Technical University Berlin

Lecture Notes

Mathematics of Deep Learning

read by Prof. Dr. Gitta Kutyniok in the winter term 2019/2020

Contents

Lecture Notes by Viktor Glombik.

Edited on August 22, 2020.

List of Figures

1 Introduction to Neural Networks and Deep Learning

1.1 Introduction to this lecture

After briefly touching on the basics of statistical learning theory we will cover the four main aspects of the mathematical theory of deep learning: expressivity, optimization, generalization and interpretability. Furthermore we will see deep learning in mathematical approaches such as inverse problems and numerical analysis of partial differential equations.

The goal of this course will be to learn the possibilities and limitations of deep learning in order to lay the foundation for delving further into this topic, both theoretically and practically.

Deep neural networks have recently shown impressive results in a variety of real-world applications such as self-driving cars, surveillance, legal issues, health care, speech recognition and many more.

They have already outperformed various mathematical based approaches which were fundamental in inverse problems and imaging science (denoising, edge detection, inpainting, classification, superresolution and limited-Angle Computed Tomography), numerical analysis of the Black-Scholes, Allen-Cahn and parametric PDEs and modelling. Therefore, neural networks and standard mathematical techniques have been combined (cf. [3]) by using model-based methods as far as possible and data-driven methods where necessary.

Unfortunately very few theoretical result explain the success of deep neural networks (cf. [15]): "Ali Rahimi, a researcher in artificial intelligence at Google [...], took a swipe at his field last December and received a 40-second ovation for it. [...] at an AI conference, Rahimi charged that machine learning algorithms [...] have become a form of «alchemy». Researchers, he said, do not know why some algorithms work and others don't, nor do they have rigorous criteria for choosing one AI architecture over another". [4]

For brevity' sake we will write $[n] := \{1, \dots, n\}$ for $n \in \mathbb{N}$.

1.2 Deep Neural Networks

Artificial Neurons & Neural Networks

The idea behind artificial neural networks is to mimic the functionality of the human brain and can be traced back to McCulloch and Pitts in 1943, who were interested in developing an algorithmic approach to learning.

definition 1.2.1 (Artificial neuron)
An artificial neuron with weights $w \in \mathbb{R}^d$, bias $b \in \mathbb{R}$ and activation function $\varrho : \mathbb{R} \to \mathbb{R}$ is a function

$$\mathbb{R}^n \to \mathbb{R},\ x \mapsto \varrho(\langle x, w \rangle - b).$$

artificial neuron
activation function

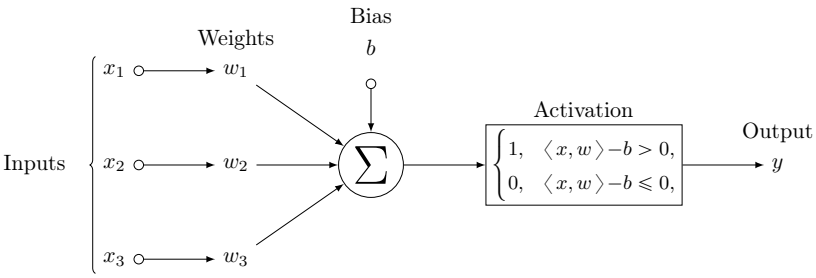


Fig. 1: An artificial neuron.

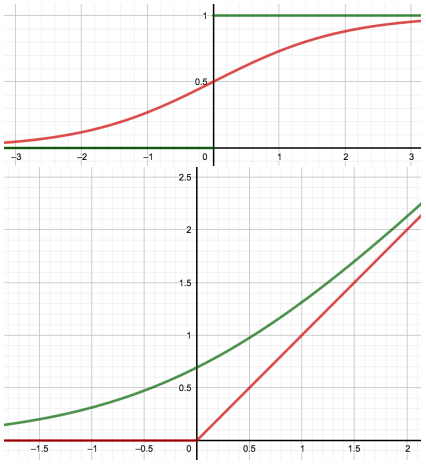


Fig. 2: Activation functions. [1]

Example 1.2.2 (Activation functions)

- Heaviside function $\mathbb{1}_{[0,\infty)}$,
- Sigmoid function $x \mapsto (1 + e^{-x})^{-1}$,
- Rectifiable Linear Unit (ReLU) $x \mapsto \max(0, x)$,
- Softmax function $x \mapsto \ln(1 + e^x)$. ◇

definition 1.2.3 (Artificial Neural Network Types)
An artificial neural network (ANN) is a graph consisting of artificial neurons called feed-forward ANN if it is direct and acyclic and recurrent ANN else.

artificial neural network

Deep Neural Networks

The start of the Age of Data at the turn of the century saw a drastic improvement of computing power and vast amounts of data being available. This gave rise to deep neural networks (cf. Y. LeCun et al., ca 2000).

definition 1.2.4 ((Deep) Neural Network (DNN))
The map

$$\mathbb{R}^d \to \mathbb{R}^{N_L},\ x \mapsto T_L \varrho(T_{L-1} \varrho(\dots \varrho(T_1(x))))$$

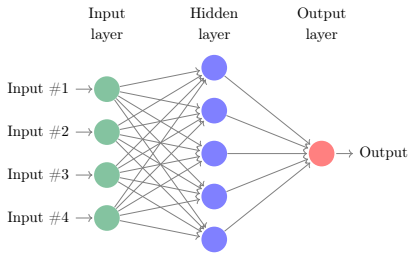


Fig. 3: An artificial neural network.

is called DNN, where $d := N_0 \in \mathbb{N}$ is the dimension of input layer, L the number of layers, N_ℓ the number of neurons in layer $\ell \in [L]$, $\varrho : \mathbb{R} \to \mathbb{R}$ the activation function and $(T_\ell : \mathbb{R}^{N_{\ell-1}} \to \mathbb{R}^{N_\ell})_{\ell=1}^L$ are affine linear maps.

Example 1.2.5 (DNN Architecture: Affine linear maps)
The affine linear maps T_ℓ are defined by a matrix $A_\ell \in \mathbb{R}^{N_{\ell-1} \times N_\ell}$ and an affine part $b_\ell \in \mathbb{R}^{N_\ell}$ via $T_\ell(x) := A_\ell x + b_\ell$. ◇

In the example on the right

$$A_1 := \begin{pmatrix} a_1^1 & a_2^1 & 0 \\ 0 & 0 & a_3^1 \\ 0 & 0 & a_4^1 \end{pmatrix} \quad \text{and} \quad A_2 := \begin{pmatrix} a_1^2 & a_2^2 & 0 \\ 0 & 0 & a_3^2 \end{pmatrix}.$$

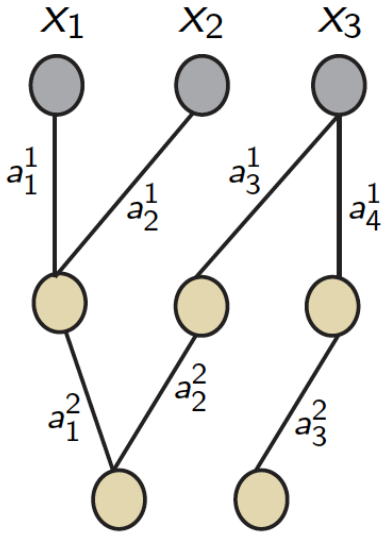


Fig. 4: Neural network architecture. [2]

Example 1.2.6 (Training of neural networks)
Let $\mathcal{M}$ be a manifold and $(x_i, f(x_i))_{i=1}^m$ samples of a function $f : \mathcal{M} \to [K]$, aiming to classify an input $x \in \mathcal{M}$ into one of K classes.

We now select d, L, $(N_\ell)_{\ell=1}^L$ and ϱ, collectively forming the architecture of a DNN. Now we learn the affine linear functions $(T_\ell)_{\ell=1}^L$ by

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

where $\mathcal{L}$ is a loss function, $\mathcal{R}$ a regularization term and $\lambda > 0$ (cf. 1.2.9) yielding the network

$$\Phi_{(A_\ell,b_\ell)_\ell} : \mathbb{R}^d \to \mathbb{R}^{N_L}, \; x \mapsto T_L \varrho(T_{L-1}\varrho(\dots \varrho(T_1(x))))$$

This is often solved by stochastic gradient descent (cf. subsection 1.3), in pursuit of

$$\Phi_{(A_\ell,b_\ell)_\ell} \approx f.$$ ◇

Four Fundamental Questions concerning DNNs

- Expressivity (⇝ Applied Harmonic Analysis, Approximation Theory) — Expressivity
  - How powerful is the network architecture?
  - Can it represent the correct functions?
- Learning (⇝ Differential Geometry, Optimal Control, Optimization) — Learning
  - Why does the algorithm yield reasonable results?
  - What are sensible starting values?
- Generalization (⇝ Learning Theory, Optimization, Statistics) — Generalization
  - Why do DNNs preform well on unseen data?
  - What impact does the depth of the network have?
- Interpretability (⇝ Information Theory, Uncertainty Quantification) — Interpretability

– Why did a trained DNN reach a certain decision?

– Which of the inputs components contributes most?

Mathematical Learning Theory

definition 1.2.7 (Learning (T. Mitchell, 1997))
A computer program learns from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

While this definition lacks mathematical precision, we can give some examples for T, E and P:

Example 1.2.8 (Task, experience, performance measure)
A task could be

- classification: compute $f : \mathbb{R}^n \to [k]$, mapping a data point to its class (i.e. hand-written digit recognition), — classification
- regression: compute $f : \mathbb{R}^n \to \mathbb{R}$ prediction numerical value (i.e. future prices), — regression
- density estimation: learn a probability density $p : \mathbb{R} \to \mathbb{R}^+$ on the test data space to i.e. find corrupted data or detect anomalies.

The experience is typically given by the data set. We mainly distinguish into two types of learning:

- supervised: labelled data ($\rightsquigarrow$ classification). — supervised
- unsupervised: Data is unlabelled and the algorithm is tasked with find a structure in the data ($\rightsquigarrow$ clustering). Think of a classification task, in which you do not know the classes the data points in the (test) data set belong to. — unsupervised

The usual performance measure is accuracy: the proportion of data points for which the model (function) outputs the correct value.

This can be evaluated by cross-validation: the data is split into subsets called cross-validation folds and then trained on one and tested on the other. ◇ — cross-validation

Example 1.2.9 (Linear Regression)
We want to predict the function $f : \mathbb{R}^n \to \mathbb{R}$ (task) and split our data set (experience) into a training set $\mathbf{z} := ((x_i^{\text{train}}, y_i^{\text{train}})_{i=1}^m \subset \mathbb{R}^n \times \mathbb{R}$ and the test set $((x_i^{\text{test}}, y_i^{\text{test}})_{i=1}^m \subset \mathbb{R}^n \times \mathbb{R}$.

For the learning algorithm we define the hypothesis space

$$\mathcal{H} := \operatorname{span}((\varphi)_{i=1}^{\ell}) \subset \mathcal{C}(\mathbb{R}^n)$$

and the empirical error/risk by the mean squared error (MSE) — mean squared error

$$\mathcal{E}_{\mathbf{z}}(f) := \frac{1}{m} \sum_{i=1}^{m} \left(\hat{f}(x_i^{\text{train}}) - y_i^{\text{train}} \right)^2 .$$

and then find the empirical target function

$$f_{\mathcal{H},\mathbf{z}} := \arg\min_{f\in\mathcal{H}} \mathcal{E}_{\mathbf{z}}(f). \tag{1}$$

Every $f \in \mathcal{H}$ can be expressed as $f = \sum_{i=1}^{\ell} w_i\varphi_i$. For sake of brevity we set $\mathbf{y} := (y_i^{\text{train}})_{i=1}^m$, $\mathbf{w} := (w_i)_{i=1}^\ell$ and $\mathbf{A} := (\varphi_j(x_i^{\text{train}}))_{i,j} \in \mathbb{R}^{m\times\ell}$ so we can write $\mathcal{E}_{\mathbf{z}}(f) = \|\mathbf{A}\mathbf{w} - \mathbf{y}\|^2$.

Letting $\hat{\mathbf{w}} := \arg\min_{\mathbf{w}\in\mathbb{R}^\ell} \mathcal{E}_{\mathbf{z}}(f)$ being the minimiser, the sought estimate (= solution of the regression task) is given by

$$\hat{f} := \sum_{i=1}^{\ell} (\hat{\mathbf{w}})_i\varphi_i. \qquad \diamond$$

The general statistical learning problem

We can now state a mathematically precise definition of learning.

definition 1.2.10 (Statistical Learning Problem)
Let (Ω, Σ, σ) be a probability space, $X : \Omega \to \mathbb{R}^n$ and $Y : \Omega \to \mathbb{R}^k$ be random vectors and $\mathcal{X}$ and $\mathcal{Y}$ be the images of X and Y, where we assume that $\mathcal{X} \subset \mathbb{R}^n$ is compact. For a loss function $\ell : \mathbb{R}^k \times \mathbb{R}^k \to [0,\infty]$ and a measurable $f : \mathcal{X} \to \mathcal{Y}$ define the error

$$\mathcal{E}(f) := \mathbb{E}\left[\ell(f(X), y)\right] = \int_\Omega \ell(f(X(\omega), Y(\omega))\,\mathrm{d}\sigma(\omega).$$

The learning problem is to find $f_{\mathcal{H}} := \arg\min_{f:\mathcal{X}\to\mathcal{Y}} \mathcal{E}(f)$. learning problem

Remark In regression we have $k = 1$. A typical loss function is the squared error $\ell(x, y) := (x - y)^2$.

definition 1.2.12 (Notation: Probability measures)
For $x \in \mathcal{X}$ let $\sigma(y \mid x)$ be the conditional probability measure on $\mathcal{Y}$ (with respect to x) and $\sigma_{\mathcal{X}}$ the marginal probability measure on $\mathcal{X}$.

We have $\sigma_{\mathcal{X}}(S) = \sigma(\pi_{\mathcal{X}}^{-1}(S))$, where $\pi_{\mathcal{X}} : \mathcal{X}\times\mathcal{Y} \to \mathcal{X}$, $(x, y) \mapsto x$. Then for every integrable function $\varphi : Z \to \mathbb{R}$

$$\int_{\mathcal{X}\times\mathcal{Y}} \varphi(x, y)\,\mathrm{d}\sigma(x, y) = \int_{\mathcal{X}} \left(\int_{\mathcal{Y}} \varphi(x, y)\,\mathrm{d}\sigma(y \mid x)\right) \mathrm{d}\sigma_{\mathcal{X}}(x)$$

holds. **WHAT IS Z???**

definition 1.2.13 (Regression function)
We define the regression function by

$$f_\sigma : \mathcal{X} \to \mathcal{Y},\ x \mapsto \int_{\mathcal{Y}} y \, \mathrm{d}\sigma(y \mid x).$$

Theorem 1.2.1: Regression

The regression function f_σ is a minimiser of $\mathcal{E}$ over $L^2(\mathcal{X}, \sigma_\mathcal{X})$:

$$\mathcal{E}(f) = \|f - f_\sigma\|^2_{L^2(\mathcal{X},\sigma_\mathcal{X})} + \mathcal{E}(f_\sigma) \quad \forall f \in L^2(\mathcal{X}, \sigma_X)$$

As the distribution σ is unknown, the target function can't be computed.

definition 1.2.14 (Hypothesis space, target function)
A subspace $\mathcal{H} \subset \mathcal{C}(\mathcal{X}, \mathcal{Y})$ is called hypothesis space.

hypothesis space

Example 1.2.15 Homogeneous polynomials spaces or reproducing kernel Hilbert spaces are commonly used hypothesis spaces. ◇

In reality we only have access to the evidence data. We thus assume a given sample $S := ((x_k, y_k))_{k=1}^N$ drawn i.i.d. according to σ and try to estimate $f_\mathcal{H}$ from S.

definition 1.2.16 (Empirical error / target function)
The empirical error of f with respect to loss ℓ and sample S is

$$\mathcal{E}_S(f) := \frac{1}{N} \sum_{k=1}^{N} \ell(f(x_k), y_k).$$

An empirical target function $f_{\mathcal{H},S} \in \mathcal{H}$ is a minimiser of $\mathcal{E}_S$ over $\mathcal{H}$.

The error of the empirical target function can be decomposed as

$$\mathcal{E}(f_{\mathcal{H},S}) = \underbrace{\mathcal{E}(f_{\mathcal{H},S}) - \mathcal{E}(f_\mathcal{H})}_{\substack{\text{generalization/} \\ \text{sample error}}} + \underbrace{\mathcal{E}(f_\mathcal{H})}_{\substack{\text{approx.} \\ \text{error}}} \quad \text{(Bias-Variance decomposition)}$$

Remark 1.2.17 The approximation error $\mathcal{E}(f_\mathcal{H})$ is only dependent on $\mathcal{H}$ while the generalization error also depends on size of the sample S.

Typically we observe the following behavior *for a fixed sample size*

$\mathcal{H}$ / error	generalization	approximation
larger	increase	decrease
smaller	decrease	increase

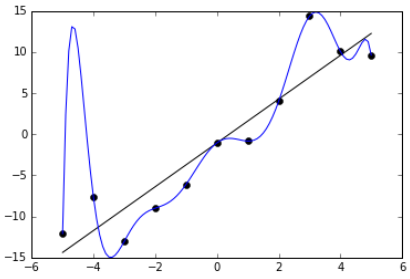


Fig. 5: Noisy (roughly linear) data is fitted to a linear function and a polynomial function. Although the polynomial function is a perfect fit, the linear function can be expected to generalize better: if the two functions were used to extrapolate beyond the fitted data, the linear function should make better predictions. [10]

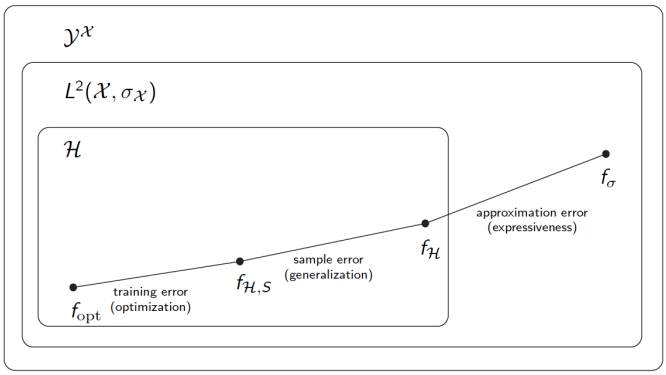


Fig. 6: Three types of errors in learning problems. We aim for a universally best method. [2]

Deep Neural Networks Enter the Stage

DEFINITION 1.2.18 (DEEP NEURAL NETWORK II)

In the setting of definition 1.2.4 the map

$$R_\varrho : \mathbb{R}^d \to \mathbb{R}^{N_L},\ \Phi(x) \mapsto \Phi_{(A_\ell, b_\ell)_{\ell=1}^L}(x)$$

is the realization of Φ with activation function ϱ and $\Phi := ((A_k, b_k))_{k=1}^L$ the neural network architecture

realization
architecture

We call $N(\Phi) := d + \sum_{\ell=1}^L N_\ell$ the number of neurons, $L(\Phi) := L$ the number of layers, $M(\Phi) := \sum_{\ell=1}^L \|A_\ell\|_0 + \|b_\ell\|_0$, where $\|\cdot\|_0$ denotes the number of non-zero entries.

For $d \in \mathbb{N}$, $M, L, N \in \mathbb{N} \cup \{\infty\}$ and $R \in \mathbb{R}$ we denote by $\mathcal{NN}_{d,M,N,L}^R$ the set of neural networks Φ with input dimension d, $N_L = 1$ and $M(\Phi) \leqslant M$ and $N(\Phi) \leqslant N$, where all weights are bound by R.

We call Φ sparsely connected if $M(\Phi)$ is small, shallow if L is small and deep if L is large.

Example 1.2.19 (Notation $\mathcal{NN}_{d,M,N,L}^R$) Let Φ be given by

$$A_1 := \begin{pmatrix} 1 & 2 & 0 \\ 0 & 0 & 5 \\ 0 & 0 & 4 \end{pmatrix},\ b_1 := \begin{pmatrix} 2 \\ 3 \\ 4 \end{pmatrix},\ A_2 := \begin{pmatrix} 2 & 6 & 0 \\ 0 & 0 & 7 \end{pmatrix},\ \text{and } b_2 := \begin{pmatrix} 8 \\ 8 \end{pmatrix}.$$

Then $\Phi \in \mathcal{NN}_{3,7,8,2}^8$.

An artificial neuron is a realization of a NN $\Phi \in \mathcal{NN}_{d,d,d+1,1}$. ◇

We will now cover the two basic operations on neural networks.

Lemma 1.2.20 (Concatenation)

Let $L_1, L_2 \in \mathbb{N}$ and $\Phi^{(i)} := \left((A_k^{(i)}, b_k^{(i)})\right)_{k=1}^{L_i}$ for $i \in \{1, 2\}$ be neural network such that the input layer of $\Phi^{(1)}$ and the output layer of $\Phi^{(2)}$ have the same dimensions. For any activation function ϱ

- $R_\varrho\left(\Phi^{(1)} \circ \Phi^{(2)}\right) = R_\varrho(\Phi^{(1)}) \circ R_\varrho(\Phi^{(2)})$

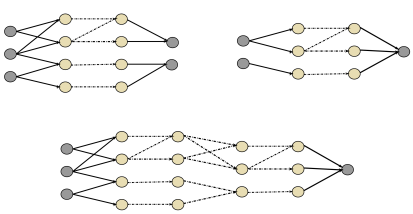


Fig. 7: Concatenation of neural networks. [2]

- $L\left(\Phi^{(1)} \circ \Phi^{(2)}\right) = L(\Phi^{(1)}) + L(\Phi^{(2)}) - 1$

hold, where $\Phi^{(1)} \circ \Phi^{(2)}$ *is defined as*

$$\left(\left((A_k^{(2)}, b_1^{(2)})\right)_{k=1}^{L_2-1}, (A_1^{(1)}A_{L_2}^{(2)}, A_1^{(1)}b_{L^2}^{(2)} + b_1^{(1)}), \left((A_k^{(1)}, b_k^{(1)})\right)_{k=2}^{L_1}\right).$$

Lemma 1.2.21

Let $L \in \mathbb{N}$ *and* $\Phi^{(i)}$ *both have* L *layers. For all activation functions* ϱ *and* $x \in \mathbb{R}^d$

- $R_\varrho(P(\Phi^{(1)}, \Phi^{(2)})(x) = \left(R_\varrho(\Phi^{(1)})(x), R_\varrho(\Phi^{(2)})(x)\right)$,
- $L\left(P(\Phi^{(1)}, \Phi^{(2)})\right) = L(\Phi^{(1)})$,
- $M\left(P(\Phi^{(1)}, \Phi^{(2)}\right) = M(\Phi^{(1)}) + M(\Phi^{(2)})$

holds, where $P(\Phi^{(1)}, \Phi^{(2)}) := \left((\hat{A}_k, \hat{b}_k)_{k=1}^{L}\right)$ *and*

$$\hat{A}_1 := \begin{pmatrix} A_1^{(1)} \\ A_1^{(2)} \end{pmatrix}, \text{ and } \hat{A}_\ell := \begin{pmatrix} A_\ell^{(1)} & 0 \\ 0 & A_\ell^{(2)} \end{pmatrix}, \hat{b}_\ell := \begin{pmatrix} b_\ell^{(1)} \\ b_\ell^{(2)} \end{pmatrix} \quad \forall \ell \in [L].$$

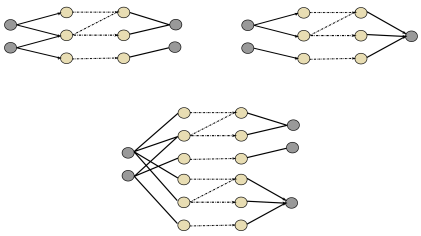


Fig. 8: Parallelization of neural networks. [2]

Lemma 1.2.22 (The identity with neural networks)

Define $\Phi^{(\mathrm{id})} := ((A_1, 0), (A_2, 0))$ *with* $A_1 := (\mathrm{id}_{\mathbb{R}^d}, -\mathrm{id}_{\mathbb{R}^d})^T$ *and* $A_2 := A_1^T$. *Then* $R_\varrho(\Phi^{(\mathrm{id})}) \equiv \mathrm{id}_{\mathbb{R}^d}$ *for* $\varrho := ReLU$.

Remark 1.2.23 Therefore different architectures can lead to the same realization; $R_\varrho(\Phi) = R_\varrho(\Phi \circ \Phi^{(\mathrm{id})})$.

1.3 Training of deep neural networks

Let $d := N_0$, $(N_k)_{k=1}^L \subset \mathbb{N}$ and ϱ an activation function. Define the hypothesis space

$$\mathcal{H} := \left\{R_\varrho(\Phi) : \Phi = ((A_k, b_k))_{k=1}^L, \ A_\ell \in \mathbb{R}^{N_{\ell-1}, N_\ell}, b_\ell \in \mathbb{R}^{N_\ell}\right\}.$$

Given samples $\mathbf{z} := ((x_i, y_i))_{i=1}^m \subset \mathbb{R}^d \times \mathbb{R}^{N_L}$, we aim to find the empirical target function

$$f_z := \arg\min_{f \in \mathcal{H}} \sum_{i=1}^{M} \mathcal{L}(f, x_i, y_i),$$

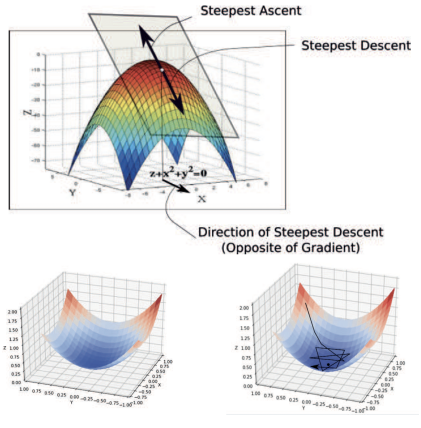
where $f : \mathcal{C}(\mathbb{R}^d, \mathbb{R}^{N_L}) \times \mathbb{R}^d \times \mathbb{R}^{N_L} \to \mathbb{R}$ is a loss function; e.g. the MSE.

loss function

As optimization approach we chose gradient descent: For $F : \mathbb{R}^N \to \mathbb{R}$ one uses the iteration

$$u_{n+1} \leftarrow u_n - \eta \nabla F(u_n), \quad n \in \mathbb{N},$$

where $\eta \in \mathbb{R}$ is the step size, a hyperparameter and the gradient ∇ is taken componentwise.



For our problem we set $F \coloneqq \sum_{i=1}^{m} \mathcal{L}(f, x_i, y_i)$ and $u \coloneqq ((A_\ell, b_\ell))_{\ell=1}^{L}$. In order to find $\nabla_{((A_\ell,b_\ell))_{\ell=1}^{L}} F$ we need to compute

$$\frac{\partial L(f,x,y)}{\partial (A_\ell)_{i,j}} \quad \text{and} \quad \frac{\partial L(f,x,y)}{\partial (b_\ell)_i}$$

for all i, j, ℓ.

This can be achieved by backpropagation: Compute a_ℓ, z_ℓ for $\ell \in \{0, \dots, L\}$ **WHAT ARE THEY??** Now set $\delta_L \coloneqq 2(f(x) - y)$. Then

backpropagation

$$\frac{\partial \mathcal{L}(f,x,y)}{\partial A_L} = \delta_L a_{L-1}^T \quad \text{and} \quad \frac{\partial \mathcal{L}(f,x,y)}{\partial b_L} = \delta_L.$$

For $\ell = L - 1$ to 1 now set

$$\delta_\delta \coloneqq \operatorname{diag}(\varrho'(z_\ell)) A_{\ell+1}^T \cdot \delta_{\ell+1}.$$

Then we have

$$\frac{\partial \mathcal{L}(f,x,y)}{\partial A_\ell} = \delta_L a_{\ell-1}^T \quad \text{and} \quad \frac{\partial \mathcal{L}(f,x,y)}{\partial b_\ell} = \delta_\ell.$$

and can return $\nabla_{(A_\ell,b_\ell)_{\ell=1}^{\ell}} \mathcal{L}(f, x, y)$.

Our goal therefore is to find a stationary point of $F \coloneqq \sum_{i=1}^{m} F_i : \mathbb{R}^N \to \mathbb{R}$, where $F_i \coloneqq \mathcal{L}(f, x_i, y_i)$.

An algorithm realizing this takes as input data a neural network f and a loss function $\mathcal{L}$. After initializing an starting value u_0 and $n = 0$, while the error is large, choose $i^* \in [m]$ uniformly at random and update $u_{n+1} \leftarrow u_n - \eta \nabla F_{i*}$ and $n + 1 \leftarrow n$. Finally, return u_n.

2 Expressivity

Expressivity is concerned with the following questions.

- Which architecture to choose for a particular application?
- What is the expressive power of a given architecture?
- What effect has the depth of a neural network in this respect?
- What is the complexity of the approximating neural network?
- What are suitable function spaces to consider?
- Can deep neural networks beat the curse of dimensionality?

Mathematically, we ask under which conditions on a neural network Φ and an activation function ϱ every function from a prescribed function class $\mathcal{C}$ can be arbitrarily well approximated, i.e. $\|R_\varrho(\Phi) - f\| \leqslant \varepsilon$ for all $f \in \mathcal{C}$.

2.1 Viewpoint of approximation theory

Let a function class $\mathcal{C} \subset L^2(\mathbb{R}^d)$ and a function system $(\varphi)_{i\in I} \subset L^2(\mathbb{R}^d)$ be given. In order to measure the suitability of the function system for uniform approximation of $\mathcal{C}$ we introduce the

DEFINITION 2.1.1 (M-TERM BEST APPROXIMATION)
The error of best M-term approximation for $f \in \mathcal{C}$, $\|f - f_M\|_2$, is

$$\inf\left\{\left\|f - \sum_{i\in I_M} c_i\varphi_i\right\|_2 : I_M \subset I, |I_M| = M, (c_i)_{i\in I_M} \subset \mathbb{R}\right\}.$$

DEFINITION 2.1.2 (OPTIMAL APPROXIMATION RATE)
The largest $\gamma > 0$ such that

$$\sup_{f\in\mathcal{C}} \|f - f_M\|_2 = \mathcal{O}(M^{-\gamma}) \quad \text{as } M \to \infty$$

is the optimal (sparse) approximation rate of $\mathcal{C}$ by $(\varphi_i)_{i\in I}$.

We now introduce wavelets and their associated transform, which is applied in the compression standard JPEG200.

DEFINITION 2.1.3 ($\mathbb{R}$-WAVELET)
Let $\varphi \in L^2(\mathbb{R})$ a scaling function and $\psi \in L^2(\mathbb{R})$ a wavelet. Then the associated wavelet system is

$$\{\varphi(x-m) : m \in \mathbb{Z}\} \cup \{2^{j/2}\psi(2^j x - m) : j \geqslant 0, m \in \mathbb{Z}\}.$$

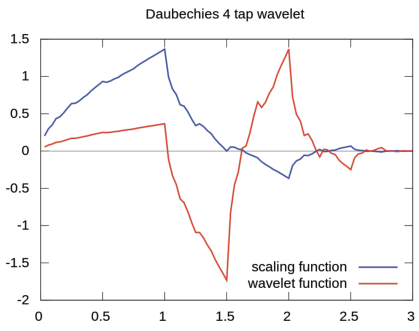


Fig. 9: A wavelet. [2]

Definition 2.1.4 ($\mathbb{R}^2$-Wavelet)
The the associated wavelet system is

$$\{\varphi^{(1)}(x-m)\}_{m\in\mathbb{Z}} \cup \{2^{j/2}\psi^{(i)}(2^j x - m) : j \geqslant 0, m \in \mathbb{Z}, i \in \{1,2,3\}\},$$

where

$$\varphi^{(1)}(x) := \varphi(x_1)\varphi(x_2),\ \psi^{(1)}(x) := \varphi(x_1)\psi(x_2),$$
$$\psi^{(2)}(x) := \psi(x_1)\varphi(x_2),\ \psi^{(3)}(x) := \psi(x_1)\psi(x_2)$$

The wavelet transform is given by

$$f \mapsto \big((\langle f, \varphi_m \rangle)_m, (\langle f, \psi_{j,m,i} \rangle)_{j,m,i}\big). \tag{2}$$

In order to find mathematical models for images we see that their key components are usually shapes separated by an edge, i.e. an singularity.

Definition 2.1.5 (Cartoon-like functions (Donoho))
The set of cartoon-like functions is

$$\mathcal{E}^2(\mathbb{R}^2) := \{f \in L^2(\mathbb{R}^2) : f = f_0 + f_1 \cdot \mathbb{1}_B\},$$

where $\varnothing \neq B \subset [0,1]^2$ is simply connected with $\mathcal{C}^2$-boundary and bounded curvature and the $f_i \in \mathcal{C}^2(\mathbb{R}^2)$ are supported in $[0,1]^2$ with $\|f_i\|_{\mathcal{C}^2} \leqslant 1$, $i \in \{0,1\}$.

Theorem 2.1.1: Donoho (2001)

Let $(\psi_\lambda)_\lambda \subset L^2(\mathbb{R}^2)$. Allowing only polynomial depth search, we have the following optimal behavior for $f \in \mathcal{E}^2(\mathbb{R}^2)$:

$$\|f - f_N\|_2 \asymp N^{-1} \quad \text{as } N \to \infty$$

Unfortunately, Donoho **(TODO: Citation)** also proved that the optimal rate for any directional representation system, of which wavelets are just one example, the optimal approximation rate is 2, so wavelets cannot represent cartoon-like functions well. This is due to the isotropic structure of wavelets: they are scaled by $2^{j/2}$ in both directions equally.

Definition 2.1.6 (Shearlets (Kutyniok, Labate, 2006))
For $j \in \mathbb{Z}$ let $A_j := \operatorname{diag}\left(2^j, 2^{j/2}\right)$ and $S_j := \left(\begin{smallmatrix} 1 & j \\ 0 & 1 \end{smallmatrix}\right)$ and ψ be a wavelet. A shearlet is the map $\psi_{j,k,m}(x) := 2^{3j/4}\psi(\underbrace{S_k A_j x - m}_{\text{affine linear}})$.

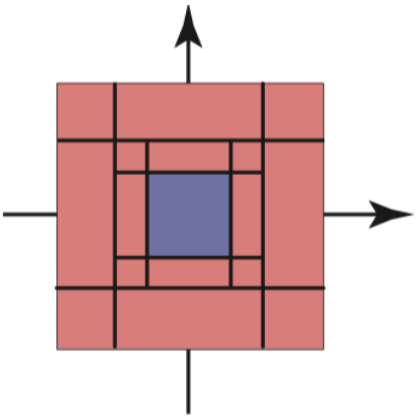


Fig. 10: Tiling of the plane induced by wavelets. [2]

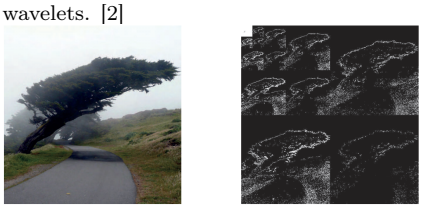


Fig. 11: The wavelet transform. [2]

cartoon-like functions

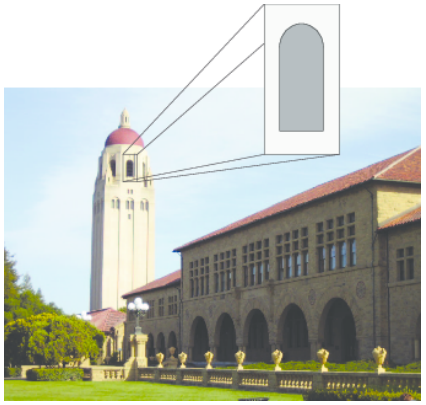


Fig. 12: Anisotropic (dominated by an edge) features govern natural images.

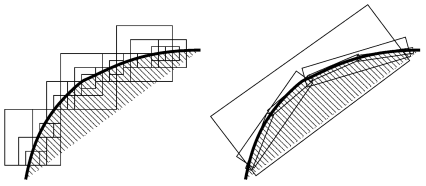


Fig. 13: Intuitive explanation for why wavelets don't preform as well as shearlets.

shearlet

definition 2.1.7 (Cone-adapted shearlet system)
The cone-adapted shearlet system $\mathcal{SH}(\varphi, \psi, \tilde{\psi})$ generated by $\varphi, \psi, \tilde{\psi} \in L^2(\mathbb{R}^2)$ is the union of

$$\{\psi(\cdot - m) : m \in \mathbb{Z}\},$$
$$\{2^{3j/4}\psi(S_k A_j x - m) : j \geqslant 0, |k| \leqslant \lceil 2^{j/2} \rceil, m \in \mathbb{Z}^2\}$$
$$\{2^{3j/4}\tilde{\psi}(\tilde{S}_k \tilde{A}_j x - m) : j \geqslant 0, |k| \leqslant \lceil 2^{j/2} \rceil, m \in \mathbb{Z}^2\},$$

where $\tilde{A}_j := \operatorname{diag}(2^{j/2}, 2^j)$ and $\tilde{S}_k := S_k^T$.

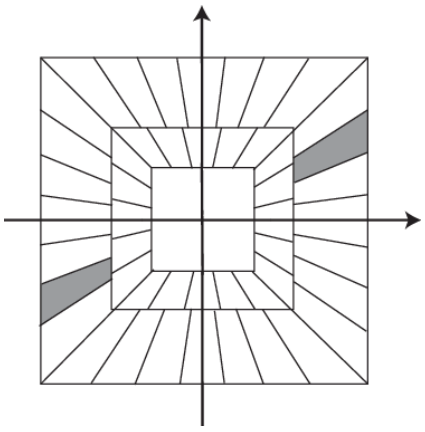


Fig. 14: The tiling of the frequency domain induced by cone adapted shearlets.

Theorem 2.1.2: Shearlet frame is parseval frame

For classical shearlets $\psi, \tilde{\psi}$ and all $f \in L^2(\mathbb{R}^2)$ we have ([28])

$$\|f\|_2^2 = \sum_{\sigma \in \mathcal{SH}(\varphi,\psi,\tilde{\psi}} |\langle f, \sigma \rangle|^2.$$

Theorem 2.1.3: Compactly supported shearlets are optimally sparse

Let $\varphi, \psi, \tilde{\psi} \in L^2(\mathbb{R}^2)$ be compactly supported and $\hat{\psi}, \hat{\tilde{\psi}}$ satisfy certain decay conditions. Then $\mathcal{SH}(\varphi, \psi, \tilde{\psi})$ provides an optimally sparse approximation of $f \in \mathcal{E}^2(\mathbb{R}^2)$, i.e.

$$\|f - f_N\|_2 \lesssim N^{-1}(\log(N))^{\frac{3}{2}} \quad \text{as } N \to \infty$$

The associated algorithms (2 & 3D (parallelized) fast shearlet transform) have been implemented mostly at TU Berlin, i.e. in `Matlab` (Kutynio, Lim, Reisenhofer; 2013), `Julia` (Loarca; 2017), `Python` (Look; 2018) and `Tensorflow` (Kutyniok, Loarca; 2019), see also www.shearlab.org.

Lets get back to expressivity of DNNs. In general, we ask which complexity $M(\Phi)$ (cf. definition 1.2.18) a neural network Φ, which approximates a function f up to $\varepsilon > 0$, needs to have.

2.2 Universality results

5.11.19 (?)

We will now see the universality of shallow neural networks; that every continuous function on a compact set can be arbitrarily well approximated with a neural network with one single hidden layer in a theorem by Cybenko (1989) and Hornik (1991).

Remark 2.2.1 Assume ϱ is a polynomial of degree $q \in \mathbb{N}$. Then $\varrho(Ax+b)$ is also a polynomial of degree q, hence $R_\varrho(\Phi)$ is also a polynomial with degree less than or equal to $L \cdot q$. In this case, $\mathcal{C}(\mathbb{R}^d)$ cannot be approximated well.

Theorem 2.2.1: Universal Approximation Thm

Let $\varrho : \mathbb{R} \to \mathbb{R}$ be continuous but not a polynomial and $K \subset \mathbb{R}^d$ a compact set. Then for any continuous function $f : \mathbb{R}^d \to \mathbb{R}^{N_L}$ and every $\varepsilon > 0$ there exists $M, N \in \mathbb{N}$ and $\Phi \in \mathcal{NN}_{d,M,N,2}$ with

$$\sup_{x \in K} |R_\varrho(\Phi)(x) - f(x)| \leqslant \varepsilon. \tag{3}$$

Proof. We prove that for $d \geqslant 1$ and $\varrho \in \mathcal{C}(\mathbb{R}, \mathbb{R})$ the following are equivalent:

1. ϱ is not a polynomial.
2. $\overline{S} = \mathcal{C}(K)$, where $S := \operatorname{span}(\varrho(\langle w, x \rangle - b) : w \in \mathbb{R}^d, b \in \mathbb{R})$. **DONT WE NEED SOME DOMAIN-RESTRICTIONS? Otherwise the RHS will contain more functions surely.**

"1 $\implies$ 2":

a. First consider $d = 1$ and smooth ϱ. By 1 there exists a $x_0 \in \mathbb{R}$ such that $\varrho^{(k)}(-x_0) \neq 0$ for all $k \in \mathbb{N}$. By a theorem of Stone-Weierstrass it suffices to show that S is dense in the polynomials on $\mathbb{R}$, as they are dense in $\mathcal{C}(K)$.

We have $\varrho(hx - x_0) \xrightarrow{h \to 0} \varrho(-x_0)$, thus the constant functions are well approximated. Furthermore,

$$\underbrace{\frac{1}{h}\big(\varrho((\lambda + h)x - x_0) - \varrho(x - x_0)\big)}_{\xrightarrow{h \to 0} x\varrho'(\lambda x - x_0)} \xrightarrow{h, \lambda \to 0} x p'(-x_0)$$

holds, thus linear functions and therefore, by linearity **(WHAT ABOUT MULTIPLICATIVITY??)** polynomials are well approximated.

b. Now consider arbitrary $d \geqslant 1$, smooth ϱ and

$$\operatorname{span}(g(\langle w, x \rangle - b) : w \in \mathbb{R}^d, b \in \mathbb{R}, g \in \mathcal{C}(K)) \subset \mathcal{C}(K),$$

HIER IST ES NÄMLICH $g \in \mathcal{C}(K)$ nicht $\mathcal{C}(\mathbb{R}, \mathbb{R})$!!which is dense. Thus, every $f \in \mathcal{C}(K)$ can we approximated by

$$\sum_{k=1}^{N} a_i g_i(\langle w_i, x \rangle - b_i), \quad a_i, b_i \in \mathbb{R}, w_i \in \mathbb{R}^d, g_i \in \mathcal{C}(K).$$

By a $t \mapsto g_i(t - b_i)$ can be well approximated for all $t \in \mathbb{R}$.

c. For non-smooth ϱ use a mollification $(g_\varepsilon)_{\varepsilon > 0}$ such that $g * g_\varepsilon \xrightarrow{\varepsilon \searrow 0} \varrho$.

Another version of this proof can be found in the original paper, [7]. □

Generally, neural networks have huge approximation power: The following theorem by Maiorov and Pinkus (1999) doesn't restrict the size of the weights!

Theorem 2.2.2: Universal network theorem

There exists an activation function $\varrho : \mathbb{R} \to \mathbb{R}$ such that for any $d \in \mathbb{N}$, compact $K \subset \mathbb{R}^d$, continuous $f : K \to \mathbb{R}$ and $\varepsilon > 0$ there exist $M, N \in \mathbb{N}$ (only dependent on d) and $\Phi \in \mathcal{NN}_{d,M,N,3}$ such that (3) holds.

2.3 Lower bounds for approximation

26.11.19

The question we will try to answer in this chapter are

- How well functions can be approximated by neural networks with few non-zero weights ($\rightsquigarrow$ sparse connectivity)?
  - Can we derive a lower bound on the necessary number of weights
  - Can we construct neural networks which attain this bound?
- Do neural networks approximate as good as wavelets and shearlets?

definition 2.3.1 (Vapnik-Chervoenkis dimension)

Let X be a set, $S \subset X$ and $H \subset \{h : X \to \{0,1\}\}$. Define the restriction of the function class H to S $H|_S := \{h|_S : h \in H\}$. The VC dimension of H is

VC dimension

$$\operatorname{VCdim}(H) := \sup\left\{m \in \mathbb{N} : \sup_{|S| \leqslant m} |H|_S| = 2^m\right\}.$$

The VC dimension is a tool for understanding the classification capabilities of a function class as $2^m = |\{0,1\}^m|$ is the number of possible combinations of binary labels on a set of m points. The VC dimension of H is the largest integer m such that there exists a set $S \subset X$ containing only m points such that $H|_S$ has the maximum possible cardinality, given by 2^m.

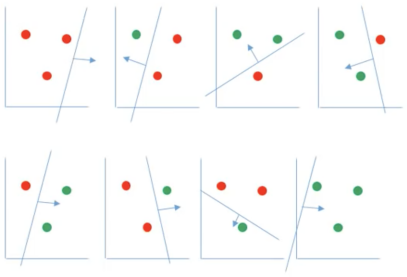


Fig. 15: All eight possible ways to shatter three non-collinear points in the plane. [8]

Example 2.3.2 (VC-Dimension) Let $X := \mathbb{R}^2$, $h := \mathbb{1}_{\mathbb{R}^+}$ and

$$H := \left\{h\left(\left\langle \frac{\cos(\theta)}{\sin(\theta)}, \cdot - t\right\rangle\right) : \theta \in [0, \pi], t \in \mathbb{R}^2\right\}.$$

Then H has VC-dimension 3 as at most 3 (non-collinear) points can be shattered by a linear classifier from H.

A hypothesis class with infinite VC-dimension is $(t \mapsto \alpha \sin(\omega t))_{\alpha,\omega \in \mathbb{R}}$. $\diamond$

Theorem 2.3.1: VC dimension of DNNs [29])

Let ϱ be a piecewise polynomial with p pieces of degree at most ℓ, $h := \chi_{\mathbb{R}^+}$ and

$$H_{N,M,d,L} := \{h \circ R_\varrho(\Phi) : \Phi : \mathcal{NN}_{d,M,N,L}\}$$

for $N, M, d \in \mathbb{N}$. Then

$$\mathrm{VCdim}(H_{N,M,d,L}) \leqslant 2ML\log_2\left(\frac{4MLpN}{\ln(2)}\right) + 2ML^2\log_2(\ell+1) + 2L$$

holds.

Corollary 2.3.3

For fixed p and ℓ and $N \leqslant M$ this implies

$$\mathrm{VCdim}(H_{N,M,d,L}) \in \mathcal{O}(ML\log_2(M) + ML^2).$$

Theorem 2.3.2: Smooth function approximation [5]

Let $L \in \mathbb{N}$ and $H := \{h \in \mathcal{C}^n([0,1]^d) : \|h\|_{\mathcal{C}^n} \leqslant 1\}$ and ϱ the ReLU. If

$$\forall \varepsilon > 0\ \forall h \in H\ \exists \Phi \in \mathcal{NN}_{d,M(\varepsilon),M(\varepsilon),L} : \|h - R_\varrho(\Phi)\|_\infty < \varepsilon,$$

holds,

$$M(\varepsilon) \in \Omega\left(\varepsilon^{-\frac{d}{n(1+\delta)}}\right)$$

follows for *all* $\delta > 0$.

Proof. For convinience let $\eta := 1 + \delta$. Towards contradiction assume that for all $h \in H$ there exists a $\Phi_h \in \mathcal{NN}_{d,M,M,\ell}$ with $\ell \leqslant L$ such that

$$\|h - R_\varrho(\Phi_h)\|_\infty \leqslant \frac{1}{8}M^{-\frac{\eta n}{d}} \tag{4}$$

for some $\delta > 0$.

Let $m := M^\eta$ and $(x_k)_{k=1}^m \subset [0,1]^d$ such that $|x_i - x_j| \geqslant M$ for all $i, j \in [m]$. Around the x_i construct $\varphi_i \in H$ with disjoint support and diameter $M^{-\frac{\eta}{d}}$. Additionally require

$$\varphi_i(x_i) = \frac{1}{2} \cdot M^{-\frac{\eta}{d}} \quad \forall i \in [m] \tag{5}$$

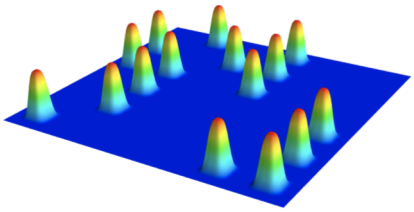


Fig. 16: A visualization of the φ_i. [5]

For every $g : \{x_1, \ldots, x_m\} \to \mathbb{R}$ there exist $(d_k)_{k=1}^m \subset \{0,1\}^m$ such that

$$g(x) = 2M^{\frac{\eta n}{d}} \sum_{i=1}^m d_i \varphi_i(x).$$

By (4) there exists a neural network $\Phi \in \mathcal{NN}_{d,M,M,\ell}$ such that

$$\left\| \sum_{i=1}^m d_i\varphi_i(x) - R_\varrho(\Phi) \right\|_\infty \leqslant \frac{1}{8}M^{-\frac{\eta n}{d}} \tag{6}$$

WOHER BLEIBT DER VORFAKTOR VON g??? By (5) and (6) we have

$$2M^{\frac{\eta n}{d}} R_\varrho(\Phi)(x_i) - \frac{1}{2} \begin{cases} > 0, & \text{if } g(x_i) = 1, \\ \leqslant 0, & \text{if } g(x_i) = 0 \end{cases}$$

for all $i \in [m]$, hence

$$\boxed{\chi_{\mathbb{R}^+}\left(2M^{\frac{\eta}{d}} R_\varrho(\Phi) - \frac{1}{2}\right) \equiv g \quad \text{on } \{x_1, \ldots, x_m\}}$$

holds. By corollary 2.3.3 we have

$$m = M^\eta \in \mathcal{O}(M \log(M)) \ \forall \delta > 0,$$

which is a contradiction. □

definition 2.3.4 (Rate distortion theory)
Let $d \in \mathbb{N}$, $\Omega \subset \mathbb{R}^d$ and $\mathcal{C} \subset L^2(\Omega)$.

1. For $\ell \in \mathbb{N}$ the sets

$$\mathcal{E}^\ell := \left\{E : \mathcal{C} \to \{0,1\}^\ell\right\}, \quad \mathcal{D}^\ell := \left\{D : \{0,1\}^\ell \to L^2(\Omega)\right\}$$

are the sets of binary en(de)coders of length ℓ. — binary en(de)coders

2. An encoder-decoder pair $(E, D) \in \mathcal{E}^\ell \times \mathcal{D}^\ell$ achieves distortion $\varepsilon > 0$ over $\mathcal{C}$ if $\mathfrak{D}(E, D) := \sup_{f \in \mathcal{C}} \|D(E(f)) - f\|_{L^2} \leqslant \varepsilon$ holds. — encoder-decoder pair, distortion

3. For $\varepsilon > 0$ we define the minimal code length — minimal code length

$$L(\varepsilon, \mathcal{C}) := \min\left\{\ell \in \mathbb{N} : \exists (E, D) \in \mathcal{E}^\ell \times \mathcal{D}^\ell : \mathfrak{D}(E, D) \leqslant \varepsilon\right\}$$

and the optimal exponent — optimal exponent

$$\gamma^*(\mathcal{C}) := \inf\left\{\gamma \in \mathbb{R} : L(\varepsilon, \mathcal{C}) \in \mathcal{O}\left(\varepsilon^{-\gamma}\right)\right\}.$$

Example 2.3.5 (Optimal exponent)
Theorem 2.1.1 states that $\gamma^*(\mathcal{E}^2(\mathbb{R}^2)) \geqslant 1$. If $\mathcal{C} \subset B^s_{p,q}(\mathbb{R}^d)$ is bounded, we have $\gamma^*(\mathcal{C}) = \frac{d}{s}$. ◇

Example 2.3.6 (Mapping functions to bit sequences)
Let

$$\mathcal{C} := \{f \in \mathcal{C}([0,1],[0,1]) : |f(x) - f(y)| \leqslant |x - y|\}.$$

For $(x_k)_{k=1}^n \subset [0,1]^n$ with $x_1 < \ldots < x_n$ define

$$\mathcal{E}^\ell : \mathcal{C} \to \{0,1\}^\ell, \ f \mapsto \left\{\mathrm{bin}_{\frac{\ell}{n}}(f(x_1)), \ldots, \mathrm{bin}_{\frac{\ell}{n}}(f(x_n))\right\},$$

where $\mathrm{bin}_n(c)$ is the binary approximation / representation of c of length n. Define $\mathcal{D}^\ell : \{0,1\}^\ell \to L^2([0,1])$ by

$$\mathcal{D}^\ell(c) := \sum_{i=1}^N \mathrm{Num}\left(c\left(\frac{\ell(i-1)}{n}\right), \ldots, c\left(\frac{\ell(i)}{n}\right)\right) \mathbb{1}_{[x_{i-1}, x_i]},$$

where $\mathrm{Num}(c)$ is the real value associated with the bit sequence c.

The distortion of $(\mathcal{E}^\ell, \mathcal{D}^\ell)$ is bounded by

$$2^{\frac{\ell}{n}} \max_{k=1}^n (|x_k - x_{k+1}|, |x_k - 1|).$$

Now choosing

$$n \sim \frac{\ell}{\log(\ell)} \qquad \text{and} \qquad |x_i - x_{i+1}| \sim \frac{1}{n}$$

yields a distortion of $\frac{\log(\ell)}{\ell}$. ◇

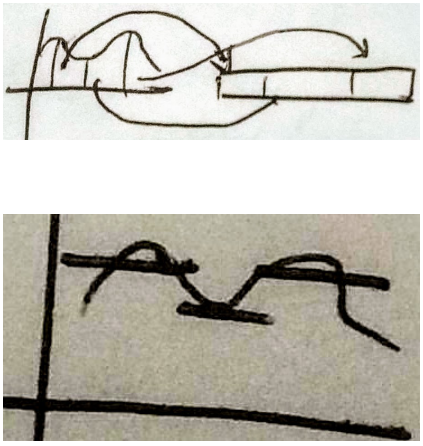


Fig. 17: Visualization of the encoder and decoder.

Theorem 2.3.3: γ^* and N-term approximation

For $\mathcal{C} \subset L^2(\mathbb{R}^d)$ the optimal N-term approximation rate is $\frac{1}{\gamma^*(\mathcal{C})}$.

Theorem 2.3.4: A fundamental lower bound [30]

Let $d \in \mathbb{N}$, $\varrho : \mathbb{R} \to \mathbb{R}$, $c > 0$, $\mathcal{C} \subset L^2(\mathbb{R}^d)$ and

$$\text{Learn} : \left(0, \frac{1}{2}\right) \times \mathcal{C} \to \mathcal{NN}_{d,\infty,\infty,\infty}$$

be an abstract learning procedure such that all weights of $\text{Learn}(\varepsilon, f)$ can be encoded with $-c\log_2(\varepsilon)$ bits. If $\mathcal{D}(\text{Learn}(\cdot, \varepsilon), R_\varrho) \leqslant \varepsilon$ holds,

$$\sup_{\varepsilon \in \left(0,\frac{1}{2}\right)} \varepsilon^\gamma \sup_{f \in \mathcal{C}} M(\text{Learn}(\varepsilon, f)) = \infty$$

holds for all $\gamma < \gamma^*(\mathcal{C})$.

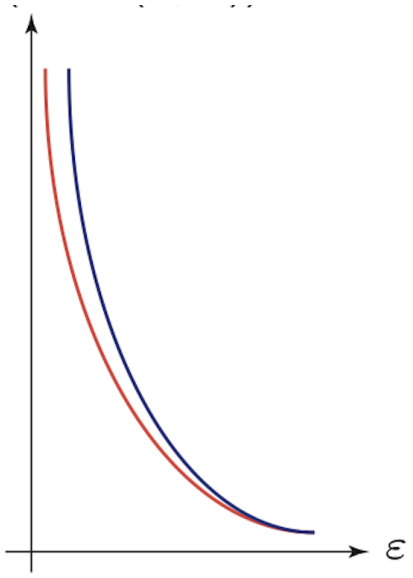


Fig. 18: The red line represents $\varepsilon^{-\gamma}$ and the blue one $\text{Learn}(\varepsilon, \mathcal{C})$.

Remark 2.3.7 The statement is equivalent to

$$\nexists C > 0 : \sup_{f \in \mathcal{C}} M(\text{Learn}(\varepsilon, f)) \leqslant C\varepsilon^{-\gamma} \quad \forall \varepsilon > 0 \ \forall \gamma < \gamma^*(\mathcal{C}).$$

Therefore, the number of edges is bounded below by $\varepsilon^{-\gamma}$.

Remark 2.3.8 This bound is in terms of the edges, hence the sparsity of the connectivity, not the neurons. However, the number of neurons is always bounded up to a uniform constant by the number of edges.

28.11.2019

Proof. Let $f \in \mathcal{C}$ and $\varepsilon > 0$. Towards contradiction assume there exists a $C > 0$ with

$$M := M(\text{Learn}(\varepsilon, f)) \leqslant C\varepsilon^{-\gamma}, \quad \gamma < \gamma^*(\mathcal{C}).$$

Each weight is encoded by $-c\log_2(\varepsilon)$ bits and $N(\text{Learn}(\varepsilon, f)) \leqslant 2M$. Then $\text{Learn}(\varepsilon, f)$ can be encoded by

$$\underbrace{2M\log_2(2M)}_{\substack{\text{position} \\ \text{of weights}}} + \underbrace{-c\log_2(\varepsilon)M}_{\substack{\text{encoding} \\ \text{weights}}} \leqslant \tilde{C}\varepsilon^{-\gamma}\log\left(\varepsilon^{-1}\right) =: C(\varepsilon, \gamma).$$

bits without loosing information. Thus there exists an invertible encoder

$$\mathfrak{T}_\varepsilon : \mathcal{NN}_{d,L,\lfloor \tilde{c}\cdot\varepsilon^{-\gamma}\rfloor,\lfloor \bar{c}\cdot\varepsilon^{-\gamma}\rfloor} \to \{0,1\}^{C(\varepsilon,\gamma)}.$$

Now set

$$\mathcal{E}_\varepsilon : \mathcal{C} \to \{0,1\}^{C(\varepsilon,\gamma)}, \ f \mapsto \mathfrak{T}_\varepsilon(\text{Learn}(\varepsilon, f)),$$
$$\mathcal{D}_\varepsilon : \{0,1\}^{C(\varepsilon,\gamma)} \to L^2(\mathbb{R}), \ c \mapsto R_\varrho(\mathfrak{T}_\varepsilon^{-1}(c)).$$

By construction, for all $f \in \mathcal{C}$

$$\begin{aligned}\| \mathcal{D}_\varepsilon(\mathcal{E}_\varepsilon(f)) - f\|_{L^2} &= \|R_\varrho(\cancel{\mathfrak{T}_\varepsilon^{-1}}(\cancel{\mathfrak{T}_\varepsilon}(\mathrm{Learn}(\varepsilon, f)))) - f\|_{L^2} \\ &= \|R_\varrho(\mathrm{Learn}(\varepsilon, f)) - f\|_{L^2} < \varepsilon\end{aligned}$$

holds. We therefore have constructed an encoder-decoder pair of distortion ε and bit length $-\varepsilon^\gamma \log(\varepsilon^{-1})$. But $\gamma^*(\mathcal{C})$ is the smallest γ such that there exists an encoder-decoder pair of length $c\varepsilon^{-\gamma}$. This is a contradiction. □

But what happens in the border case $\gamma = \gamma^*(\mathcal{C})$?

2.4 Upper bounds and optimality

We investigate whether we can construct neural networks with attain the bounds proven above and how wavelets and shearlets compare to neural networks in this regard.

definition 2.4.1 (Approximation method)
Let $K \subset \mathbb{R}^d$ be a compact set. A family of continuous coefficient mappings $\mathcal{A}_N : L^2(K) \to \mathbb{R}^N$, $N \in \mathbb{N}$ and reconstruction mappings $\mathcal{R}_N : \mathbb{R}^N \to L^2(K)$ is called approximation method.

approximation method

Example 2.4.2 $\mathcal{A}_n(g)$ could be the first n shearlet coefficients of g. ⋄

Our larger goal is to prove the following bold statement:

Theorem 2.4.1
Let $(\mathcal{A}_N, \mathcal{R}_N)_{N\in\mathbb{N}}$ be an approximation method and $\varrho \in \mathcal{C}(\mathbb{R}, \mathbb{R})$ not a polynomial. Then for every $N \in \mathbb{N}$ and every $g \in L^2(K)$ there exists a neural network Φ with $M(\Phi) \leqslant N \operatorname{polylog}(N)$ and

$$\|g - R_\varrho(\Phi)\|_{L^2(K)} \leqslant \|g - (\mathcal{R}_N \circ \mathcal{A}_N)(g)\|_{L^2(K)}.$$

Therefore, even a ReLU neural network will outperform any approximation method.

ReLU Calculus

Notice that the ReLU activation function is not much different from using any other piece-wise linear activation function with finitely many breakpoints: one can replace one network by an equivalent one but having another activation function while only increasing the number of units and weights by constant factors ([5], proposition 1).

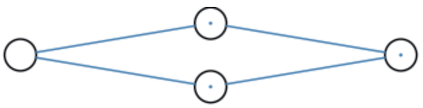


Fig. 19: A neural network representing the hat function. [2]

Define the hat function

$$g : \mathbb{R} \to [0,1],\ x \mapsto (1-|2x-1|)\,\mathbb{1}_{[0,1]}(x) = \begin{cases} 2x, & \text{if } x \in \left(0, \frac{1}{2}\right), \\ 2(1-x), & \text{if } x \in \left[\frac{1}{2}, 1\right], \\ 0, & \text{else.} \end{cases}$$
$$= \mathrm{ReLU}\left(2\,\mathrm{ReLU}(x) - 4\,\mathrm{ReLU}\left(x - \frac{1}{2}\right)\right)$$
$$= 2\min(x, 1-x)\,\mathbb{1}_{[0,1]}(x).$$

and the sawtooth function $g_m := \underbrace{g \circ \ldots \circ g}_{m \text{ times}}$ [6].

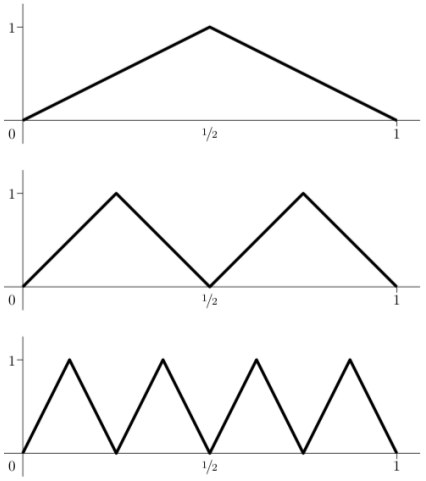


Fig. 20: The function g_m for $m \in \{0,1,2\}$. [6], Subsection 2.2.

Lemma 2.4.3 (Square function [5])
For all $\varepsilon > 0$ there exists a neural network Φ with $\mathcal{O}(\log(\varepsilon^{-1}))$ layers and nodes such that $\sup_{x\in[0,1]} |x^2 - R_{\mathrm{ReLU}}(\Phi(x))| \leqslant \varepsilon$.

Proof. We have $x^2 = x - \sum_{m=1}^{\infty} 2^{-2m} g_m(x)$. Defining $f_k(x) := x - \sum_{m=1}^{k} 2^{-2k} g_k(x)$ yields $\|x^2 - f_k(x)\| = 2^{-2(k+1)}$ [1] and f_k can be realized by a neural network with $\mathcal{O}(k)$ layers and nodes. Setting $\varepsilon = 2^{-2(k+1)}$ yields $k \in \mathcal{O}\left(\log_2\left(\frac{1}{\varepsilon}\right)\right)$. □

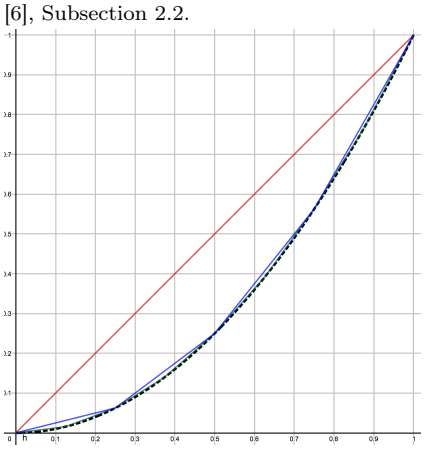


Fig. 21: The functions f_k for $k \in \{1,2\}$ and x^2. [1]

Lemma 2.4.4 (Multiplication [5])
For $\varepsilon > 0$ there exists a neural network Φ with $\mathcal{O}(\log(\varepsilon^{-1}))$ layers and nodes such that $\sup_{(x_1,x_2)\in[0,1]^2} |x_1 x_2 - R_{\mathrm{ReLU}}(\Phi)(x_1,x_2)| \leqslant \varepsilon$.

Proof. This follows directly from $x_1 x_2 = \frac{1}{2}\left[(x_1+x_2)^2 - (x_1^2 + x_2^2)\right]$ and the previous lemma. □

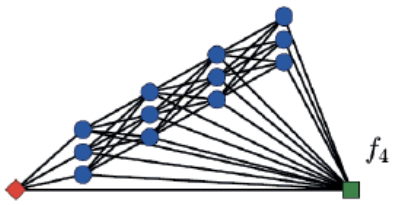


Fig. 22: Neural network realizing multiplication. (really?) [5], subsection 3.1.

Lemma 2.4.5 (Polynomials [5])
For $\varepsilon > 0$, there exists a neural network Φ with $\mathcal{O}(\mathrm{polylog}(\varepsilon^{-1}))$ layers and nodes such that $\sup_{x\in[0,1]} \left|\sum_{k=0}^{\ell} a_i x^i - R_{\mathrm{ReLU}}(\Phi)(x)\right| \leqslant \varepsilon$.

Proof. Monomials can be approximated by iteratively approximating the square and multiplication with the identity (cf. lemma 1.2.22). Therefore the result follows as neural networks are closed with respect to linear combinations. □

Lemma 2.4.6 (Smooth functions I [5])
Suppose that f is Gevrey, i.e. there exist $C, R, \sigma > 0$ such that $\sup_{x\in[0,1]} \left|f^{(k)}(x)\right| \leqslant C R^k (k!)^\sigma$ for all $k \in \mathbb{N}$. Then for all $\varepsilon > 0$ there exists a neural network Φ with $\mathcal{O}\left(\mathrm{polylog}(\varepsilon^{-1})\right)$ layers and nodes such that $\sup_{x\in[0,1]} |f(x) - R_{\mathrm{ReLU}}(\Phi)(x)| \leqslant \varepsilon$.

[1] It suffices to consider the last interval $\left[1-2^{-k}, 1\right]$ as $\frac{\mathrm{d}^2}{\mathrm{d}x^2}(x^2) > 0$. The linear interpolation of x^2 in this interval is $\bar{f}_k(x) := (2-2^{-k})x + 2^{-k} - 1$. By differentiating (maximum at $x = 1 - 2^{-(k+1)}$) one obtains

$$\|x^2 - \bar{f}_k(x)\|_\infty = \sup_{x\in[1-2^{-k},1]} (2-2^{-k})x + 2^{-k} - 1 - x^2 = 2^{-2k-2}.$$

Proof. As f can be well approximated by local polynomials, this follows from lemma 2.4.5. □

Theorem 2.4.2: Smooth functions II [5]

Let f satisfy $\|f\|_{\mathcal{C}^n([0,1])} \leqslant 1$. Then for all $\varepsilon > 0$ there exists a neural network Φ with $\mathcal{O}(\log(\varepsilon^{-1}))$ layers and $\mathcal{O}(\varepsilon^{-\frac{1}{n}})$ nodes such that $\sup_{x\in[0,1]} |f(x) - R_{\mathrm{ReLU}}(\Phi)(x)| \leqslant \varepsilon$.

Proof. (Sketch) As f can be approximated by local polynomials, we can use lemma 2.4.5 and argue the optimality by using VC-dimension. □

Remark 2.4.7 This bound is optimal in view of theorem 2.3.2.

2.5 Approximation by Wavelets, Shearlets and more

definition 2.5.1 (Affine System)

Let $d \in \mathbb{N}$, $(A_j)_{j\in\mathbb{N}} \subset \mathrm{GL}(\mathbb{R}^d)$, $(\psi_k)_{k=1}^S \subset L^2(\mathbb{R}^d)$ be compactly supported. Then

$$\left\{ |A_j|^{\frac{1}{2}} \psi_s(A_j x - b) : s \in [S], b \in \mathbb{Z}^d, j \in \mathbb{N} \right\}$$

is an affine system.

Remark 2.5.2 The factor $\sqrt{|A_j|}$ ensures all φ_i, where i is an enumeration of the indices in the above set, have equal norm.

Example 2.5.3 Wavelets and shearlet systems are affine. ◇

Theorem 2.5.1: Memory-optimal NNs [30]

Let $\Omega \subset \mathbb{R}^d$ be bounded and $(\varphi_i)_i \subset L^2(\Omega)$ an affine system with $(\psi_s)_{s=1}^S$ defined as before and $\varrho : \mathbb{R} \to \mathbb{R}$ an activation function. Assume there exists $\varepsilon > 0$ such that for all $D > 0$ and s

$$\exists \Phi_{D,\varepsilon} \in \mathcal{NN}_{s,C,2C,L} : \|\psi_s - R_\varrho(\Phi_{D,\varepsilon})\|_{L^2([-D,D]^d)} \leqslant \varepsilon.$$

Let $\mathcal{C} \subset L^2(\mathbb{R}^d)$. If there exist $\varepsilon > 0$, $M \in \mathbb{N}$ and $f \in L^2(\Omega) \cap \mathcal{C}$ such that there exist coefficients $(d_i)_{i=1}^M$ with

$$\left\| f - \sum_{i=1}^{M} d_i \varphi_i \right\| \leqslant \varepsilon,$$

then there exists a deep neural network Φ with $\mathcal{O}(M)$ edges such that $\|f - R_\varrho(\Phi)\| \leqslant 2\varepsilon$.

This produces memory-optimal DNNs.

Proof. We have $\|\psi_s - R_\varrho(\Phi_s)\| \leqslant \varepsilon$ with $M(\Phi_s) \leqslant C$. Since the map $x \mapsto A_j x - b$ is affine linear we have $\left\|\psi_i - R_\varrho(\tilde{\Phi}_i)\right\| \leqslant \varepsilon$ and $M(\tilde{\Phi}_i) \leqslant C$. Look at picture to the right, $M(\Phi) \in \mathcal{O}(M)$.

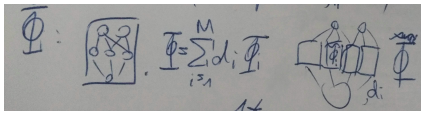


Fig. 23: TODO

$$\|f - R_\varrho(\Phi)\| \overset{\triangle \neq}{\leqslant} \left\|f - \sum_{i=1}^{M} d_i \Phi_i\right\| + \left\|\sum_{i=1}^{M} d_i \Phi_i - R_\varrho(\Phi)\right\| \leqslant 2\varepsilon. \qquad \square$$

Corollary 2.5.4 (Memory optimal NNs and affine systems)
If an affine system $(\varphi_i)_i \subset L^2(\mathbb{R}^d)$ satisfies that

- *for each i there exists a neural network Φ_i with at most $C > 0$ edges such that $\varphi_i = R_\varrho(\Phi_i)$*
- *there exists a $\tilde{C} > 0$ such that for all $f \in \mathcal{C} \subset L^2(\mathbb{R}^d)$ we have $\left\|f - \sum_{i=1}^{M} f_i \varphi_i\right\| \leqslant \tilde{C} M^{-\frac{1}{\gamma^*(\mathcal{C})}}$,*

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

Recall that if a neural network stems from a fixed learning procedure Learn, for all $\gamma < \gamma^*(\mathcal{C})$, there does not exist a $C > 0$ such that

$$\sup_{f \in \mathcal{C}} M(\text{Learn}(\varepsilon, f)) \leqslant C\varepsilon^{-\gamma} \quad \forall \varepsilon > 0.$$

Proof. By the previous theorem there exists a neural network Φ with $R_\varrho(\Phi) = \sum_{i=1}^{M} f_i \varphi_i$ and $M(\varphi) \in \mathcal{O}(M)$. We have

$$\Bigg\| f - \underbrace{\sum_{i=1}^{M} f_i \varphi_i}_{\substack{= R_\varrho(\Phi) \text{ with} \\ M(\Phi) \in \mathcal{O}(M)}} \Bigg\| \leqslant \tilde{C} M^{-\frac{1}{\gamma^*(\mathcal{C})}} =: \varepsilon \iff M \in \mathcal{O}(\varepsilon^{-\gamma^*(\mathcal{C})}) \qquad \square$$

In summary we showed the memory optimal neural networks can be constructed by mimicking N-term approximation.

We now have completed nearly every step of our road map for the section:

1. Find model function class $\mathcal{C} \subset L^2(\mathbb{R}^2)$. $\rightsquigarrow$ cartoon-like functions.
2. Find an associate representation system ($\rightsquigarrow$ shearlets) with the following properties:
   - provides optimally sparse approximations for $\mathcal{C}$ ($\rightsquigarrow$ has been proven)
   - its elements can be realized by a neural network with controlled number of edges. This is still missing!

In order to approximate the ψ_s we construct wavelet generators (LeCun; 1987), (Shaham, Cloninger, Coifman; 2017): Let ϱ be the ReLU and define $t(x) := \varrho(x) - \varrho(x-1) - \varrho(x-2) + \varrho(x-3)$, which can be constructed with a two-layer network. Now $\varphi(x, y) := \varrho(t(x) + t(y) - 1)$ yields a 2D bump function. Summing up shifted versions of φ, which can be realized by just one additional layer yields a function ψ_s with vanishing moments.

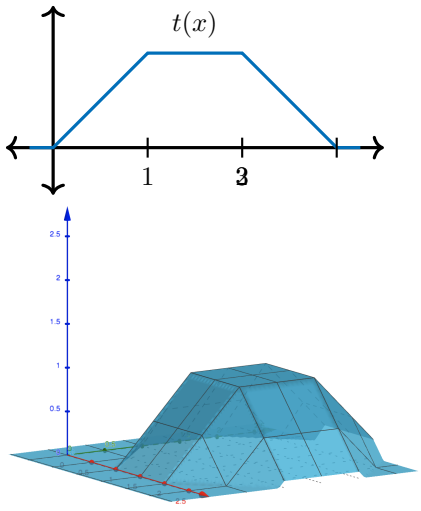


Fig. 24: The functions $\varphi(x, y)$ and

Those ψ will not be differentiable. A smoothed version of a ReLU leads to appropriate shearlet generators.

We will now see that function classes which are optimal representable by shearlets, etc. are also optimally approximated by memory-efficient neural networks with a parallel architecture:

Theorem 2.5.2: Optimal approximation [30]

Let ϱ be an admissable smooth activation function and $\varepsilon > 0$. Then there exists $C_\varepsilon > 0$ such that for all $f \in \mathcal{E}^2(\mathbb{R}^2)$ and $N \in \mathbb{N}$ there exists a $\Phi \in \mathcal{NN}_{2,\mathcal{O}(N),\mathcal{O}(N),3}$ satisfying

$$\|f - R_\varrho(\Phi)\|_{L^2} \leqslant C_\varepsilon N^{-1+\varepsilon}.$$

Remark 2.5.5 This is the optimal rate; hence the first bound is sharp!

Some numerics

Typically weights are learnt by backpropagation. This raises the following question: Does this lead to the optimal sparse connectivity?

In our setup we consider a fixed network ReLU-architecture, learning specific functions with stochastic gradient descent. We analyze the learnt subnetworks to conduct an analysis of the connection between approximation error and number of edges.

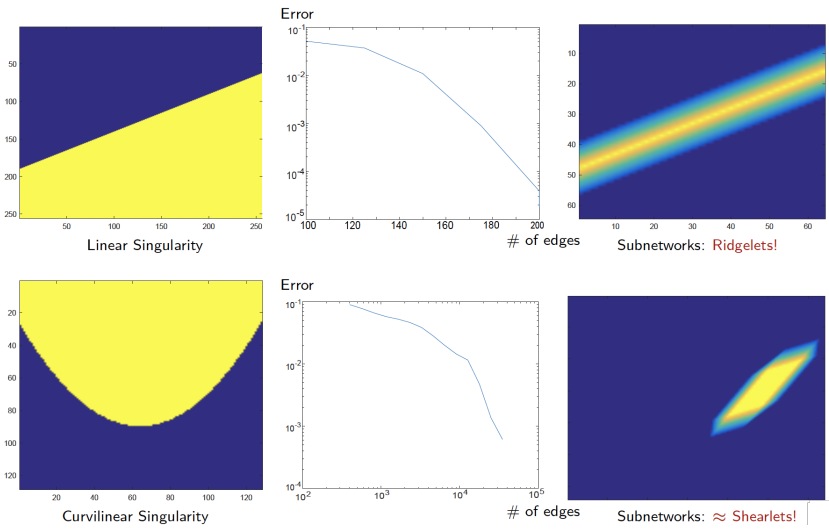


Fig. 25: Numerical experiments with ReLU and backpropagation. [Source??]

Feel free to verify that we have answered all questions posed at the beginning of subsection 2.3.

2.6 Impact of Depth

03.12.19

Our next theorem, in rough terms, states than

"There exists a simple (approximately radial) function on $\mathbb{R}^d$, expressible by a 3-layer neural network of width polynomial in the dimension d, which cannot be arbitrarily well approximated by 2-layer networks, unless their width is exponential in d."

Remark 2.6.1 This result holds for virtually all known activation functions, including rectified linear units, sigmoids and thresholds (i.e $\mathbb{1}_{[0,\infty)}$). It shows that depth - even if increased by 1 - can be exponentially more valuable than width for standard feedforward neural networks.

Assumption 1: The activation function ϱ is measurable and satisfies

$$|\varrho(x)| \leqslant C(1+|x|^\alpha) \quad \forall x \in \mathbb{R},\ C, \alpha > 0.$$

Assumption 2: There exists a constant $c_\varrho \geqslant 1$ such that for any Lipschitz function $f : \mathbb{R} \to \mathbb{R}$ with Lipschitz constant L which is constant outside a bounded interval $[-R, R]$ and for any $\delta > 0$ there exist scalars $a, \alpha_i, \beta_i, \gamma_i$, $i \in [w]$ with $w \leqslant c_\varrho \frac{RL}{\delta}$ and

$$\sup_{x \in \mathbb{R}^d} \left| f(x) - a - \sum_{k=1}^{w} \alpha_k \varrho(\beta_k x - \gamma_k) \right| \leqslant \delta.$$

The second assumption guarantees that f can be approximated by a neural network with one hidden layer.

Theorem 2.6.1: From two to three layers [31]

If ϱ satisfies the two assumptions above then there exist universal constants $c, C > 0$ such that the following holds: For every dimension $d > C$ there exists a probability measure μ on $\mathbb{R}^d$ and a function $g : \mathbb{R}^d \to \mathbb{R}$ with the following properties:

1. g is bounded in $[-2, 2]$, supported on $\{x \in \mathbb{R}^d : \|x\| \leqslant C\sqrt{d}\}$ and expressible by a 3-layer network of width $Cc_\varrho d^{\frac{19}{4}}$.
2. Every function f expressible by a 2-layer network of width at most ce^{cd} satisfies $\mathbb{E}_{x\sim\mu}[(f(x) - g(x))^2] \geqslant c$.

Proof. 3-layer case: Consider radial functions $g(x) = h(\|x\|_2)$, where $h : \mathbb{R} \to \mathbb{R}$. By assumption 2, $x \mapsto x^2$ can be approximated by a linear combination of neurons. Then approximate $\|x\|_2^2 = \sum_{i=1}^d x_i^2$ on any bounded domain. The next layer approximates h. $\rightsquigarrow$ 3 layers.

2-layer case: μ corresponds to a probability density function φ^2. The condition (ii) is equivalent to

$$\int (f(x) - g(x))^2 \varphi^2(x)\, \mathrm{d}x = \|f\varphi - g\varphi\|_2^2 \overset{(\mathrm{P})}{=} \|\widehat{f\varphi} - \widehat{g\varphi}\|_2^2 \geqslant c,$$

where (P) is the Plancherel theorem.

Choose $\varphi := \mathbb{1}_{\overline{B}(0,1)} = \left(\frac{Rd}{\|x\|}\right)^{\frac{d}{2}} J_{\frac{d}{2}}(2\pi Rd\|x\|)$, where J_k is a Bessel function, which is a density function. If f is expressible by a 2-layer network, $\hat{f}$ has a special form: For this, consider

$$f : \mathbb{R} \to \mathbb{R},\ x \mapsto \sum_{i=1}^{k} f_i(\langle v_i, x \rangle).$$

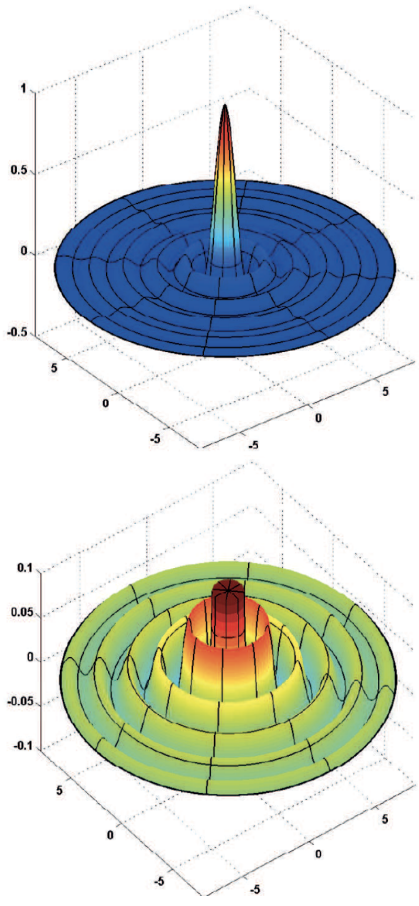


Fig. 26: Illustration of φ for $d = 2$. [2]

Then (using distribution theory) it holds that $\operatorname{supp}(\hat{f}) = \bigcup_{i=1}^{k} \operatorname{span}(v_i)$. Then

$$\widehat{f\varphi} = \hat{f} \circ \hat{\varphi} = \hat{f} \circ \mathbb{1}_{\overline{B}(0,1)}$$

and therefore

$$\operatorname{supp}(\widehat{f\varphi}) = \bigcup_{i=1}^{k} \left(\operatorname{span}(v_i) \cdot \overline{B}(0,1)\right) =: T,$$

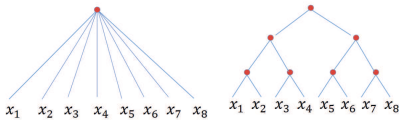
where T is a union of tubes through the origin with width 1. Impact of dimension: Unless k is exponentially large i.i.d., T is very sparse at large distances from the origin, i.e.

$$\frac{\operatorname{Vol}(T \cap S^{d-1})}{\operatorname{Vol}(S^{d-1})} \leqslant k e^{-d}.$$

Goal: Find g, where $g\varphi$ is radial and $g\hat{\varphi}$ has most of its energy away from zero.

Main technical part: Ansatz: $\varphi(x) \cdot \hat{g}(x) = \sum \varepsilon_i c_i g_i$, where g_i are carefully chosen concerning $\hat{g}_i$. □

We will now take a look at compositorial functions and a theorem from [32], which roughly states that "Deep (hierarchical) networks can approximate the class of compositional functions with the same accuracy as shallow networks but with exponentially lower number of (training) parameters."



compositional functions
Fig. 27: Visual intuition for the theorem. [2]

definition 2.6.2 (Compositional function)
A function $f : \mathbb{R}^d \to \mathbb{R}$ following the structure of a binary tree

$$f(x_1, \ldots, x_n) = h_1(h_2(h_3(x_1, x_2), h_4(x_3, x_4)), \ldots)$$

for appropriate functions $h_i : \mathbb{R}^2 \to \mathbb{R}$ is called compositional.

definition 2.6.3 ($\mathcal{W}_{r,d}$, $\mathcal{W}_{H,r,2}$, $\mathcal{S}_n$ and $\mathcal{D}_n$)
Let $\mathcal{W}_{r,d}$ be set of all functions with continuous partial derivatives up to order r such that

$$\sum_{|k| \leqslant r} \|D^k f\| \leqslant 1$$

and $\mathcal{W}_{H,r,2}$ be the class of compositional functions in $\mathcal{W}_{r,d}$.

Let $\mathcal{S}_n := \mathcal{NN}_{d,\infty,d+n+1,2}$ and $\mathcal{D}_n$ the class of DNNs from $\mathcal{NN}_{d,\infty,\infty,\infty}$ whose realizations the structure of a compositional function with the analog of functions h_i being in $\mathcal{S}_n$.

Theorem 2.6.2: Compositional function approximation [32]

Let $\varrho \in \mathcal{C}^\infty(\mathbb{R})$ and not a polynomial on any subinterval of $\mathbb{R}$. For $f \in \mathcal{W}_{r,d}$

$$\inf_{\Phi \in \mathcal{S}_n} \|f - R_\varrho(\Phi)\| = \mathcal{O}(n^{-r/q})$$

and for $f \in \mathcal{W}_{H,r,2}$

$$\inf_{\Phi \in \mathcal{D}_n} \|f - R_\varrho(\Phi)\| = \mathcal{O}(n^{-r/2})$$

hold.

Remark 2.6.4 (Corollary?) If we only know r and want to guarantee an accuracy of ε, we need a shallow network with $\mathcal{O}(\varepsilon^{-q/r})$ trainable parameters.

If we assume a hierarchical structure on the target function, then to guarantee an accuracy of ε, a corresponding deep network requires $\mathcal{O}(\varepsilon^{-2/r})$ trainable parameters.

Because of time constraint we only briefly covered convolutions neural networks, so this section will not be include in here (for now). See slide 49 - 59 from the lecture.

3 Learning

3.1 Training in overparameterized regimes

10.12.19

In the overparameterized regime the number of neurons exceeds the number of samples, enabling zero training error and therefore a globally optimal solution and convergence of SGD in high probability over the initialization.

Consider the following two-layer network: Let ϱ be the ReLU activation function, $(w_r)_{r=1}^n \subset \mathbb{R}^d$ the weight vectors of the first layers, W the associated matrix and $a \in \mathbb{R}^n$ the output weights. **TODO: BILD** Consider the neural network $\Phi = (W, a)$ and its realization

$$R_\varrho(\Phi) : \mathbb{R}^d \to \mathbb{R},\ x \mapsto \frac{1}{\sqrt{n}} \sum_{r=1}^n a_r \varrho(\langle w_r, x \rangle)$$

WO KOMMT DER GEWICHTENDE FAKTOR HER? For samples $((x_k, y_k))_{k=1}^m \subset \mathbb{R}^d \times \mathbb{R}$ we want to minimize the empirical loss function

$$F(W, a) := \sum_{i=1}^m \frac{1}{2} \big(R_\varrho(W, a)(x_i) - y_i\big)^2$$

First, consider

$$W(k+1) = W(k) - \eta \frac{\partial F(W(k), a)}{\partial W(k)}, \tag{7}$$

where $\eta > 0$ is the step size. Here we have

$$\frac{\partial F(W, a)}{\partial w_r} = \frac{1}{\sqrt{n}} \sum_{i=1}^m \big(R_\varrho(W, a)(x_i) - y_i\big) a_r x_i \mathbb{1}_{\{\langle w_r, x_i \rangle \geqslant 0\}}(?),$$

which is neither smooth nor convex.

Instead of gradient descent we use gradient flow, which is gradient descent with infinitesimal step size: Instead of (7) consider the differential equation

gradient flow

$$\frac{\mathrm{d}w_r(t)}{\mathrm{d}t} = -\frac{\partial F(W(t), a)}{\partial w_r(t)}.$$

For brevity's sake let

$$u_i(t) := R_\varrho(W(t), a)(x_i)$$

for $i \in [m]$ be the prediction on input x_i at times t and $u(t) := (u_i(t))_{i=1}^n$ be prediction vector at times t. Further let

$$H^\infty := \mathbb{E}_{w \sim \mathcal{N}(0,1)} \left[\langle x_i, x_j \rangle \mathbb{1}_{\{\langle w, x_i \rangle, \langle x, w_j \rangle \geqslant 0\}}\right]$$

by the Gram-Matrix induced by ϱ and random initialization.

We assume that $\lambda_0 := \lambda_{\min}(H^\infty) > 0$, which is fulfilled if for all $i \neq j$ we have $x_i \nparallel x_j$ ([26], theorem 3.1). Note for most real world datasets, no two inputs are parallel, so our assumption holds in general.

Theorem 3.1.1: Convergence Rate of Gradient Flow ([26], Theorem 3.2)

Suppose the assumption holds and $\|x_i\|_2 = 1$ and $|y_i| \leqslant C$ holds for a constant C and all $i \in [m]$. Choose the number of hidden neurons as

$$n = \Omega\left(\frac{m^6}{\lambda_0^4 \delta^3}\right)$$

and i.i.d. initialize $w_r \sim \mathcal{N}(0,1)$ and $a_r \sim \mathrm{unif}[\{\pm 1\}]$ for $r \in [n]$. Then with probability at least $1-\delta$ over the initialization we have

$$\|u(t) - y\|_2^2 \leqslant \exp(-\lambda_0 t)\|u(0) - y\|_2^2.$$

Proof. (1) **Dynamics of predictions.** We have

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}u_i(t) &= \sum_{r=1}^{n} \left\langle \frac{\partial R_\varrho(W(t),a)(x_i)}{\partial w_r(t)}, \frac{\mathrm{d}w_r(t)}{\mathrm{d}t} \right\rangle \\ &= \ldots = \sum_{j=1}^{m} (y_j - u_j(t)) H_{i,j}(t),\end{aligned}$$

where

$$H_{i,j} = \frac{\langle x_i, x_j \rangle}{n} \sum_{r=1}^{n} \mathbb{1}_{\{\langle x_i, w_r\rangle, \langle x_j, w_r\rangle > 0\}}(r).$$

Thus,

$$\frac{\mathrm{d}}{\mathrm{d}t}u(t) = H(t)(y - u(t)). \tag{8}$$

(2) We have

$$\|H(0) - H^\infty\|_2 = \mathcal{O}\left(\frac{1}{\sqrt{n}}\right) = \|H(t) - H(0)\|_2 \quad \forall t \geqslant 0,$$

i.e. the higher the overparametrization (larger u) the more the dynamics of the predictions are governed by H^∞.

(3) If $n \in \Omega\left(\frac{m^2}{\lambda_0^2}\log\left(\frac{m}{\delta}\right)\right)$ with probability at least $1-\delta$, we have $\|H(0) - H^\infty\|_2 \leqslant \frac{\lambda_0}{4}$ and $\lambda_{\min}(H(0)) \geqslant \frac{3}{4}\lambda_0$.

This is proven with concentration bounds. **TODO: WHATS THAT?**

(4) $H(t)$ is stable in terms of $W(t)$: If w_r are i.i.d. from $\mathcal{N}(0,1)$, then with probability $1-\delta$ we have: for all $(w_k(t))_{k=1}^n \subset \mathbb{R}^d$ with

$$\|w_r(0) - w_r(t)\|_2 \leqslant \frac{\delta\lambda_0}{m^2} =: R \tag{9}$$

for some c and all r. Then

$$\|H(t) - H(0)\|_2 \leqslant \frac{\lambda_0}{4} \quad \text{and} \quad \lambda_{\min}(H(t)) \geqslant \frac{\lambda_0}{2}.$$

This can see be considering the event

$$A_{i,r} := \{\exists w : \|w - w_r(0)\| = R, \mathbb{1}_{\{\langle x_r, w_r(0)\rangle \geqslant 0\}} \neq \mathbb{1}_{\{\langle x_i, w_r(0)\rangle \geqslant 0\}}\}.$$

Then

$$P(A_{i,r}) = P_{z\sim\mathcal{N}(0,1)}(|z| < R) \leqslant \frac{2R}{\sqrt{2\pi}} \tag{10}$$

holds. Then,

$$\mathbb{E}\left[\left|H_{i,j}(0) - H_{i,j}(t)\right|\right] \leqslant \frac{1}{n}\sum_{r=1}^{n} \mathbb{E}\left[\mathbb{1}_{\{A_{i,r}\cap A_{j,r}\}}(r)\right] \overset{(10)}{\leqslant} \frac{4R}{\sqrt{2\pi}}.$$

Summing over $i, j \in [m]$ and applying Markov's inequality we get that with probability of at least $1-\delta$

$$\sum_{i,j=1}^{m} \left|H_{i,j}(t) - H_{i,j}(z)\right| \leqslant \frac{4m^2 R}{\sqrt{2\pi}\delta}.$$

We then use matrix perturbation to get the result:

$$\begin{aligned}\|H(t) - H(0)\|_2 &\leqslant \|H(t) - H(0)\|_F \\ &\leqslant \sum_{i,j=1}^{m} |H_{i,j}(t) - H_{i,j}(z)| \leqslant \frac{4m^2 R}{\sqrt{2\pi}\delta} \overset{(9)}{=} \frac{4\lambda_0}{\sqrt{2\pi}}.\end{aligned}$$

Hence,

$$\lambda_{\min}(H(t)) \geqslant \left(\lambda_{\min}(H(0)) - \frac{4\lambda_0}{\sqrt{2\pi}}\right) \geqslant \frac{\lambda_0}{2}.$$

(5) If $s \in [0,t]$, $\lambda_{\min}(H(s)) \geqslant \frac{\lambda_0}{2}$, then

$$\|y - u(t)\|_2^2 \leqslant e^{-\lambda t}\|y - u(0)\|_2^2 \tag{11}$$

and $\|w_r(t) - w_r(0)\| \leqslant \sqrt{\frac{m}{\pi}}\frac{\|y-u(0)\|_2}{\lambda_0} =: R'$.

The proof of this fact relies on using (8). It gives

$$\frac{\mathrm{d}}{\mathrm{d}t}\|y - u(t)\|_2^2 = -2(y-u(t))^T(y-u(t)) \leqslant -\lambda_0\|y-u(t)\|_2^2,$$

where the inequality comes from $\lambda_{\min}(H(s)) \geqslant \frac{\lambda}{2}$. This implies

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(e^{\lambda_0 t}\|y-u(t)\|_2^2\right) \leqslant 0,$$

so $e^{\lambda_0 t}\|y-u(t)\|_2^2$ is decreasing in t, implying (11).

(6) If $R' < R$ then

- $\lambda_{\min}(H(t)) \geqslant \frac{\lambda_0}{2}$ for all $t \geqslant 0$.
- $\|w_r(t) - w_r(0)\|_2 \leqslant R'$ for all r.
- (11).

We have $R' < R$ if n large enough. □

We now consider the differential equations

$$\frac{\mathrm{d}w_r(t)}{\mathrm{d}t} = -\frac{\partial F(W(t), a(t))}{\partial w_r(t)} \quad \text{and} \quad \frac{\mathrm{d}w_r(t)}{\mathrm{d}t} = -\frac{\partial F(W(t), a(t))}{\partial a_r(t)}$$

Theorem 3.1.2: Jointly training both layers [26]

Let the previous assumptions as above be fulfilled and the number of hidden neurons be $n = \Omega\left(\frac{m^6 \log(m/\delta)}{\lambda_0^4 \delta^3}\right)$. Further initialize $w_r \sim \mathcal{N}(0,1)$ and $a_r \sim \mathrm{unif}[\{\pm 1\}]$ for $r \in [n]$. Then with probability at

least $1-\delta$ over the initialization we have

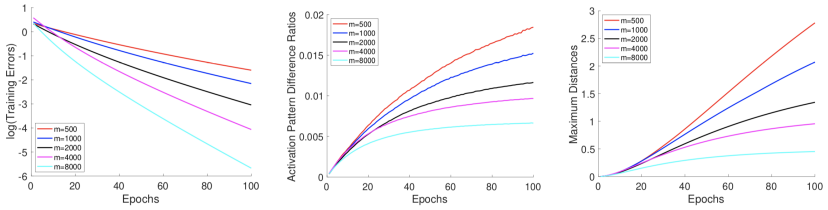
$$\|u(t)-y\|_2^2 \leqslant e^{-\lambda_0 t}\|u(0)-y\|_2^2.$$

The proof uses similar argument as before.

Theorem 3.1.3: Discrete Time Analysis [26]

Let the previous assumptions be fulfilled and the number of hidden neurons be $n = \Omega\left(\frac{m^6}{\lambda_0^4 \delta^3}\right)$. Further initialize as above and set the step size $\eta := \mathcal{O}\left(\frac{\lambda_0}{m^2}\right)$. Then with probability at least $1-\delta$ over the initialization we have

$$\|u(k)-y\|_2^2 \leqslant \left(1-\frac{\eta\lambda_0}{2}\right)^k \|u(0)-y\|_2^2 \qquad \text{for } k \in \mathbb{N}\,.$$



(a) Convergence rates.

(b) Percentiles of pattern changes.

(c) Maximum distances from initialization.

Fig. 28: Some numerics. [26]

A similar result as before holds for deep neural networks. More precisely, the authors show that gradient descent achieves zero training loss in polynomial time for a deep over-parameterized residual neural network (ResNet) [27].

4 Generalization

4.1 Empirical Risk Minimization

Aiming to find an alternative definition of the complexity of a function class we will define its covering number, describing how well it can be approximated with simple objects such as ε-balls.

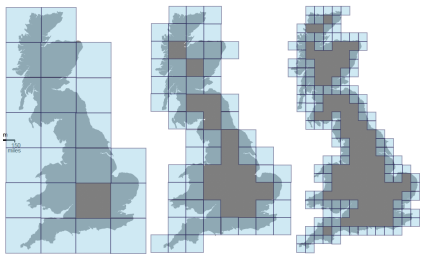


Fig. 29: Illustration of the concept of covering number with ε-balls with respect to the uniform norm. [2]

DEFINITION 4.1.1 (COVERING NUMBER)

Let X be a metric space and $\varepsilon > 0$. We define

$$\mathcal{N}(X, \varepsilon) := \min \left\{ \ell \in \mathbb{N} : \exists (x_k)_{k=1}^{\ell} \subset X : X \subset \bigcup_{k=1}^{\ell} B(x_k, \varepsilon) \right\}$$

DEFINITION 4.1.2 (PACKING NUMBER)

The packing number is

$$\mathcal{P}(X, \varepsilon) := \max\{k \in \mathbb{N} : \exists (x_\ell)_{\ell=1}^{k} : \|x_i - x_j\| > \varepsilon \ \forall i \neq j\}.$$

Lemma 4.1.3

We have

$$\mathcal{P}(X, 2\varepsilon) \leqslant \mathcal{N}(X, \varepsilon) \leqslant \mathcal{P}(X, \varepsilon) \leqslant \left(1 + \frac{2}{\varepsilon}\right)^d$$

THEOREM 4.1.1: GENERALIZATION ERROR BOUND [SOURCE???]

Let $\mathcal{H}$ be a hypothesis class and assume that for all $f \in \mathcal{H}$ we have $|f(X) - Y| \leqslant M$ almost everywhere. The for all $\varepsilon > 0$

$$\mathbb{P}\big(\mathcal{E}(f_{\mathcal{H},S}) - \mathcal{E}(f_{\mathcal{H}}) \leqslant \varepsilon\big) \geqslant 1 - \mathcal{N}\left(\mathcal{H}, \frac{\varepsilon}{8M}\right) \cdot 2\exp\left(-\frac{m\varepsilon^2}{8M^4}\right)$$

holds.

WHAT ARE X and Y???Notice that this bound makes sense as for many samples, i.e. large m as the term $e^{-Cm\varepsilon^2}$ decreases, yielding a better bound, while $\mathcal{N}\left(\mathcal{H}, \frac{\varepsilon}{8M}\right)$ increases with growing M, yielding a smaller and therefore worse lower bound.

Corollary 4.1.4 (Sufficient number)

A generalization error smaller than or equal to $\varepsilon \geqslant 0$ is achieved by

$$m \geqslant \varepsilon^{-2} \ln\big(\mathcal{N}(\mathcal{H}, c\varepsilon)\big)$$

with high probability for some $c > 0$.

Unfortunately, the complexity measure given above is independent of the data distribution and the samples hence very pessimistic.

Motivation of Rademacher complexity

For generalization we need to bound

$$\sup_{f\in\mathcal{H}} \left| \mathbb{E}\left[\ell(f(x), y)\right] - \mathcal{E}_S(f) \right|.$$

In cross-validation you split the samples S into S' and S'' (of approximately the same size) and estimate the generalization error of f by

cross-validation

$$\frac{2}{m}\left(\sum_{(x,y)\in S'} \ell(f(x), y) - \sum_{(x,y)\in S''} \ell(f(x), y) \right) = \frac{2}{m} \sum_{i=1}^{m} \sigma_i \ell(f(x_i), y_i) =: (\star).$$

We denote $\sigma := (\sigma_i)_{i=1}^m$, where

$$\sigma_i = \begin{cases} 1, & \text{if } (x_i, y_i) \in S', \\ -1, & \text{if } (x_i, y_i) \in S'' \end{cases}$$

If we now choose ℓ to be the hinge loss $\ell(f(x), y) := \frac{1 - f(x)\cdot y}{2}$ and without loss of generality the labels to be $y_i \in \{\pm 1\}$ one obtains

hinge loss

$$(\star) = \frac{1}{m} \sum_{i=1}^{m} \sigma_i (1 - f(x_i) y_i).$$

Now consider a random split, so σ_i becomes a random variable. Hence

$$\mathbb{E}_\sigma \left[\sup_{f\in\mathcal{H}} \left| \frac{1}{m} \sum_{i=1}^{m} \sigma_i (1 - f(x_i) y_i) \right| \right] \leqslant \mathcal{O}\left(m^{-\frac{1}{2}}\right) + \underbrace{\mathbb{E}_\sigma \left[\sup_{f\in\mathcal{H}} \frac{1}{m} \sum_{I=1}^{m} \tilde{\sigma}_i f(x_i) \right]}_{=:(\dagger)},$$

where $\tilde{\sigma}_i := y_i \sigma_i$. The term (†) measures how well the hypothesis class is able to fit random labels to the samples.

The $\mathcal{O}\left(m^{-\frac{1}{2}}\right)$ terms follows directly from Van Zuijlen's bound [9]

$$\mathbb{P}\left(\frac{1}{\sqrt{m}} \sum_{i=1}^{m} \sigma_i \leqslant 1 \right) \geqslant \frac{1}{2},$$

where $\mathbb{P}(\sigma_i = 1) = \mathbb{P}(\sigma_i = -1) = \frac{1}{2}$.

definition 4.1.5 (Rademacher complexity)

The Rademacher complexity of a function class $\mathcal{F}$ with respect to the sample $S := (x_i)_{i=1}^m$ is

$$\mathbb{R}_S(\mathcal{F}) := \mathbb{E}_\sigma \left[\sup_{f\in\mathcal{H}} \frac{1}{m} \left| \sum_{i=1}^{m} \sigma_i f(x_i) \right| \right],$$

where the expectation is taken over all Rademacher random variables $\sigma := (\sigma_i)_{i=1}^m \subset \{\pm 1\}^m$ chosen uniformly at random.

The Rademacher complexity can be used to derive data-dependent upper-bounds on the learnability of function classes. Intuitively, a function class with smaller Rademacher complexity is easier to learn.

Statistical learning theory tells us to choose $\mathcal{F} := \mathcal{H}$.

Example 4.1.6 (Linear classifiers' Rademacher complexity)
Consider the hypothesis space of unbiased linear classifiers

$$\mathcal{F}_{a,b} := \{\mathbb{R}^d \supset \{y : \|y\|_2 \leqslant a\} \ni x \mapsto \langle v, x \rangle : \|v\|_2 \leqslant b\}.$$

One obtains

$$\begin{aligned}
\mathbb{R}_S(\mathcal{F}_{a,b}) &= \mathbb{E}_\sigma\left[\sup_{\|v\|_2\leqslant b}\frac{1}{m}\left|\sum_{i=1}^m \langle v, \sigma_i x_i\rangle\right|\right] \overset{\triangle\neq}{\underset{\text{(CS)}}{\leqslant}} \frac{b}{m}\,\mathbb{E}_\sigma\left[\left\|\sum_{i=1}^m \sigma_i x_i\right\|_2\right] \\
&\overset{\text{(L)}}{=} \frac{b}{m}\,\mathbb{E}_\sigma\left[\sqrt{\sum_{i,j=1}^m \sigma_i\sigma_j\langle x_i, x_j\rangle}\right] && \|\cdot\| = \sqrt{\langle\cdot,\cdot\rangle} \\
&\overset{\text{(J)}}{\leqslant} \frac{b}{m}\sqrt{\sum_{i,j=1}^m \mathbb{E}_\sigma[\sigma_i\sigma_j]\langle x_i, x_j\rangle} \\
&\leqslant \frac{b}{m}\sqrt{\mathbb{E}_\sigma\left[\sum_{i=1}^m \|x_i\|_2^2\right]} \leqslant \frac{ba}{\sqrt{m}}, && \mathbb{E}[\sigma_i\sigma_j] = 0 \text{ if } i\neq j
\end{aligned}$$

where (L) is the linearity of the inner product on $\mathbb{R}^d$ and (J) is Jensen's inequality $\mathbb{E}[f(X)] \leqslant f(\mathbb{E}[X])$ for concave f. ◇

Example 4.1.7 Consider the hypothesis space of unbiased neural networks

$$\mathcal{F}_{L,B_1,\dots,B_L} := \{\{y : \|y\|_2 \leqslant 1\} \ni x \mapsto A_L\varrho\left(A_{L-1}\dots\varrho(A_1 x)\right)\}$$

with $\|A_i\|_F \leqslant B_i$ for all $i \in [L]$. A theorem [11] states

$$\mathcal{R}_S(\mathcal{F}_{L,B_1,\dots,B_L}) \leqslant m^{-\frac{1}{2}}\left(\sqrt{2\log(2)}d + 1\right)\prod_{i=1}^L B_i. \qquad \diamond$$

Theorem 4.1.2: Generalization Bound [12]

Let $\mathcal{F} \subset \{f : \mathcal{X} \to [0,1]\}$ be a function class, X is a random variable on $\mathcal{X}$ and $S := (x_i)_{i=1}^m$ be i.i.d. drawn from $\mathcal{X}$ according to X. Then with possibility at least $1-\delta$ for all $f \in \mathcal{F}$ we have

$$\sup_{f\in\mathcal{F}}\left|\mathbb{E}[f] - \frac{1}{m}\sum_{i=1}^m f(x_i)\right| \leqslant 2\,\mathcal{R}_S(\mathcal{F}) + \mathcal{O}\left(\sqrt{\frac{\ln(\delta^{-1})}{m}}\right).$$

Notice that if we choose $f = \ell(h(x), y)$ the left term becomes $\mathbb{E}[f] - \mathcal{E}_S(f)$.

Understanding deep learning requires rethinking generalization: the following table from [13] shows that state of the art networks can fit random labels. Therefore, their Rademacher complexity is 1. The classical theory (left diagram) in the diagram below does not explain generalization!

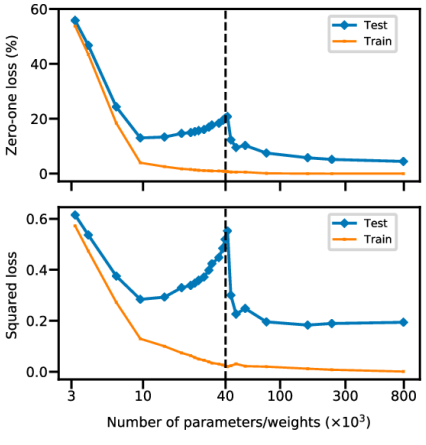
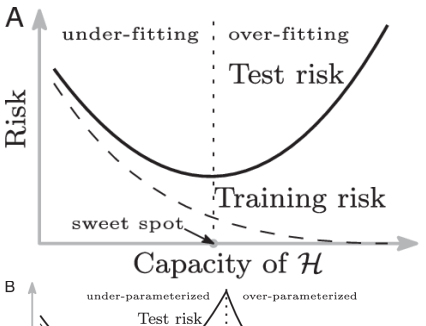


Fig. 30: Sometimes a "double descent curve" is observed. [14]



model	# params	random crop	weight decay	train accuracy	test accuracy
Inception	1,649,402	yes	yes	100.0	89.05
		yes	no	100.0	89.31
		no	yes	100.0	86.03
		no	no	100.0	85.75
(fitting random labels)		no	no	100.0	9.78
Inception w/o BatchNorm	1,649,402	no	yes	100.0	83.00
		no	no	100.0	82.00
(fitting random labels)		no	no	100.0	10.12
Alexnet	1,387,786	yes	yes	99.90	81.22
		yes	no	99.82	79.66
		no	yes	100.0	77.36
		no	no	100.0	76.07
(fitting random labels)		no	no	99.82	9.86
MLP 3x512	1,735,178	no	yes	100.0	53.35
		no	no	100.0	52.39
(fitting random labels)		no	no	100.0	10.48
MLP 1x512	1,209,866	no	yes	99.80	50.39
		no	no	100.0	50.51
(fitting random labels)		no	no	99.34	10.61

Fig. 32: The training and test accuracy of various model on the CIFAR10 dataset. Performance with and without data augmentation and weights decay are compared. The results of fitting random labels are also included. [13, 14]

5 Interpretability

07.01

Motivation. Given a trained neural network, we don't know what the training data was nor how it was trained. The goal of this section is to provide tools to open this black box.

Remark 5.0.1 Even if we know the training data and training procedure, due to the lack of a comprehensive theory, we don't have this information.

Example 5.0.2 Often, job applications are prescreened by neural networks. If one is rejected, one is interested in the reason for this decision.

In hospitals, neural networks are used to suggest treatments. Certainly patients and doctors alike are interested in the reasoning behind them.$\diamond$

Therefore, our goal is to derive human-like explanations from a neural network, even though defining "human-like" is tricky.

Interpretability of a Classification Decision

The approach that probably comes to mind first to investigate which features are most relevant for the decision. This can be achieved by treating every pixel of an image separately (the current standard). It would obviously be desirable to also consider combinations of pixels in order to incorporate additional knowledge about the data.

One is also interested in the certainty of the decision.

As seen on the right one can construct relevance maps. But this doesn't have to provide sufficient knowledge about a network decision.

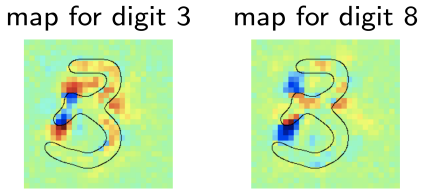


Fig. 33: Red regions indicate pixel relevant for deciding this image to be the number displayed above, while blue pixels indicate regions relevant for deciding against this number.

Previous relevance mapping methods include gradient based methods such as Sensitivity Analysis [17] and SmoothGrad [16], backwards propagation based methods such as Guided Backprop [18], Layer-wise Relevance Propagation [19] and Deep Taylor [20], surrogate model based methods such as LIME (Local Interpretable Model-agnostic Explanations) [22] and game theoretic methods such as Shapley values [23, 24] and SHAP (Shapley Additive Explanations) [34].

Sensitivity Analysis

Let $\mathcal{C} := (c_k)_{k=1}^n$ be a set of classes. Consider a DNN with realization

$$R_\varrho(\Phi) : \mathbb{R}^d \to \mathbb{R}^n, \ x \mapsto (x_{c_k})_{k=1}^n,$$

where $x_{c_i} \in \mathbb{R}$ describes to which extend x belongs to class c_i.

Define $(R_\varrho(\Phi))_{c_i} := x_{c_i}$ and assume this map is piecewise differentiable.

DEFINITION 5.0.3 (SENSITIVITY MAP)
With the above notation, let

$$M_\mathcal{C}(x) := \frac{\partial R_\varrho(\Phi)_{c_i}(x)}{\partial x}$$

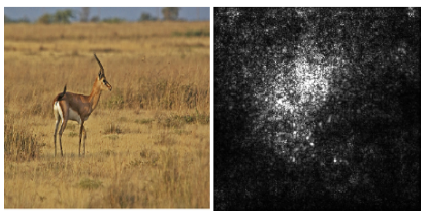


Fig. 34: An image and its sensitivity map M_c. Here $|C| = 1$. The light regions indicate where the algorithm identifies a lot of gazelle ($= c_1$) and the dark regions where there is little of it: Lighter pixels indicate partial derivatives with higher absolute values. [16]

Remark 5.0.4 $M_{\mathcal{C}}$ represents how much a small change in each pixel of x makes to the classification score x_{c_i} for the classes from $\mathcal{C}$.

The problems in the sensitivity map in figure Fig. 34 are evident:

- it looks more like noise because of its scattered nature,
- it is very far from a human explanation; correlation with regions from the image picked by a human are rough at best.

This might be because the partial derivative fluctuate greatly (Fig. 35). A solution is local smoothing [16]: Instead of M_c we now use

$$\widehat{M}_c(x) := \frac{1}{n}\sum_{i=1}^{n} M_c(x_i) + \mathcal{N}(0, \sigma^2),$$

where the x_i are from a neighborhood from x.

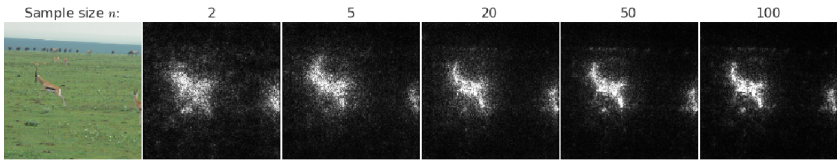


Fig. 36: Effect of sample size on the estimated gradient for inception. 10% noise was applied to each image. [16]

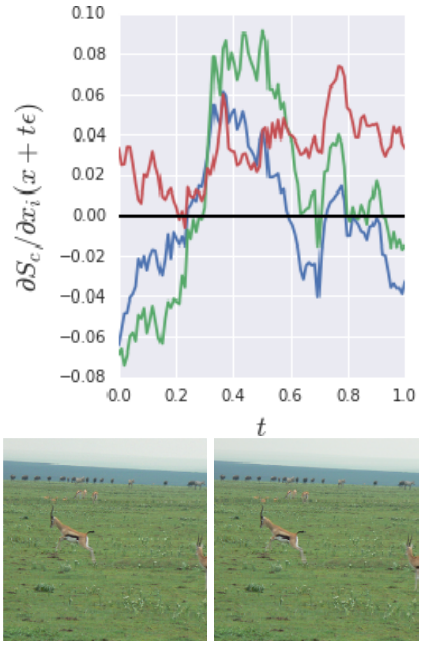


Fig. 35: The partial derivative of M_c with respect to the RGB values of a single pixel as a fraction of the maximum entry in the gradient vector, $\max_i \frac{\partial M_c}{\partial x_i}(t)$, (top) as one moves away from a baseline image x (left) to a fixed location x^+ (right) is one random sample from $\mathcal{N}(0, 0.012)$. The final image is indistinguishable to a human from the origin image. [16]

Layerwise- Relevance Propagation (LRP)

A realization of a neural network takes in an image x and returns $R_\varrho(\Phi)(x)$. Denote the neurons of the ℓ-th layer by $(R_{\ell,j})_{j=1}^{N_\ell}$. Then

$$R_{\ell,j} = \sum_{k=1}^{N_{\ell+1}} \frac{a_{\ell,j}(w_{\ell,j})_k}{\sum_n a_{\ell,n}(w_{\ell,n})_j} R_{\ell+1,k},$$

where w are the weights and the a are constants tuned in the algorithm. In other words: a neurons (and therefore its relevance to the output) can be reconstructed by the neurons of the following layer and the weights.

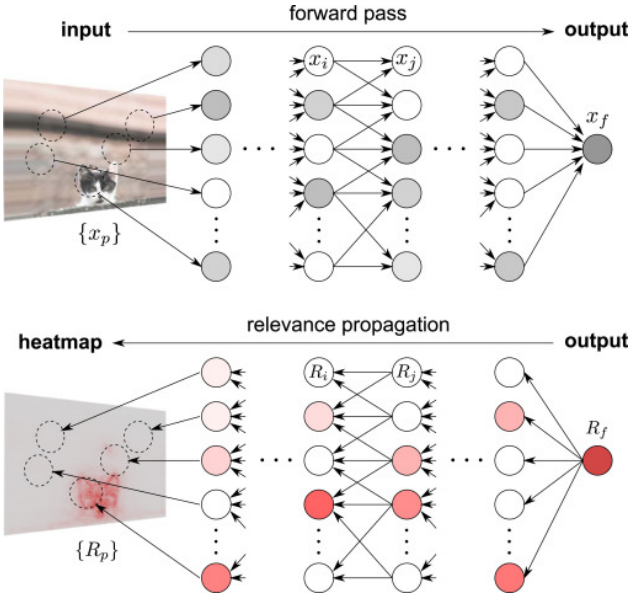


Fig. 37: Visualization of backpropagation. [21]

Deep Taylor

For a realization $R_\varrho(\Phi) : \mathbb{R}^d \to \mathbb{R}$ we want to determine the relevance of each pixel x_k of $x = (x_k)_{k=1}^d$ for the output $R_\varrho(\Phi)(x)$.

definition 5.0.5 (Relevance maps (RM))
A collection of functions $M_p : \mathbb{R}^d \to \mathbb{R}$ is called relevance maps of $(x_k)_{k=1}^d$ if

(1) $(M_p)_p$ is non-negative, i.e. $M_p \geqslant 0$,

(2) $(M_p)_p$ is conservative, i.e if $---\sum_p M_p(x) = R_\varrho(\Phi)(x)$.

Remark 5.0.6 In sensitivity analysis we have $M_p(x) = \frac{\mathrm{d}}{\mathrm{d}x} R_\varrho(\Phi)(x)$, which doesn't satisfy (1) nor (2). **WHAT ABOUT LRP???**

The following definition comes from [21]:

definition 5.0.7 (Taylor decomposition RM)
Assume $R_\varrho(\Phi) \in \mathcal{C}^1$ and $R_\varrho(\Phi)(\hat{x}) = 0$ for some $\hat{x}$. Then

$$M_p(x) := \frac{\partial}{\partial x_p} R_\varrho(\Phi)(\hat{x})(x_p - \hat{x}_p)$$

is the Taylor decomposition relevance map.

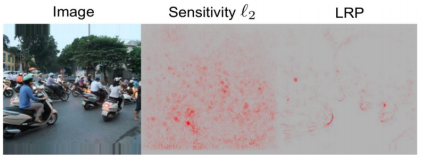


Fig. 38: Numerical experiment for sensitivity vs. LRP. **SOURCE???**

Remark 5.0.8 Consider the standard Taylor expansion

$$R_\varrho(\Phi)(x) = \underbrace{R_\varrho(\Phi)(\hat{x})}_{=0} + \nabla R_\varrho(\Phi)(\hat{x})^T(x - \hat{x}) + \underbrace{\mathcal{O}(\|x - \hat{x}\|^2)}_{\text{can be } <0},$$

which is almost conservative.

The Taylor decomposition relevance map can be efficiently computer by backpropagation.

Fehlt: Bild: numerical experiments I, II (Folie 11/12) Note to Slide 11: the black lines are artificial and aren't created by the algorithm.

5.1 Towards a more mathematical understanding

Our main goal is to understand decisions of "black-box" predictors. We face the following challenges:

- What exactly is relevance in a mathematical sense?
  - Rigorous definition of relevance by information theory.
- What is a good relevance map?
  - ⇝ Formulation of interpretability as optimization problem.
  - ⇝ Theoretical analysis of complexity.
- How to compare different relevance maps?
  - ⇝ Canonical framework for comparison.

The relevance mapping problem

Let $\Phi : [0,1]^d \to [0,1]$ be a classification function, where high values indicate high relevance and $x \in [0,1]^d$ be an input signal.

We want to to determine the most relevant components of x for the prediction $\Phi(x)$. We choose the relevant components $S \subset [d]$, which should be small. We consider $S^{\complement}$ to be non-relevant. Hence the only two relevance values are 0 (not relevant) and 1 (relevant).

Original image x Relevant components S Non-relevant components S^c

Fig. 39: TODO. [2]

Rate-Distortion explanation (RDE)

Let say Alice has a message x a neural network Φ with its realisation $R_\varrho(\Phi)$. She now transmits the sparse relevant part of x, S, to Bob, wo also posses Φ. In order to apply it to S, the "missing" parts have to be filled as well, which is done by random noise η. We call Bob's whole message y and aim for $R_\varrho(\Phi)(y) \approx R_\varrho(\Phi)(x)$.

definition 5.1.1 (Rate-Distortion function)

The distortion is

$$\mathbb{E}\left[\frac{1}{2}\big(R_\varrho(\Phi)(y) - R_\varrho(\Phi)(x)\big)\right]$$

and the rate-distortion function

$$R(\varepsilon) := \min_{S\subset[d]} \{|S| : D(S) \leqslant \varepsilon\}. \qquad (|S| \text{ rate})$$

We call the S part of x ε-relevant.

Problem relaxation

??? \ setting	discrete	continuous
relevant set	$S \subset [d]$	$s \in [0,1]^d$
variable	$y = x_s + \eta_{S^c}$	$y = s \odot x + (1-s) \odot \eta$
distortion	$D(S)$	$D(S)$
rate	$\lvert S\rvert$	$\lVert s\rVert_{\ell_1}$

Here, $\odot$ denotes entry wise multiplication.

Relevance problem $\min_{s\in[0,1]^d} D(s) + \lambda\lVert s\rVert_{\ell_1}$, where $\lambda > 0$ is a regularization parameter.

Finding a minimizer of $R(\varepsilon)$ or even approximating it is very hard.

definition 5.1.2 (Distortion, Obfuscation)

We call

$$D(s) := \mathbb{E}\left[\frac{1}{2}\big(\Phi(x) - \Phi(y)\big)^2\right] = \frac{1}{2}\big(\Phi(x) - \mathcal{E}[\Phi(y)]\big)^2 + \frac{1}{2}\operatorname{cov}[\Phi(y)]$$

distortion.

$$\mathbb{E}[y] = s \odot x + (1-s) \odot \mathbb{E}[n], \quad \operatorname{cov}[y] = \operatorname{diag}(1-s)\operatorname{cov}[n]\operatorname{diag}(1-s)$$

????

We have $\mathbb{E}[y], \operatorname{cov}[y] \xrightarrow{\Phi} \mathbb{E}[\Phi(y)], \operatorname{cov}[\varphi(y)]$.

The generic approach is to estimate using sample mean and sample covariance, which is possible for any classifier function Φ but might require large number of samples.

The neural network approach uses the compositional structure of Φ, propagating the distribution though the layers, projecting to a simple family of distributions at each layer.

NOTE: low rank and diagonal are density filtering methods

Hardness results

We consider the binary case. The following theorems [35] show that find a minimizer of $R(\varepsilon)$ and even the approximation of it is hard:

Theorem 5.1.1: Hardness result

Let Φ and x be given. For $k \in \{1, \dots, d\}$ and $\varepsilon < \frac{1}{4}$, deciding where $R(\varepsilon) \leqslant k$ is $\mathrm{NP}^{\mathrm{PP}}$-complete.
For $\alpha \in (0, 1)$, approximating $R(\varepsilon)$ to within a factor of $d^{1-\alpha}$ is NP-hard.

Many important problems in artificial intelligence belong to the class $\mathrm{NP}^{\mathrm{PP}}$, i.e planning under uncertainties, finding maximum a-posteriori configurations in graphical models or maximizing utility functions in Bayesian networks.

Whereas the class NP can be thought of as the set of problems solvable by guessing the answer and checking it in polynomial time, the class NP^{PP} can be thought of as the set of problems solvable by guessing the answer and checking it using a probabilistic polynomial time (PP) computation. [36]

TODO: numerics

6 Inverse Problems

When solving an inverse problem, one tries to recover the original data from a transformed version!

Example 6.0.1 (Inverse problems) In inpainting one tries to recovery from incomplete images. In medicine, magnetic resonance imaging recovers from point samples of the FOURIER transform. When extracting features, one separates the image into different features. ◇

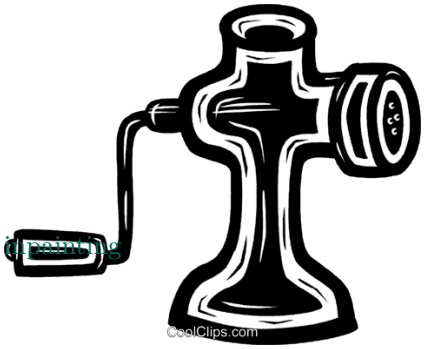


Fig. 40: Recovering meat from minced meat is a difficult inverse problem.

DEFINITION 6.0.2 (INVERSE PROBLEM)
Let X, Y be BANACH spaces and $K : X \to Y$ be linear. Given $y \in Y$ seeking $x \in X$ fulfilling $Kx = y$ is called an inverse problem.

Example 6.0.3 (Inverse problems II)
In inpainting we have

$$K : L^2([0,1]^2) \to L^2([0,1]^2),\ f \mapsto f \cdot \mathbb{1}_\Omega$$

for $\Omega \subset [0,1]^2$. In magnetic resonance imaging we have

$$K : L^2([0,1]^2) \cap L^1([0,1])^2 \to L^2([0,1]^2),\ f \mapsto (\hat{f}(\lambda))_{\lambda \in \Lambda},$$

where $\Lambda \subset [0,1]^2$ is discrete. for feature extraction we have

$$K : L^2([0,1]^2) \times L^2([0,1]^2) \to L^2([0,1]^2),\ (f,g) \mapsto f + g. \qquad \diamond$$

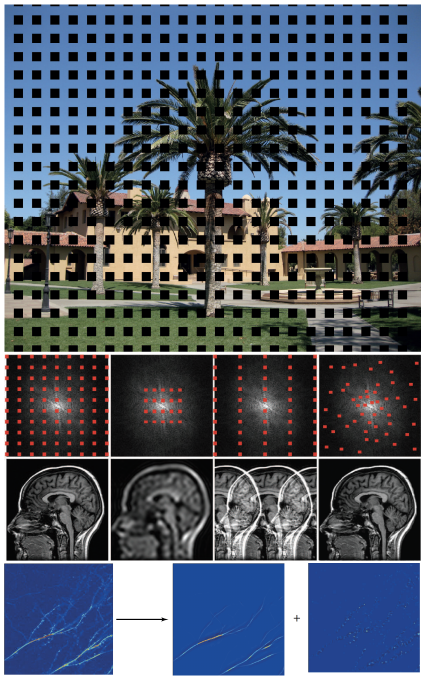


Fig. 41: Visualization of inpainting, MRI and feature extraction. [37, 38]

Example 6.0.4 (Limited Angle-(Computed) Tomography)
A CT scanner (applied in dental CT, breast tomosynthesis and electron tomography) samples the RADON transform

$$\mathcal{R} f(\varphi, s) := \int_{L(\varphi,s)} f(x)\, \mathrm{d}S(x),$$

where $L(\varphi, s) := \{x \in \mathbb{R}^2 : \langle x \cos(\varphi) + x_2 \sin(\varphi) = s\}$, $\varphi \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ and $s \in \mathbb{R}$.

This becomes an challenging inverse problem if $\mathcal{R} f(\cdot, s)$ is only sampled on $[-\varphi, \varphi] \subsetneq \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$. ◇

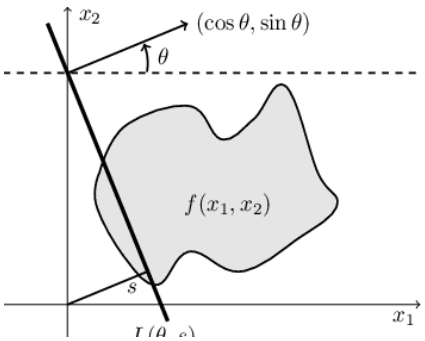


Fig. 42: Geometric setup of the RADON transform. [3]

The wellposedess conditions by are HADAMARD

1. Existence: For all $y \in Y$ there exists a $x \in X$ such that $Kx = y$.
2. Uniqueness: The above x is unique.
3. Stability: $x_n \to x$ follows from $Kx_n \to Kx$.

Most problems are not well-posed. They are called ill-posed. They can have the following problems:

1. $y \notin R(T)$, meaning there does not exists a solution.

(2) $N(T) \neq \{0\}$, meaning there exists infinitely many solutions.

(3) K^+ (as defined below) is not continuous.

The solution for (1) is trying to minimize $\|Kx - y\|$ and the solution to (2) is minimizing $\|x\|$ also:

DEFINITION 6.0.5 (TYPES OF SOLUTIONS)
We call $x \in X$ least square solution if $\|Kx - y\| = \min_{z \in X} \|Kz - y\|$ holds and minimum norm solution if also $\|x\| = \min_{z \in X} \|z\|$ holds.

THEOREM 6.0.1: MOORE-PENROSE-INVERSE
There exists a $K^+ : Y \supset D(K^+) := R(K) \oplus R(K)^\perp \to X$ such that for $y \in D(K^+)$ the equation $Kx = y$ has a minimum norm solution $x^+ = K^+ y$.

We can solve problem (3) with

DEFINITION 6.0.6 (REGULARIZATION)
A family $(J_\alpha : Y \to X)_{\alpha > 0}$ of linear continuous operators is called Regularization of K^+ if $R_\alpha \xrightarrow{\alpha \searrow 0} K^+$ holds pointwise in $D(K^+)$.

Regularization

DEFINITION 6.0.7 (TIKHONOV REGULARIZATION)
The TIKHONOV regularization is defined by

$$J_\alpha(y) := \min_{x \in X} \|Kx - y\|^2 + \alpha \mathcal{R}(x)$$

and is called standard if $\mathcal{R} = \|\cdot\|^2$.

TIKHONOV regularization

This allows use to find an approximate solution $x^{(\alpha)} \in X$, $\alpha > 0$ of an ill-posed inverse problem $Kx = y$ by minimizing J_α. The regularization term $\mathcal{R}$ ensures continuous dependence on the data and incorporate properties of the solution.

But how do we obtain prior information and how do we incorporate it in the solver? There are several types of approaches: take-from-shelf, handcrafting or learning.

Example 6.0.8 (Regularization term)
For vectors: the ℓ_2 or ℓ_1- (sparsity) norm. For functions there is the TV-norm of $f : [0,1] \to \mathbb{R}$:

$$\|f\|_{\mathrm{TV}} := \sup_k \sum_{i=0}^{n_k - 1} |f(x_{i+1} - f(x_i)|$$

or the SOBOLEV norm of $f : \Omega \to \mathbb{R}$:

$$\|f\|_{\mathcal{W}^{k,p}(\Omega)} := \left(\sum_{|\alpha| \leqslant k} \|D^\alpha f\|^p_{L^p(\Omega)} \right)^{\frac{1}{p}}. \qquad \diamond$$

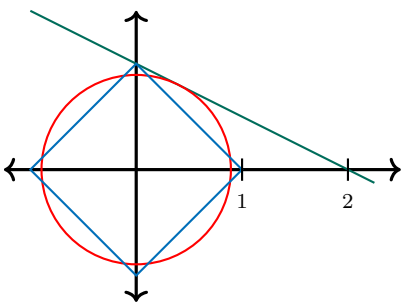


Fig. 43: The point on the green line closest to the origin differs when considering the ℓ_1-or the ℓ_2-norm, whose balls are drawn. One can see the ℓ_1-minimization "pushes points towards the axis".

Example 6.0.9 (Sparse Regularization) Recall the wavelet transform (2). For each class of data, there exists a sparsifying system, which is a **TODO????**. From figure ?? we see than to attain sparseness we want to use the ℓ_1 norm: we solve the ill-posed inverse problem $Kf = g$ by

$$f^\alpha := \arg\min_f \|Kf - g\|^2 + \alpha \cdot \|(\langle f, \psi_{j,m}\rangle)_{j,m}\|_1$$

We can find such a system by using an existing one such as from applied harmonic analysis: Wavelets, Ridgelets, Curvelets, Shearlets, etc. *or* construct a novel system based on information about the data *or* learn a system by an dictionary learning algorithm (cf. K-SVD [39]). ◇

Solving inverse problems with deep learning

A first approach from 2004 [40] trained a feedforward neural network $\mathcal{NN}_\theta : \mathbb{R}^m \to \mathbb{R}^n$ such that $\mathcal{NN}_\theta(y) \approx x$ held. This was done one CPU with out CNNs. Therefore, only small toy-examples could be used. This was seen more as a proof of concept. **TODO BIlDER.**

Later another approach [41] based on ISTA (see below) used unroll/unfold iterations and casts as a "CNN-like" neural network.

In 2012 the fist CNN `AlexNet` wins a classification challenge [42]. Since then all winners are CNN-based by a huge margin.

In the field of denoising (which is an inverse problem) there have been similar improvements (Jain et al.; 2009, Zhang et al.; 2017, Elad et al.; 2016).

Newer approaches work like this: Given training samples $(f_i, g_i)_{i=1}^N$ following the forward model $g_i = Kf_i + \eta$ determine a reconstruction operator T_θ such that $g = Kf + \eta$ implies that $T_\theta(g) \approx f$. Then, T_θ is parametrized by $\theta \in \mathbb{R}^p$ and learned from the training data. We can the evaluate the quality of T_θ by testing on the test data $(f_i, g_i)_{i=N+1}^\ell$ following the forward model.

For CT, we can use reconstruction by filtered backprojection (FBP):

$$f(x) \approx \int_0^\pi (Rf(\varphi, \cdot) * h)(\langle x, n_\varphi \rangle)\, \mathrm{d}\varphi,$$

where $\hat{h}(k) = |k|$.

A typical deep learning approaches to inverse problems is denoising direct inversion (Ye et al.; 2016, Unser et. al.; 2017). Here, one does the direct inversion with FBPs and then trains the CNN to remove noise, the intuition being that the CNN learns structured noise/artifacts. Without taking the FBP, the CNN would need to learn the physics of CT, which is more complicated. **(TODO BILD)**

Theorem 6.0.2

Let $L_0 := L(\varphi_0, s_0)$ be a line in the plane. Let $(x_0, \xi_0) \in WF(f)$ such that $x_0 \in L_0$ and ξ_0 is a normal vector to L_0.

- The singularity of f at (x_0, ξ_0) causes a unique singularity

in Rf at (φ_0, s_0).
- Singularities of f **TODO**

ISTA (Iterative soft-thresholding algorithm) [25]

21.01.2019

Solve

$$\min_{x\in\mathbb{R}^n} \|Ax-y\|^2 + \alpha\|x\|_{\ell^1},$$

where $A : \mathbb{R}^n \to \mathbb{R}^m$ is the discrete version of K by the iteration

$$x_{k+1} := S_{\alpha t}(x_k - 2tA^T(Ax_k - y)),$$

where $t \in \mathbb{R}$ is the step size and

$$S_\lambda : \mathbb{R}^n \to \mathbb{R}^n,\ (S_\lambda(x))_i := (|x_i| - \lambda)_+ \cdot \operatorname{sgn}(x_i)$$

the shrinkage operator.

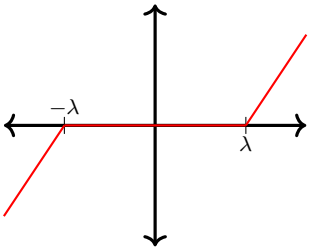


Fig. 44: The shrinkage operator for $\lambda = \frac{3}{2}$.

Douglas-Rachford

Define $\gamma > 0$ and the proximal operator

$$\operatorname{prox}_f(v) := \arg\min_z f(z) + \frac{1}{2}\|z-v\|^2$$

in order to solve

$$\min_{x\in\mathbb{R}^d} \|Ax-y\|^2 + \alpha\mathcal{R}(x)$$

by the "splitting" iteration

$$x_{k+1} := \operatorname{prox}_{\gamma\alpha\mathcal{R}}(v_k), \qquad v_{k+1} := v_k + \operatorname{prox}_{\gamma\|A\cdot-y\|^2}(2x_{k+1} - v_k) - x_{k+1}.$$

If $\mathcal{R} := \|\cdot\|^2_{\ell^1}$, the x-update is soft-thresholding.

TODO

Wavefront sets

A point $x \in \mathbb{R}^2$ is called singular point for f if there exists a $\varphi \in \mathcal{C}_0^\infty(\mathcal{U}_x)$, where $\mathcal{U}_x$ is a neighborhood of x and $\varphi(x) \neq 0$ such that $\widehat{f\varphi} \notin \mathcal{C}_0^\infty(\mathbb{R}^2)$.

singular point

A pair (x, θ) is called regular directed point if there exists neighborhoods of $\mathcal{U}_x$ and V_θ or x and θ, respectively and a $\gamma \in \mathcal{C}_0^\infty(\mathbb{R}^2)$ such that $\gamma|_{\mathcal{U}_x} \equiv 1$ and

regular directed point

$$\forall N\ \exists C_N > 0 : |(\widehat{f\gamma})(\xi)| \leqslant C_N(1+|\xi|)^{-N}\ \forall \xi \in \mathbb{R}^2, \frac{\xi_2}{\xi_1} \in V_\theta$$

The wavefront set consists of all point which are not regular directed points. (rapid decay of fourier transform $\implies$ function smooth in that direction)

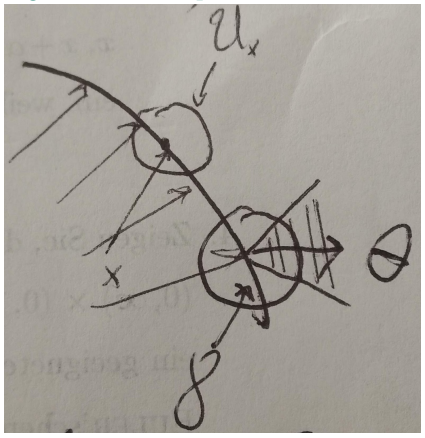


Fig. 45: TODO

15/16 Example $K : X \to Y$ is inpainting. $(x_i, Kx_i)_i$ by fitting $(Kx_i, x_i)_i$.

18 η is noise

19 Backprojection should be $\approx$ also no details were explained.

7 Partial differential equations

06.02.2020

Partial Differential Equations arise in basically any branch of science and engineering. They are key for modeling numerous types of (natural) phenomena such as sound, heat, diffusion, electrostatics, fluid dynamics, elasticity and gravitation.

7.1 Mathematics of PDEs

DEFINITION 7.1.1 ((PARTIAL) DIFFERENTIAL EQUATION)
Let $x \in \Omega \subset \mathbb{R}^d$. Then

$$\mathcal{L}\left(x, u(x), \frac{\partial u(x)}{\partial x_1}, \dots, \frac{\partial u(x)}{\partial x_d}, \frac{\partial^2 u(x)}{\partial x_1^2}, \dots\right) = 0 \tag{12}$$

is a partial differential equation. Given a function $\mathcal{L}$ we want to find a solution $u : \Omega \to \mathbb{R}$.

Example 7.1.2 (Ordinary differential equation) For $d = 1$ (12) is called ordinary and could be given by

$$0 = \mathcal{L}(u(x), u'(x)) := u(x) - u'(x).$$

For $\Omega := \mathbb{R}$ the solutions are given by $u(x) := ce^x$ for $c \in \mathbb{R}$. ◇

Example 7.1.3 (PDEs: POISSON, Heat, Wave) The partial differential equation

$$\Delta u(x) = \sum_{i=1}^{d} \frac{\partial^2 u(x)}{\partial x_i^2} = f(x)$$

is called the POISSON-equation, POISSON-equation

$$\frac{\partial u(x,t)}{\partial t} - \alpha \Delta_x u(x,t) = 0$$

is called the heat equation and heat equation

$$\frac{\partial^2 u(x,t)}{\partial t^2} - \Delta_x u(x,t) = 0$$

the wave equation, where $t \in [0, T]$ and $\alpha \in \mathbb{R}$. wave equation

They represent the three different types of PDEs (in that order):

- elliptic PDEs usually model time-independent (=stationary) problems and (their solutions) describe a state of minimal energy. elliptic
- parabolic PDEs are non-stationary elliptic equations. parabolic
- hyperbolic PDEs typically describe waves and their propagation. ◇ hyperbolic

Remark 7.1.4 (Motivation) Does there exist a solution? Consider $\mathcal{L}(x, u(x), u'(x)) = f$. If we aim for $u \in \mathcal{C}^2(\mathbb{R})$, then we need $f \in \mathcal{C}(\mathbb{R})$.

In most applications those regularity assumptions are unrealistic. In order to relax those assumptions we introduce the concept of weak solutions. weak solutions

From now on let $\Omega \subset \mathbb{R}^d$ be open and bounded.

Example 7.1.5 (Poisson's equation) For $f : \Omega \to \mathbb{R}$ consider

$$(P) \begin{cases} \Delta u = f & \text{on } \Omega \\ u = 0 & \text{on } \delta\Omega. \end{cases}$$

Let $d = 1$ and assume there exists a solution $u \in \mathcal{C}^2(\Omega)$. Then by partial integration we obtain

$$\int_\Omega u''(x)\varphi(x)\,\mathrm{d}x = \boxed{-\int_\Omega u'(x)\varphi'(x)\,\mathrm{d}x = \int_\Omega f(x)\varphi(x)\,\mathrm{d}x}$$

for $\varphi \in \mathcal{C}_0^\infty(\Omega)$.

The advantage of the boxed weak formulation is that $\Delta u(x) = f(x)$ doesn't have to hold everywhere and $u \in \mathcal{C}^2(\Omega)$ is not required anymore.$\diamond$ weak formulation

definition 7.1.6 (weak derivative)
Let $\alpha \in \mathbb{N}_0^d$. Then $v = D^\alpha u : \Omega \to \mathbb{R}$ is called weak derivative of u if or all $\varphi \in \mathcal{C}_0^\infty(\Omega)$ it holds that weak derivative

$$(-1)^{|\alpha|} \int_\Omega u(x) D^\alpha \varphi(x)\,\mathrm{d}x = \int_\Omega v(x)\varphi(x)\,\mathrm{d}x.$$

In the following that the functions involved are sufficiently integrable, which in most cases will mean that they are in $L^2(\Omega)$.

definition 7.1.7 (L^2-Sobolev Spaces)
For $m \in \mathbb{N}$ let

$$\mathcal{H}^m(\Omega) := \{u : \Omega \to \mathbb{R}, D^\alpha \text{ exists } \forall |\alpha| \leqslant m\}$$
$$\mathcal{H}_0^m(\Omega) := \{u \in \mathcal{H}^m(\Omega) : u|_{\delta\Omega} = 0\}.$$

Theorem 7.1.1: Scalar product on $\mathcal{H}^m$ and $\mathcal{H}_0^m$

The spaces $\mathcal{H}^m(\Omega)$ and $\mathcal{H}_0^m(\Omega)$ are Hilbert spaces via

$$\langle u, v \rangle := \sum_{|\alpha| \leqslant m} \int_\Omega D^\alpha u(x) D^\alpha v(x)\,\mathrm{d}x.$$

definition 7.1.8 (Weak solution to Poisson's equation)
A function $u \in \mathcal{H}_0^1(\Omega)$ is called weak solution of (P) if

$$-\int_\Omega \nabla u(x) \nabla v(x)\,\mathrm{d}x = \int_\Omega f(x)v(x)\,\mathrm{d}x$$

holds for all $v \in \mathcal{H}_0^1(\Omega)$.

More generally we can consider a bilinear form $b : \mathcal{H}^m(\Omega) \times \mathcal{H}^m(\Omega) \to \mathbb{R}$ and search for a $u \in \mathcal{H}^m(\Omega)$ fulfilling

$$b(u,v) = \int_\Omega f(x)v(x) \quad \forall v \in H^m(\Omega). \tag{13}$$

Theorem 7.1.2: Lax-Milgram (1954)

Let b be symmetric, bounded and coercive, i.e. $b(v,v) \geqslant c\|v\|^2$ for $c > 0$. Then there exists a unique solution $u \in \mathcal{H}^m(\Omega)$ to (13).

b is bounded if $|b(u,v)| \leqslant C\|u\|\|v\|$, $C > 0$ and symmetric if $b(u,v) = b(v,u)$.

Corollary 7.1.9
For suitable $f : \Omega \to \mathbb{R}$ Poisson's equation has a unique solution.

7.2 Discretization

Usually finding the exact solution is infeasible, thus we discretize the equation and the underlying space.

Commonly methods include finite element methods, finite difference, finite volume methods and frame-based approaches.

High-Fidelity Approximation

The Garlerkin method solves

$$b(u^h, v) = \int_\Omega f(x)v(x)\,\mathrm{d}x \quad \forall v \in U^h,$$

where $U^h \subset H^m(\Omega)$ is a D-dimensional subspace called high-fidelity discretization instead of (13).

Lemma 7.2.1 (Céa (1964))
Up to a constant u^h is the best approximation of u by elements in U^h.

Let $(\varphi_i)_{i=1}^D$ be a basis for U^h. Then we have

$$b\left(\sum_{i=1}^D u_i^h \varphi_i, \sum_{j=1}^D v_j \varphi_j\right) = \int_\Omega f(x) \sum_{j=1}^D v_j \varphi_j(x)\,\mathrm{d}x,$$

which we can – using the bilinearity of b – rewrite as

$$\sum_{i=1}^D u_i^h b(\varphi_i, \varphi_j) - \int_\Omega f(x)\varphi_j(x)\,\mathrm{d}x = 0 \quad \forall j \in [D].$$

Therefore,

$$u_h = \sum_{i=1}^{D} u_i^h \varphi_i = \left[(b(\varphi_j, \varphi_i))_{i,j=1}^{D}\right]^{-1} \left(\int_\Omega f(x)\varphi_i(x)\,\mathrm{d}x\right)_{i=1}^{D} \in \mathbb{R}^d .$$

We have reduce the high-dimensional problem to a problem from linear algebra, matrix inversion.

7.3 Parametric PDEs

Parameter dependent families of PDEs arise in basically any branch of science and engineering such as complex design problems, inverse problems, optimization tasks and uncertainty quantification.

The number of parameters can be finite (i.e. physical properties such as domain geometry) or infinite (i.e. when modelling of random stochastic diffusion field).

Given a differential equation $\mathcal{L}(u_y, y) = f_y$ for parameters y, the parametric map is $\mathcal{Y} \to \mathcal{H}$, $y \mapsto u_y$.

In our setting we will consider parameter-dependent equations of the form

$$b_y(u_y, v) = \int_\Omega f_y(x)v(x)\,\mathrm{d}x \qquad \forall y \in \mathcal{Y},\ v \in \mathcal{H},$$

where $\mathcal{Y} \subset \mathbb{R}^p$ is the compact parameter set (p large), $\mathcal{H}$ is a Sobolev space, $b_y : \mathcal{H} \times \mathcal{H} \to \mathbb{R}$ is symmetric, uniformly coercive (uniformly in y???) and uniformly continuous bilinear form, $f_y : \Omega \to \mathbb{R}$ is the uniformly bounded, parameter-dependent right-hand side and $u_y \in \mathcal{H}$ is the solution.

We assume the solution manifold $(\mathcal{Y}) := \{u_y : y \in \mathcal{Y}\} \subset \mathcal{H}$ to be compact.

Example 7.3.1 (Parametric Poisson's equation)
For $f : \Omega \to \mathbb{R}$ consider

$$\begin{cases} \nabla(a \cdot \nabla u) = f & \text{in } \Omega, \\ u = 0 & \text{on } \delta\Omega, \end{cases} \qquad \text{where } a \in \mathcal{A} \subset \{g : \Omega \to \mathbb{R}, \text{bounded}\}.$$

If $\mathcal{A}$ is compact there exists functions $(g_i)_{i=1}^\infty$ such that for every $a \in \mathcal{A}$ there exist $(y_i)_{i=1}^\infty \subset \mathbb{R}$ with $a = \sum_{i=1}^\infty y_i g_i$. Since we are solving numerically anyway, we restrict ourselves to the case that $a = \sum_{i=1}^p y_i g_i$ for some potentially very large $p \in \mathbb{N}$. ◇

Multi-query situation

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

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

Applications include design optimization, optimal control, routine analysis, uncertainty quantification and inverse problems.

As the computational cost often is much too high, we use the reduced basis method, using that the solution is compact and low-dimensional.

reduced basis method

TODO

Reduced Basis Method $U^{rb} \subset U^h \subset \mathcal{H}$ with dimensions $d(\varepsilon), D$ and possibly ∞, respectively.

Reduced basis ψ_i, high fidelity basis φ_i. V is the transformation matrix between them.

$u_y^{rb} = (B_y^{rb})^{-1} f_y^{rb}$.

Theorem 7.3.1: Discrete Version [33]

We assume that

- For all $\varepsilon > 0$ there exists $d(\varepsilon) \ll D$, $V \in \mathbb{R}^{D \times d(\varepsilon)}$ such that for all $y \in \mathcal{Y}$ there exists $B_y^{rb} \in \mathbb{R}^{d(\varepsilon) \times d(\varepsilon)}$ with
$$\|V B_y^{rb} V^T f_y^h u_y^h\| \leqslant \varepsilon.$$
- There exist ReLU neural networks Φ^B and Φ^f of size $\mathcal{O}(\text{poly}(p)(d(\varepsilon))^2 \,\text{polylog}(\varepsilon))$ such that for all $y \in \mathcal{Y}$ we have
$$\|\Phi^B - B_y^{rb}\|, \|\Phi^f - f_y^{rb}\| \leqslant \varepsilon.$$
Then there exists a ReLU neural network Φ of size $\mathcal{O}((d(\varepsilon))^3 \,\text{polylog}(\varepsilon) + D + \text{poly}(p)(d(\varepsilon))^2 \,\text{polylog}(\varepsilon))$ such that
$$\|\Phi - u_y^h\| \leqslant \varepsilon \quad \forall y \in \mathcal{Y}\,. \tag{14}$$

Proof. (1) **Scalar multiplication.** By [5] scalar multiplication on $[-1, 1]$ can be ε-approximated by a NN of size $\mathcal{O}(\log(\varepsilon^{-1}))$.

(2) **Matrix multiplication.** A matrix multiplication of two $d \times d$-matrices can be performed by $\mathcal{O}(d^3)$ scalar multiplications. Therefore, matrix multiplication can be ε-approximated by a NN of size $\mathcal{O}((d(\varepsilon))^3 \cdot \log(\varepsilon^{-1}))$.

(3) **Matrix inversion.** Neural networks can approximate matrix polynomials and therefore the Inversion operators as
$$\sum_{k=0}^{n} A^k \xrightarrow{n\to\infty} (I - A)^{-1}.$$
Therefore, matrix inversion can be approximated by a neural network of size $\mathcal{O}((d(\varepsilon))^3 \cdot \log^q(\varepsilon^{-1}))$ for some $q > 0$.

(4) **Approximating u_y^h.** We can use both assumptions to approximate
$$V(B_y^{rb})^{-1} V^T f_y^h = f_y^{rb}$$
by a neural network. □

Theorem 7.3.2: Continuous version [33]

Let $(\psi_i)_{i=1}^{d(\varepsilon)}$ denote the reduced basis and assume that there exists ReLU neural networks $(\Phi_i)_{i=1}^{d(\varepsilon)}$ of size $\mathcal{O}(\text{polylog}(\varepsilon))$ such that $\|\Phi_i - \psi_i\|_{\mathcal{H}} \leqslant \varepsilon$ for all $i \in [d(\varepsilon)]$.
Then there exists a ReLU neural network Φ of size

$\mathcal{O}((d(\varepsilon))^3 \operatorname{polylog}(\varepsilon) + \operatorname{poly}(p)(d(\varepsilon))^2 \operatorname{polylog}(\varepsilon))$ such that (14) holds.

Remark 7.3.2 These hypothesis are fulfilled for diffusion and linear elasticity equations.

Folie 44

$$x'(t) = \varrho(A(t)x(t) - b(t)), \qquad x(0) = x_0$$

References

[1] Selfmade in GeoGebra6.

[2] Lecture Slides by Prof. Gitta Kutyniok.

[3] Tatiana A Bubba, Gitta Kutyniok, Matti Lassas, Maximilian März, Wojciech Samek, Samuli Siltanen and Vignesh Srinivasan.
Learning The Invisible: A Hybrid Deep Learning-Shearlet Framework for Limited Angle Computed Tomography
Inverse Problems, Volume 35, Number 6, 31.05.2019.

[4] Matthew Hutson.
AI researchers allege that machine learning is alchemy.
Science Magazine, 03.05.2018.

[5] Dmitry Yarotsky.
Error bounds for approximations with deep ReLU networks. Neural networks Volume 94, Pages 103-114, 2016.

[6] Matus Telgarsky.
Representation Benefits of Deep Feedforward Networks.
Arvix, 2016.

[7] George Cybenko.
Approximation by superpositions of a sigmoidal function.
Mathematics of Control, Signals and Systems, Volume 2, Pages 303-314, 1989.

[8] Screenshot of What is VC Dimension | Example vc dimension of Line Hypothesis class on YouTube at 7:01.

[9] Martien C.A. van Zuijlen.
On a conjecture concerning the sum of independent Rademacher random variables.
Arxiv, 2011.

[10] Wikipedia page for overfitting.

[11] Noah Golowich, Alexander Rakhlin, Ohad Shamir.
Size-Independent Sample Complexity of Neural Networks.
Proceedings of Machine Learning Research, Volume 75, Pages 1–3, 2018.

[12] Chapter 26 in Shalev-Shwartz, Shai; Ben-David, Shai (2014). Understanding Machine Learning – from Theory to Algorithms.

[13] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, Oriol Vinyals.
Understanding deep learning requires rethinking generalization.
ArXiv, 2019

[14] Mikhail Belkin, Daniel Hsu, Siyuan Ma, Soumik Mandal.
Reconciling modern machine learning practice and the bias-variance trade-off.
ArXiv, 2019

[15] Michael Elad (CS, Technion).
Deep, deep trouble Deep Learnings Impact on Image Processing, Mathematics, and Humanity.
SIAM News.

[16] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda B. Viégas, Martin Wattenberg.
SmoothGrad: Removing Noise by Adding Noise.
2017.

[17] Baehrens, Schroeter, Harmeling, Kawanabe, Hansen, Müller, 2010

[18] Springenberg, Dosovitskiy, Brox, Riedmiller, 2015

[19] Bach, Binder, Montavon, Klauschen, Müller, Samek, 2015

[20] G. Montavon, W. Samek, K. Müller.
Methods for interpreting and understanding deep neural networks.
Digital Signal Processing 73, 1–15, 2018

[21] Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, Klaus-Robert Müller
Explaining nonlinear classification decisions with deep Taylor decomposition.
Pattern Recognition, Volume 65, May 2017, Pages 211-222

[22] Ribeiro, Singh, Guestrin, 2016

[23] (Shapley, 1953)

[24] Kononenko, Štrumbelj, 2010

[25] Ingrid Daubechies, M. Defrise and C. De Mol.
An iterative thresholding algorithm for linear inverse problems with a sparsity constraint.
Communications on Pure and Applied Mathematics, 57(11), 1413-1457, November 2004.

[26] Simon S. Du, Xiyu Zhai, Barnabas Poczos, Aarti Singh.
Gradient Descent Provably Optimizes Over-parameterized Neural Networks.
Published as a conference paper at International Conference on Learning Representations 2019.

[27] Simon S. Du, Jason D. Lee, Haochuan Li, Liwei Wang, Xiyu Zhai, Gradient Descent Finds Global Minima of Deep Neural Networks, ICML 2019, (see also arXiv:1811.03804).

[28] D. Labate, G. Kutyniok and G. Weiss, W.-Q. Lim.
Sparse multidimensional representation using shearlets.
Proc. of SPIE conference on Wavelet Applications in Signal and Image Processing XI, San Diego, USA, (2005).

[29] M.Anthony and P. Bartlett.
Neural Network Learning: Theoretical Foundations.
Cambridge University Press, Cambridge, UK, (2009).

[30] Bölcskei, Grohs, Kutyniok and Peterson.

Optimal Approximation with Sparsely Connected Deep Neural Networks.
SIAM Journal on Mathematics of Data Science, 2019, Volume 1, Number 1, pages 8–45.

[31] R. Eldan and O. Shamir.
The Power of Depth for Feedforward Neural Networks.
JMLR: Workshop and Conference Proceedings 49 (2016), 1–34.

[32] T. Poggio, H. Mhaskar, L. Rosasco, B. Miranda, and Q. Liao.
Why and when can deep-but not shallow-networks avoid the curse of dimensionality: A review.
International Journal of Automation and Computing 14(5) (2017), 503–519.

[33] Gitta Kutyniok, Philipp Petersen, Mones Raslan, Reinhold Schneider.
A Theoretical Analysis of Deep Neural Networks and Parametric PDEs.
ArXiv

[34] Scott Lundberg, Su-In Lee.
A Unified Approach to Interpreting Model Predictions. ArXiv, 2017.

[35] Stephan Wäldchen, Jan Macdonald, Sascha Hauch, and Gitta Kutyniok.
The Computational Complexity of Understanding Network Decisions.
ArXiv, 2019.

[36] M. Littman, J. Goldsmith, M. Mundhenk.
The computational complexity of probabilistic planning.
Journal of artificial intelligence research 9 (1998), pages 1-36.

[37] Michael Lustig, David L. Donoho, Juan M. Santos, and John M. Pauly.
Compressed Sensing MRI: A look at how CS can improve on current imaging techniques.
IEEE SIGNAL PROCESSING MAGAZINE, MARCH 2008.

[38] Gitta Kutyniok and Wang-Q Lim.
Image Separation Using Wavelets and Shearlets. Curves and surfaces. 7th international conference, Avignon, France, June 24–30, 2010. Revised selected papers

[39] Michal Aharon, Michael Elad, and Alfred Bruckstein.
K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation.
IEEE TRANSACTIONS ON SIGNAL PROCESSING, VOL. 54, NO. 11, NOVEMBER 2006.

[40] Paschalis P et al.
Tomographic image reconstruction using artificial neural networks.
Nucl. Instrum. Methods 527 211–5,2004.

[41] K. Gregor and Y. LeCun.
Learning fast approximations of sparse coding Proceedings of the 27th

International Conference on Machine Learning (ICML-10), 2010, pp. 399–406.

[42] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov.
Improving neural networks by preventing co-adaptation of feature detectors.
ArXiv.

Möglichkeiten für Bib:

On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation, 2015

Explaining nonlinear classification decisions with deep Taylor decomposition, 2017

Layer-Wise Relevance Propagation for Deep Neural Network Architectures, 2016

Index