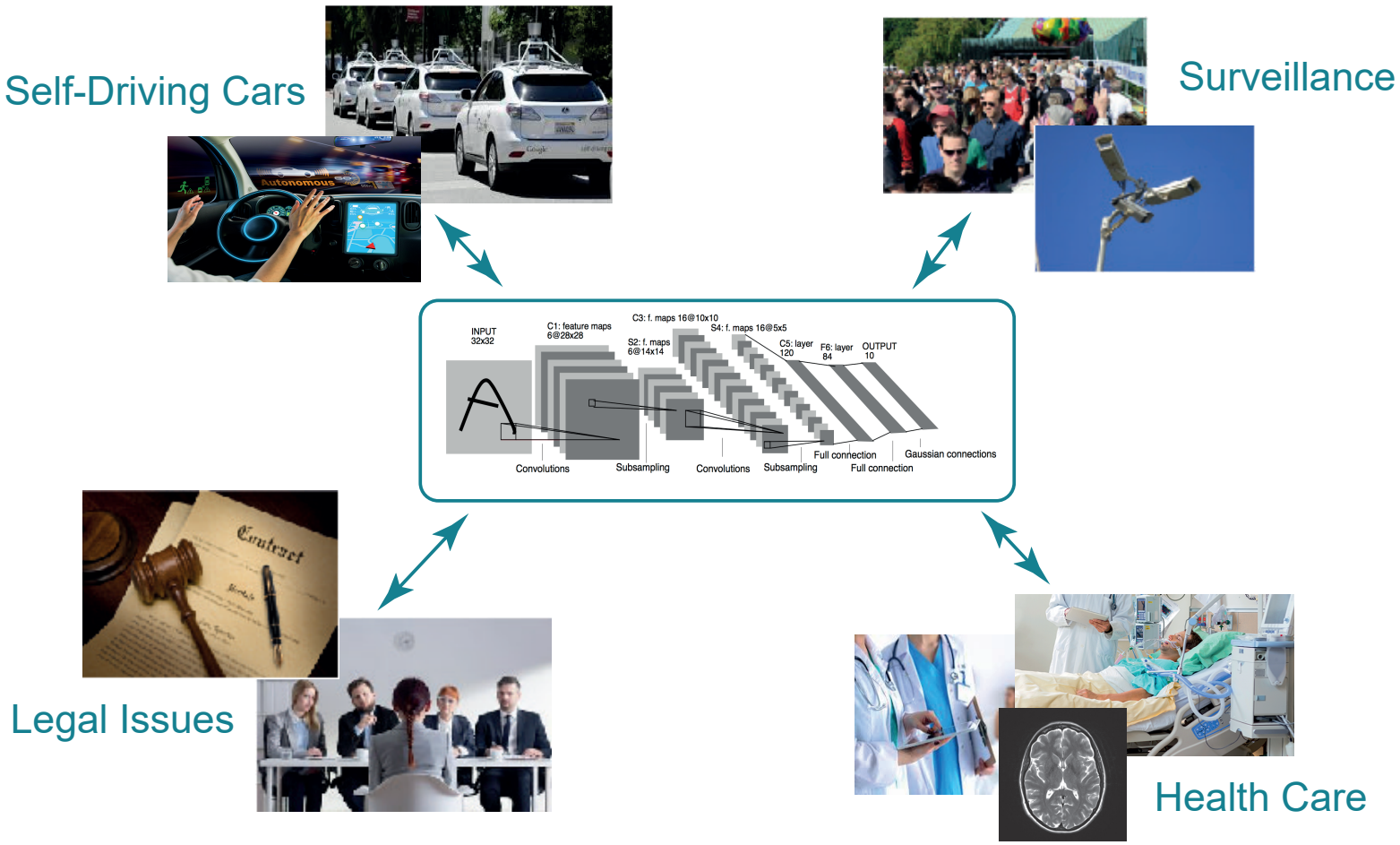
Theory of Deep Learning

Gitta Kutyniok

(Technische Universität Berlin)

Spring School Series "Models and Data"
DASIV SmartState Center, USC, March 17–20, 2019

The Impact of Deep Learning



Deep Learning = Artificial Intelligence?

AlphaGo Zero Shows Machines Can Become Superhuman Without Any Help

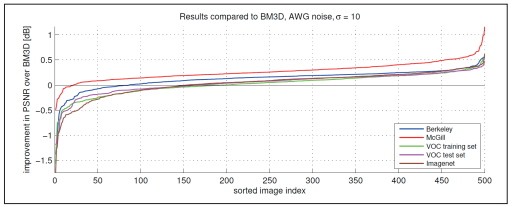
"AlphaGo wasn't the best Go player on the planet for very long. A new version of the masterful AI program has emerged, and it's a monster. In a head-to-head matchup, AlphaGo Zero defeated the original program by 100 games to none."

'...AlphaGo Zero...started with nothing but a blank board and the rules of the game. It learned simply by playing millions of games against itself, using what it learned in each game to improve."

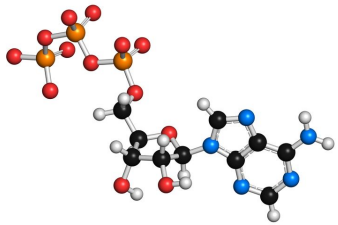
MIT Technology Review (Oct. 2017)

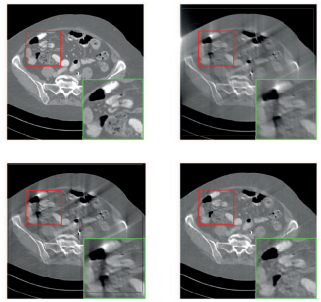
Further Applications of Deep Neural Networks

Some Examples from Areas in Mathematics...



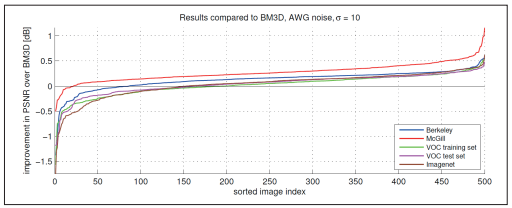
- Imaging Sciences.
  $\rightsquigarrow$ *Image denoising (Burger, Schuler, Harmeling; 2012).*
- PDE Solvers.
  $\rightsquigarrow$ *Schrödinger equation (Rupp, Tkatchenko, Müller, von Lilienfeld; 2012).*
- Inverse Problems.
  $\rightsquigarrow$ *Limited-angle tomography (Bubba, K, Lassas, März, Samek, Siltanen, Srinivan; 2018).*



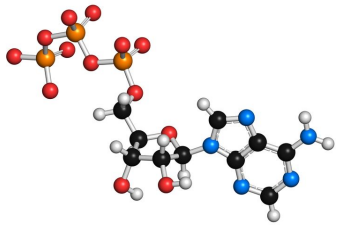


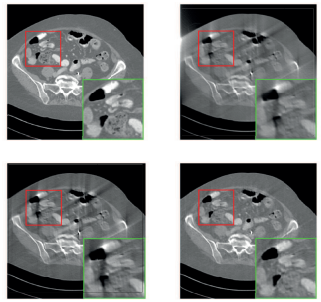
Further Applications of Deep Neural Networks

Some Examples from Areas in Mathematics...



- Imaging Sciences.
  $\rightsquigarrow$ *Image denoising (Burger, Schuler, Harmeling; 2012).*
- PDE Solvers.
  $\rightsquigarrow$ *Schrödinger equation (Rupp, Tkatchenko, Müller, von Lilienfeld; 2012).*
- Inverse Problems.
  $\rightsquigarrow$ *Limited-angle tomography (Bubba, K, Lassas, März, Samek, Siltanen, Srinivan; 2018).*





Deep, Deep Trouble:

Deep Learnings Impact on Image Processing, Mathematics, and Humanity

Michael Elad (CS, Technion), SIAM News, 2017

A Need for Theory...?

First Appearance of Neural Networks

Key Task of McCulloch and Pitts (1943):

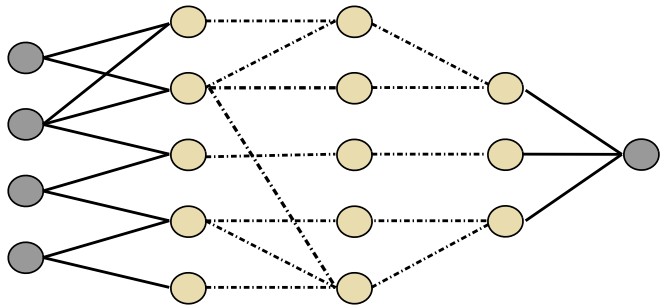
- Develop an algorithmic approach to learning.
- Mimicking the functionality of the human brain.

Goal: Artifical Intelligence!

Neural Networks in Mathematical Terms

Definition:

Assume the following notions:



- $d \in \mathbb{N}$: Dimension of input layer.
- L: Number of layers.
- N: Number of neurons.
- $\sigma : \mathbb{R} \to \mathbb{R}$: (Non-linear) function called *rectifier*.
- $W_\ell : \mathbb{R}^{N_{\ell-1}} \to \mathbb{R}^{N_\ell}$, $\ell = 1, \dots, L$: Affine linear maps $(x \mapsto Ax + b)$

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

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

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

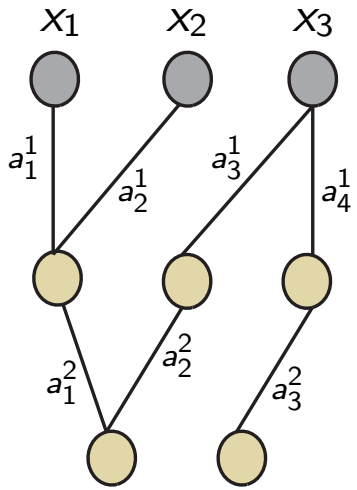
Looking closer...

Remark: The affine linear map W_ℓ is 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

$$W_\ell(x) = A_\ell x + b_\ell.$$

$$A_1 = \begin{pmatrix} a_1^1 & a_2^1 & 0 \\ 0 & 0 & a_3^1 \\ 0 & 0 & a_4^1 \end{pmatrix}$$

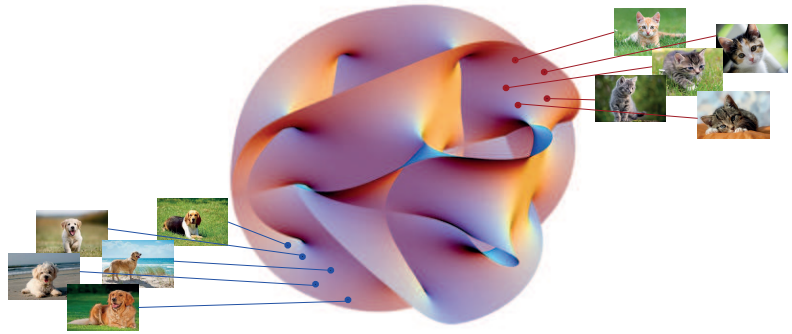
$$A_2 = \begin{pmatrix} a_1^2 & a_2^2 & 0 \\ 0 & 0 & a_3^2 \end{pmatrix}$$



Training of Deep Neural Networks

High-Level Set Up:

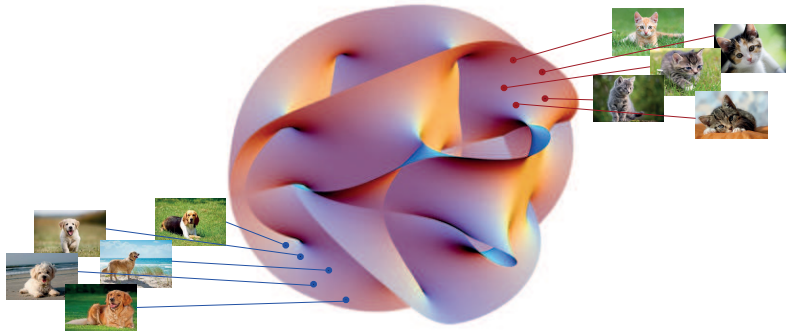
- Samples $(x_i, f(x_i))_{i=1}^m$ of a function such as $f : \mathcal{M} \to \{1, 2, \dots, K\}$.



Training of Deep Neural Networks

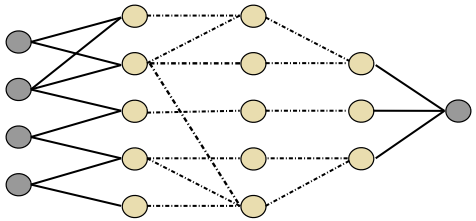
High-Level Set Up:

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



- Select an architecture of a deep neural network, i.e., a choice of d, L, $(N_\ell)_{\ell=1}^L$, and σ.

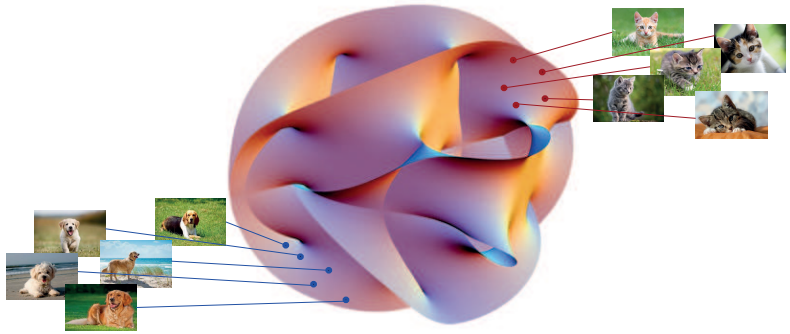
  Sometimes selected entries of the matrices $(A_\ell)_{\ell=1}^L$, i.e., weights, are set to zero at this point.



Training of Deep Neural Networks

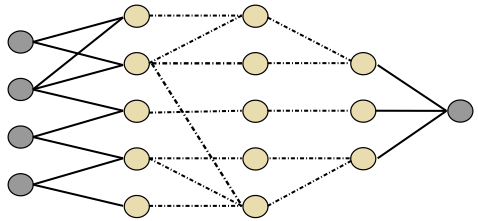
High-Level Set Up:

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



- Select an architecture of a deep neural network, i.e., a choice of d, L, $(N_\ell)_{\ell=1}^L$, and σ.

  Sometimes selected entries of the matrices $(A_\ell)_{\ell=1}^L$, i.e., weights, are set to zero at this point.



- Learn the affine-linear functions $(W_\ell)_{\ell=1}^L = (A_\ell \cdot + b_\ell)_{\ell=1}^L$ by

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

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

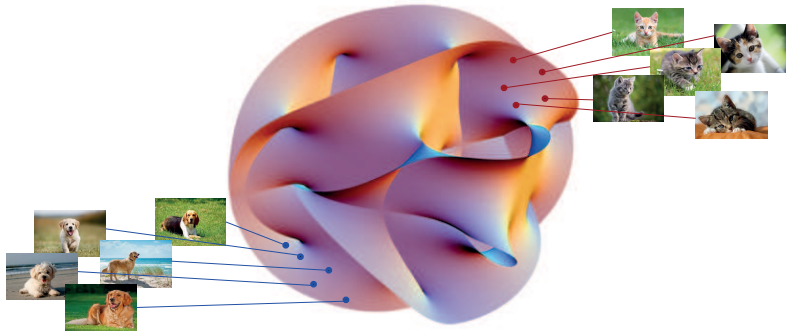
$$\Phi_{A_\ell, b_\ell}(x) = W_L \sigma(W_{L-1} \sigma(\dots \sigma(W_1(x))).$$

This is often done by stochastic gradient descent.

Training of Deep Neural Networks

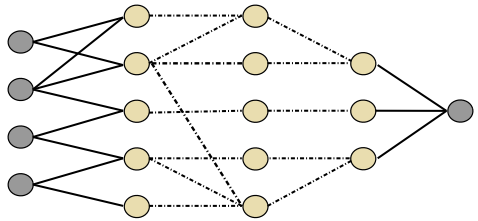
High-Level Set Up:

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



- Select an architecture of a deep neural network, i.e., a choice of d, L, $(N_\ell)_{\ell=1}^L$, and σ.

  Sometimes selected entries of the matrices $(A_\ell)_{\ell=1}^L$, i.e., weights, are set to zero at this point.



- Learn the affine-linear functions $(W_\ell)_{\ell=1}^L = (A_\ell \cdot + b_\ell)_{\ell=1}^L$ by

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

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

$$\Phi_{A_\ell, b_\ell}(x) = W_L \sigma(W_{L-1} \sigma(\dots \sigma(W_1(x))).$$

This is often done by stochastic gradient descent.

Goal: $\Phi \approx f$

Second Appearance of Neural Networks

Key Observations by Y. LeCun et al. (around 2000):

- Drastic improvement of computing power.
  - $\rightsquigarrow$ *Networks with hundreds of layers can be trained.*
  - $\rightsquigarrow$ *Deep Neural Networks!*
- Age of Data starts.
  - $\rightsquigarrow$ *Vast amounts of training data is available.*

Second Appearance of Neural Networks

Key Observations by Y. LeCun et al. (around 2000):

- Drastic improvement of computing power.
  $\rightsquigarrow$ *Networks with hundreds of layers can be trained.*
  $\rightsquigarrow$ *Deep Neural Networks!*
- Age of Data starts.
  $\rightsquigarrow$ *Vast amounts of training data is available.*

Current Situation:

- Setting up a deep neural network for a particular application is more or less trail-and-error and based on experience.
- Training a deep neural network is very unpredictable.
- Almost no knowledge about why a deep neural network mades a decision.

Danger of Deep Neural Networks?

AI researchers allege that machine learning is alchemy

"Ali Rahimi, a researcher in artificial intelligence (AI) at Google in San Francisco, California, took a swipe at his field last December–and received a 40-second ovation for it. Speaking at an AI conference, Rahimi charged that machine learning algorithms, in which computers learn through trial and error, 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..."

"For example, he says, they adopt pet methods to tune their AIs' "learning rates" how much an algorithm corrects itself after each mistake–without understanding why one is better than others. In other cases, AI researchers training their algorithms are simply stumbling in the dark. For example, they implement what's called "stochastic gradient descent" in order to optimize an algorithm's parameters for the lowest possible failure rate. Yet despite thousands of academic papers on the subject, and countless ways of applying the method, the process still relies on trial and error..."

Science (May 2018)

Fundamental Questions concerning Deep Neural Networks

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

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

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

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

Fundamental Questions concerning Deep Neural Networks

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

  $\rightsquigarrow$ *Applied Harmonic Analysis, Approximation Theory, ...*
- *Learning:*
  - Why does the current learning algorithm produce anything reasonable?
  - What are good starting values?

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

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

Fundamental Questions concerning Deep Neural Networks

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

  $\rightsquigarrow$ *Applied Harmonic Analysis, Approximation Theory, ...*
- *Learning:*
  - Why does the current learning algorithm produce anything reasonable?
  - What are good starting values?

  $\rightsquigarrow$ *Differential Geometry, Optimal Control, Optimization, ...*
- *Generalization:*
  - Why do deep neural networks perform that well on data sets, which do not belong to the input-output pairs from a training set?
  - What impact has the depth of the network?

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

Fundamental Questions concerning Deep Neural Networks

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

  $\rightsquigarrow$ *Applied Harmonic Analysis, Approximation Theory, ...*
- *Learning:*
  - Why does the current learning algorithm produce anything reasonable?
  - What are good starting values?

  $\rightsquigarrow$ *Differential Geometry, Optimal Control, Optimization, ...*
- *Generalization:*
  - Why do deep neural networks perform that well on data sets, which do not belong to the input-output pairs from a training set?
  - What impact has the depth of the network?

  $\rightsquigarrow$ *Learning Theory, Optimization, Statistics, ...*
- *Explainability:*
  - Why did a trained deep neural network reach a certain decision?
  - Which components of the input do contribute most?

Fundamental Questions concerning Deep Neural Networks

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

  $\rightsquigarrow$ *Applied Harmonic Analysis, Approximation Theory, ...*
- *Learning:*
  - Why does the current learning algorithm produce anything reasonable?
  - What are good starting values?

  $\rightsquigarrow$ *Differential Geometry, Optimal Control, Optimization, ...*
- *Generalization:*
  - Why do deep neural networks perform that well on data sets, which do not belong to the input-output pairs from a training set?
  - What impact has the depth of the network?

  $\rightsquigarrow$ *Learning Theory, Optimization, Statistics, ...*
- *Explainability:*
  - Why did a trained deep neural network reach a certain decision?
  - Which components of the input do contribute most?

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

Outline

1. 1. Lecture: Mathematical Learning Theory
2. 1. Lecture: Expressivity of Deep Neural Networks
   - Classical Results
   - (Optimal) Bounds for Approximation
3. 1. Lecture: Learning and Generalization of Deep Neural Networks
   - Stochastic Gradient Descent
   - Thoughts about a Theory
4. 1. Lecture: Explainability of Deep Neural Networks
   - What are the Goals?
5. 2. Lecture: Applications to Inverse Problems
   - Sparse Regularization
   - Deep Learning meets Inverse Problems
   - Learning (only) the Invisible: A Hybrid Approach

Mathematical Learning Theory

What is Learning?

Definition by T. Mitchell (1997):

"A computer program is said to learn 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."

$\rightsquigarrow$ *Needs certainly to be made mathematically precise!*

Examples for Task T

Classification Task:

Compute a function $f : \mathbb{R}^n \to \{1, \ldots, k\}$, which maps a data point $x \in \mathbb{R}^n$ to the class k.

$\rightsquigarrow$ Handwritten Digits, ...

Regression Task:

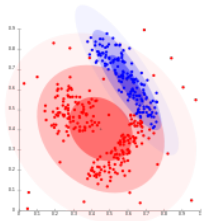
Compute a function $f : \mathbb{R}^n \to \mathbb{R}$, which hence predicts a numerical value.

$\rightsquigarrow$ Prediction of future prices of securities, ...

Density Estimation Task:

Learn a probability density $p : \mathbb{R} \to \mathbb{R}^+$, which can be interpreted as a probability distribution on the space the test data was drawn from.

$\rightsquigarrow$ Finding corrupted data, determining anomalies in data,...



Example for Experience E

Experience as a Data Set:

The experience is typically given by a data set containing many data points such as $x_i \in X$ for all $i = 1, \ldots, m$.

Two Cases:

- *Supervised learning:*
  - Each data point is associated with a label.
    $\rightsquigarrow$ Think of a classification task, in which you know the classes the data points in the (test) data set belong to.
- *Unsupervised learning:*
  - The data point are not labeled.
    $\rightsquigarrow$ Think of a classification task, in which you do not know the classes the data points in the (test) data set belong to.

Example for Performance Measure P

Accuracy as Performance Measure:

The performance is typically measured by the proportion of data points, for which the model (function) outputs the correct value.

Cross-Validation:

The data set is often split into two sets:

- *Training set:*
  This is used to learn the function or density.
- *Test set:*
  This is used to measure the performance of the model.

Linear Regression as one Example, I

Task T:

Predict the function $f : \mathbb{R}^n \to \mathbb{R}$.

Experience E:

We split our data set into

- training set $((x_i^{train}, y_i^{train}))_{i=1}^m \subseteq \mathbb{R}^n \times \mathbb{R}$,
- test set $((x_i^{test}, y_i^{test}))_{i=1}^m \subseteq \mathbb{R}^n \times \mathbb{R}$.

Performance Measure P:

We evaluate the performance of an estimator $\hat{f} : \mathbb{R}^n \to \mathbb{R}$ as the *mean squared error*

$$\frac{1}{m} \sum_{i=1}^{m} |\hat{f}(x_i^{test}) - y_i^{test}|^2.$$

Linear Regression as one Example, II

Learning Algorithm:

- Define a *hypothesis space*

$$\mathcal{H} := \operatorname{span}\{\varphi_1, \dots, \varphi_\ell\} \subseteq C(\mathbb{R}^n).$$

- Given training data

$$\mathbf{z} := ((x_i^{train}, y_i^{train}))_{i=1}^m \subseteq \mathbb{R}^n \times \mathbb{R}.$$

- Define the *empirical error/risk* for some $f : \mathbb{R}^n \to \mathbb{R}$ by

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

Find the *empirical target function*

$$f_{\mathcal{H},\mathbf{z}} := \operatorname{argmin}_{f \in \mathcal{H}} \mathcal{E}_{\mathbf{z}}(f).$$

Linear Regression as one Example, III

Computing the Empirical Target Function:

- Note that every $f \in \mathcal{H}$ can be written as $\sum_{i=1}^{\ell} w_i \varphi_i$.
- We set

$$\mathbf{y} := (y_i^{train})_{i=1}^m \quad \text{and} \quad \mathbf{w} := (w_i)_{i=1}^{\ell}.$$

- Let

$$\mathbf{A} = (\varphi_j(x_i^{train}))_{i,j} \in \mathbb{R}^{m \times \ell}.$$

- With this notation, we obtain

$$\mathcal{E}_{\mathbf{z}}(f) = \|\mathbf{A}\mathbf{w} - \mathbf{y}\|^2.$$

- Let

$$\hat{\mathbf{w}} := \operatorname{argmin}_{\mathbf{w} \in \mathbb{R}^{\ell}} \mathcal{E}_{\mathbf{z}}(f).$$

Then the seeked estimate (= solution of the regression task) is

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

The General Statistical Learning Problem

A Mathematician's Definition of Learning

Definition (Statistical Learning Problem): Let

- $(\Omega, \Sigma, \varrho)$ be a probability space,
- $X : \Omega \to \mathbb{R}^n$ and $Y : \Omega \to \mathbb{R}^k$ random vectors,
- $\mathcal{X}$ and $\mathcal{Y}$ be the images of X and Y, (assume $\mathcal{X} \subseteq \mathbb{R}^n$ compact),
- loss function $\ell : \mathbb{R}^k \times \mathbb{R}^k \to [0, \infty]$.

For a (measurable) function $f : \mathcal{X} \to \mathcal{Y}$, define the error

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

Then the corresponding *learning problem* is to find

$$g = \operatorname{argmin}_{f : \mathcal{X} \to \mathcal{Y}} \mathcal{E}(f).$$

A Mathematician's Definition of Learning

Definition (Statistical Learning Problem): Let

- $(\Omega, \Sigma, \varrho)$ be a probability space,
- $X : \Omega \to \mathbb{R}^n$ and $Y : \Omega \to \mathbb{R}^k$ random vectors,
- $\mathcal{X}$ and $\mathcal{Y}$ be the images of X and Y, (assume $\mathcal{X} \subseteq \mathbb{R}^n$ compact),
- loss function $\ell : \mathbb{R}^k \times \mathbb{R}^k \to [0, \infty]$.

For a (measurable) function $f : \mathcal{X} \to \mathcal{Y}$, define the error

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

Then the corresponding *learning problem* is to find

$$g = \operatorname{argmin}_{f : \mathcal{X} \to \mathcal{Y}} \mathcal{E}(f).$$

Simplification:

- Regression case $k = 1$.
- Squared error loss function $\ell(\tilde{y}, y) = (\tilde{y} - y)^2$.

The Regression Function

Definition: For $x \in \mathcal{X}$ let $\varrho(y|x)$ be the *conditional probability measure* on $\mathcal{Y}$ (with respect to x) and $\varrho_{\mathcal{X}}$ be the *marginal probability measure* on $\mathcal{X}$. We have

$$\varrho_{\mathcal{X}}(S) = \varrho(\pi_{\mathcal{X}}^{-1}(S)),$$

where $\pi_{\mathcal{X}} : \mathcal{X} \times \mathcal{Y} \to \mathcal{X}, \ (x, y) \mapsto x$. Then also

$$\int_{\mathcal{X}\times\mathcal{Y}} \phi(x,y) d\varrho(x,y) = \int_{\mathcal{X}} \left(\int_{\mathcal{Y}} \phi(x,y) d\varrho(y|x) \right) d\varrho_{\mathcal{X}}(x)$$

for every integrable function $\phi : Z \to \mathbb{R}$. Define the *regression function* by

$$f_\varrho : \mathcal{X} \to \mathcal{Y}, \quad f_\varrho(x) = \int_{\mathcal{Y}} y d\varrho(y|x), \quad \text{for } x \in \mathcal{X}.$$

The Regression Function

Definition: For $x \in \mathcal{X}$ let $\varrho(y|x)$ be the *conditional probability measure* on $\mathcal{Y}$ (with respect to x) and $\varrho_{\mathcal{X}}$ be the *marginal probability measure* on $\mathcal{X}$. We have

$$\varrho_{\mathcal{X}}(S) = \varrho(\pi_{\mathcal{X}}^{-1}(S)),$$

where $\pi_{\mathcal{X}} : \mathcal{X} \times \mathcal{Y} \to \mathcal{X}, \ (x, y) \mapsto x$. Then also

$$\int_{\mathcal{X}\times\mathcal{Y}} \phi(x,y) d\varrho(x,y) = \int_{\mathcal{X}} \left(\int_{\mathcal{Y}} \phi(x,y) d\varrho(y|x) \right) d\varrho_{\mathcal{X}}(x)$$

for every integrable function $\phi : Z \to \mathbb{R}$. Define the *regression function* by

$$f_\varrho : \mathcal{X} \to \mathcal{Y}, \quad f_\varrho(x) = \int_{\mathcal{Y}} y d\varrho(y|x), \quad \text{for } x \in \mathcal{X}.$$

Theorem: The regression function f_ϱ is a minimizer of the error $\mathcal{E}$ over $L^2(\mathcal{X}, \varrho_{\mathcal{X}})$, more precisely

$$\mathcal{E}(f) = \|f - f_\varrho\|^2_{L^2(\mathcal{X},\varrho_{\mathcal{X}})} + \mathcal{E}(f_\varrho), \quad \text{for any } f \in L^2(\mathcal{X}, \varrho_{\mathcal{X}}).$$

The Target Function

Definition:

- Let $\mathcal{H}$ be a subspace of $C(\mathcal{X}, \mathcal{Y})$. We then call $\mathcal{H}$ *hypothesis* or *model space*.
- A *target function* $f_{\mathcal{H}} \in \mathcal{H}$ is a minimizer of the error $\mathcal{E}$ over the hypothesis space $\mathcal{H}$, that is

$$f_{\mathcal{H}} = \text{argmin}_{f \in \mathcal{H}} \mathcal{E}(f).$$

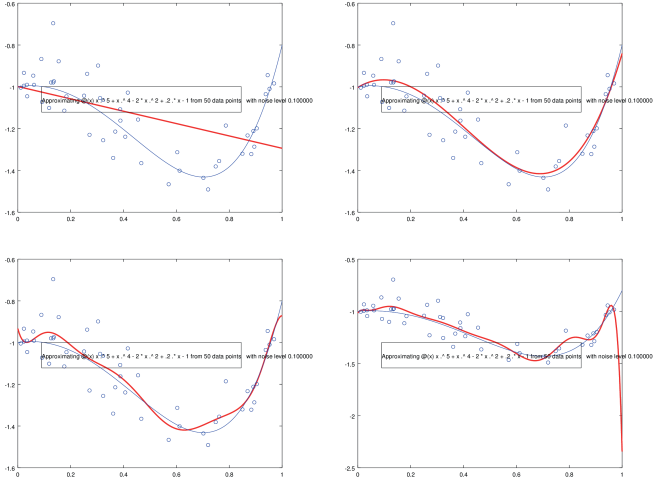
Remark:

The distribution ϱ is not known, hence the target function can not be computed.

Examples of Hypotheses Spaces:

- Space of homogenous polynomials
- Reproducing Kernel Hilbert Spaces RKHS
- ...

Underfitting versus Overfitting



The Empirical Target Function

The True Situation:

We only have access to the evidence data. Hence we have to

- assume we are given a sample $S = ((x_1, y_1), \ldots, (x_N, y_n))$ and
- assume S was drawn i.i.d. according to ϱ.

The goal is to estimate $f_{\mathcal{H}}$ from S.

Definition:

- The *empirical error* of f with respect to loss ℓ and sample data S is defined as

$$\mathcal{E}_S(f) = \frac{1}{N}\sum_{i=1}^{N} \ell(f(x_i), y_i) = \frac{1}{N}\sum_{i=1}^{N} (f(x_i) - y_i)^2.$$

- An *empirical target function* $f_{\mathcal{H},S} \in \mathcal{H}$ is a minimizer of the empirical error $\mathcal{E}_S$ over the hypothesis space $\mathcal{H}$, that is

$$f_{\mathcal{H},S} = \operatorname{argmin}_{f\in\mathcal{H}} \mathcal{E}_S(f).$$

The Bias-Variance Decomposition

The error of the empirical target function can be decomposed:

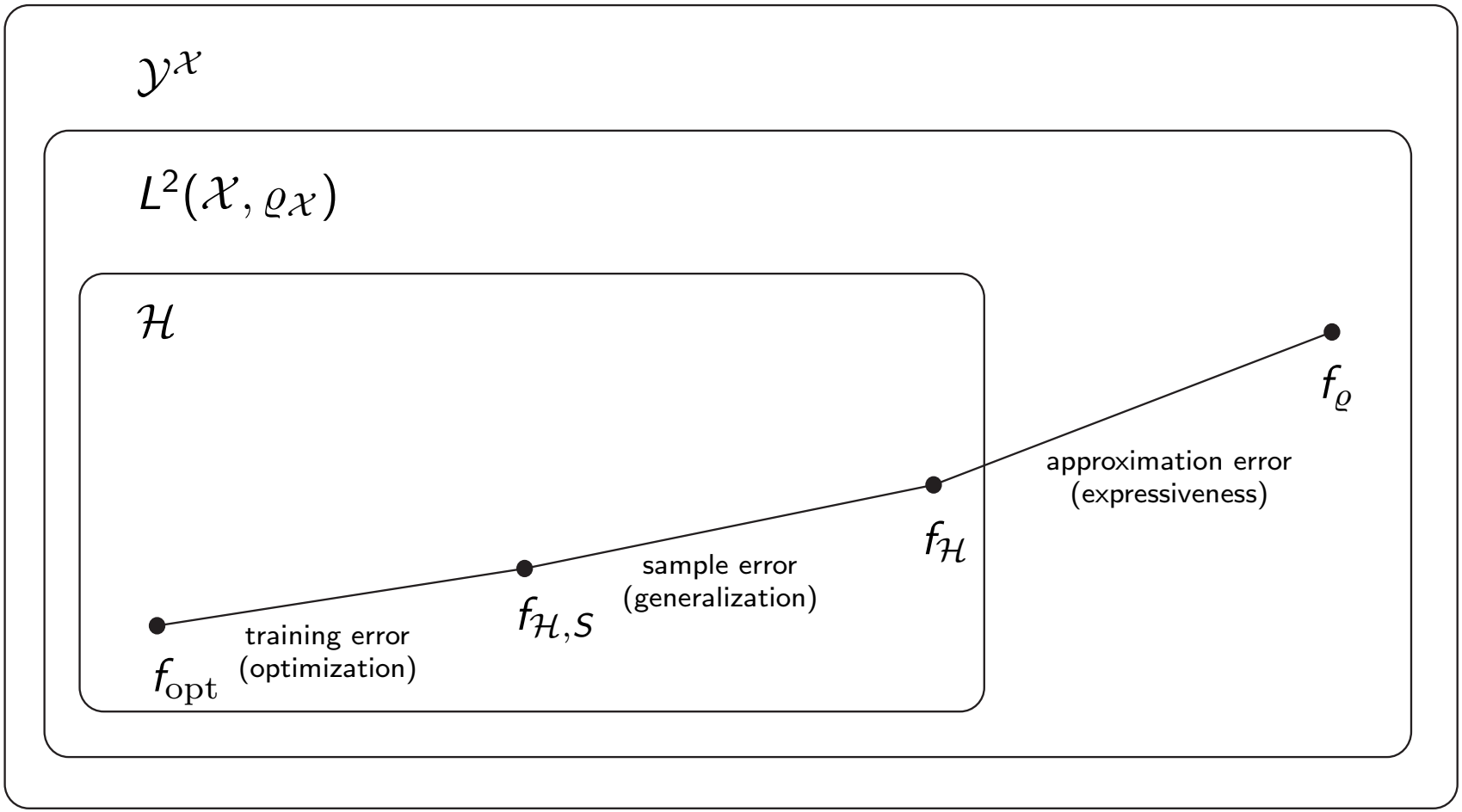
$$\mathcal{E}(f_{\mathcal{H},S}) = \underbrace{\mathcal{E}(f_{\mathcal{H},S}) - \mathcal{E}(f_{\mathcal{H}})}_{\text{sample error}} + \underbrace{\mathcal{E}(f_{\mathcal{H}})}_{\text{approximation error}}$$

Remark:

- The approximation error is only affected by the choice of the hypothesis space $\mathcal{H}$.
- The sample error depends on the size of the sample S but also on the choice of $\mathcal{H}$.
- We have the following typical behaviour for a fixed sample size:

	sample error	approximation error
larger $\mathcal{H}$	*increases*	*decreases*
smaller $\mathcal{H}$	*decreases*	*increases*

Three Types of Errors in Learning Problems



Universally Best Method?

Approaches:

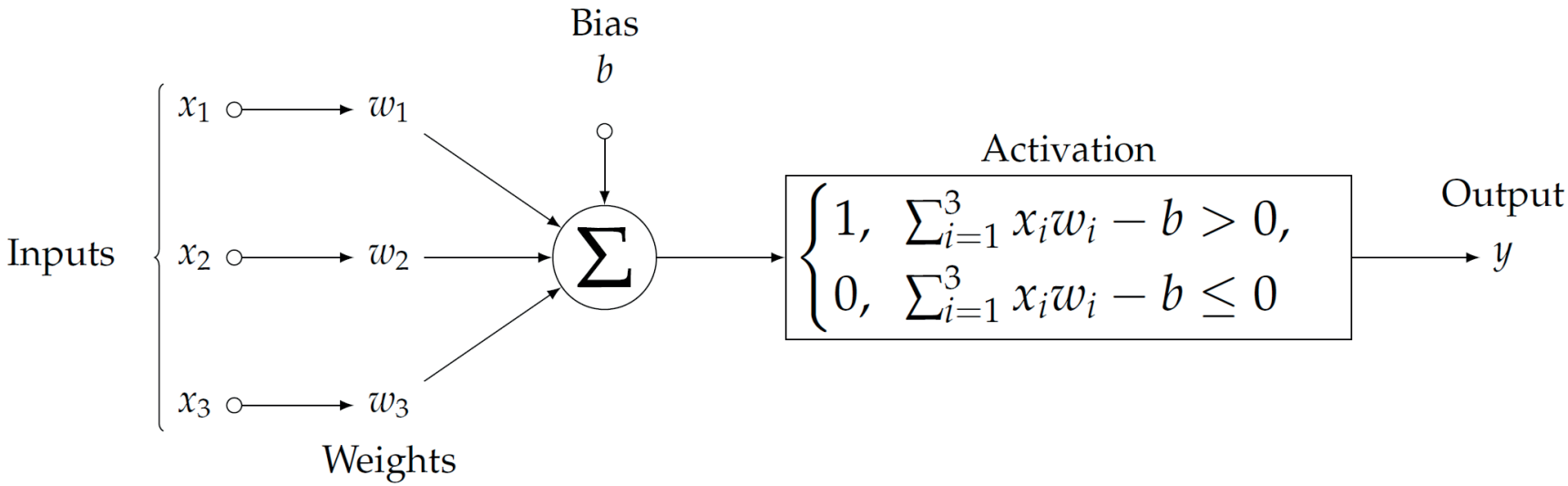
- (Regularized) least squares
- Maximum A Posteriori (MAP) estimate
- Principal Component Analysis (PCA)
- (Kernel) Support Vector Machines (SVM)
- ...

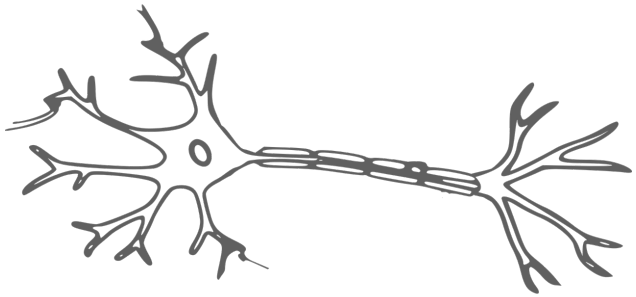
Aim for a universally best method!!!

Deep Neural Networks Enter the Stage

Artificial Neurons

Mimic the human brain!





Artificial Neurons

Definition: An *artificial neuron* with *weights* $w_1, ..., w_n \in \mathbb{R}$, *bias* $b \in \mathbb{R}$ and *activation function (rectifier)* $\sigma : \mathbb{R} \to \mathbb{R}$ is defined as the function $f : \mathbb{R}^n \to \mathbb{R}$ given by

$$f(x_1, ..., x_n) = \sigma\left(\sum_{i=1}^{n} x_i w_i - b\right) = \sigma(\langle x, w \rangle - b),$$

where $w = (w_1, ..., w_n)$ and $x = (x_1, ..., x_n)$.

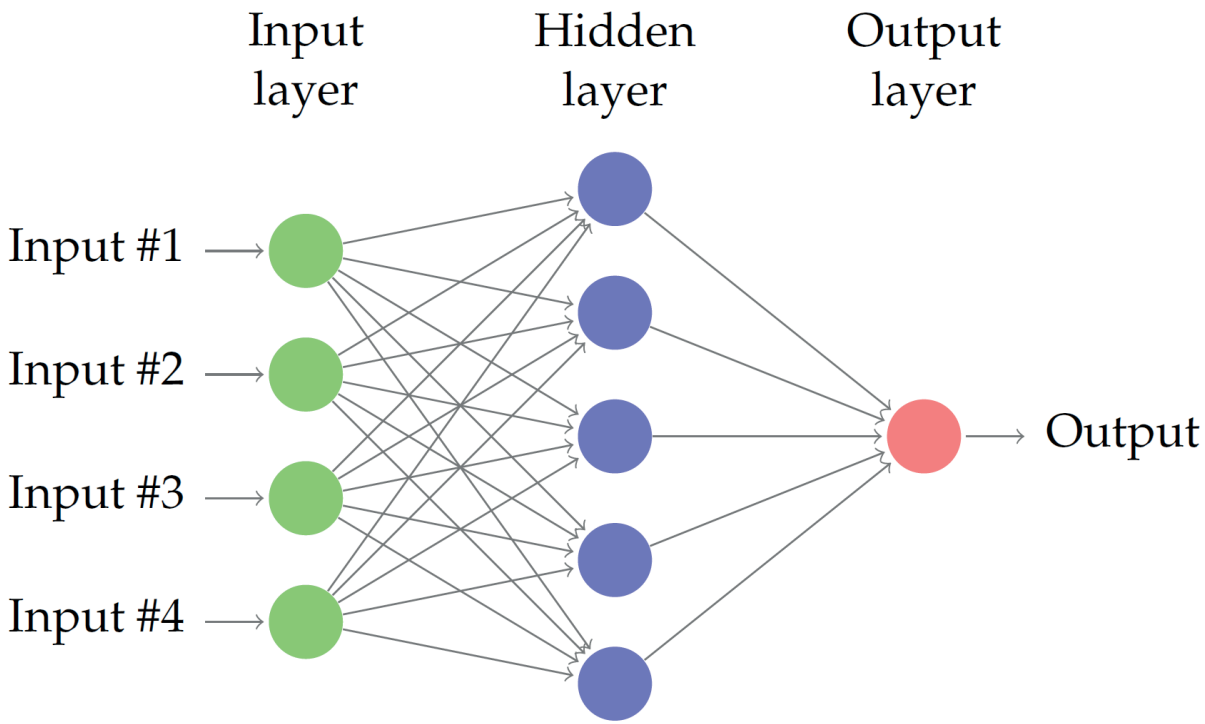
Examples of Activation Functions:

- Heaviside function $\sigma(x) = \begin{cases} 1, & x > 0, \\ 0, & x \leq 0. \end{cases}$
- Sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$.
- Rectifiable Linear Unit (ReLU) $\sigma(x) = \max\{0, x\}$.
- Softmax function $\sigma(x) = \ln(1 + e^x)$.

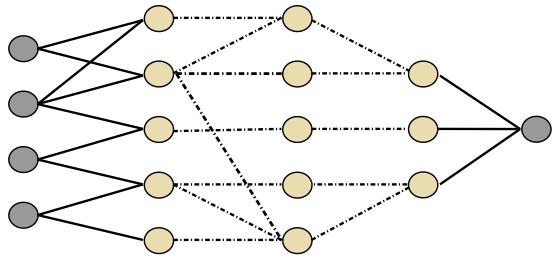
Artificial Neural Network

Definition:

An *artificial neural network* is a graph which consists of artificial neurons. A *feed-forward neural network* is a directed, acyclic graph. All other neural networks are called *recurrent neural networks*.



Key Notions, I



Definition: Let

- $d \in \mathbb{N}$ be the input dimension,
- $L \in \mathbb{N}$ be the number of layers,
- $N_0, N_1, ..., N_L$ the number of neurons in each layer and $N_0 := d$,
- $A_l \in \mathbb{R}^{N_l \times N_{l-1}}, l = 1, ..., L$ be the weights of the edges
- $b_l \in \mathbb{R}^{N_l}, l = 1, ..., L$ be the biases, and
- $\sigma : \mathbb{R} \to \mathbb{R}$ be the (non-linear) activation function/rectifier.

Then

$$\Phi = ((A_l, b_l))_{l=1}^{L}$$

is called *neural network* ("*architecture*") and the map

$$R_\sigma(\Phi) : \mathbb{R}^d \to \mathbb{R}^{N_L}, \quad R_\sigma(\Phi)(x) := x_L,$$

where $x_0 := x$, $x_l := \sigma(A_l x_{l-1} - b_l)$, $l = 1, ...L-1$, and $x_L := A_L x_{L-1} - b_L$ is called the *realization of Φ with activation function σ*.

Key Notions, II

Definition (continued):

We further call

- $N(\Phi) := d + \sum_{l=1}^{L} N_L$ the total number of neurons,
- $L(\Phi) := L$ the number of layers,
- $M(\Phi) := \sum_{l=1}^{L} \|A_l\|_0$ the number of weights (edges) where $\|\cdot\|_0$ is the number of non-zero entries.

We say that

- Φ is *sparsely connected*, if $M(\Phi)$ is small,
- Φ is a *shallow neural network*, if $L(\Phi)$ is small,
- Φ is a *deep neural network*, if $L(\Phi)$ is large.

For $d \in \mathbb{N}$ and $M, L, N \in \mathbb{N} \cup \{\infty\}$, we denote by

$$\mathcal{NN}_{d,M,N,L}$$

the set of neural networks Φ with input dimension d, $N_L = 1$ and

$$M(\Phi) \leq M, N(\Phi) \leq N, L(\Phi) \leq L.$$

Key Notions, III

Definition (continued):

If the size of the weights are a concern, we denote by

$$\mathcal{NN}^R_{d,M,N,L}$$

the set of neural networks Φ with input dimension d, $N_L = 1$, with

$$M(\Phi) \leq M, N(\Phi) \leq N, L(\Phi) \leq L,$$

and with all weights bounded by R.

Doing Nothing...

Lemma: Define

$$\Phi^{\mathrm{Id}} := ((A_1, b_1), (A_2, b_2))$$

with

$$A_1 = \begin{pmatrix} \mathrm{Id}_{\mathbb{R}^d} \\ -\mathrm{Id}_{\mathbb{R}^d} \end{pmatrix}, \quad b_1 = b_2 = 0, \quad A_2 = (\mathrm{Id}_{\mathbb{R}^d}, -\mathrm{Id}_{\mathbb{R}^d}).$$

Then

$$R_{ReLU}(\Phi^{\mathrm{Id}})(x) = x \quad \text{for all } x \in \mathbb{R}^d.$$

Remark: Let Φ be a neural network with input dimension d. Then

$$R_{ReLU}(\Phi) = R_{ReLU}(\Phi \circ \Phi^{\mathrm{Id}}),$$

i.e., different architectures can lead to the same realization.

Fundamental Questions concerning Deep Neural Networks

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

  $\rightsquigarrow$ *Applied Harmonic Analysis, Approximation Theory, ...*
- *Learning:*
  - Why does the current learning algorithm produce anything reasonable?
  - What are good starting values?

  $\rightsquigarrow$ *Differential Geometry, Optimal Control, Optimization, ...*
- *Generalization:*
  - Why do deep neural networks perform that well on data sets, which do not belong to the input-output pairs from a training set?
  - What impact has the depth of the network?

  $\rightsquigarrow$ *Learning Theory, Optimization, Statistics, ...*
- *Explainability:*
  - Why did a trained deep neural network reach a certain decision?
  - Which components of the input do contribute most?

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

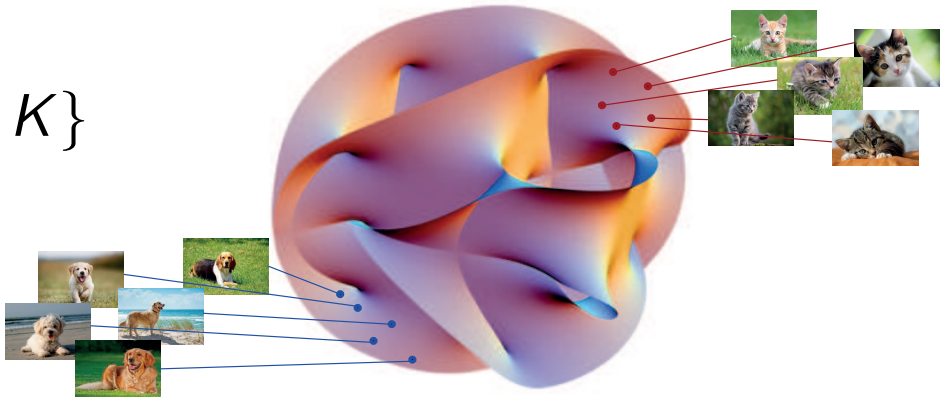
Expressivity

Main Task:

Deep neural networks approximate highly complex functions typically based on given sampling points.

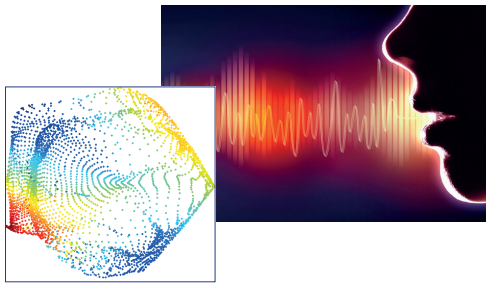
- Image Classification:

$$f : \mathcal{M} \to \{1, 2, \ldots, K\}$$



- Speech Recognition:

$$f : \mathbb{R}^{S_1} \to \mathbb{R}^{S_2}$$



Main Research Goal

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 natural function spaces for DNN applications?

Mathematical Problem:

Under which conditions on a neural network Φ and an activation function σ can every function from a prescribed function class $\mathcal{C}$ be arbitrarily well approximated, i.e.

$$\|R_\sigma(\Phi) - f\|_\infty \leq \varepsilon, \qquad \text{for all } f \in \mathcal{C}.$$

Universal Approximation Theorem

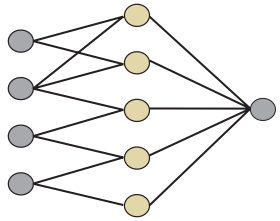
Universal Approximation Theorem (Cybenko, 1989)(Hornik, 1991) (Pinkus, 1999):

Let $\sigma : \mathbb{R} \to \mathbb{R}$ be continuous, but not a polynomial. Also, fix $d \geq 1$, $L = 2$, $N_L \geq 1$, and a compact set $K \subseteq \mathbb{R}^d$. Then, for any continuous $f : \mathbb{R}^d \to \mathbb{R}^{N_L}$ and every $\varepsilon > 0$, there exist $M, N \in \mathbb{N}$ and $\Phi \in \mathcal{NN}_{d,M,N,2}$ with

$$\sup_{x \in K} |R_\sigma(\Phi)(x) - f(x)| \leq \varepsilon.$$

...there exist $N \in \mathbb{N}$, $a_k, b_k \in \mathbb{R}$, $w_k \in \mathbb{R}^d$ such that

$$\sup_{x \in K} |\sum_{k=1}^{N} a_k \sigma(\langle w_k, x \rangle - b_k) - f(x)| \leq \varepsilon.$$



Universal Approximation Theorem

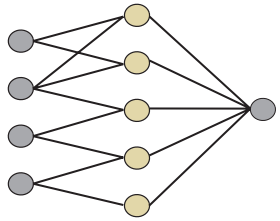
Universal Approximation Theorem (Cybenko, 1989)(Hornik, 1991) (Pinkus, 1999):

Let $\sigma : \mathbb{R} \to \mathbb{R}$ be continuous, but not a polynomial. Also, fix $d \geq 1$, $L = 2$, $N_L \geq 1$, and a compact set $K \subseteq \mathbb{R}^d$. Then, for any continuous $f : \mathbb{R}^d \to \mathbb{R}^{N_L}$ and every $\varepsilon > 0$, there exist $M, N \in \mathbb{N}$ and $\Phi \in \mathcal{NN}_{d,M,N,2}$ with

$$\sup_{x \in K} |R_\sigma(\Phi)(x) - f(x)| \leq \varepsilon.$$

...there exist $N \in \mathbb{N}$, $a_k, b_k \in \mathbb{R}$, $w_k \in \mathbb{R}^d$ such that

$$\sup_{x \in K} |\sum_{k=1}^{N} a_k \sigma(\langle w_k, x \rangle - b_k) - f(x)| \leq \varepsilon.$$



Interpretation: Every continuous function can be approximated up to an error of $\varepsilon > 0$ with a neural network with a single hidden layer and with $O(N)$ neurons.

Idea of Proof

- Restrict to $N_L = 1$ and claim: For $d \geq 1, \sigma$ continuous, $\sigma : \mathbb{R} \to \mathbb{R}$ TFAE:
  (i) $\operatorname{span}\{\sigma(\langle w, x\rangle - b) : w \in \mathbb{R}^d, b \in \mathbb{R}\}$ is dense $C(K, \mathbb{R})$.
  (ii) σ is not a polynomial.
- Now: (ii)$\Rightarrow$ (i) for $d = 1$ and a smooth rectifier σ.
- Since σ is not a polynomial, there exists one $x_0 \in \mathbb{R}$ with

$$\sigma^{(k)}(-x_0) \neq 0 \text{ for all } k.$$

- Constant functions can be arbitrarily well approximated:

$$\sigma(hx - x_0) \to \sigma(-x_0) \text{ as } h \to 0,$$

- Linear functions can be arbitrarily well approximated:

$$\underbrace{\frac{\sigma((\lambda + h)x - x_0) - \sigma(x - x_0)}{h}}_{\to x\sigma'(\lambda x - x_0) \text{ for } h \to 0} \to x \cdot \sigma'(-x_0), \qquad \text{as } h, \lambda \to 0,$$

$\rightsquigarrow$ Any polynomial can be well approximated, then use Stone-Weierstraß Thm.

$\rightsquigarrow$ Finally, extend to d arbitrary and σ not smooth.

Universal Approximation Theorem

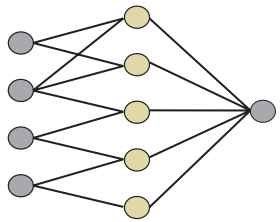
Universal Approximation Theorem (Cybenko, 1989)(Hornik, 1991) (Pinkus, 1999):

Let $\sigma : \mathbb{R} \to \mathbb{R}$ be continuous, but not a polynomial. Also, fix $d \geq 1$, $L = 2$, $N_L \geq 1$, and a compact set $K \subseteq \mathbb{R}^d$. Then, for any continuous $f : \mathbb{R}^d \to \mathbb{R}^{N_L}$ and every $\varepsilon > 0$, there exist $M, N \in \mathbb{N}$ and $\Phi \in \mathcal{NN}_{d,M,N,2}$ with

$$\sup_{x \in K} |R_\sigma(\Phi)(x) - f(x)| \leq \varepsilon.$$

...there exist $N \in \mathbb{N}$, $a_k, b_k \in \mathbb{R}$, $w_k \in \mathbb{R}^d$ such that

$$\sup_{x \in K} |\sum_{k=1}^{N} a_k \sigma(\langle w_k, x \rangle - b_k) - f(x)| \leq \varepsilon.$$



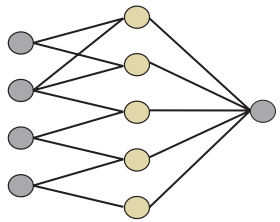
Interpretation: Every continuous function can be approximated up to an error of $\varepsilon > 0$ with a neural network with a single hidden layer and with $O(N)$ neurons.

Universal Approximation Theorem

Universal Approximation Theorem (Cybenko, 1989)(Hornik, 1991) (Pinkus, 1999):

Let $\sigma : \mathbb{R} \to \mathbb{R}$ be continuous, but not a polynomial. Also, fix $d \geq 1$, $L = 2$, $N_L \geq 1$, and a compact set $K \subseteq \mathbb{R}^d$. Then, for any continuous $f : \mathbb{R}^d \to \mathbb{R}^{N_L}$ and every $\varepsilon > 0$, there exist $M, N \in \mathbb{N}$ and $\Phi \in \mathcal{NN}_{d,M,N,2}$ with

$$\sup_{x \in K} |R_\sigma(\Phi)(x) - f(x)| \leq \varepsilon.$$



...there exist $N \in \mathbb{N}$, $a_k, b_k \in \mathbb{R}$, $w_k \in \mathbb{R}^d$ such that

$$\sup_{x \in K} |\sum_{k=1}^{N} a_k \sigma(\langle w_k, x \rangle - b_k) - f(x)| \leq \varepsilon.$$

Interpretation: Every continuous function can be approximated up to an error of $\varepsilon > 0$ with a neural network with a single hidden layer and with $O(N)$ neurons.

What is the connection between ε and N?

One Size Fits All?

"Universal Network Theorem" (Maiorov and Pinkus, 1999):
There exists an activation function $\sigma : \mathbb{R} \to \mathbb{R}$ such that for any $d \in \mathbb{N}$, $K \subset \mathbb{R}^d$ compact, $f : K \to \mathbb{R}$ continuous, and any $\varepsilon > 0$, we find an associated neural network Φ with *two hidden layers of fixed size only dependent on dimension d* such that

$$\sup_{x \in K} |R_\sigma(\Phi)(x) - f(x)| \leq \varepsilon.$$

One Size Fits All?

"Universal Network Theorem" (Maiorov and Pinkus, 1999):
There exists an activation function $\sigma : \mathbb{R} \to \mathbb{R}$ such that for any $d \in \mathbb{N}$, $K \subset \mathbb{R}^d$ compact, $f : K \to \mathbb{R}$ continuous, and any $\varepsilon > 0$, we find an associated neural network Φ with *two hidden layers of fixed size only dependent on dimension d* such that

$$\sup_{x \in K} |R_\sigma(\Phi)(x) - f(x)| \leq \varepsilon.$$

The weights can be arbitrarily huge!

Function Approximation in a Nutshell

Goal: Given $\mathcal{C} \subseteq L^2(\mathbb{R}^d)$ and $(\varphi_i)_{i\in I} \subseteq L^2(\mathbb{R}^d)$. Measure the suitability of $(\varphi_i)_{i\in I}$ for uniformly approximating functions from $\mathcal{C}$.

Definition: The *error of best M-term approximation* of some $f \in \mathcal{C}$ is given by

$$\|f - f_M\|_{L^2(\mathbb{R}^d)} := \inf_{I_M \subset I, \#I_M = M, (c_i)_{i\in I_M}} \|f - \sum_{i\in I_M} c_i \varphi_i\|_{L^2(\mathbb{R}^d)}.$$

The largest $\gamma > 0$ such that

$$\sup_{f\in\mathcal{C}} \|f - f_M\|_{L^2(\mathbb{R}^d)} = O(M^{-\gamma}) \qquad \text{as } M \to \infty$$

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

Approximation accuracy $\leftrightarrow$ *Complexity of approximating system in terms of sparsity*

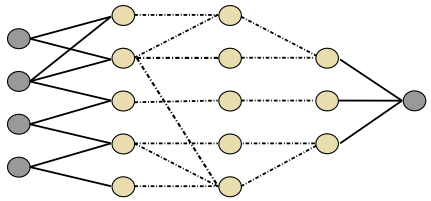
Approximation with Sparse Deep Neural Networks

Definition: Given $\mathcal{C} \subseteq L^2(\mathbb{R}^d)$ and fixed σ. Then $\mathcal{C}$ has the *effective approximation rate* $\gamma^{eff}(\mathcal{C}, \sigma) > 0$, if there exists a polynomial π such that with

$$\Gamma_M(f) := \inf_{\Phi \in \mathcal{NN}_{d,M,N,\pi(\log(M))}^{\pi(M)}} \|f - R_\sigma(\Phi)\|_{L^2(\mathbb{R}^d)}, \quad M \in \mathbb{N},\ f \in \mathcal{C},$$

we have

$$\sup_{f \in \mathcal{C}} \Gamma_M(f) = O(M^{-\gamma^{eff}(\mathcal{C},\sigma)}) \qquad \text{as } M \to \infty.$$



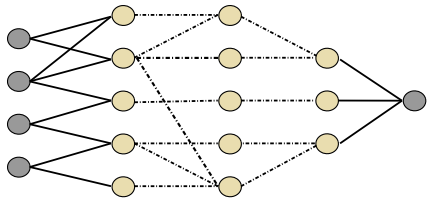
Approximation with Sparse Deep Neural Networks

Definition: Given $\mathcal{C} \subseteq L^2(\mathbb{R}^d)$ and fixed σ. Then $\mathcal{C}$ has the *effective approximation rate* $\gamma^{eff}(\mathcal{C}, \sigma) > 0$, if there exists a polynomial π such that with

$$\Gamma_M(f) := \inf_{\Phi \in \mathcal{NN}_{d,M,N,\pi(\log(M))}^{\pi(M)}} \|f - R_\sigma(\Phi)\|_{L^2(\mathbb{R}^d)}, \quad M \in \mathbb{N},\ f \in \mathcal{C},$$

we have

$$\sup_{f \in \mathcal{C}} \Gamma_M(f) = O(M^{-\gamma^{eff}(\mathcal{C},\sigma)}) \qquad \text{as } M \to \infty.$$



Approximation accuracy $\leftrightarrow$ *Complexity of approximating DNN in terms of memory efficiency*

Non-Exhaustive List of Previous Results

Approximation by NNs with one Single Hidden Layer:

- Bounds in terms terms of nodes and sample size (Barron; 1993, 1994).
- Localized approximations (Chui, Li, and Mhaskar; 1994).
- Fundamental lower bound on approximation rates (DeVore, Oskolkov, and Petrushev; 1997)(Candès; 1998).
- Lower bounds on the sparsity in terms of number of neurons (Schmitt; 1999).
- Approximation using specific rectifiers (Cybenko; 1989).
- Approximation of specific function classes (Mhaskar and Micchelli; 1995), (Mhaskar; 1996).

Non-Exhaustive List of Previous Results

Approximation by NNs with one Single Hidden Layer:

- Bounds in terms terms of nodes and sample size (Barron; 1993, 1994).
- Localized approximations (Chui, Li, and Mhaskar; 1994).
- Fundamental lower bound on approximation rates (DeVore, Oskolkov, and Petrushev; 1997)(Candès; 1998).
- Lower bounds on the sparsity in terms of number of neurons (Schmitt; 1999).
- Approximation using specific rectifiers (Cybenko; 1989).
- Approximation of specific function classes (Mhaskar and Micchelli; 1995), (Mhaskar; 1996).

Approximation by NNs with Multiple Hidden Layers:

- Approximation with sigmoidal rectifiers (Hornik, Stinchcombe, and White; 1989).
- Approximation of continuous functions (Funahashi; 1998).
- Approximation of functions together and their derivatives (Nguyen-Thien and Tran-Cong; 1999).
- Relation between one and multi layers (Eldan and Shamir; 2016), (Mhaskar and Poggio; 2016).
- Approximation by DDNs versus best M-term approximations by wavelets (Shaham, Cloninger, and Coifman; 2017).

Impact of Depth: Adding one Layer

Theorem (Eldan, Shamir; 2016):

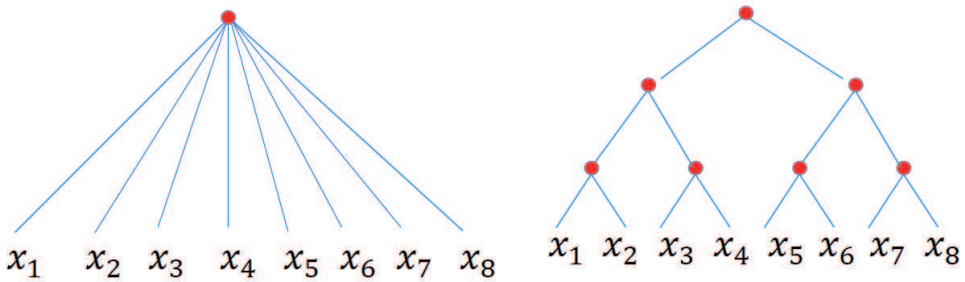
"There exists a simple (approximately radial) function on $\mathbb{R}^d$, expressible by a 3-layer neural network of width polynomial in the dimension d, which cannot be arbitrarily well approximated by 2-layer networks, unless their width is exponential in d."

Remark:

- It shows that depth – even if increased by 1 – can be exponentially more valuable than width for standard feedforward neural networks.
- Key idea of proof:
  - Approximating radial function: First the squared norm function, then the univariate function acting on the norm $\rightsquigarrow$ Easy with 3 layers!
  - But approximating radial functions with 2-layers $\rightsquigarrow$ Difficult (see Ron's talk)!

Impact of Depth: Compositorial Functions

Theorem (Mhaskar, Liao, Poggio; 2017):
"Deep (hierarchical) networks can approximate the class of compositional functions $f(x_1, ...x_n) = h_1(h_2(h_3(x_1, x_2), h_4(x_3, x_4)), ...)$ with the same accuracy as shallow networks but with exponentially lower number of (training) parameters."



Impact of Depth: Approximation Spaces

Definition (Gribonval, K, Nielsen, Voigtlaender; 2019):
Given a depth growth function

$$\mathcal{L} : n \in \mathbb{N} \mapsto \mathcal{L}(n) \in \mathbb{N} \cup \{\infty\}$$

we define, for instance, *approximation spaces* $A_q^\alpha(X, (\Sigma_n)_n)$ *(see definition in Ron's talk) associated with deep neural networks*, where X is a quasi-Banach space of whose elements are functions $f : \Omega \to \mathbb{R}^k$ and

$$\text{“}\Sigma_n := R_\sigma(\mathcal{NN}_{d,n,\infty,\mathcal{L}(n)})\text{”}.$$

Results (Gribonval, K, Nielsen, Voigtlaender; 2019):

- “The expressiveness grows considerable with depth.”
- “Skip connections (such as in ResNets) do not change the expressiveness.”
- “The expressiveness of polynomial activation functions saturates at degree 2.”

Lower Bounds for Approximation

Vapnik-Chervonenkis Dimension, I

Definition:

Let X be a set, $S \subset X$, and let $H \subseteq \{h : X \to \{0,1\}\}$ be a set of binary valued maps on X. We define

$$H_{|S} := \{h_{|S} : h \in H\},$$

which, in words, is the *restriction of the function class H to S*. The *VC dimension* of H is now defined as

$$\mathrm{VCdim}(H) := \sup \left\{ m \in \mathbb{N} : \sup_{|S| \leq m} |H_{|S}| = 2^m \right\}.$$

Vapnik-Chervonenkis Dimension, I

Definition:

Let X be a set, $S \subset X$, and let $H \subseteq \{h : X \to \{0,1\}\}$ be a set of binary valued maps on X. We define

$$H_{|S} := \{h_{|S} : h \in H\},$$

which, in words, is the *restriction of the function class H to S*. The *VC dimension* of H is now defined as

$$\mathrm{VCdim}(H) := \sup \left\{ m \in \mathbb{N} : \sup_{|S| \leq m} |H_{|S}| = 2^m \right\}.$$

Intuition:

- This is a tool for understanding the classification capabilities of a function class.
- 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.

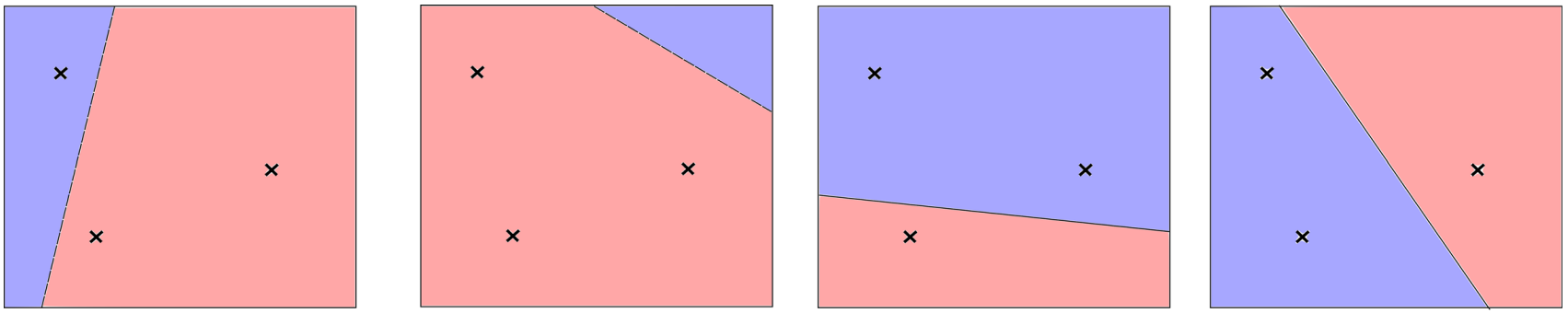
Vapnik-Chervonenkis Dimension, II

Example: Let $X = \mathbb{R}^2$.

1. Let $H = \{0, \chi_\Omega\}$ for some fixed non-empty set $\Omega \subset \mathbb{R}^2$. Then $VCdim(H) = 1$.
2. Let $h = \chi_{\mathbb{R}^+}$ and

$$H = \left\{ h_{\theta,t} := h\left(\left\langle \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix}, \bullet - t \right\rangle \right) \;\middle|\; \theta \in [-\pi, \pi], t \in \mathbb{R}^2 \right\}.$$

Then H is the set of all linear classifiers. If S contains 3 points in general position, then $|H_{|S}| = 8$. On the other hand, 4 points cannot be shattered by H.



Vapnik-Chervonenkis Dimension, III

Theorem (Anthony, Bartlett; 2009): Let σ be piecewise polynomial with p pieces of degree at most ℓ, $h = \chi_{\mathbb{R}^+}$, and for $N, M, d \in \mathbb{N}$ we define

$$H_{N,M,d,L} := \{h \circ \mathrm{R}_\sigma(\Phi) \ : \ \Phi \in \mathcal{NN}_{d,M,N,L}\}.$$

Then

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

Vapnik-Chervonenkis Dimension, III

Theorem (Anthony, Bartlett; 2009): Let σ be piecewise polynomial with p pieces of degree at most ℓ, $h = \chi_{\mathbb{R}^+}$, and for $N, M, d \in \mathbb{N}$ we define

$$H_{N,M,d,L} := \{h \circ \mathrm{R}_\sigma(\Phi) \ : \ \Phi \in \mathcal{NN}_{d,M,N,L}\}.$$

Then

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

Theorem (Yarotsky; 2017): Let $n, d, L \in \mathbb{N}$ and $H = \{h \in C^n([0,1]^d) : \|h\|_{C^n} \leq 1)$. Further let σ be the ReLU. If for every $\varepsilon > 0$ and every $h \in H$ there exists a neural network $\Phi \in \mathcal{NN}_{d,M(\varepsilon),M(\varepsilon),L}$ such that

$$\|h - \mathrm{R}_\sigma(\Phi)\|_\infty < \varepsilon,$$

then, for all $\delta > 0$,

$$M(\varepsilon) = \Omega(\varepsilon^{-d/n(1+\delta)}).$$

Rate Distortion Theory

Key Ingredient from Information Theory (Rate Distortion Theory):
The optimal exponent $\gamma^*(\mathcal{C})$ is a measure of complexity of the function class $\mathcal{C}$:
"The optimal exponent describes the dependence of the code length for encoding the function class on the required approximation quality."

Precise Definition:
With $E : L^2(\mathbb{R}^d) \to \{0,1\}^\ell$, $D : \{0,1\}^\ell \to L^2(\mathbb{R}^d)$, set

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

and then

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

A Fundamental Lower Bound

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

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

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

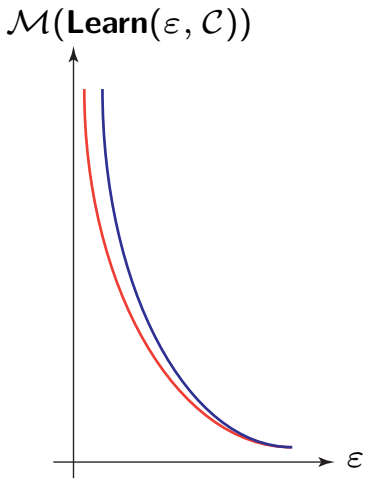
(1) Each weight of $\textbf{Learn}(\varepsilon, f)$ can be encoded with $< -c \log_2(\varepsilon)$ bits,

(2) and

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

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

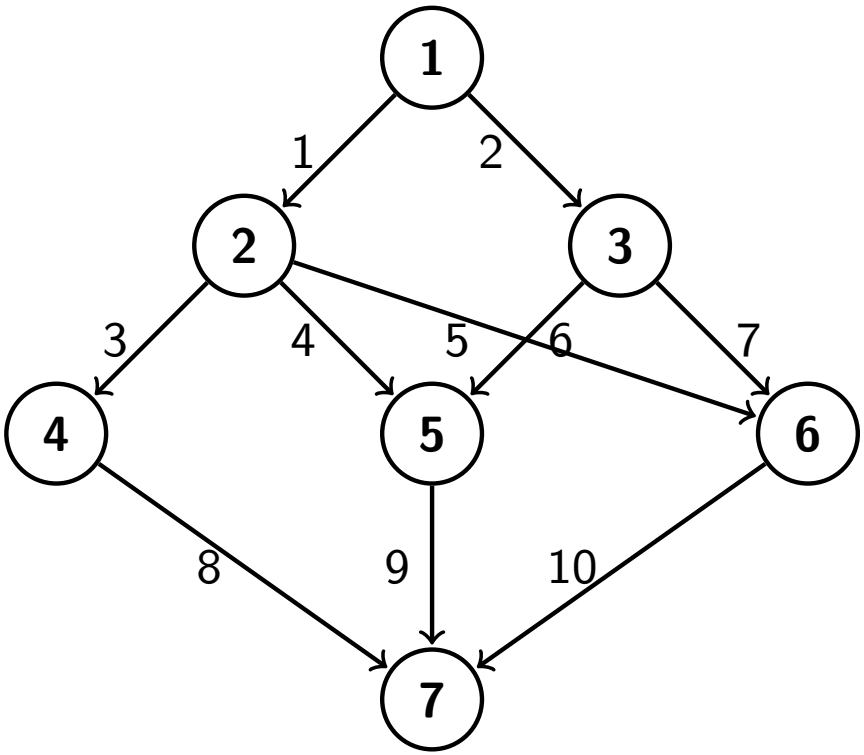
$$\sup_{f \in \mathcal{C}} \mathcal{M}(\textbf{Learn}(\varepsilon, f)) \leq C\varepsilon^{-\gamma} \quad \text{for all } \varepsilon > 0,$$



with $\mathcal{M}(\textbf{Learn}(\varepsilon, f))$ the number of non-zero weights in $\textbf{Learn}(\varepsilon, f)$.

Idea of Proof

Every network with M edges can be encoded in a bit string of length $O(M)$.



Encode:

- # layers,
- # neurons in each layer,
- for each neuron in chronological order # the number of children,
- for each neuron in chronological order the indices of children,
- in chronological order the weights of edges.

A Fundamental Lower Bound

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

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

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

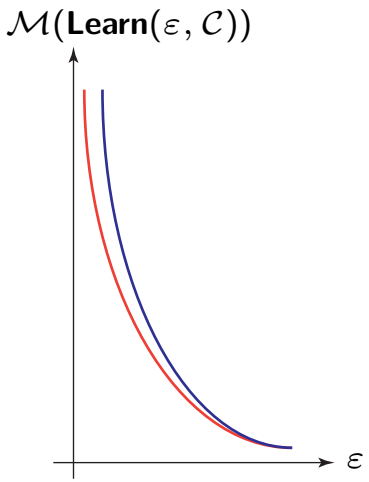
(1) Each weight of $\textbf{Learn}(\varepsilon, f)$ can be encoded with $< -c \log_2(\varepsilon)$ bits,

(2) and

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

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

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



with $\mathcal{M}(\textbf{Learn}(\varepsilon, f))$ the number of non-zero weights in $\textbf{Learn}(\varepsilon, f)$.

A Fundamental Lower Bound

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

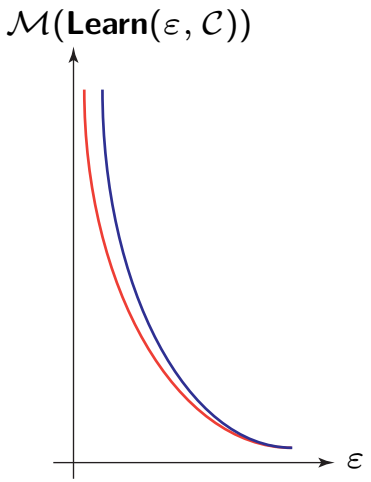
$$\textbf{Learn} : (0,1) \times \mathcal{C} \to \mathcal{NN}_{\infty,\infty,d,\rho}$$

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

(1) Each weight of $\textbf{Learn}(\varepsilon, f)$ can be encoded with $< -c \log_2(\varepsilon)$ bits,

(2) and

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



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

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

with $\mathcal{M}(\textbf{Learn}(\varepsilon, f))$ the number of non-zero weights in $\textbf{Learn}(\varepsilon, f)$.

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

Optimally Sparse Deep Neural Networks

DNNs and Representation Systems, I

Question:

Can we exploit approximation results with representation systems?

DNNs and Representation Systems, I

Question:

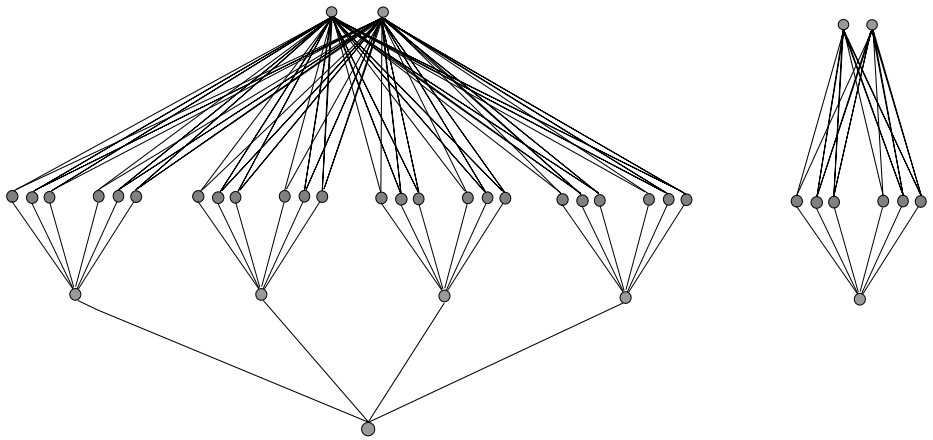
Can we exploit approximation results with representation systems?

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

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

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

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



DNNs and Representation Systems, II

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

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

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

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

Proof:

- There exists a network Φ with $O(M)$ edges with $\Phi = \sum_{i\in I_M} c_i\varphi_i$.
- Set $\varepsilon = \tilde{C} M^{-1/\gamma^*(\mathcal{C})}$ and solve for the number of edges M, yielding

$$M = O(\varepsilon^{-\gamma^*(\mathcal{C})}).$$

DNNs and Representation Systems, II

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

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

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

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

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

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

Road Map

Construction of optimally sparse deep neural networks:

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

(2) Determine an associated representation system with the following properties:
   - The elements of this system can be realized by a neural network with controlled number of edges.
   - This system provides optimally sparse approximations for $\mathcal{C}$.

DNNs have as much expressive power as most classical systems!

Road Map

Construction of optimally sparse deep neural networks:

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

(2) Determine an associated representation system with the following properties:
   - The elements of this system can be realized by a neural network with controlled number of edges.
   - This system provides optimally sparse approximations for $\mathcal{C}$.

DNNs have as much expressive power as most classical systems!

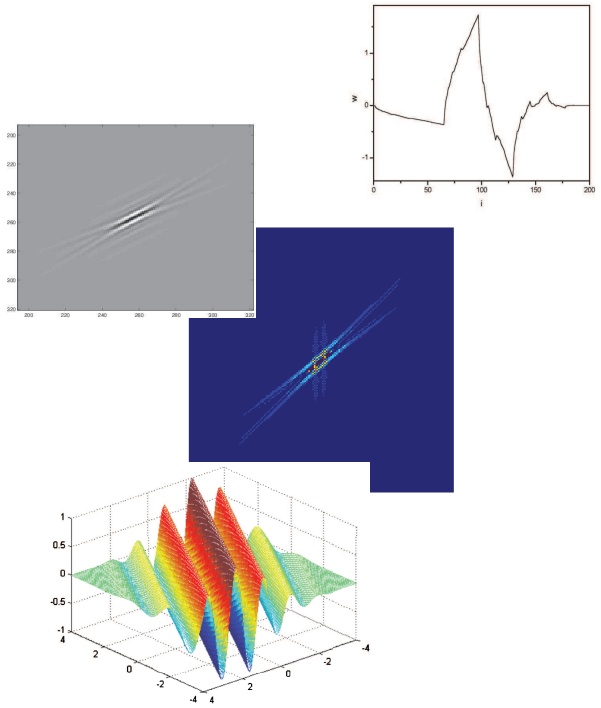
$\rightsquigarrow$ *But this does not yet control the size of the weights!*

Applied Harmonic Analysis

Representation systems designed by Applied Harmonic Analysis concepts have established themselves as a standard tool in applied mathematics, computer science, and engineering.

Examples:

- Wavelets.
- Ridgelets.
- Curvelets.
- Shearlets.
- ...



Key Property:

Fast Algorithms combined with Sparse Approximation Properties!

Affine Transforms

Building Principle:

Many systems from applied harmonic analysis such as

- wavelets,
- ridgelets,
- shearlets,

constitute affine systems:

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

Affine Transforms

Building Principle:

Many systems from applied harmonic analysis such as

- wavelets,
- ridgelets,
- shearlets,

constitute affine systems:

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

Realization by Neural Networks:

The following conditions are equivalent:

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

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

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

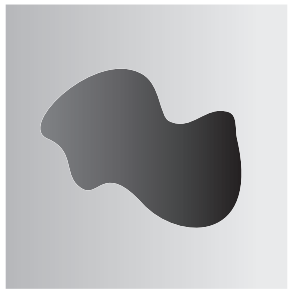
The Class of Cartoon-Like Functions

Definition (Donoho; 2001)(Grohs, Keiper, K, and Schäfer; 2016):
Let $\alpha \in [\frac{1}{2}, 1]$ and $\nu > 0$. We then define the class of *α-cartoon-like functions* by

$$\mathcal{E}^{\frac{1}{\alpha}}(\mathbb{R}^2) = \{f \in L^2(\mathbb{R}^2) : f = f_1 + \chi_B f_2\},$$

where $B \subset [0,1]^2$ with $\partial B \in C^{\frac{1}{\alpha}}$, and the functions f_1 and f_2 satisfy $f_1, f_2 \in C_0^{\frac{1}{\alpha}}([0,1]^2)$, $\|f_1\|_{C^{\frac{1}{\alpha}}}, \|f_2\|_{C^{\frac{1}{\alpha}}}, \|\partial B\|_{C^{\frac{1}{\alpha}}} < \nu$.

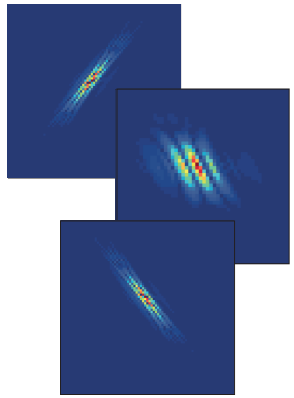
Illustration:



Constructing α-Shearlets

Shearlets (K, Labate; 2006):

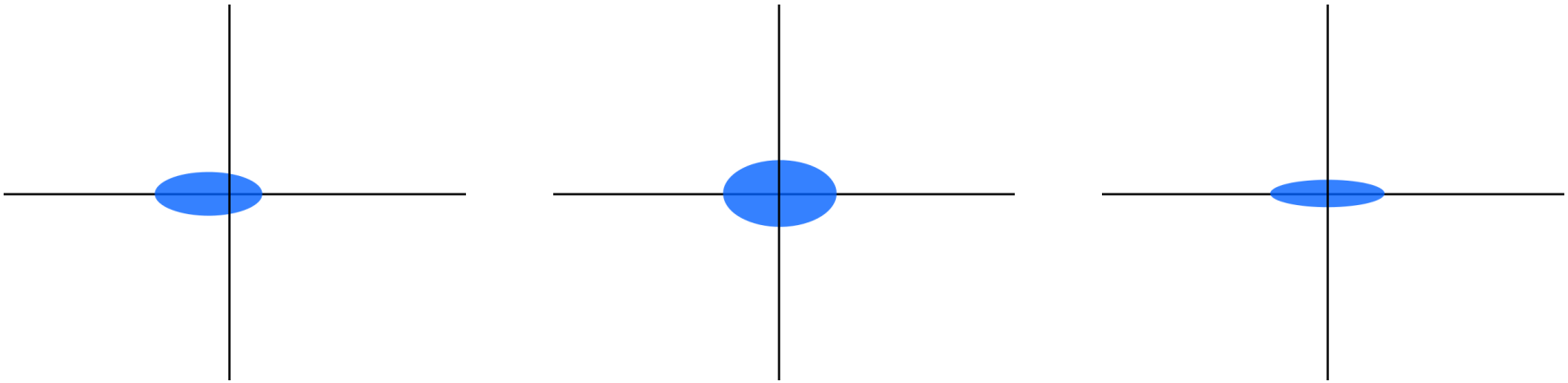
$$A_j := \begin{pmatrix} 2^j & 0 \\ 0 & 2^{j/2} \end{pmatrix}, \qquad S_k := \begin{pmatrix} 1 & k \\ 0 & 1 \end{pmatrix}, \quad j, k \in \mathbb{Z}.$$



Then

$$\psi_{j,k,m} := 2^{\frac{3j}{4}} \psi(S_k A_j \cdot - m).$$

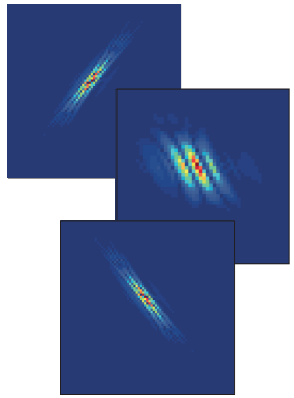
Notice: *$x \mapsto S_k A_j x - m$ is an affine-linear map!*



Constructing α-Shearlets

Shearlets (K, Labate; 2006):

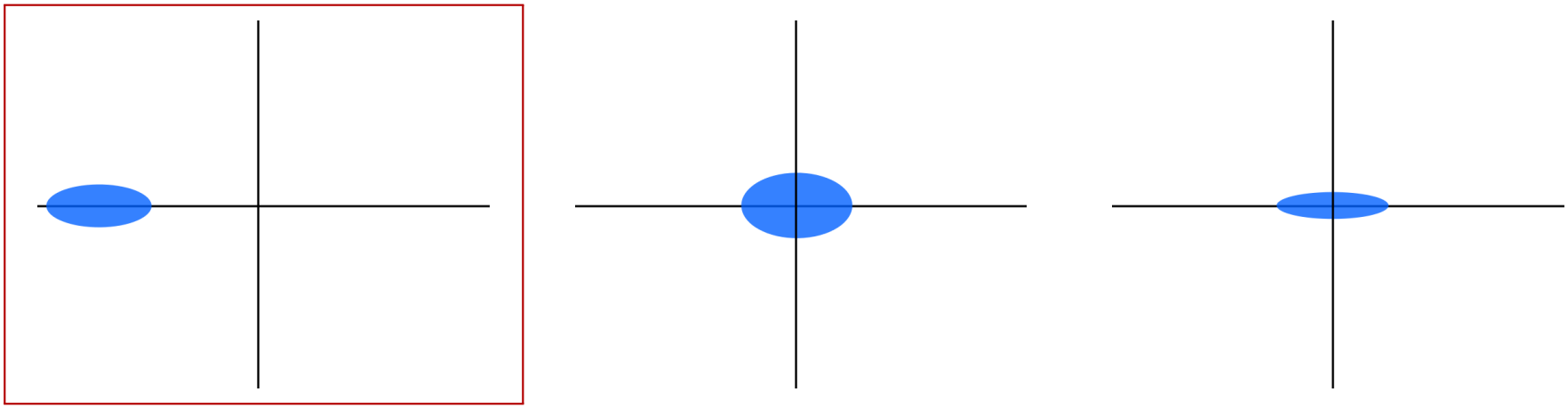
$$A_j := \begin{pmatrix} 2^j & 0 \\ 0 & 2^{j/2} \end{pmatrix}, \qquad S_k := \begin{pmatrix} 1 & k \\ 0 & 1 \end{pmatrix}, \quad j, k \in \mathbb{Z}.$$



Then

$$\psi_{j,k,m} := 2^{\frac{3j}{4}} \psi(S_k A_j \cdot - m).$$

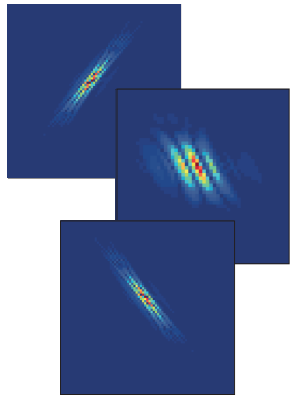
Notice: *$x \mapsto S_k A_j x - m$ is an affine-linear map!*



Constructing α-Shearlets

Shearlets (K, Labate; 2006):

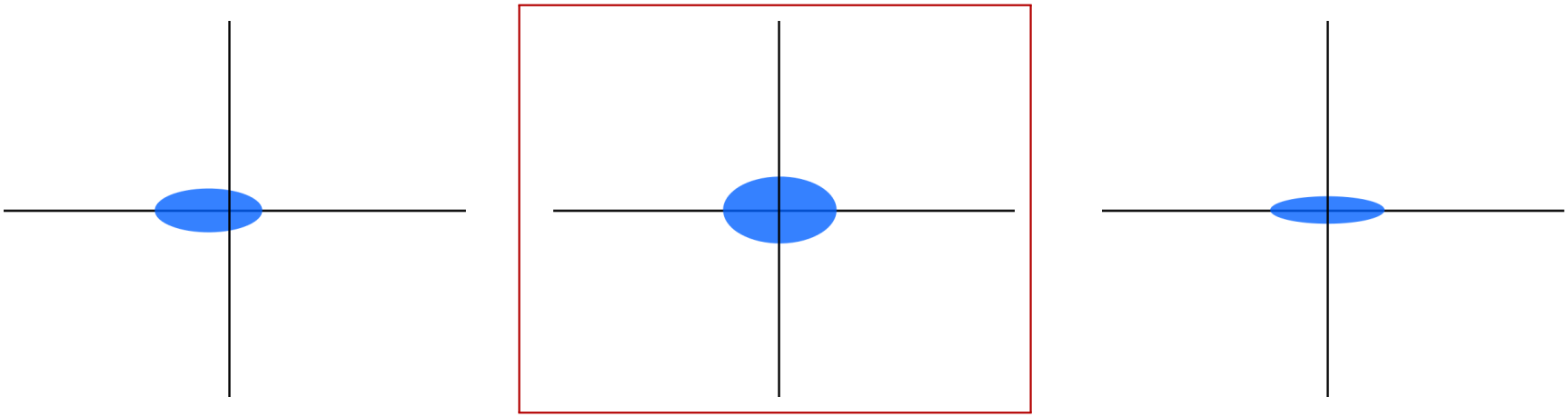
$$A_j := \begin{pmatrix} 2^j & 0 \\ 0 & 2^{j/2} \end{pmatrix}, \qquad S_k := \begin{pmatrix} 1 & k \\ 0 & 1 \end{pmatrix}, \quad j, k \in \mathbb{Z}.$$



Then

$$\psi_{j,k,m} := 2^{\frac{3j}{4}} \psi(S_k A_j \cdot - m).$$

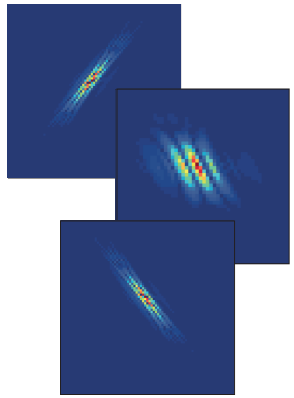
Notice: *$x \mapsto S_k A_j x - m$ is an affine-linear map!*



Constructing α-Shearlets

Shearlets (K, Labate; 2006):

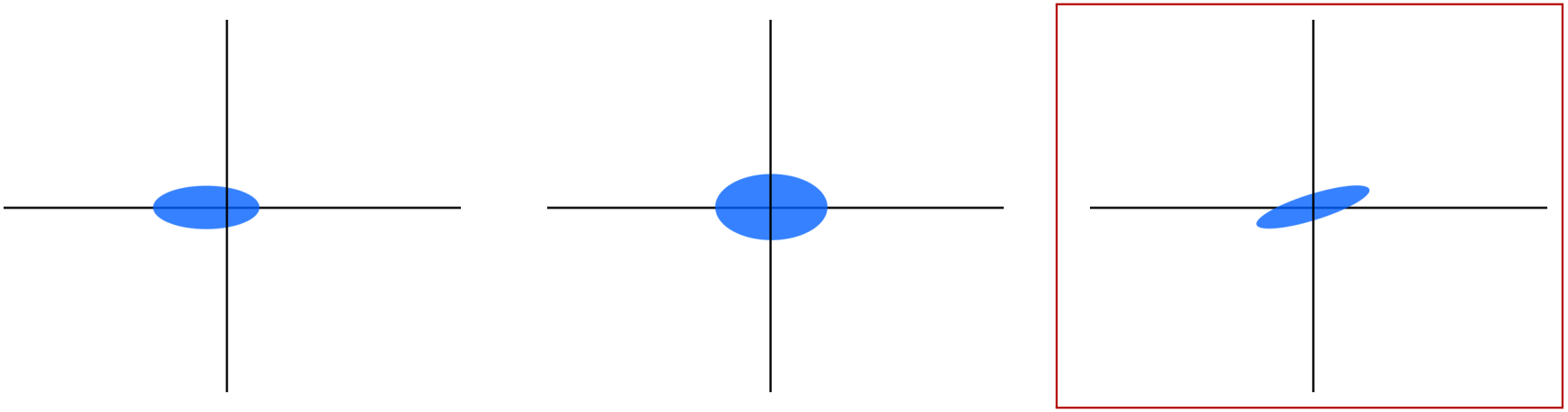
$$A_j := \begin{pmatrix} 2^j & 0 \\ 0 & 2^{j/2} \end{pmatrix}, \qquad S_k := \begin{pmatrix} 1 & k \\ 0 & 1 \end{pmatrix}, \quad j, k \in \mathbb{Z}.$$



Then

$$\psi_{j,k,m} := 2^{\frac{3j}{4}} \psi(S_k A_j \cdot - m).$$

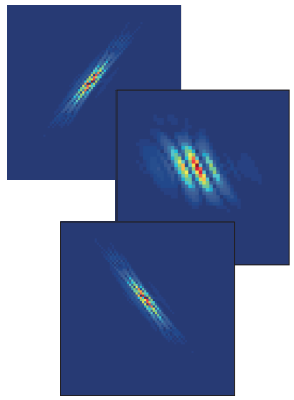
Notice: *$x \mapsto S_k A_j x - m$ is an affine-linear map!*



Constructing α-Shearlets

Shearlets (K, Labate; 2006):

$$A_j := \begin{pmatrix} 2^j & 0 \\ 0 & 2^{j/2} \end{pmatrix}, \qquad S_k := \begin{pmatrix} 1 & k \\ 0 & 1 \end{pmatrix}, \quad j, k \in \mathbb{Z}.$$



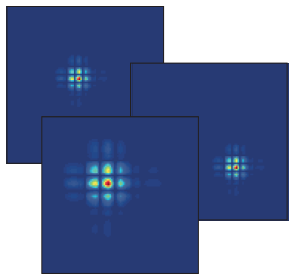
Then

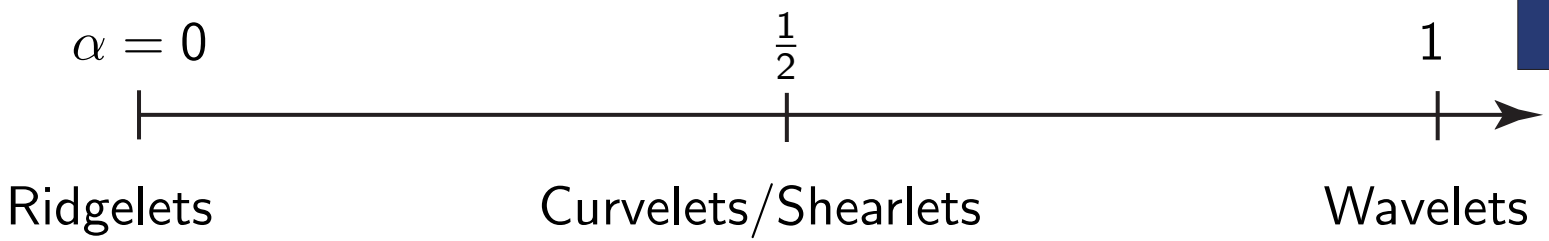
$$\psi_{j,k,m} := 2^{\frac{3j}{4}} \psi(S_k A_j \cdot - m).$$

Extension to α-Shearlets (Grohs, Keiper, K, and Schäfer; 2016):

For $j \in \mathbb{Z}$, define

$$A_{\alpha,j} = \begin{pmatrix} 2^j & 0 \\ 0 & 2^{\alpha j} \end{pmatrix}.$$





Optimal Sparse Approximation with α-Shearlets

Theorem (Grohs, Keiper, K, and Schäfer; 2016)(Voigtlaender; 2017):
Let $\alpha \in [\frac{1}{2}, 1]$, let $\phi, \psi \in L^2(\mathbb{R}^2)$ be sufficiently smooth and compactly supported, and let ψ have sufficiently many vanishing moments. Also set $\widetilde{\psi}(x_1, x_2) := \psi(x_2, x_1)$ for all $x_1, x_2 \in \mathbb{R}$.
Then there exists some $c^* > 0$ such that, for every $\varepsilon > 0$, there exists a constant $C_\varepsilon > 0$ with

$$\|f - f_N\|_{L^2(\mathbb{R}^2)} \leq C_\varepsilon N^{-\frac{1}{2\alpha}+\varepsilon} \qquad \text{for all } f \in \mathcal{E}^{\frac{1}{\alpha}}(\mathbb{R}^2),$$

where f_N is a best N-term approximation with respect to $\mathcal{SH}_\alpha(\varphi, \psi, \widetilde{\psi}, c)$ and $0 < c < c^*$.

This is the (almost) optimal sparse approximation rate!

Road Map

General Approach:

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

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

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

Road Map

General Approach:

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

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

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

Road Map

General Approach:

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

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

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

Road Map

General Approach:

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

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

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

Road Map

General Approach:

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

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

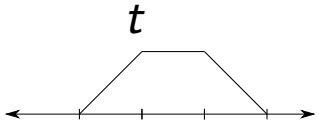
- The elements of this system can be realized by a neural network with controlled number of edges.
  $\rightsquigarrow$ *Still to be analyzed!*
- This system provides optimally sparse approximations for $\mathcal{C}$.
  $\rightsquigarrow$ *This has been proven!*

Construction of Generators

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

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

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

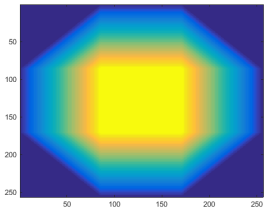


$\rightsquigarrow$ *t can be constructed with a 2 layer network.*

- Observe that

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

yields a 2D bump function.



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

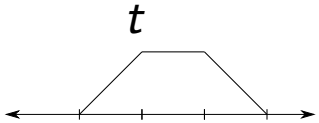
$\rightsquigarrow \psi$ *can be realized by a 3 layer neural network.*

Construction of Generators

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

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

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

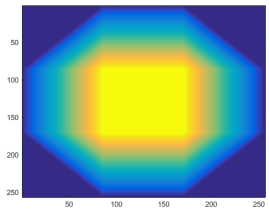


$\rightsquigarrow$ *t can be constructed with a 2 layer network.*

- Observe that

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

yields a 2D bump function.



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

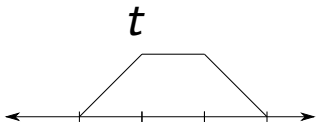
$\rightsquigarrow \psi$ *can be realized by a 3 layer neural network.*

This cannot yield differentiable functions ψ!

Construction of Generators

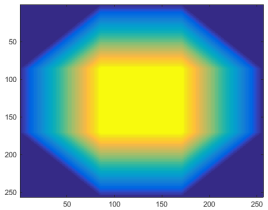
Wavelet generators (LeCun; 1987), (Shaham, Cloninger, Coifman; 2017):

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



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

- Observe that
$$\phi(x_1, x_2) := \rho(t(x_1) + t(x_2) - 1)$$
yields a 2D bump function.



- Summing up shifted versions of ϕ yields a function ψ with vanishing moments.
$\rightsquigarrow \psi$ *can be realized by a 3 layer neural network.*

Our Construction: Use a smoothed version of a ReLU.

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

Optimal Approximation

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

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

Remark: The topology and quantized weights of this network can be stored with $C \cdot N \cdot \text{polylog}(N)$ bits.

Function classes which are optimal representable by affine systems are also optimally effectively approximated by memory-efficient neural networks with a parallel architecture!

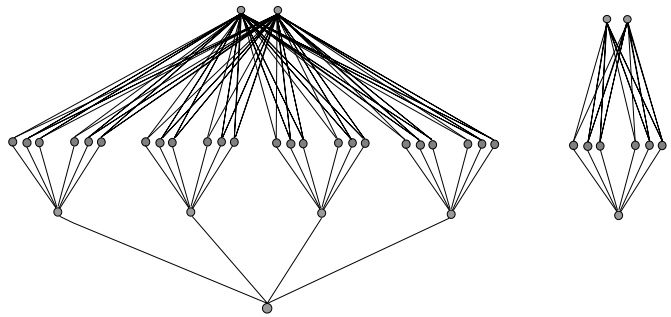
Some Numerics

Typically weights are learnt by backpropagation. This raises the following question:

Does this lead to the optimal sparse connectivity?

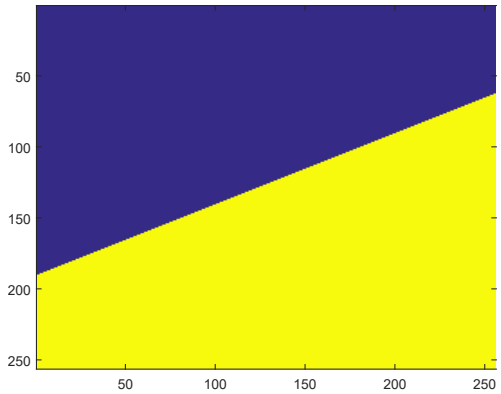
Our setup:

- Fixed network topology with ReLUs.
- Specific functions to learn.
- Learning through SGD.
- Analyze the learnt subnetworks.

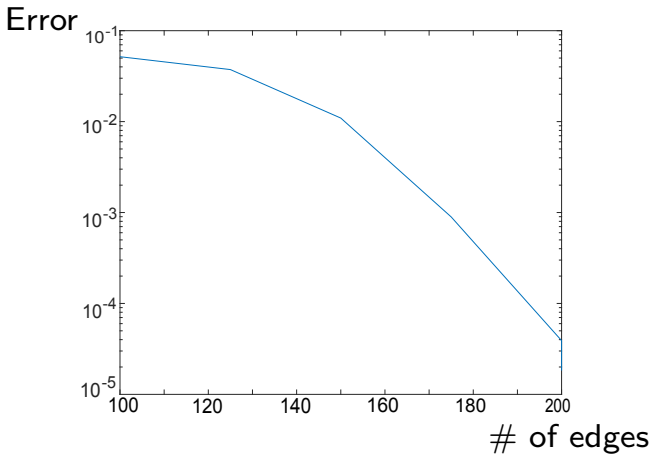


- Analysis of the connection between approximation error and number of edges.

Example: Function 1

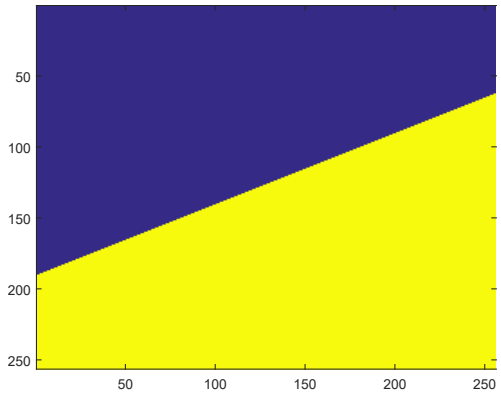


Linear Singularity

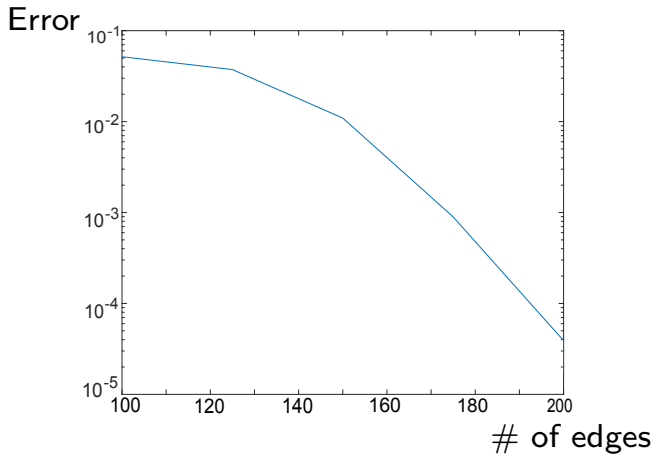


Observation: The decay is exponential. This is expected if the network is a sum of 0-shearlets, which are ridgelets.

Example: Function 1

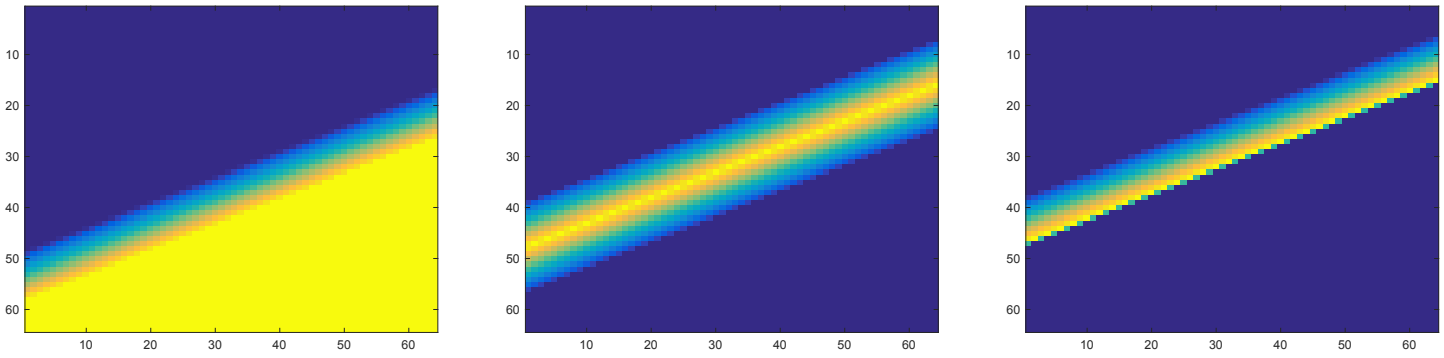


Linear Singularity



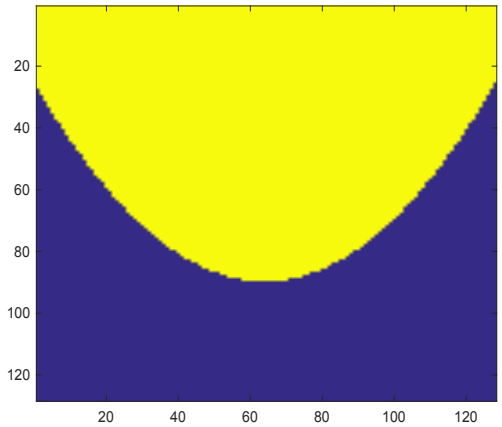
Observation: The decay is exponential. This is expected if the network is a sum of 0-shearlets, which are ridgelets.

Examples of Subnetworks:

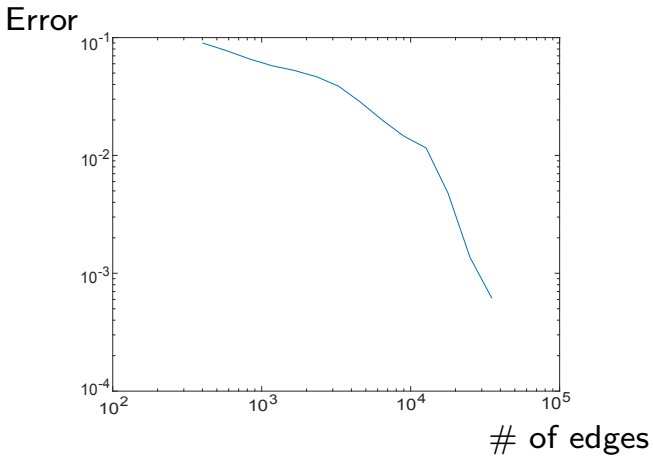


These have indeed the shape of ridgelets!

Example: Function 2

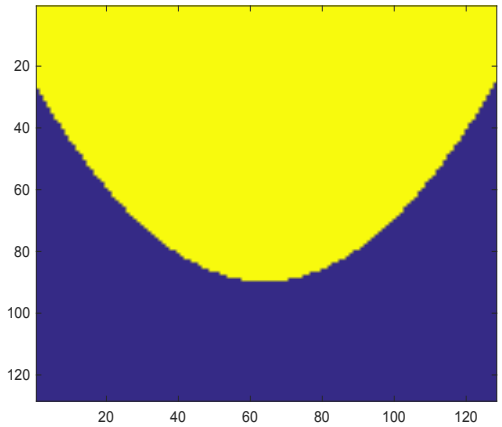


Curvilinear Singularity

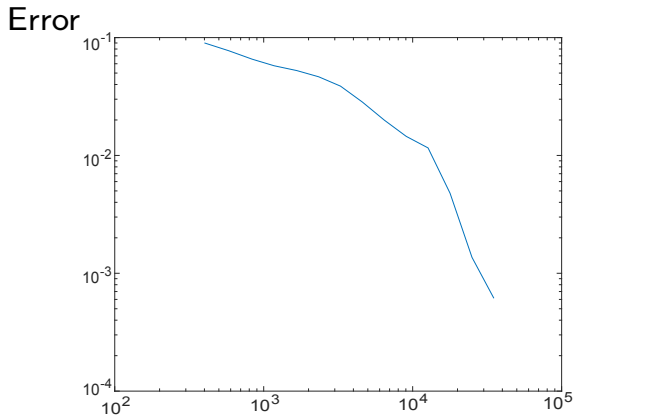


Observation: The decay is of the order M^{-1}. This is expected if the network is a sum of $\frac{1}{2}$-shearlets.

Example: Function 2

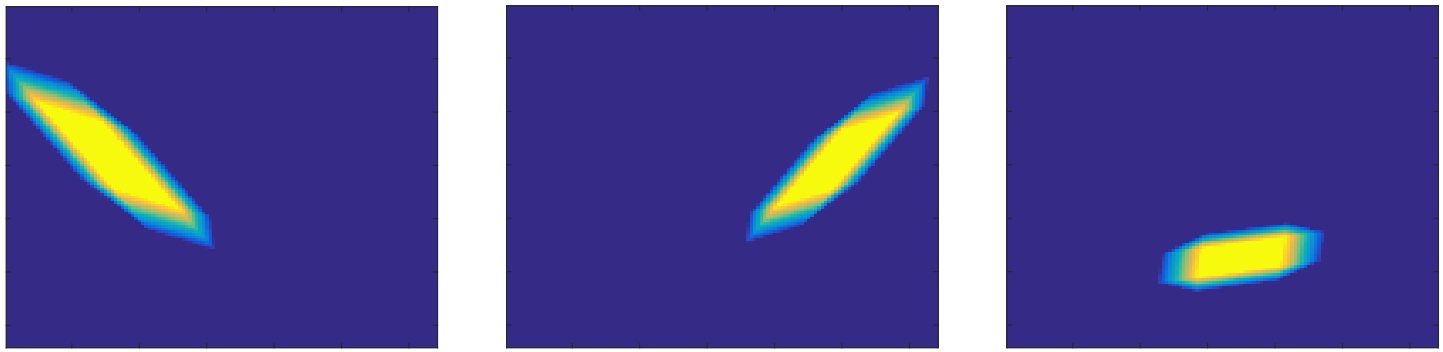


Curvilinear Singularity



Observation: The decay is of the order M^{-1}. This is expected if the network is a sum of $\frac{1}{2}$-shearlets.

Examples of Subnetworks:



These seem to be indeed anisotropic!

Fundamental Questions concerning Deep Neural Networks

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

  $\rightsquigarrow$ *Applied Harmonic Analysis, Approximation Theory, ...*
- *Learning:*
  - Why does the current learning algorithm produce anything reasonable?
  - What are good starting values?

  $\rightsquigarrow$ *Differential Geometry, Optimal Control, Optimization, ...*
- *Generalization:*
  - Why do deep neural networks perform that well on data sets, which do not belong to the input-output pairs from a training set?
  - What impact has the depth of the network?

  $\rightsquigarrow$ *Learning Theory, Optimization, Statistics, ...*
- *Explainability:*
  - Why did a trained deep neural network reach a certain decision?
  - Which components of the input do contribute most?

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

Problem Setting

Let $d = N_0 \in \mathbb{N}$ and $N_1, ..., N_L, L \in \mathbb{N}$ and let σ be a rectifier. Then consider the hypothesis space

$$\mathcal{H} := \{R_\sigma(\Phi) : \Phi = ((A_1, b_1), ..., (A_L, b_L)),\ A_l \in \mathbb{R}^{N_{l-1}, N_l},\ b_l \in \mathbb{R}^{N_l}\}.$$

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

$$f_z := f_{\mathcal{H},z} = \text{argmin}_{f \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^{m} (f(x_i) - y_i)^2.$$

Problem Setting

Let $d = N_0 \in \mathbb{N}$ and $N_1, ..., N_L, L \in \mathbb{N}$ and let σ be a rectifier. Then consider the hypothesis space

$$\mathcal{H} := \{R_\sigma(\Phi) : \Phi = ((A_1, b_1), ..., (A_L, b_L)),\ A_l \in \mathbb{R}^{N_{l-1}, N_l},\ b_l \in \mathbb{R}^{N_l}\}.$$

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

$$f_z := f_{\mathcal{H},z} = \text{argmin}_{f \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^m (f(x_i) - y_i)^2.$$

More General Task: One can also consider a more general case such as

$$f_z = \text{argmin}_{f \in \mathcal{H}} \sum_{i=1}^m \mathcal{L}(f, x_i, y_i)$$

where $\mathcal{L} : C(\mathbb{R}^d, \mathbb{R}^{N_L}) \times \mathbb{R}^d \times \mathbb{R}^{N_L} \to \mathbb{R}$ is a *loss function*.

Gradient Descent

Optimization Approach: A simple optimization method is *gradient descent*. For $F : \mathbb{R}^N \to \mathbb{R}$, this amounts to

$$u_{n+1} \leftarrow u_n - \eta \nabla F(u_n) \text{ for all } n \in \mathbb{N}$$

where $\nabla F(u) = (\frac{\partial F}{\partial u_1}(u), \dots \frac{\partial F}{\partial u_n}(u))$ and η is the step size.

Gradient Descent

Optimization Approach: A simple optimization method is *gradient descent*. For $F : \mathbb{R}^N \to \mathbb{R}$, this amounts to

$$u_{n+1} \leftarrow u_n - \eta \nabla F(u_n) \text{ for all } n \in \mathbb{N}$$

where $\nabla F(u) = (\frac{\partial F}{\partial u_1}(u), \dots \frac{\partial F}{\partial u_n}(u))$ and η is the step size.

In our problem... we have

$$F = \sum_{i=1}^{m} \mathcal{L}(f, x_i, y_i) \text{ and } u = ((A_l, b_l))_{l=1}^{L}.$$

Since

$$\nabla_{((A_l,b_l))_{l=1}^{L}} F = \sum_{i=1}^{m} \nabla_{((A_l,b_l))_{l=1}^{L}} \mathcal{L}(f, x_i, y_i),$$

we need to compute

$$\frac{\partial \mathcal{L}(f,x,y)}{\partial (A_l)_{i,j}} \quad \text{and} \quad \frac{\partial \mathcal{L}(f,x,y)}{\partial (b_l)_i} \quad \text{for all } i, j, l.$$

Backpropagation

Data: A neural network f, a loss function $\mathcal{L}$, points x, y.

Result: The matrices $\nabla_{(A_l,b_l)_{l=1}^L} \mathcal{L}(f,x,y)$.

Algorithm:

Compute a_l, z_l for $l = 0, \ldots L$;

Set $\delta_L := 2(f(x) - y)$;

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

for $l = L-1$ to 1 **do**

$$\delta_l := \mathrm{diag}(\sigma'(z_l)) A_{l+1}^T \cdot \delta_{l+1};$$

$$\text{Then } \frac{\partial \mathcal{L}(f,x,y)}{\partial A_l} = \delta_l a_{l-1}^T \text{ and } \frac{\partial \mathcal{L}(f,x,y)}{\partial b_l} = \delta_l;$$

return $\nabla_{(A_l,b_l)_{l=1}^L} \mathcal{L}(f,x,y)$.

Stochastic Gradient Descent

Goal: Find a stationary point of

$$F = \sum_{i=1}^{m} F_i : \mathbb{R}^N \to \mathbb{R} \text{ where } F_i = \mathcal{L}(f, x_i, y_i).$$

Data: A neural network f, a loss function $\mathcal{L}$.

Result: A point u_n.

Algorithm:

- Set starting value u_0 and $n = 0$.
- **while** (error is large), **do**
  - Pick $i^* \in \{1, ..., m\}$ uniformly at random;
  - Update $u_{n+1} \leftarrow u_n - \eta \nabla F_{i^*}$;
  - Set $n + 1 \leftarrow n$;

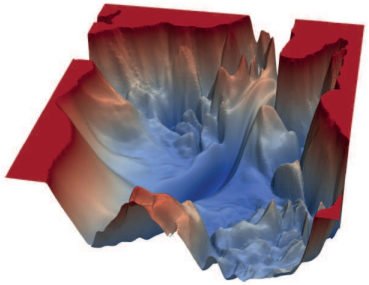
return u_n.

$\rightsquigarrow$ *Mini-Batch!*

Learning

Type of Results:

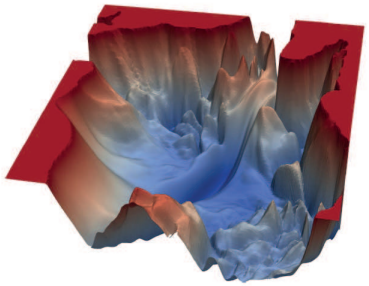
- Understanding the energy landscape: Global minima versus local minima.
- Designing new training algorithms.



Learning

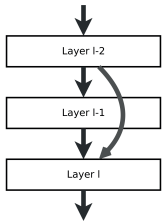
Type of Results:

- Understanding the energy landscape: Global minima versus local minima.
- Designing new training algorithms.



Approach initiated by (Haber, Ruthotto; 2017):

A Residual Neural Network has the following form for each layer:



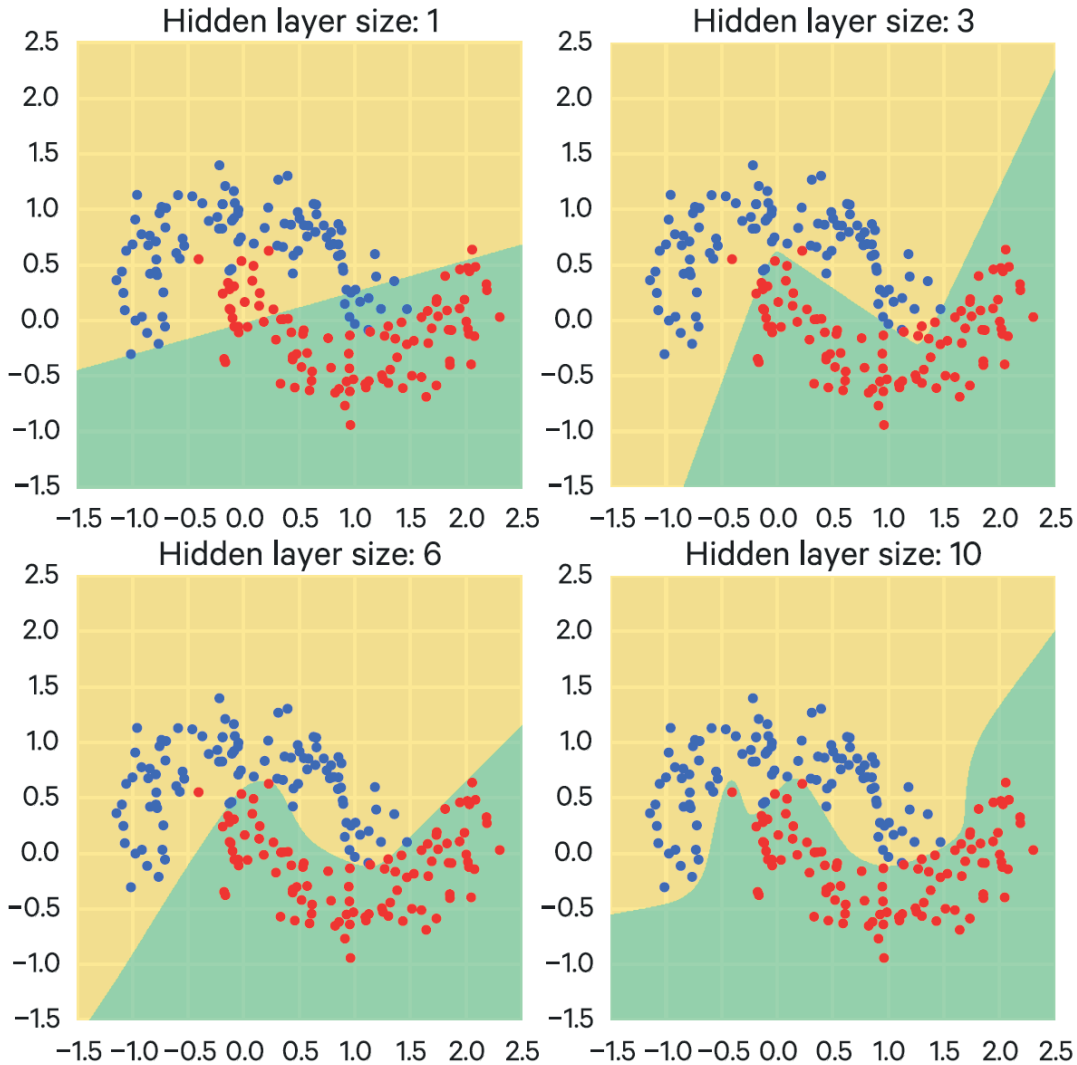
$$Y_{j+1} = Y_j + F(\theta^{(j)}, Y_j)$$

The ResNet can be seen as a forward Euler discretization (with a fixed step size of $\delta_t = 1$) of the initial value problem

$$\begin{aligned} \partial_t Y(\theta, t) &= F(\theta(t), Y(t)), \quad \text{for } t \in (0, T], \\ Y(\theta, 0) &= Y_0. \end{aligned}$$

$\rightsquigarrow$ *Stable parabolic (hyperbolic) DNNs!*

Generalization



No convincing theory yet!

Fundamental Questions concerning Deep Neural Networks

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

  $\rightsquigarrow$ *Applied Harmonic Analysis, Approximation Theory, ...*
- *Learning:*
  - Why does the current learning algorithm produce anything reasonable?
  - What are good starting values?

  $\rightsquigarrow$ *Differential Geometry, Optimal Control, Optimization, ...*
- *Generalization:*
  - Why do deep neural networks perform that well on data sets, which do not belong to the input-output pairs from a training set?
  - What impact has the depth of the network?

  $\rightsquigarrow$ *Learning Theory, Optimization, Statistics, ...*
- *Explainability:*
  - Why did a trained deep neural network reach a certain decision?
  - Which components of the input do contribute most?

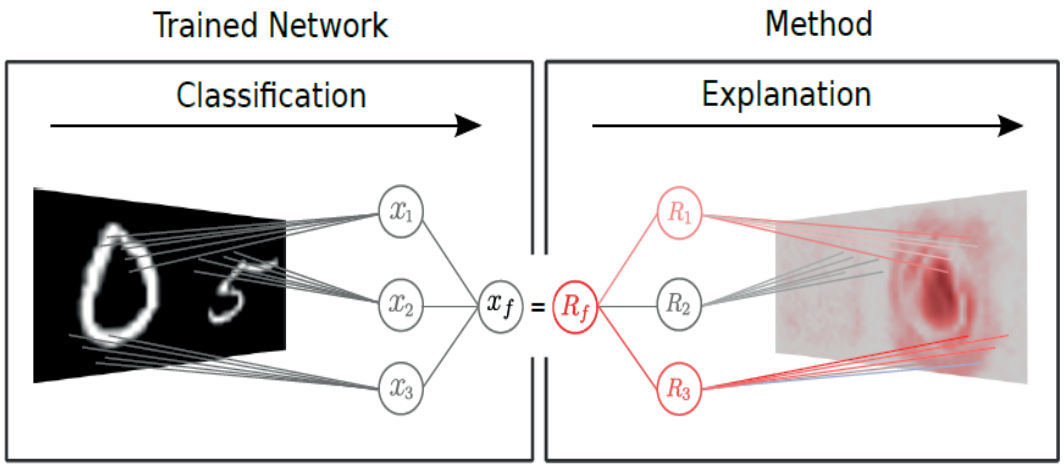
  $\rightsquigarrow$ *Information Theory, Uncertainty Quantification, ...*

Explainability

Main Questions:

Given a trained deep neural network.

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



Relevance Maps

Goal: Consider the realization of a neural network

$$f : \mathbb{R}^d \to \mathbb{R}.$$

Determine the relevance of each x_p of $x = (x_1, \ldots x_d)$ for the output $f(x)$.

Definition:

A collection $R = (R_p)_{p=1}^d$ of functions $R_p : \mathbb{R}^d \to \mathbb{R}$ is called *relevance map*.

- R is *non-negative*, if $R_p(x) \geq 0$ for all p, x.
- R is *conservative* with respect to f, if

$$\sum_p R_p(x) = f(x).$$

- R is *consistent*, if it is both non-negative and conservative.

Remark: Consistency implies that the decision $f(x)$ is distributed among the input pixels and only the contribution towards the decision is counted (not against it).

Sensitivity Analysis

Definition:

Assume f is continuously differentiable. Then

$$R_p(x) := \left(\frac{\partial f(x)}{\partial x_p}\right)^2$$

is a relevance map called *sensitivity analysis*.

Remarks:

- This relevance map is non-negative, but not conservative or consistent.
- Sensitivity analysis only uses ∇f, but not the decision $f(x)$. It answers the question *Changing which pixels makes the image look less/more like a cat?"*, but not *"Which pixels make the image a cat?"*.

Taylor Decomposition Relevance Map

Definition (Müller et al.; 2017):

Assume f is continuously differentiable and $\tilde{x}$ a suitably chosen root point of f, for instance $f(\tilde{x}) = 0$. Then

$$R_p(x) := \frac{\partial f(\tilde{x})}{\partial x_p}(x_p - \tilde{x}_p)$$

is the *Taylor decomposition relevance map*.

Remarks:

- The idea is to choose a root point $\tilde{x}$ near x which is neutral with respect to f in the sense of $f(x) = 0$.
- Up to second-order terms, this relevance map is conservative:

$$\begin{aligned} f(x) &= f(\tilde{x}) + \nabla f(\tilde{x})^T (x - \tilde{x}) + O(\|x - \tilde{x}\|^2) \\ &= \sum_p R_p(x) + O(\|x - \tilde{x}\|^2). \end{aligned}$$

- This relevance map can be efficiently computed by starting at $f(x)$ and going reversely through the network.

Explainability

"Easier" Subproblem:

Given $\Phi : \{0,1\}^n \to \{0,1\}$, $x \in \{0,1\}^n$, $k \in \mathbb{N}$, and $\delta > 0$. Compute a set of significant pixels $S \subset \{0,1\}^n$, $\#(S) \leq k$ with

$$\frac{\#\{y \in \{0,1\}^n : x_S = y_S \text{ and } \Phi(x) = \Phi(y)\}}{\#\{y \in \{0,1\}^n : x_S = y_S\}} \geq \delta.$$

Theorem (Wäldchen, Macdonald, Hauch, K; 2019):

"This problem is NN^{PP}-complete and NP-hard to approximate."

Explainability

"Easier" Subproblem:

Given $\Phi : \{0,1\}^n \to \{0,1\}$, $x \in \{0,1\}^n$, $k \in \mathbb{N}$, and $\delta > 0$. Compute a set of significant pixels $S \subset \{0,1\}^n$, $\#(S) \leq k$ with

$$\frac{\#\{y \in \{0,1\}^n : x_S = y_S \text{ and } \Phi(x) = \Phi(y)\}}{\#\{y \in \{0,1\}^n : x_S = y_S\}} \geq \delta.$$

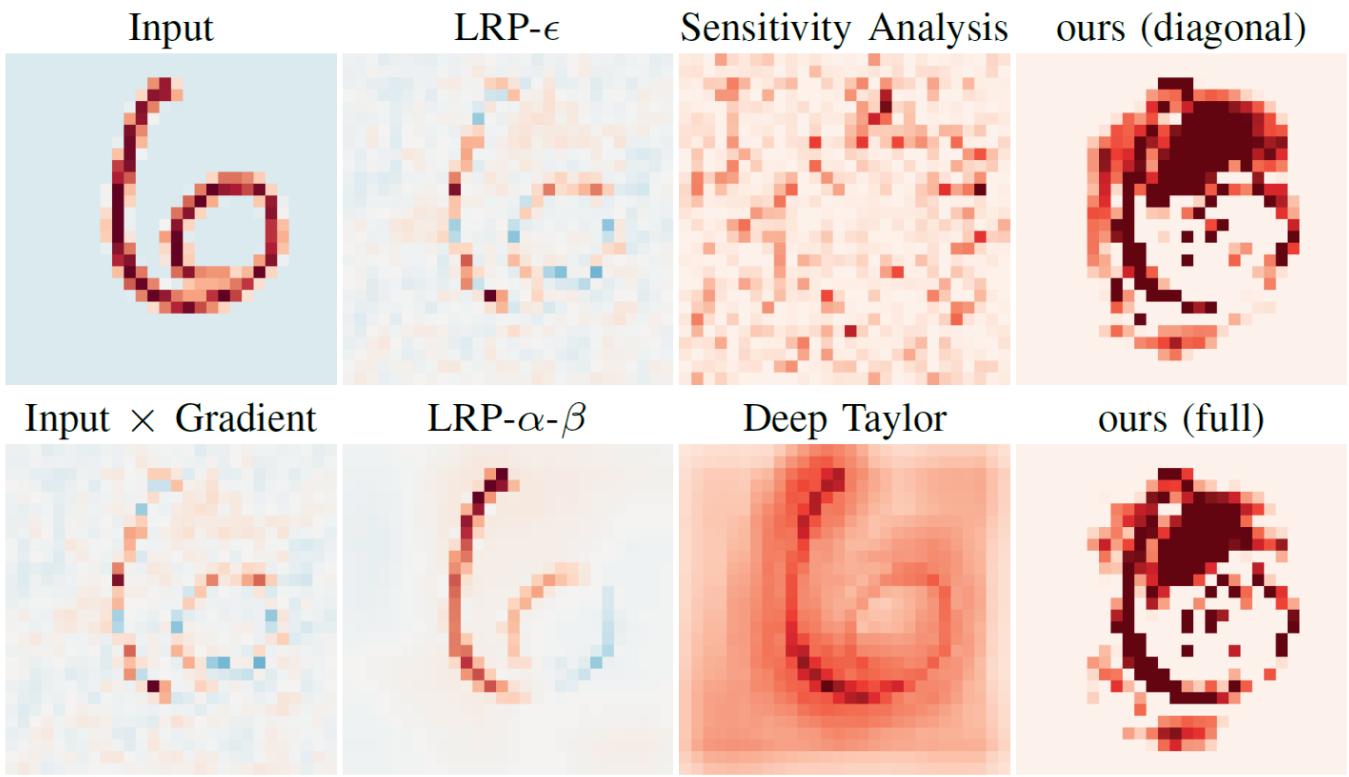
Theorem (Wäldchen, Macdonald, Hauch, K; 2019):

"This problem is NN^{PP}-complete and NP-hard to approximate."

$\rightsquigarrow$ *Solution strategy of a relaxed optimization problem based on assumed density filtering (Wäldchen, Macdonald, Hauch, K; 2019)!*

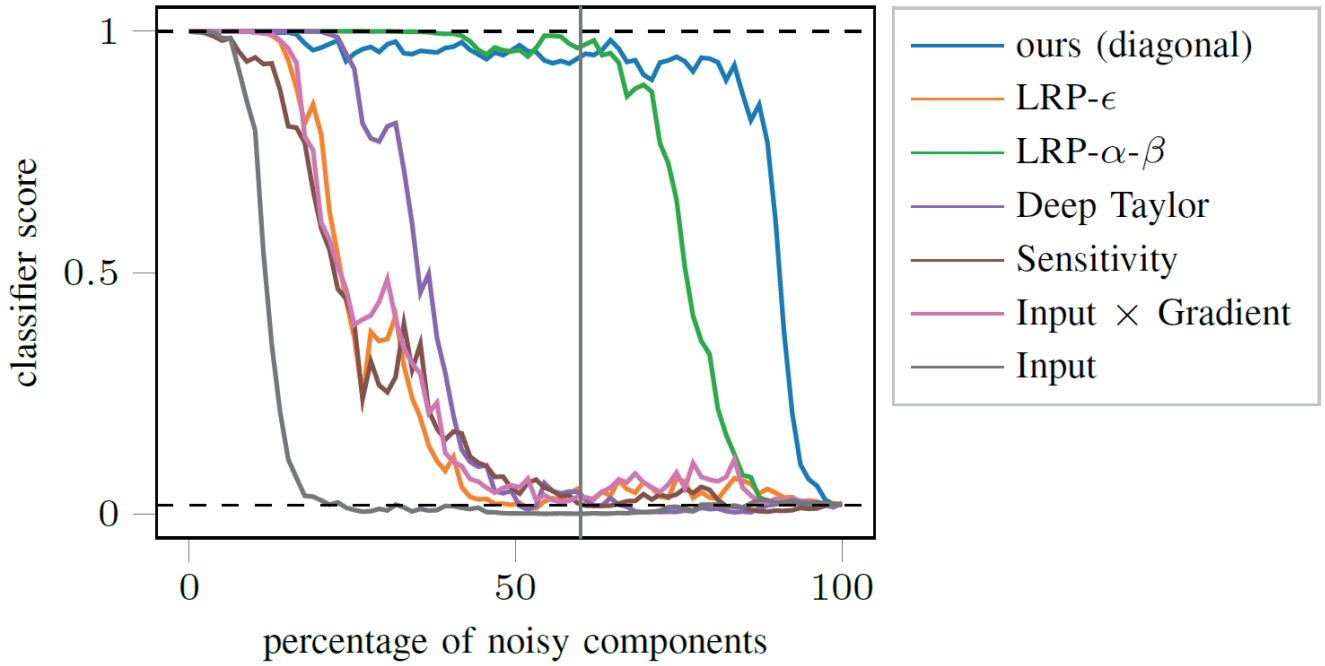
Numerical Results, I

(Wäldchen, Macdonald, Hauch, K; 2019)



Numerical Results, II

(Wäldchen, Macdonald, Hauch, K; 2019)



Fundamental Questions concerning Deep Neural Networks

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

  $\rightsquigarrow$ *Applied Harmonic Analysis, Approximation Theory, ...*
- *Learning:*
  - Why does the current learning algorithm produce anything reasonable?
  - What are good starting values?

  $\rightsquigarrow$ *Differential Geometry, Optimal Control, Optimization, ...*
- *Generalization:*
  - Why do deep neural networks perform that well on data sets, which do not belong to the input-output pairs from a training set?
  - What impact has the depth of the network?

  $\rightsquigarrow$ *Learning Theory, Optimization, Statistics, ...*
- *Explainability:*
  - Why did a trained deep neural network reach a certain decision?
  - Which components of the input do contribute most?

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

THANK YOU!

References available at:

www.math.tu-berlin.de/∼kutyniok

Code available at:

www.ShearLab.org

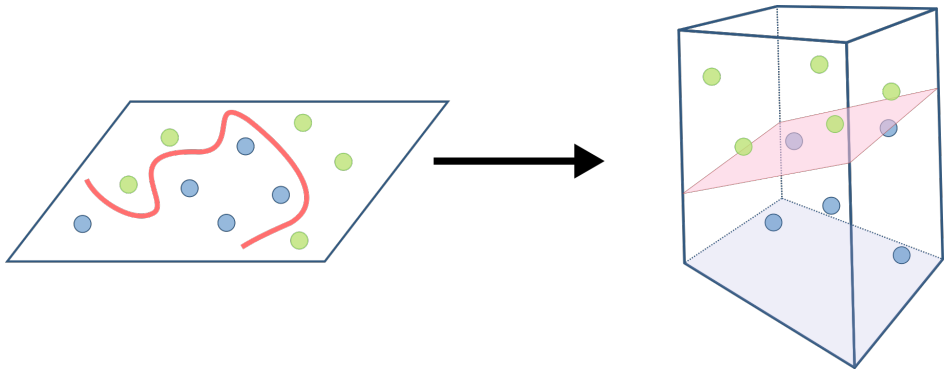
Related Books:

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

Example for Hypothesis Space

Intuition behind Feature Maps:

- Suppose there exists a suitable similarity measure $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ on $\mathcal{X}$ and we would like to search for the closest points to some $x \in \mathcal{X}$.
- Assume there exists a map $\Phi : \mathcal{X} \to \mathbb{R}^n$ which linearizes K as $K(x, x') = \langle \Phi(x), \Phi(x') \rangle$.



Question: For which K does there exist such a feature map?

Mercer Kernels

Definition:

Fix some map $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$.

- K is called *symmetric*, if $K(x, x') = K(x', x)$ for all $x, x' \in \mathcal{X}$.
- Let $\mathbf{x} := \{x_1, ..., x_k\} \subseteq \mathcal{X}$. Then the matrix $K(\mathbf{x}) \in \mathbb{R}^{k \times k}$ with entries $K(x_i, x_j)$ for $i, j = 1, ...k$ is called a *Gramian* of K in $\mathbf{x}$.
- K is called *positive semi-definite*, if everyone of its Gramians is always positive-semidefinite.
- K is a *Mercer Kernel*, if it is symmetric, positive semi-definite and continuous.

Example:

- Let $f : \mathbb{R}_0^+ \to \mathbb{R}$ be strictly monotone and let K be defined by $K(x, x') := f(\|x - x'\|^2)$. This is a Mercer kernel.
- In particular, the *Gaussian kernel*

$$K(x, x') = e^{\frac{-\|x-x'\|^2}{c^2}}, \quad c > 0,$$

is a Mercer kernel.

Reproducing Kernel Hilbert Spaces

Theorem (Mercer Theorem):

Let K be a Mercer kernel. Then there exists a unique Hilbert space $\mathcal{H}_K$ of functions defined on $\mathcal{X}$ mapping into $\mathbb{R}$ with

(i) the functions $K_x : x' \to K(x, x')$ belong to $\mathcal{H}_K$ for all $x \in \mathcal{X}$.

(ii) $\overline{\text{span}}\{K_x : x \in \mathcal{X}\} = \mathcal{H}_K$.

(iii) For all $f \in \mathcal{H}_K$ and $x \in \mathcal{X}$, we have $f(x) = \langle f, K_x \rangle_{\mathcal{H}_K}$, which in particular means that $K(x, x') = \langle K_x, K_{x'} \rangle$ for all $x, x' \in \mathcal{X}$.

The spaces $\mathcal{H}_K$ are called *Reproducing Kernel Hilbert Spaces (RKHS)*.

◂ Return