$	

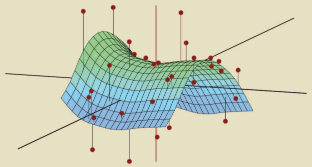


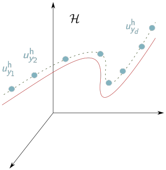


Statistical Learning Problem = Parametric Problem?

Comparison/Similarities:

Statistical Learning Problem	Parametric Problem
Learn $f : X \to Y$	Learn $\mathcal{Y} \ni y \mapsto u_y \in \mathcal{H}$
Distribution on $X \times Y$	PDE
Loss function $\mathcal{L}\colon Y \times Y \to \mathbb{R}^+$	
Training data $(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^N$	
Training phase $\sum_{i=1}^N \mathcal{L}(f(\mathbf{x}_i), \mathbf{y}_i)$	

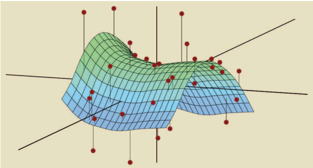


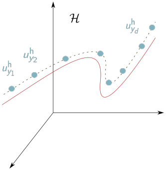


Statistical Learning Problem = Parametric Problem?

Comparison/Similarities:

Statistical Learning Problem	Parametric Problem
Learn $f : X \to Y$	Learn $\mathcal{Y} \ni y \mapsto u_y \in \mathcal{H}$
Distribution on $X \times Y$	PDE
Loss function $\mathcal{L}$: $Y \times Y \to \mathbb{R}^+$	Metric on state space
Training data $(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^N$	
Training phase $\sum_{i=1}^N \mathcal{L}(f(\mathbf{x}_i), \mathbf{y}_i)$	

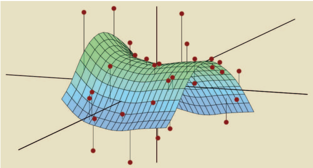


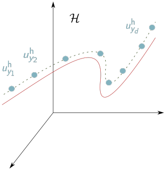


Statistical Learning Problem = Parametric Problem?

Comparison/Similarities:

Statistical Learning Problem	Parametric Problem
Learn $f: X \to Y$	Learn $\mathcal{Y} \ni y \mapsto u_y \in \mathcal{H}$
Distribution on $X \times Y$	PDE
Loss function $\mathcal{L}: Y \times Y \to \mathbb{R}^+$	Metric on state space
Training data $(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^N$	Snapshots
Training phase $\sum_{i=1}^N \mathcal{L}(f(\mathbf{x}_i), \mathbf{y}_i)$	

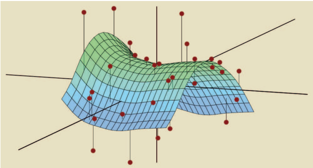


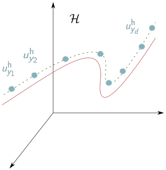


Statistical Learning Problem = Parametric Problem?

Comparison/Similarities:

Statistical Learning Problem	Parametric Problem
Learn $f : X \to Y$	Learn $\mathcal{Y} \ni y \mapsto u_y \in \mathcal{H}$
Distribution on $X \times Y$	PDE
Loss function $\mathcal{L}$: $Y \times Y \to \mathbb{R}^+$	Metric on state space
Training data $(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^N$	Snapshots
Training phase $\sum_{i=1}^N \mathcal{L}(f(\mathbf{x}_i), \mathbf{y}_i)$	Offline phase





Our Results: Discrete Version

Theorem (K, Petersen, Raslan, Schneider; 2019):
We assume the following:

- For all $\varepsilon > 0$, there exists $d(\varepsilon) \ll D$, $\mathbf{V} \in \mathbb{R}^{D \times d(\varepsilon)}$, such that for all $y \in \mathcal{Y}$ there exists $\mathbf{B}_y^{\mathrm{rb}} \in \mathbb{R}^{d(\varepsilon) \times d(\varepsilon)}$ with
$$\|\mathbf{V}(\mathbf{B}_y^{\mathrm{rb}})^{-1} V^T \mathbf{f}_y^{\mathrm{h}} - \mathbf{u}_y^{\mathrm{h}}\| \leq \varepsilon.$$
- There exist ReLU neural networks Φ^B and Φ^f of size $O(\mathrm{poly}(p) d(\varepsilon)^2 \mathrm{polylog}(\varepsilon))$ such that, for all $y \in \mathcal{Y}$,
$$\|\Phi^B - \mathbf{B}_y^{\mathrm{rb}}\| \leq \varepsilon \quad \text{and} \quad \|\Phi^f - V^T \mathbf{f}_y^{\mathrm{h}}\| \leq \varepsilon.$$

Then there exists a ReLU neural network Φ of size $O(d(\varepsilon)^3 \mathrm{polylog}(\varepsilon) + D + \mathrm{poly}(p) d(\varepsilon)^2 \mathrm{polylog}(\varepsilon))$ such that
$$\|\Phi - \mathbf{u}_y^{\mathrm{h}}\| \leq \varepsilon \qquad \text{for all } y \in \mathcal{Y}.$$

Our Results: Discrete Version

Theorem (K, Petersen, Raslan, Schneider; 2019):

We assume the following:

- For all $\varepsilon > 0$, there exists $d(\varepsilon) \ll D$, $\mathbf{V} \in \mathbb{R}^{D \times d(\varepsilon)}$, such that for all $y \in \mathcal{Y}$ there exists $\mathbf{B}_y^{\mathrm{rb}} \in \mathbb{R}^{d(\varepsilon) \times d(\varepsilon)}$ with
$$\|\mathbf{V}(\mathbf{B}_y^{\mathrm{rb}})^{-1} V^T \mathbf{f}_y^{\mathrm{h}} - \mathbf{u}_y^{\mathrm{h}}\| \leq \varepsilon.$$
- There exist ReLU neural networks Φ^B and Φ^f of size $O(\mathrm{poly}(p) d(\varepsilon)^2 \mathrm{polylog}(\varepsilon))$ such that, for all $y \in \mathcal{Y}$,
$$\|\Phi^B - \mathbf{B}_y^{\mathrm{rb}}\| \leq \varepsilon \quad \text{and} \quad \|\Phi^f - V^T \mathbf{f}_y^{\mathrm{h}}\| \leq \varepsilon.$$

Then there exists a ReLU neural network Φ of size $O(d(\varepsilon)^3 \mathrm{polylog}(\varepsilon) + D + \mathrm{poly}(p) d(\varepsilon)^2 \mathrm{polylog}(\varepsilon))$ such that
$$\|\Phi - \mathbf{u}_y^{\mathrm{h}}\| \leq \varepsilon \qquad \text{for all } y \in \mathcal{Y}.$$

Extremely fast computation of the parametric map,
while beating the curse of dimensionality!

Our Results: Continuous Version

Theorem (K, Petersen, Raslan, Schneider; 2019):

Let $(\psi_i)_{i=1}^{d(\varepsilon)}$ denote the reduced basis. We assume in addition the following:

- There exist ReLU neural networks $(\Phi_i)_{i=1}^{d(\varepsilon)}$ of size $O(\text{polylog}(\varepsilon))$ such that $\|\Phi_i - \psi_i\|_{\mathcal{H}} \leq \varepsilon$ for all $i = 1, \ldots, d(\varepsilon)$.

Then there exists a ReLU neural network Φ of size $O(d(\varepsilon)^3 \text{polylog}(\varepsilon) + \text{poly}(p) d(\varepsilon)^2 \text{polylog}(\varepsilon))$ such that

$$\|\Phi - u_y\|_{\mathcal{H}} \leq \varepsilon \qquad \text{for all } y \in \mathcal{Y}.$$

Our Results: Continuous Version

Theorem (K, Petersen, Raslan, Schneider; 2019):
Let $(\psi_i)_{i=1}^{d(\varepsilon)}$ denote the reduced basis. We assume in addition the following:

- There exist ReLU neural networks $(\Phi_i)_{i=1}^{d(\varepsilon)}$ of size $O(\text{polylog}(\varepsilon))$ such that $\|\Phi_i - \psi_i\|_{\mathcal{H}} \leq \varepsilon$ for all $i = 1, \ldots, d(\varepsilon)$.

Then there exists a ReLU neural network Φ of size $O(d(\varepsilon)^3\text{polylog}(\varepsilon) + \text{poly}(p)d(\varepsilon)^2\text{polylog}(\varepsilon))$ such that

$$\|\Phi - u_y\|_{\mathcal{H}} \leq \varepsilon \qquad \text{for all } y \in \mathcal{Y}.$$

Remark: The hypotheses are fulfilled, for example, by

- Diffusion equations,
- Linear elasticity equations.

Key Idea of the Proof

Main Task: Approximate $\mathbf{V}(\mathbf{B}_y^{\mathrm{rb}})^{-1}\mathbf{V}^T\mathbf{f}_y^{\mathrm{h}}$ by a ReLU neural network and control its size!

Key Idea of the Proof

Main Task: Approximate $\mathbf{V}(\mathbf{B}_y^{\mathrm{rb}})^{-1}\mathbf{V}^T\mathbf{f}_y^{\mathrm{h}}$ by a ReLU neural network and control its size!

Step 1 (Scalar Multiplication from Yarotsky; 2017):
For $g(x) := \min\{2x, 2-2x\}$ and $g_s := g \circ \ldots \circ g$ (s times), we have

$$x^2 = \lim_{n\to\infty} x - \sum_{s=1}^{n} \frac{g_s(x)}{2^{2s}} \quad \text{for all } x \in [0,1].$$

Key Idea of the Proof

Main Task: Approximate $\mathbf{V}(\mathbf{B}_y^{\mathrm{rb}})^{-1}\mathbf{V}^T\mathbf{f}_y^{\mathrm{h}}$ by a ReLU neural network and control its size!

Step 1 (Scalar Multiplication from Yarotsky; 2017):
For $g(x) := \min\{2x, 2-2x\}$ and $g_s := g \circ \ldots \circ g$ (s times), we have

$$x^2 = \lim_{n\to\infty} x - \sum_{s=1}^{n} \frac{g_s(x)}{2^{2s}} \quad \text{for all } x \in [0,1].$$

Also, g can be represented by a neural network due to

$$g(x) = 2\rho(x) - 4\rho(x - \frac{1}{2}) + 2\rho(x-2) \quad \text{for all } x \in [0,1].$$

Key Idea of the Proof

Main Task: Approximate $\mathbf{V}(\mathbf{B}_y^{\mathrm{rb}})^{-1}\mathbf{V}^T\mathbf{f}_y^{\mathrm{h}}$ by a ReLU neural network and control its size!

Step 1 (Scalar Multiplication from Yarotsky; 2017):
For $g(x) := \min\{2x, 2-2x\}$ and $g_s := g \circ \ldots \circ g$ (s times), we have

$$x^2 = \lim_{n\to\infty} x - \sum_{s=1}^{n} \frac{g_s(x)}{2^{2s}} \quad \text{for all } x \in [0,1].$$

Also, g can be represented by a neural network due to

$$g(x) = 2\rho(x) - 4\rho(x - \frac{1}{2}) + 2\rho(x-2) \quad \text{for all } x \in [0,1].$$

Moreover,

$$xz = 1/4((x+z)^2 - (x-z)^2) \quad \text{for all } x, z \in \mathbb{R}.$$

$\Longrightarrow$ *Scalar multiplication on $[-1,1]^2$ can be ε-approximated by a neural network of size $\mathcal{O}(\log_2(1/\varepsilon))$.*

Key Idea of the Proof

Step 2 (Multiplication):

A matrix multiplication of two matrices of size $d \times d$ can be performed by d^3 scalar multiplications.

$\Longrightarrow$ *Matrix multiplication can be ε-approximated by a neural network of size $\mathcal{O}(d(\varepsilon)^3 \log_2(1/\varepsilon))$.*

Key Idea of the Proof

Step 2 (Multiplication):

A matrix multiplication of two matrices of size $d \times d$ can be performed by d^3 scalar multiplications.

$\Longrightarrow$ *Matrix multiplication can be ε-approximated by a neural network of size $\mathcal{O}(d(\varepsilon)^3 \log_2(1/\varepsilon))$.*

Step 3 (Inversion):

- Neural networks can approximate matrix polynomials.
- Neural networks can the inversion operator $\mathbf{A} \mapsto \mathbf{A}^{-1}$ using

$$\sum_{s=0}^{m} \mathbf{A}^s \longrightarrow (\mathbf{Id}_{\mathbb{R}^d} - \mathbf{A})^{-1} \quad \text{as } m \to \infty.$$

$\Longrightarrow$ *Matrix inversion can be ε-approximated by a neural network of size $\mathcal{O}(d(\varepsilon)^3 \log_2^q(1/\varepsilon))$ for a constant $q > 0$.*

Key Idea of the Proof

Step 4 (Discrete Parametric Map w.r.t Reduced Basis):

- Now use the assumptions on $\mathbf{B}_y^{\mathrm{rb}}$ and $\mathbf{f}_y^{\mathrm{rb}}$.

$\Longrightarrow$ *The map* $y \mapsto (\mathbf{B}_y^{\mathrm{rb}})^{-1}\mathbf{f}_y^{\mathrm{rb}}$ *can be* ε*-approximated by a neural network* Φ^{rb} *of size* $\mathcal{O}(d(\varepsilon)^3 \log_2^q(1/\varepsilon) + poly(p) d(\varepsilon)^2 \log_2^q(1/\varepsilon))$.

Key Idea of the Proof

Step 4 (Discrete Parametric Map w.r.t Reduced Basis):

- Now use the assumptions on $\mathbf{B}_y^{\mathrm{rb}}$ and $\mathbf{f}_y^{\mathrm{rb}}$.

$\Longrightarrow$ *The map $y \mapsto (\mathbf{B}_y^{\mathrm{rb}})^{-1}\mathbf{f}_y^{\mathrm{rb}}$ can be ε-approximated by a neural network Φ^{rb} of size $\mathcal{O}(d(\varepsilon)^3 \log_2^q(1/\varepsilon) + poly(p)d(\varepsilon)^2 \log_2^q(1/\varepsilon))$.*

For Theorem 1:

- Now use the assumption that every element from the reduced basis can be approximately represented in the high-fidelity basis.
- Consider then $\mathbf{V} \circ \Phi^{\mathrm{rb}}$.

$\Longrightarrow$ *The discrete parametric map can be ε-approximated by a neural network of size $\mathcal{O}(d(\varepsilon)^3 \log_2^q(1/\varepsilon) + d(\varepsilon)D + poly(p)d(\varepsilon)^2 \log_2^q(1/\varepsilon))$.*

Key Idea of the Proof

Step 4 (Discrete Parametric Map w.r.t Reduced Basis):

- Now use the assumptions on $\mathbf{B}_y^{\mathrm{rb}}$ and $\mathbf{f}_y^{\mathrm{rb}}$.

$\Longrightarrow$ *The map $y \mapsto (\mathbf{B}_y^{\mathrm{rb}})^{-1}\mathbf{f}_y^{\mathrm{rb}}$ can be ε-approximated by a neural network Φ^{rb} of size $\mathcal{O}(d(\varepsilon)^3 \log_2^q(1/\varepsilon) + poly(p)d(\varepsilon)^2 \log_2^q(1/\varepsilon))$.*

For Theorem 1:

- Now use the assumption that every element from the reduced basis can be approximately represented in the high-fidelity basis.
- Consider then $\mathbf{V} \circ \Phi^{\mathrm{rb}}$.

$\Longrightarrow$ *The discrete parametric map can be ε-approximated by a neural network of size $\mathcal{O}(d(\varepsilon)^3 \log_2^q(1/\varepsilon) + d(\varepsilon)D + poly(p)d(\varepsilon)^2 \log_2^q(1/\varepsilon))$.*

For Theorem 2:

- Now use the assumption that neural networks can approximate each element of the reduced basis.

$\Longrightarrow$ *The continuous parametric map can be ε-approximated by a neural network of size $\mathcal{O}(d(\varepsilon)^3 \log_2^q(1/\varepsilon) + poly(p)d(\varepsilon)^2 \log_2^q(1/\varepsilon))$.*

Conclusions

What to take Home...?

Deep Learning:

- Impressive performance in combination with classical model-based methods (Inverse Problems, PDEs, ...) $\rightsquigarrow$ Limited-Angle CT.
- Theoretical foundation of neural networks almost entirely missing: Expressivity, Learning, Generalization, and Explainability.

Expressivity of Deep Neural Networks:

- We derive a fundamental lower bound on the complexity, which each learning algorithm has to obey.
- Neural networks are as powerful approximators as classical affine systems such as wavelets, shearlets, ...

Deep Neural Networks for Parametric PDEs:

- We theoretically show that in this setting neural networks beat the curse of dimensionality by explicably constructing such networks.
- Once the network is trained, the parametric map can be computed extremely fast.

THANK YOU!

References available at:

www.math.tu-berlin.de/∼kutyniok

Code available at:

www.ShearLab.org

Related Books:

- Y. Eldar and G. Kutyniok
  Compressed Sensing: Theory and Applications
  Cambridge University Press, 2012.
- G. Kutyniok and D. Labate
  Shearlets: Multiscale Analysis for Multivariate Data
  Birkhäuser-Springer, 2012.
- P. Grohs and G. Kutyniok
  Theory of Deep Learning
  Cambridge University Press (in preparation)