Mathematics of Deep Neural Networks

Gitta Kutyniok

(Technische Universität Berlin)

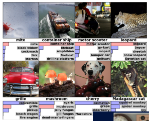
AIT Workshop of the Information Theory Group
Technische Universität Berlin, May 4, 2018

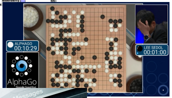
Deep Neural Networks

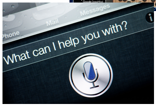
Deep neural networks have recently shown impressive results in a variety of real-world applications.

Some Examples...

- Image classification (ImageNet).
  $\rightsquigarrow$ *Hierarchical classification of images.*
- Games (AlphaGo).
  $\rightsquigarrow$ *Success against world class players.*
- Speech Recognition (Siri).
  $\rightsquigarrow$ *Current standard in numerous cell phones.*





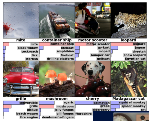


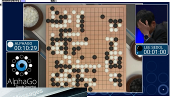
Deep Neural Networks

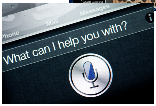
Deep neural networks have recently shown impressive results in a variety of real-world applications.

Some Examples...

- Image classification (ImageNet).
  ⇝ *Hierarchical classification of images.*
- Games (AlphaGo).
  ⇝ *Success against world class players.*
- Speech Recognition (Siri).
  ⇝ *Current standard in numerous cell phones.*







Very few theoretical results explaining their success!

More Highlights: Image Classification, II

Result of a neural network trained to produce describing sentences for images (Karpathy and Fei-Fei; 2017):

"black and white dog jumps over bar."

"young girl in pink shirt is swinging on swing."

"man in blue wetsuit is surfing on wave."

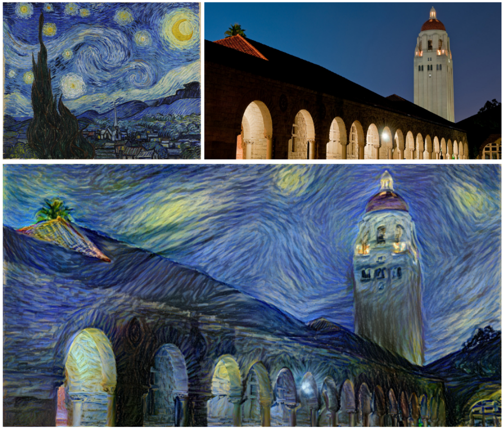
"baseball player is throwing ball in game."

"woman is holding bunch of bananas."

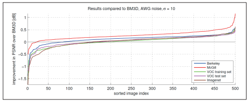
"black cat is sitting on top of suitcase."

Even more Highlights: Style Transfer

Result of a neural network trained to transfer a style (Gatys, Ecker, and Bethge; 2015):

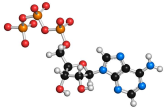


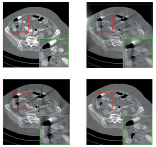
Further Applications of Deep Neural Networks



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 (K, März, Samek, Srinivasan; 2018).*

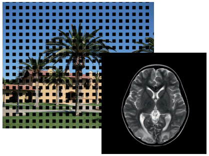




Inverse Problems and Deep Neural Networks

Examples:

- Denoising.
- Inpainting.
- Magnetic Resonance Tomography.
- ...



Generalized Tikhonov Regularization:

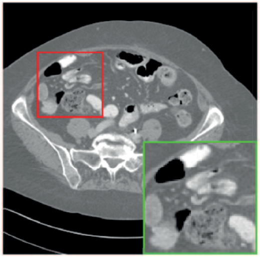
Given an ill-posed inverse problem $Kx = y$, where $K : X \to Y$, an approximate solution $x^\alpha \in X$, $\alpha > 0$, can be determined by

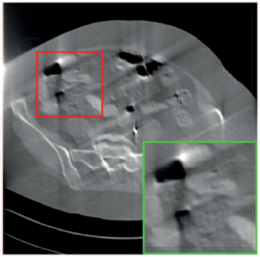
$$\min_{\tilde{x}} \|K\tilde{x} - y\|^2 + \alpha \mathcal{P}(\tilde{x}).$$

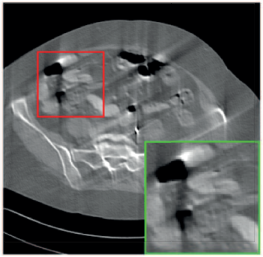
Deep Learning Approaches:

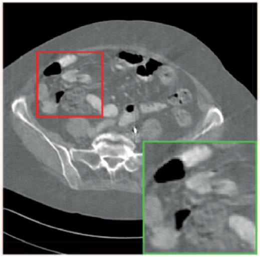
- Pure deep learning.
- Combine the "physics" of the problem with deep learning; often by invoking the adjoint operator K^*.

Limited-Angle Tomography

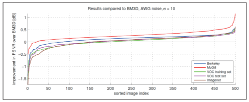

Original


Filtered Backprojection
(PSNR 24.9)


Sparse Regularization with Shearlets
(PSNR 37.3)

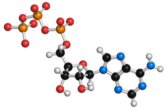

Combined Shearlet-DNN Approach
(PSNR 43.4)

Further Applications of Deep Neural Networks

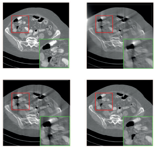


Some Examples from Areas in Mathematics...

- Imaging Sciences.
  $\rightsquigarrow$ *Image denoising (Burger, Schuler, Harmeling; 2012).*

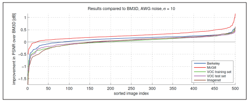


- PDE Solvers.
  $\rightsquigarrow$ *Schrödinger equation (Rupp, Tkatchenko, Müller, von Lilienfeld; 2012).*



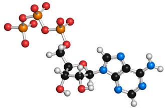
- Inverse Problems.
  $\rightsquigarrow$ *Limited-angle tomography (K, März, Samek, Srinivasan; 2018).*

Further Applications of Deep Neural Networks

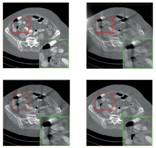


Some Examples from Areas in Mathematics...

- 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 (K, März, Samek, Srinivasan; 2018).*

Deep, Deep Trouble:

Deep Learnings Impact on Image Processing, Mathematics, and Humanity

Michael Elad (CS, Technion), SIAM News

A bit of history...

First Appearance of Neural Networks

Key Task of McCulloch and Pitts (1943):

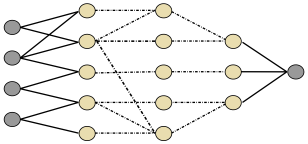
- Develop an algorithmic approach to learning.
- Mimicking the functionality of the human brain.

Goal: Artifical Intelligence!

First Neural Networks

Neural Networks:

- Neurons arranged in layers.
- Layers connected by weighted edges.
- Training algorithms learn the weights.
- Training based on a training data set.



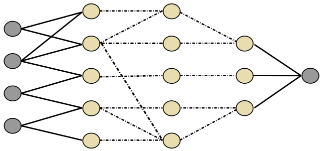
Results:

- Significant theoretical progress.
- Barely any practically interesting applications.

Neural Networks

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*.
- $W_\ell : \mathbb{R}^{N_{\ell-1}} \to \mathbb{R}^{N_\ell}$, $\ell = 1, \ldots, L$: Affine linear maps.

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

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

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

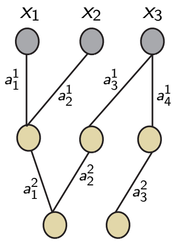
Sparse Connectivity

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

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

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

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

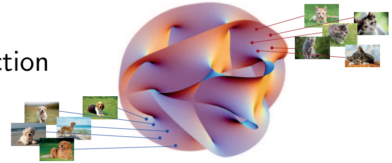


Definition: We write $\Phi \in \mathcal{NN}_{L,M,d,\sigma}$, where M number of edges with non-zero weight. A DNN with small M is called *sparsely connected*.

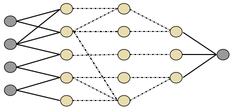
Training of Deep Neural Networks

High-Level Set Up:

- Given are (random) samples of a function such as $f : \mathcal{M} \to \{1, 2, \ldots, K\}$.



- Given 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 $(W_\ell)_{\ell=1}^L = (A_\ell \cdot + b_\ell)_{\ell=1}^L$, i.e., the weights and offsets, yielding the network

$$\Phi : \mathbb{R}^d \to \mathbb{R}^{N_L}, \quad \Phi(x) = W_L\sigma(W_{L-1}\sigma(\ldots\sigma(W_1(x))).$$

This is often done by backpropagation, a special case of 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.*

Nowadays we see various applications such as...

- ...the previously discussed ones.
- ...automatic cars.

Second Appearance of Neural Networks

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

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

Nowadays we see various applications such as...

- ...the previously discussed ones.
- ...automatic cars.

The future might see much more sensitive applications such as...

- ...medical applications.
- ...business/law applications.
- ...military applications.

Problem with Deep Neural Networks

Current Situation:

- Setting up a deep neural network for a particular application is trail-and-error.
- Training a deep neural network is very unpredictable.
- Working on applications typically require large teams.
- No knowledge about why a deep neural network made a decision.

Very few theoretical results explaining their success!

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, ...*

Outline

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!*

Example for Task T, I

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.

Handwritten Digits:

The function f should, for instance, satisfy $f($ $) = 5$.

Example for Task T, II

Regression Task:

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

Some Applications:

- Expected claim amount of an insured person.
- Prediction of future prices of securities.

Close
Close
Close
Buy
Stop Loss
Buy
Stop Loss
Buy
Stop Loss
1
2

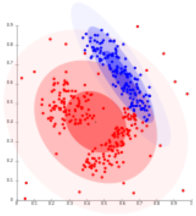
Example for Task T, III

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.

Some Applications:

- 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 it 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*

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

- 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*

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

Linear Regression as one Example, III

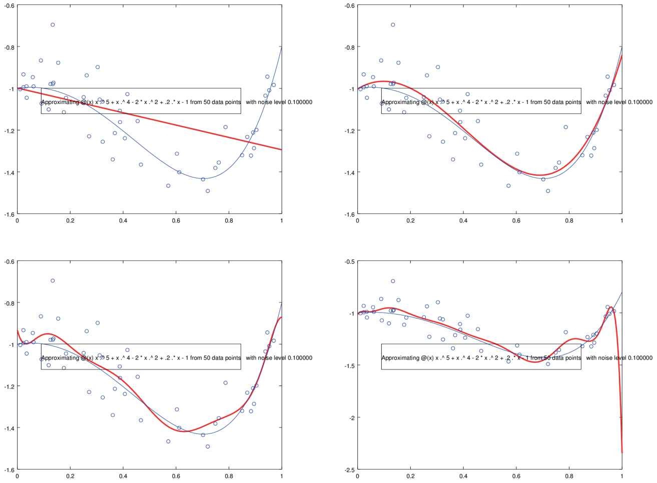
Computing the Empirical Target Function:

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

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

$$f_* := \sum_{i=1}^{\ell} (\mathbf{w}_*)_i \varphi_i.$$

Underfitting versus Overfitting



General Problem Setting

Assumption:

Let

(a) $X \subseteq \mathbb{R}^n$ be compact,

(b) $Y = \mathbb{R}^k$ (often we have $k = 1$),

(c) ϱ be a probability measure on $Z := X \times Y$.

The Learning Problem:

Find a function $f : X \to Y$, which minimizes the *(least square) error*

$$\mathcal{E}(f) := \int_Z (f(x) - y)^2 d\varrho(x, y).$$

That means that the goal is to determine

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

Solution of the Problem

Definition:

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

$$\varrho_X(S) = \varrho(\pi_X^{-1}(S)),$$

where $\pi_X : X \times Y \to X,\ (x,y) \mapsto x.$ Then also

$$\int_Z \phi(x,y) d\varrho(x,y) = \int_X \left(\int_Y \phi(x,y) d\varrho(y|x) \right) d\varrho_X(x)$$

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

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

Solution of the Problem

Definition:

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

$$\varrho_X(S) = \varrho(\pi_X^{-1}(S)),$$

where $\pi_X : X \times Y \to X,\ (x,y) \mapsto x$. Then also

$$\int_Z \phi(x,y) d\varrho(x,y) = \int_X \left(\int_Y \phi(x,y) d\varrho(y|x) \right) d\varrho_X(x)$$

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

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

Proposition:

For every integrable function $f : X \to Y$ we have

$$\mathcal{E}(f) = \int_X (f(x) - f_\varrho(x))^2 d\varrho_X(x) + \sigma_\varrho^2$$

The True Situation

Definition:

Let $z := ((x_1, y_1), ..., (x_m, y_m))$ be a sample in Z^m. We then define the *empirical error of $f : X \to Y$ with respect to z* by

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

Remarks:

- The error $\mathcal{E}(f)$ cannot be computed because we don't know the probability measure, but the empirical error $\mathcal{E}_z(f)$ can be.
- Bounding $\mathcal{E}(f) - \mathcal{E}_z(f)$ allows us to bound the actual error $\mathcal{E}(f)$ from the observed $\mathcal{E}_z(f)$ and give us some hope to approximate the regression function.

Assuming a Hypothesis Space

Definition:

- Let $\mathcal{H}$ be a compact subset of $C(X)$, equipped with the norm
$$\|f\|_\infty := \sup_{x \in X} |f(x)|.$$
We call $\mathcal{H}$ *hypothesis* or *model space*.
- The *target function* $f_\mathcal{H} \in \mathcal{H}$ is defined by
$$f_\mathcal{H} := \mathrm{argmin}_{f \in \mathcal{H}} \mathcal{E}(f).$$
- Let $z = ((x_1, y_1), ...(x_m, y_m)) \in Z^m$ be a sample. The *empirical target function* is defined as
$$f_{\mathcal{H},z} = \mathrm{argmin}_{f \in \mathcal{H}} \mathcal{E}_z(f).$$

Assuming a Hypothesis Space

Definition:

- Let $\mathcal{H}$ be a compact subset of $C(X)$, equipped with the norm
$$\|f\|_\infty := \sup_{x \in X} |f(x)|.$$
We call $\mathcal{H}$ *hypothesis* or *model space*.
- The *target function* $f_{\mathcal{H}} \in \mathcal{H}$ is defined by
$$f_{\mathcal{H}} := \operatorname{argmin}_{f \in \mathcal{H}} \mathcal{E}(f).$$
- Let $z = ((x_1, y_1), \dots (x_m, y_m)) \in Z^m$ be a sample. The *empirical target function* is defined as
$$f_{\mathcal{H},z} = \operatorname{argmin}_{f \in \mathcal{H}} \mathcal{E}_z(f).$$

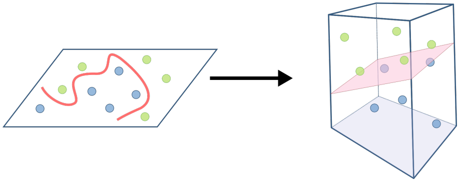
Corollary:

Under the above assumptions, we have
$$f_{\mathcal{H}} = \operatorname{argmin}_{f \in \mathcal{H}} \int_X (f(x) - f_\varrho(x))^2 d\varrho_X(x).$$

Example for Hypothesis Space

Intuition behind Feature Maps:

- Suppose there exists a suitable similarity measure $K : X \times X \to \mathbb{R}$ on X and we would like to search for the closest points to some $x \in X$.
- Assume there exists a map $\Phi : X \to \mathbb{R}^n$ 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 $K : X \times X \to \mathbb{R}$.

- K is called *symmetric*, if $K(x, x') = K(x', x)$ for all $x, x' \in X$.
- Let $\mathbf{x} := \{x_1, \ldots, x_k\} \subseteq X$. Then the matrix $K(\mathbf{x}) \in \mathbb{R}^{k \times k}$ with entries $K(x_i, x_j)$ for $i, j = 1, \ldots k$ 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 X mapping into $\mathbb{R}$ with

1. the functions $K_x : x' \to K(x, x')$ belong to $\mathcal{H}_K$ for all $x \in X$.
2. $\overline{\mathrm{span}}\{K_x : x \in X\} = \mathcal{H}_K$.
3. For all $f \in \mathcal{H}_K$ and $x \in X$, 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 $x, x' \in X$.

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

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 X mapping into $\mathbb{R}$ with

1. the functions $K_x : x' \to K(x,x')$ belong to $\mathcal{H}_K$ for all $x \in X$.
2. $\overline{\mathrm{span}}\{K_x : x \in X\} = \mathcal{H}_K$.
3. For all $f \in \mathcal{H}_K$ and $x \in X$, 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 $x, x' \in X$.

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

Proposition:
Let $\mathcal{H}_K$ be a reproducing kernel Hilbert space, where the kernel K is defined over some X, which is assumed to be compact. For $R > 0$, $B_R(0)$ is a hypothesis space.

Sample and Approximation Error

Definition:

Let $z \in Z^m$ be a sample, let $\mathcal{H}$ be a hypothesis space and let $f_{\mathcal{H}}$ and $f_{\mathcal{H},z}$ be defined as before. Then let

$$\mathcal{E}_{\mathcal{H}}(f) := \mathcal{E}(f) - \mathcal{E}(f_{\mathcal{H}})$$

be the *error of f in $\mathcal{H}$*. Then the empirical error $\mathcal{E}(f_{\mathcal{H},z})$ decomposes as

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

Remarks:

- *Sample error*: This originates from the samples not approximating the regression function well enough.
- *Approximation error*: This emerges from not choosing the right space.
- Typically, enlarging $\mathcal{H}$ will reduce the approximation error, but the sampling error increases. This is called the *Bias-Variance trade-off*.

Universally Best Method?

General Problem:

Given samples $((x_i, y_i))_{i=1}^m \in Z^m (= (X \times Y)^m)$.

- Learn f which minimizes the error $\mathcal{E}(f) := \int_Z (f(x) - y)^2 d\varrho(x, y)$.
- With this, also learn the probability distribution ϱ.

Determine according to which probability measure the samples are generated!

Universally Best Method?

General Problem:

Given samples $((x_i, y_i))_{i=1}^m \in Z^m (= (X \times Y)^m)$.

- Learn f which minimizes the error $\mathcal{E}(f) := \int_Z (f(x) - y)^2 d\varrho(x, y)$.
- With this, also learn the probability distribution ϱ.

Determine according to which probability measure the samples are generated!

Approaches:

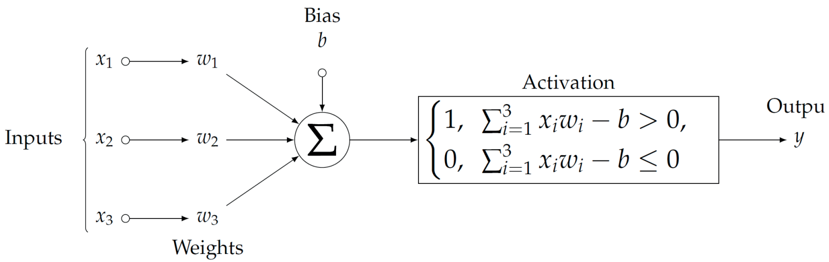
- (Regularized) least squares
- Maximum A Posteriori (MAP) estimate
- Principal Component Analysis (PCA)
- (Kernel) Support Vector Machines (SVM)
- ...

Aim for a universally best method!!!

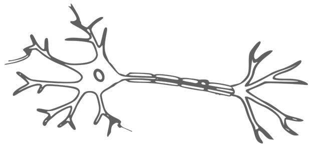
Deep Neural Networks: Basics

Artificial Neurons

Mimic the human brain!



Bias
b
x_1
w_1
Inputs
x_2
w_2
x_3
w_3
Weights
Σ
Activation
$\begin{cases} 1, & \sum_{i=1}^{3} x_i w_i - b > 0, \\ 0, & \sum_{i=1}^{3} x_i w_i - b \leq 0 \end{cases}$
Outpu
y



Artificial Neurons

Definition: An *artificial neuron* with *weights* $w_1, \ldots, w_n \in \mathbb{R}$, *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

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

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

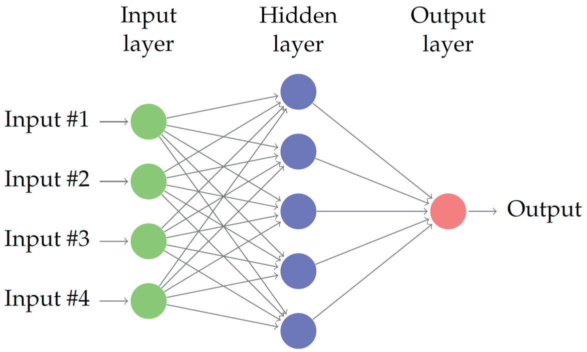
Examples of Activation Functions:

- 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:

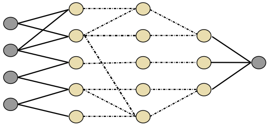
An *artificial neural network* is a graph which consists of artificial neurons. A *feed-forward neural network* is a directed, acyclic graph. All other neural networks are called *recurrent neural networks*.



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.

Getting Familiar with the Notions...

Examples:

(1) Let Φ be given by

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

Then $\Phi \in \mathcal{NN}_{d,M,N,L}$ with

Getting Familiar with the Notions...

Examples:

(1) Let Φ be given by

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

Then $\Phi \in \mathcal{NN}_{d,M,N,L}$ with

$$d = 3, \; M = 7, \; L = 2, \text{ and } N = 8.$$

Getting Familiar with the Notions...

Examples:

(1) Let Φ be given by

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

Then $\Phi \in \mathcal{NN}_{d,M,N,L}$ with

$$d = 3,\ M = 7,\ L = 2,\ \text{and}\ N = 8.$$

(2) An artificial neuron is a realization with activation function σ of a neural network $\Phi \in \mathcal{NN}_{d,M,N,L}$ with

Getting Familiar with the Notions...

Examples:

(1) Let Φ be given by

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

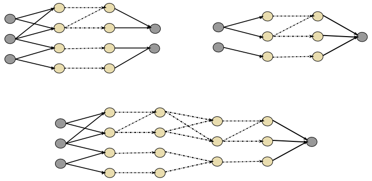
Then $\Phi \in \mathcal{NN}_{d,M,N,L}$ with

$$d = 3,\ M = 7,\ L = 2, \text{ and } N = 8.$$

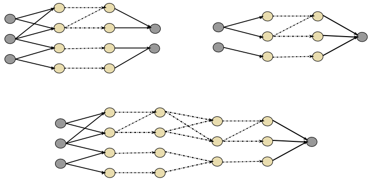
(2) An artificial neuron is a realization with activation function σ of a neural network $\Phi \in \mathcal{NN}_{d,M,N,L}$ with

$$M = d,\ L = d + 1, \text{ and } N = 1.$$

Basic Operations: Concatenation



Basic Operations: Concatenation



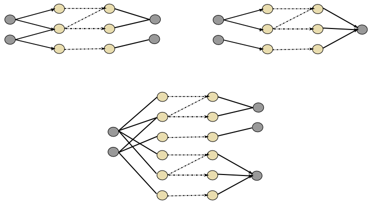
Lemma: Let $L_1, L_2 \in \mathbb{N}$ and $\Phi^i = ((A_1^i, b_1^i), ..., (A_{L_i}^i, b_{L_i}^i)), i = 1, 2$, be neural networks such that the input layer of Φ^1 has the same dimension as the output layer of Φ^2. Then, for any rectifier σ,

$$R_\sigma(\Phi^1 \circ \Phi^2) = R_\sigma(\Phi^1) \circ R_\sigma(\Phi^2) \quad \text{and} \quad L(\Phi^1 \circ \Phi^2) = L_1 + L_2 - 1,$$

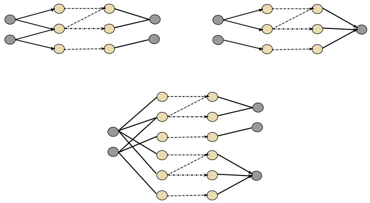
where $\Phi^1 \circ \Phi^2$ denotes the *concatenation of Φ^1 and Φ^2*:

$$((A_1^2, b_1^2), ...(A_{L_2-1}^2, b_{L_2-1}^2), (A_1^1 A_{L_2}^2, A_1^1 b_{L_2}^2 + b_1^1), (A_2^1, b_2^1), ...(A_{L_1}^1, b_{L_1}^1)).$$

Basic Operations: Parallelization



Basic Operations: Parallelization



Lemma: Let $L \in \mathbb{N}$ and $\Phi^i = ((A_1^i, b_1^i), ..., (A_L^i, b_L^i)), i = 1, 2....$ Then, for any rectifier σ and any $x \in \mathbb{R}^d$,

$$\begin{aligned} (R_\sigma(P(\Phi^1, \Phi^2)))(x) &= (R_\sigma(\Phi^1)(x), R_\sigma(\Phi^2)(x)), \\ L(P(\Phi^1, \Phi^2)) &= L(\Phi^1) + L(\Phi^2), \\ M(P(\Phi^1, \Phi^2)) &= M(\Phi^1) + M(\Phi^2), \end{aligned}$$

where $P(\Phi^1, \Phi^2)$ denotes the *parallelization of Φ^1 and Φ^2*:

$$P(\Phi^1, \Phi^2) := ((\hat{A}_1, \hat{b}_1), ..., (\hat{A}_L, \hat{b}_L)) \text{ with}$$

$$\hat{A}_1 = \begin{pmatrix} A_1^1 \\ A_1^2 \end{pmatrix}, \hat{b}_1 = \begin{pmatrix} b_1^1 \\ b_1^2 \end{pmatrix}, \hat{A}_l = \begin{pmatrix} A_l^1 & 0 \\ 0 & A_l^2 \end{pmatrix}, \hat{b}_l = \begin{pmatrix} b_l^1 \\ b_l^2 \end{pmatrix}, 1 \leq l \leq L.$$

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

$$R_{ReLU}(\Phi^1)(x) = x \quad \text{for all } x \in \mathbb{R}^d.$$

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.

Deep Neural Networks: Training

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

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

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 $\nabla F(u) = (\frac{\partial F}{\partial u_1}(u), \ldots \frac{\partial F}{\partial u_n}(u))$ 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 $l = 0, ...L$;
    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
        $\delta_l := \mathrm{diag}(\sigma'(z_l)) A_{l+1}^T \cdot \delta_{l+1};$
        Then $\frac{\partial \mathcal{L}(f,x,y)}{\partial A_l} = \delta_l a_{l-1}^T$ 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!*

Expressivity of Neural Networks: Universality

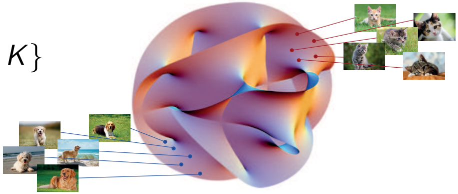
Expressivity

Main Task:

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

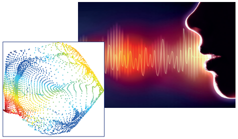
- Image Classification:

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



- 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?

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.

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

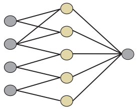
Universal Approximation Theorem

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

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

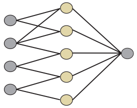


Universal Approximation Theorem

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



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

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.

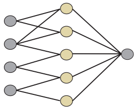
Universal Approximation Theorem

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



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

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!

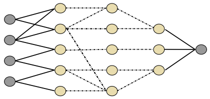
Effective Approximation

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

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

we have

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



Remark: Inspired by Donoho (1999) from classical approximation theory.

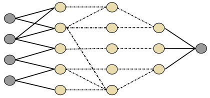
Effective Approximation

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

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

we have

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



Remark: Inspired by Donoho (1999) from classical approximation theory.

Task:

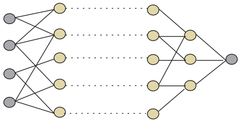
Analyze the expressive power of DNNs in terms of memory efficiency!

Expressivity of Sparsely Connected Deep Neural Networks

Sparsely Connected Deep Neural Networks

Key Problem:

- Deep neural networks employed in practice often consist of hundreds of layers.
- Training and operation of such networks pose formidable computational challenge.



$\rightsquigarrow$ *Employ deep neural networks with sparse connectivity!*

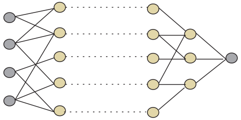
Example of Speech Recognition:

- Typically speech recognition is performed in the cloud (e.g. SIRI).
- New speech recognition systems (e.g. Android) can operate offline and are based on a sparsely connected deep neural network.

Sparsely Connected Deep Neural Networks

Key Problem:

- Deep neural networks employed in practice often consist of hundreds of layers.
- Training and operation of such networks pose formidable computational challenge.



$\rightsquigarrow$ *Employ deep neural networks with sparse connectivity!*

Example of Speech Recognition:

- Typically speech recognition is performed in the cloud (e.g. SIRI).
- New speech recognition systems (e.g. Android) can operate offline and are based on a sparsely connected deep neural network.

Key Challenge for Memory Efficient DNNs:

Approximation accuracy $\leftrightarrow$ *Complexity of approximating DNN in terms of sparse connectivity*

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 $(\varphi_i)_{i\in I}$.

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

Example: Wavelets

Definition (1D): Let $\phi \in L^2(\mathbb{R})$ be a scaling function and $\psi \in L^2(\mathbb{R})$ be a wavelet. Then the associated wavelet system is defined by

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

Example: Wavelets

Definition (1D): Let $\phi \in L^2(\mathbb{R})$ be a scaling function and $\psi \in L^2(\mathbb{R})$ be a wavelet. Then the associated wavelet system is defined by

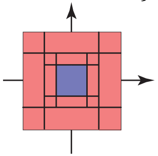
$$\{\phi(x-m) : m \in \mathbb{Z}\} \cup \{2^{j/2}\psi(2^j x - m) : j \geq 0, m \in \mathbb{Z}\}.$$

Definition (2D): A wavelet system is defined by

$$\{\phi^{(1)}(x-m) : m \in \mathbb{Z}^2\} \cup \{2^j \psi^{(i)}(2^j x - m) : j \geq 0, m \in \mathbb{Z}^2, i = 1,2,3\},$$

where

$$\phi^{(1)}(x) = \phi(x_1)\phi(x_2) \quad \text{and} \quad \begin{aligned} \psi^{(1)}(x) &= \phi(x_1)\psi(x_2), \\ \psi^{(2)}(x) &= \psi(x_1)\phi(x_2), \\ \psi^{(3)}(x) &= \psi(x_1)\psi(x_2). \end{aligned}$$



Example: Wavelets

Definition (1D): Let $\phi \in L^2(\mathbb{R})$ be a scaling function and $\psi \in L^2(\mathbb{R})$ be a wavelet. Then the associated wavelet system is defined by

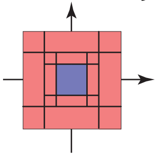
$$\{\phi(x-m) : m \in \mathbb{Z}\} \cup \{2^{j/2}\psi(2^j x - m) : j \geq 0, m \in \mathbb{Z}\}.$$

Definition (2D): A wavelet system is defined by

$$\{\phi^{(1)}(x-m) : m \in \mathbb{Z}^2\} \cup \{2^j \psi^{(i)}(2^j x - m) : j \geq 0, m \in \mathbb{Z}^2, i = 1,2,3\},$$

where

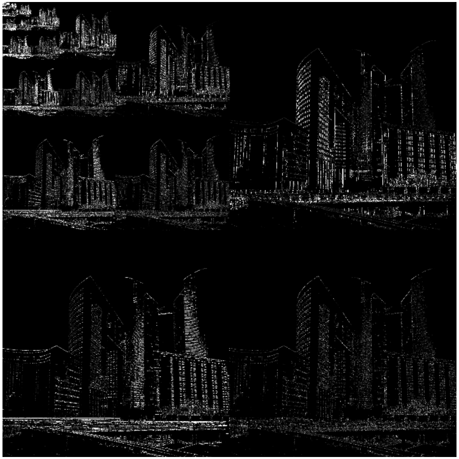
$$\phi^{(1)}(x) = \phi(x_1)\phi(x_2) \quad \text{and} \quad \begin{aligned} \psi^{(1)}(x) &= \phi(x_1)\psi(x_2), \\ \psi^{(2)}(x) &= \psi(x_1)\phi(x_2), \\ \psi^{(3)}(x) &= \psi(x_1)\psi(x_2). \end{aligned}$$



Approximation of some f with a wavelet ONB $(\psi_\lambda)_{\lambda \in \Lambda}$:

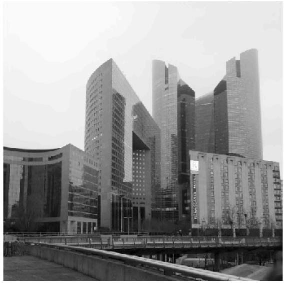
$$f_M = \sum_{\lambda \in \Lambda_M} c_\lambda \psi_\lambda, \quad \text{where } c_\lambda = \langle f, \psi_\lambda \rangle.$$

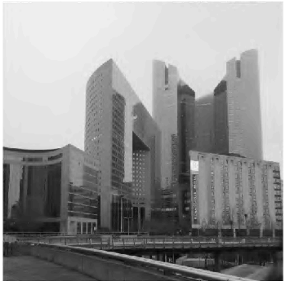
Wavelet Decomposition: JPEG2000



Wavelet Decomposition: JPEG2000

Original


25% Compression

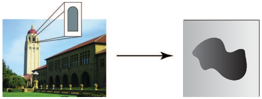

5% Compression

Fitting Model for Images

Definition (Donoho; 2001):
The set of cartoon-like functions $\mathcal{E}^2(\mathbb{R}^2)$ is defined by

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

where $B \subset [0,1]^2$ nonempty, simply connected with C^2-boundary, ∂B has bounded curvature, and $f_i \in C^2(\mathbb{R}^2)$ with supp $f_i \subseteq [0,1]^2$ and $\|f_i\|_{C^2} \leq 1$ for $i = 0, 1$.

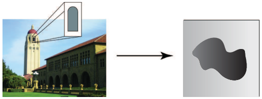


Fitting Model for Images

Definition (Donoho; 2001):
The set of cartoon-like functions $\mathcal{E}^2(\mathbb{R}^2)$ is defined by

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

where $B \subset [0,1]^2$ nonempty, simply connected with C^2-boundary, ∂B has bounded curvature, and $f_i \in C^2(\mathbb{R}^2)$ with supp $f_i \subseteq [0,1]^2$ and $\|f_i\|_{C^2} \leq 1$ for $i = 0, 1$.



Theorem:
Given a wavelet orthonormal basis $(\psi_\lambda)_{\lambda \in \Lambda} \subseteq L^2(\mathbb{R}^2)$, the decay rate of the L^2-error of best M-term approximation of $f \in \mathcal{E}^2(\mathbb{R}^2)$ is

$$\|f - f_M\|_2 \asymp M^{-\frac{1}{2}}, \quad M \to \infty, \quad \text{where } f_M = \sum_{\lambda \in \Lambda_M} c_\lambda \psi_\lambda.$$

But this is not the optimal rate (Donoho; 2001)!

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

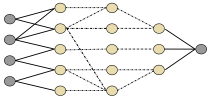
Approximation with Sparse Deep Neural Networks

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

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



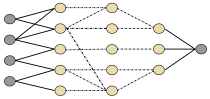
Approximation with Sparse Deep Neural Networks

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

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

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).

Challenges

(1) How many edges do we need for a certain accuracy?
$\rightsquigarrow$ *Conceptual lower bound on the number of edges of the DNN, which each learning algorithm has to obey!*

(2) Are there DNNs which are memory-optimal?
$\rightsquigarrow$ *Sharpness of the bound by explicit construction of optimal DNNs!*

(3) But what happens in practise using ReLUs and backpropagation?
$\rightsquigarrow$ *Success of certain (parallel) network topologies to reach optimal bound!*

A Lower Bound on Sparse Connectivity

Rate Distortion Theory

To use information theoretic arguments, we require the following notions from information theory:

Definition:
Let $d \in \mathbb{N}$. Then, for each $\ell \in \mathbb{N}$, we let

$$\mathfrak{E}^\ell := \{E : L^2(\mathbb{R}^d) \to \{0,1\}^\ell\}$$

denote the set of *binary encoders with length ℓ* and

$$\mathfrak{D}^\ell := \{D : \{0,1\}^\ell \to L^2(\mathbb{R}^d)\}$$

the set of *binary decoders of length ℓ.*

Rate Distortion Theory

Definition (continued):
For arbitrary $\varepsilon > 0$ and $\mathcal{C} \subset L^2(\mathbb{R}^d)$, the *minimax code length* $L(\varepsilon, \mathcal{C})$ is given by

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

and the *optimal exponent* $\gamma^*(\mathcal{C})$ is defined by

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

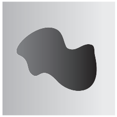
Interpretation:
The optimal exponent $\gamma^*(\mathcal{C})$ describes the dependence of the code length on the required approximation quality.

$\gamma^*(\mathcal{C})$ is a measure of the complexity of the function class!

Optimal Exponent

Example:

- Let $\mathcal{C} \subset B^s_{p,q}(\mathbb{R}^d)$ be bounded. Then we have
$$\gamma^*(\mathcal{C}) = \frac{d}{s}.$$
- (Donoho; 2001) Let $\mathcal{C} = \mathcal{E}^2(\mathbb{R}^2)$, the class of cartoon-like functions. Then we have
$$\gamma^*(\mathcal{C}) \geq 1.$$



A Fundamental Lower Bound

Theorem (Bölcskei, Grohs, K, and Petersen; 2017):
Let $d \in \mathbb{N}$, $\sigma : \mathbb{R} \to \mathbb{R}$, $c > 0$, and let $\mathcal{C} \subset L^2(\mathbb{R}^d)$. Further, let

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

satisfy 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

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

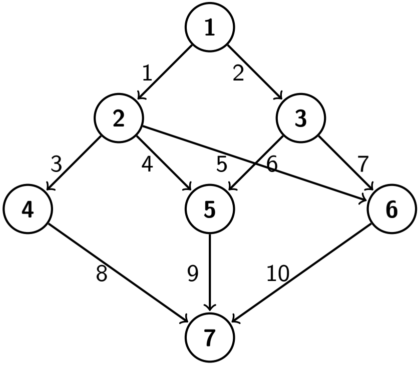
Then, for all $\gamma < \gamma^*(\mathcal{C})$,

$$\varepsilon^\gamma \sup_{f \in \mathcal{C}} \mathcal{M}(\textbf{Learn}(\varepsilon, f)) \to \infty \qquad \text{as } \varepsilon \to 0,$$

where $\mathcal{M}(\textbf{Learn}(\varepsilon, f))$ denotes 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:

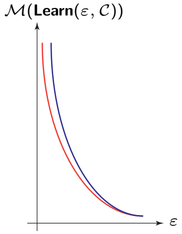
- # layers,
- # neurons in each layer,
- for each neuron in chronological order # the number of children,
- for each neuron in chronological order the indices of children,
- in chronological order the weights of edges.

A Fundamental Lower Bound

Some implications and remarks:

- 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

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



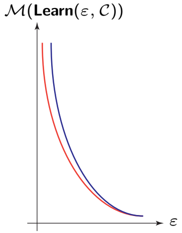
$\Rightarrow$ There exists a fundamental lower bound on the number of edges.

A Fundamental Lower Bound

Some implications and remarks:

- 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

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



$\Rightarrow$ There exists a fundamental lower bound on the number of edges.

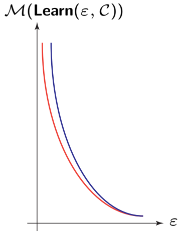
What happens for $\gamma = \gamma^*(\mathcal{C})$?

A Fundamental Lower Bound

Some implications and remarks:

- 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

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



$\Rightarrow$ There exists a fundamental lower bound on the number of edges.

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

- 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.

Back to the Effective Approximation

Corollary (Bölcskei, Grohs, K, and Petersen; 2017):
Given a function class $\mathcal{C} \subseteq L^2(\mathbb{R}^d)$ and an activation function σ with certain weak properties. Then the effective approximation rate $\gamma^{\mathit{eff}}(\mathcal{C}, \sigma)$ of $\mathcal{C}$ satisfies

$$\gamma^{\mathit{eff}}(\mathcal{C}, \sigma) \leq \frac{1}{\gamma^*(\mathcal{C})}.$$

Even though the 'classical' approximation rate can be unbounded, the effective one is bounded by the complexity of the function class!

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:

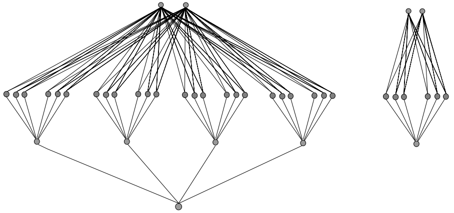
Can we exploit approximation results with representation systems?

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 $\varphi_i = R_\sigma(\Phi_i)$.

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

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



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 $\varphi_i = R_\sigma(\Phi_i)$.
- 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 $R_\sigma(\Phi) = \sum_{i\in I_M} c_i\varphi_i$.
- 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 $\varphi_i = R_\sigma(\Phi_i)$.
- 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

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

Road Map

General Approach:

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

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

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

DNNs have as much approximation power as most classical systems!

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 $\mathcal{C}$.

DNNs have as much approximation power as most classical systems!

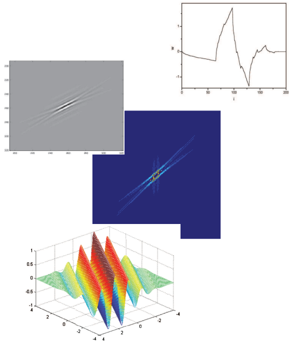
$\rightsquigarrow$ *But this does not yet control the size of the weights!*

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.

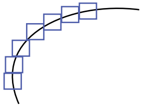
(ii) ψ can be realized by a neural network Φ_2.

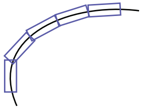
Also, Φ_1 and Φ_2 have the same number of edges up to a constant factor.

What are Shearlets?

Key Ideas of the Shearlet Construction

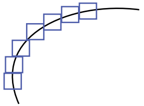
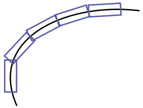
Wavelet versus Shearlet Approximation:





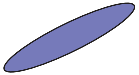
Key Ideas of the Shearlet Construction

Wavelet versus Shearlet Approximation:

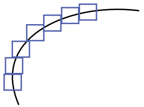
Parabolic scaling ('width ≈ length^2'):

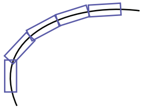
$$A_{2^j} = \begin{pmatrix} 2^j & 0 \\ 0 & 2^{j/2} \end{pmatrix}, \quad j \in \mathbb{Z}.$$



Key Ideas of the Shearlet Construction

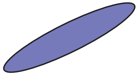
Wavelet versus Shearlet Approximation:





Parabolic scaling ('width $\approx$ length^2'):

$$A_{2^j} = \begin{pmatrix} 2^j & 0 \\ 0 & 2^{j/2} \end{pmatrix}, \quad j \in \mathbb{Z}.$$



Orientation via shearing:

$$S_k = \begin{pmatrix} 1 & k \\ 0 & 1 \end{pmatrix}, \quad k \in \mathbb{Z}.$$

Advantage:

- Shearing leaves the digital grid $\mathbb{Z}^2$ invariant.
- Uniform theory for the continuum and digital situation.

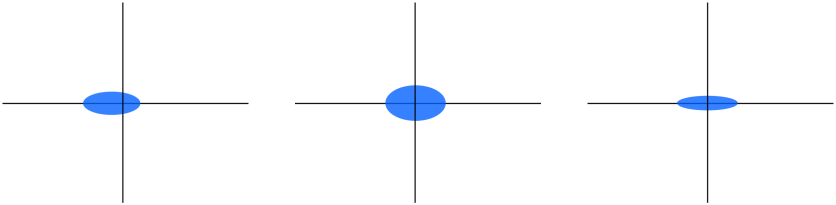
Shearlet Systems

Affine systems:

$$\{|\det M|^{1/2}\psi(M \cdot -m) : M \in G \subseteq GL_2,\ m \in \mathbb{Z}^2\}.$$

Definition (K, Labate; 2006):
For $\psi \in L^2(\mathbb{R}^2)$, the associated shearlet system is defined by

$$\{2^{\frac{3j}{4}}\psi(S_k A_{2^j} \cdot -m) \ :\ j,k \in \mathbb{Z}, m \in \mathbb{Z}^2\}.$$



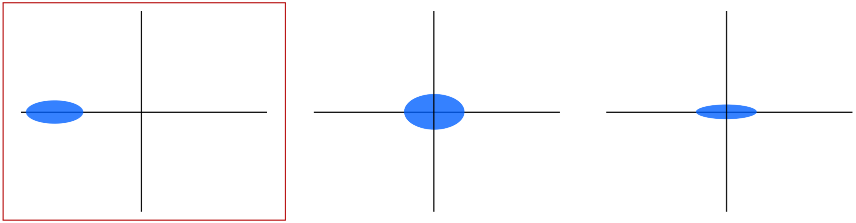
Shearlet Systems

Affine systems:

$$\{|\det M|^{1/2}\psi(M\,\cdot\,-m) : M \in G \subseteq GL_2,\ m \in \mathbb{Z}^2\}.$$

Definition (K, Labate; 2006):
For $\psi \in L^2(\mathbb{R}^2)$, the associated shearlet system is defined by

$$\{2^{\frac{3j}{4}}\psi(S_k A_{2^j}\cdot - m)\ :\ j,k \in \mathbb{Z}, m \in \mathbb{Z}^2\}.$$



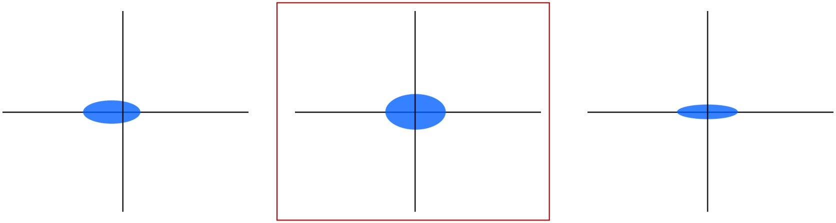
Shearlet Systems

Affine systems:

$$\{|\det M|^{1/2}\psi(M\,\cdot\,-m) : M \in G \subseteq GL_2,\ m \in \mathbb{Z}^2\}.$$

Definition (K, Labate; 2006):
For $\psi \in L^2(\mathbb{R}^2)$, the associated shearlet system is defined by

$$\{2^{\frac{3j}{4}}\psi(S_k A_{2^j}\cdot - m)\ :\ j,k \in \mathbb{Z}, m \in \mathbb{Z}^2\}.$$



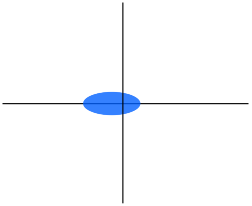
Shearlet Systems

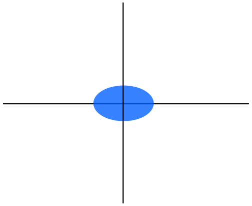
Affine systems:

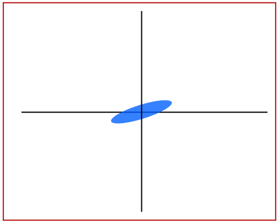
$$\{|\det M|^{1/2}\psi(M\ \cdot\ -m) : M \in G \subseteq GL_2,\ m \in \mathbb{Z}^2\}.$$

Definition (K, Labate; 2006):
For $\psi \in L^2(\mathbb{R}^2)$, the associated shearlet system is defined by

$$\{2^{\frac{3j}{4}}\psi(S_k A_{2^j} \cdot -m)\ :\ j,k \in \mathbb{Z}, m \in \mathbb{Z}^2\}.$$







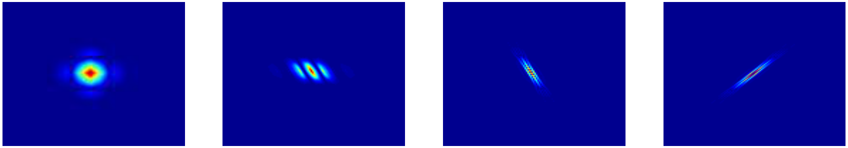
Shearlet Systems

Affine systems:

$$\{|\det M|^{1/2}\psi(M\,\cdot\,-m) : M \in G \subseteq GL_2,\ m \in \mathbb{Z}^2\}.$$

Definition (K, Labate; 2006):
For $\psi \in L^2(\mathbb{R}^2)$, the associated shearlet system is defined by

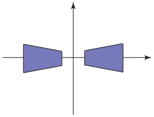
$$\{2^{\frac{3j}{4}}\psi(S_k A_{2^j}\,\cdot\,-m)\ :\ j,k \in \mathbb{Z}, m \in \mathbb{Z}^2\}.$$



Example of Classical (Band-Limited) Shearlet

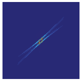
Let $\psi \in L^2(\mathbb{R}^2)$ be defined by

$$\hat{\psi}(\xi) = \hat{\psi}(\xi_1, \xi_2) = \hat{\psi}_1(\xi_1)\,\hat{\psi}_2(\tfrac{\xi_2}{\xi_1}),$$



where

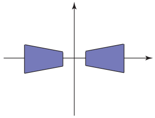
- ψ_1 wavelet, $\mathrm{supp}(\hat{\psi}_1) \subseteq [-2, -\frac{1}{2}] \cup [\frac{1}{2}, 2]$ and $\hat{\psi}_1 \in C^\infty(\mathbb{R})$.
- $\mathrm{supp}(\hat{\psi}_2) \subseteq [-1, 1]$ and $\hat{\psi}_2 \in C^\infty(\mathbb{R})$.



Example of Classical (Band-Limited) Shearlet

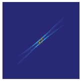
Let $\psi \in L^2(\mathbb{R}^2)$ be defined by

$$\hat{\psi}(\xi) = \hat{\psi}(\xi_1, \xi_2) = \hat{\psi}_1(\xi_1)\,\hat{\psi}_2(\tfrac{\xi_2}{\xi_1}),$$

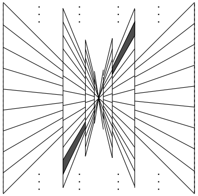


where

- ψ_1 wavelet, $\mathrm{supp}(\hat{\psi}_1) \subseteq [-2, -\frac{1}{2}] \cup [\frac{1}{2}, 2]$ and $\hat{\psi}_1 \in C^\infty(\mathbb{R})$.
- $\mathrm{supp}(\hat{\psi}_2) \subseteq [-1, 1]$ and $\hat{\psi}_2 \in C^\infty(\mathbb{R})$.



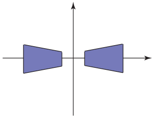
Induced tiling of Fourier domain:



Example of Classical (Band-Limited) Shearlet

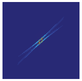
Let $\psi \in L^2(\mathbb{R}^2)$ be defined by

$$\hat{\psi}(\xi) = \hat{\psi}(\xi_1, \xi_2) = \hat{\psi}_1(\xi_1)\, \hat{\psi}_2(\tfrac{\xi_2}{\xi_1}),$$

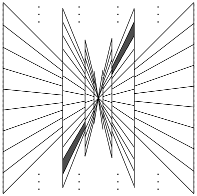


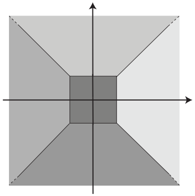
where

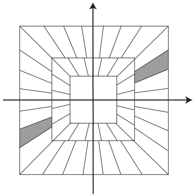
- ψ_1 wavelet, $\mathrm{supp}(\hat{\psi}_1) \subseteq [-2, -\frac{1}{2}] \cup [\frac{1}{2}, 2]$ and $\hat{\psi}_1 \in C^\infty(\mathbb{R})$.
- $\mathrm{supp}(\hat{\psi}_2) \subseteq [-1, 1]$ and $\hat{\psi}_2 \in C^\infty(\mathbb{R})$.



Induced tiling of Fourier domain:







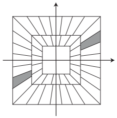
(Cone-adapted) Discrete Shearlet Systems

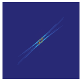
Definition (K, Labate; 2006):
The (cone-adapted) discrete shearlet system $\mathcal{SH}(c; \phi, \psi, \tilde{\psi})$, $c > 0$, generated by $\phi \in L^2(\mathbb{R}^2)$ and $\psi, \tilde{\psi} \in L^2(\mathbb{R}^2)$ is the union of

$$\{\phi(\cdot - cm) : m \in \mathbb{Z}^2\},$$

$$\{2^{3j/4}\psi(S_k A_{2^j} \cdot - cm) : j \geq 0, |k| \leq \lceil 2^{j/2} \rceil, m \in \mathbb{Z}^2\},$$

$$\{2^{3j/4}\tilde{\psi}(\tilde{S}_k \tilde{A}_{2^j} \cdot - cm) : j \geq 0, |k| \leq \lceil 2^{j/2} \rceil, m \in \mathbb{Z}^2\}.$$





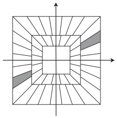
(Cone-adapted) Discrete Shearlet Systems

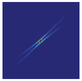
Definition (K, Labate; 2006):
The (cone-adapted) discrete shearlet system $\mathcal{SH}(c;\phi,\psi,\tilde{\psi})$, $c > 0$, generated by $\phi \in L^2(\mathbb{R}^2)$ and $\psi, \tilde{\psi} \in L^2(\mathbb{R}^2)$ is the union of

$$\{\phi(\cdot - cm) : m \in \mathbb{Z}^2\},$$

$$\{2^{3j/4}\psi(S_k A_{2^j} \cdot - cm) : j \geq 0, |k| \leq \lceil 2^{j/2} \rceil, m \in \mathbb{Z}^2\},$$

$$\{2^{3j/4}\tilde{\psi}(\tilde{S}_k \tilde{A}_{2^j} \cdot - cm) : j \geq 0, |k| \leq \lceil 2^{j/2} \rceil, m \in \mathbb{Z}^2\}.$$





Theorem (K, Labate, Lim, Weiss; 2006):
For $\psi, \tilde{\psi}$ classical shearlets, $\mathcal{SH}(1;\phi,\psi,\tilde{\psi})$ is a Parseval frame for $L^2(\mathbb{R}^2)$:

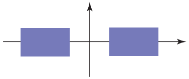
$$A\|f\|_2^2 \leq \sum_{\sigma \in \mathcal{SH}(1;\phi,\psi,\tilde{\psi})} |\langle f, \sigma \rangle|^2 \leq B\|f\|_2^2 \quad \text{for all } f \in L^2(\mathbb{R}^2)$$

holds for $A = B = 1$.

Compactly Supported Shearlets

Theorem (Kittipoom, K, Lim; 2012):

Let $\phi, \psi, \tilde{\psi} \in L^2(\mathbb{R}^2)$ be compactly supported, and let $\hat{\psi}$, $\hat{\tilde{\psi}}$ satisfy certain decay condition. Then there exists c_0 such that $\mathcal{SH}(c; \phi, \psi, \tilde{\psi})$ forms a shearlet frame with controllable frame bounds for all $c \leq c_0$.

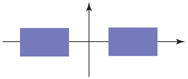


Remark: Exemplary class with $B/A \approx 4$.

Compactly Supported Shearlets

Theorem (Kittipoom, K, Lim; 2012):

Let $\phi, \psi, \tilde{\psi} \in L^2(\mathbb{R}^2)$ be compactly supported, and let $\hat{\psi}$, $\hat{\tilde{\psi}}$ satisfy certain decay condition. Then there exists c_0 such that $\mathcal{SH}(c; \phi, \psi, \tilde{\psi})$ forms a shearlet frame with controllable frame bounds for all $c \leq c_0$.

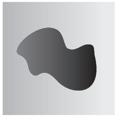


Remark: Exemplary class with $B/A \approx 4$.

Theorem (K, Lim; 2011):

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

$$\|f - f_N\|_2^2 \leq C \cdot N^{-2} \cdot (\log N)^3, \quad N \to \infty.$$



Recent Approaches to Fast Shearlet Transforms

`www.ShearLab.org`:

- Separable Shearlet Transform *(Lim; 2009)*
- Digital Shearlet Transform *(K, Shahram, Zhuang; 2011)*
- 2D&3D (parallelized) Shearlet Transform *(K, Lim, Reisenhofer; 2014)*

Additional Code:

- Filter-based implementation *(Easley, Labate, Lim; 2009)*
- Fast Finite Shearlet Transform *(Häuser, Steidl; 2014)*
- Shearlet Toolbox 2D&3D *(Easley, Labate, Lim, Negy; 2014)*

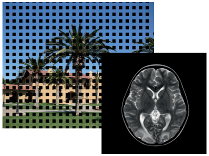
Theoretical Approaches:

- Adaptive Directional Subdivision Schemes *(K, Sauer; 2009)*
- Shearlet Unitary Extension Principle *(Han, K, Shen; 2011)*
- Gabor Shearlets *(Bodmann, K, Zhuang; 2013)*

Application to Inverse Problems

Examples:

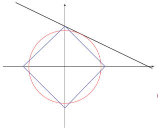
- Denoising.
- Feature Extraction.
- Inpainting.
- Magnetic Resonance Tomography.
- ...



Sparse Regularization:

Given an ill-posed inverse problem $Kx = y$, where $K : X \to Y$ and x is known to be sparsely representable by a shearlet frame $(\sigma_\eta)_\eta$, an approximate solution $x^\alpha \in X$, $\alpha > 0$, can be determined by

$$\min_{\tilde{x}} \|K\tilde{x} - y\|^2 + \alpha \|(\langle \tilde{x}, \sigma_\eta \rangle)_\eta\|_1.$$

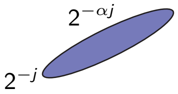


Extension to α-Shearlets

Main Idea:

- Introduction of a parameter $\alpha \in [0,1]$ to measure the amount of anisotropy.
- For $j \in \mathbb{Z}$, define

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



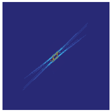
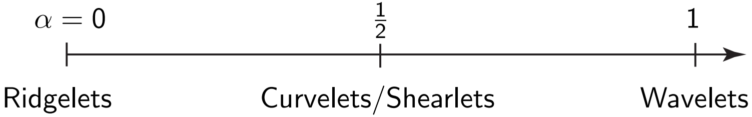


Illustration:



α-Shearlets

Definition (Grohs, Keiper, K, and Schäfer; 2016)(Voigtlaender; 2017):
For $c \in \mathbb{R}^+$ and $\alpha \in [0,1]$, the *cone-adapted α-shearlet system* $\mathcal{SH}_\alpha(\phi, \psi, \tilde{\psi}, c)$ generated by $\phi, \psi, \tilde{\psi} \in L^2(\mathbb{R}^2)$ is defined by

$$\mathcal{SH}_\alpha(\phi, \psi, \tilde{\psi}, c) := \Phi(\phi, c, \alpha) \cup \Psi(\psi, c, \alpha) \cup \tilde{\Psi}(\tilde{\psi}, c, \alpha),$$

where

$$\begin{aligned} \Phi(\phi, c, \alpha) &:= \{\phi(\cdot - m) : m \in c\mathbb{Z}^2\}, \\ \Psi(\psi, c, \alpha) &:= \{2^{j(1+\alpha)/2}\psi(S_k A_{\alpha,j} \cdot - m) : j \geq 0, |k| \leq \lceil 2^{j(1-\alpha)} \rceil, \\ &\qquad m \in c\mathbb{Z}^2, k \in \mathbb{Z}^2\}, \\ \tilde{\Psi}(\tilde{\psi}, c, \alpha) &:= \{2^{j(1+\alpha)/2}\widetilde{\psi}(S_k^T \tilde{A}_{\alpha,j} \cdot - m) : j \geq 0, |k| \leq \lceil 2^{j(1-\alpha)} \rceil, \\ &\qquad m \in c\mathbb{Z}^2, k \in \mathbb{Z}^2\}. \end{aligned}$$

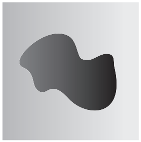
Cartoon-Like Functions

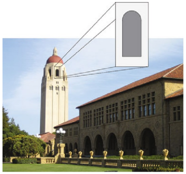
Definition (Donoho; 2001)(Grohs, Keiper, K, and Schäfer; 2016):
Let $\alpha \in [\frac{1}{2}, 1]$ and $\nu > 0$. We then define the class of *α-cartoon-like functions* by

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

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:





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

Next Task: Realize sufficiently smooth functions with sufficiently many vanishing moments with a neural network.

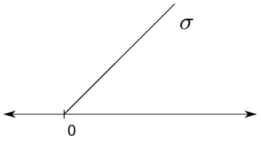
Construction of Generators

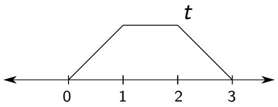
Next Task: Realize sufficiently smooth functions with sufficiently many vanishing moments with a neural network.

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

- Assume rectifiers $\sigma(x) = \max\{x, 0\}$ (ReLUs).
- Define

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



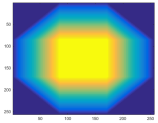


$\rightsquigarrow$ t can be constructed with a two layer network.

Construction of Wavelet Generators

Construction by (Shaham, Cloninger, and Coifman; 2017) continued:

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

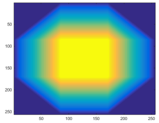


- Summing up shifted versions of ϕ yields a function ψ with vanishing moments.
- Then ψ can be realized by a 3 layer neural network.

Construction of Wavelet Generators

Construction by (Shaham, Cloninger, and Coifman; 2017) continued:

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



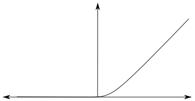
- Summing up shifted versions of ϕ yields a function ψ with vanishing moments.
- Then ψ can be realized by a 3 layer neural network.

This cannot yield differentiable functions ψ!

New Class of Rectifiers

Definition (Bölcskei, Grohs, K, Petersen; 2017):
Let $\sigma : \mathbb{R} \to \mathbb{R}^{+}$, $\sigma \in C^{\infty}(\mathbb{R})$ satisfy

$$\sigma(x) = \begin{cases} 0 & \text{for } x \leq 0, \\ x & \text{for } x \geq K, \end{cases}$$

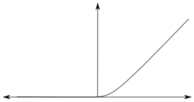


for some constant $K > 0$. Then we call σ an *admissible smooth rectifier*.

New Class of Rectifiers

Definition (Bölcskei, Grohs, K, Petersen; 2017):
Let $\sigma : \mathbb{R} \to \mathbb{R}^+$, $\sigma \in C^\infty(\mathbb{R})$ satisfy

$$\sigma(x) = \begin{cases} 0 & \text{for } x \leq 0, \\ x & \text{for } x \geq K, \end{cases}$$



for some constant $K > 0$. Then we call σ an *admissible smooth rectifier*.

Construction of 'good' generators:
Let σ be an admissible smooth rectifier, and define

$$\begin{aligned} t(x) &:= \sigma(x) - \sigma(x-1) - \sigma(x-2) + \sigma(x-3), \\ \phi(x) &:= \sigma(t(x_1) + t(x_2) - 1). \end{aligned}$$

This yields smooth bump functions ϕ, and thus smooth functions ψ with many vanishing moments.
$\rightsquigarrow$ *Leads to appropriate shearlet generators!*

Optimal Approximation

Theorem (Bölcskei, Grohs, K, and Petersen; 2017): Let σ be an admissible smooth rectifier, and let $\varepsilon > 0$. Then there exist $C_\varepsilon > 0$ such that the following holds:
For all $f \in \mathcal{E}^{\frac{1}{\alpha}}(\mathbb{R}^2)$ and $N \in \mathbb{N}$, we can construct a neural network $\Phi \in \mathcal{NN}_{3,O(N),\sigma,O(\text{polylog}(N))}$ satisfying

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

Optimal Approximation

Theorem (Bölcskei, Grohs, K, and Petersen; 2017): Let σ be an admissible smooth rectifier, and let $\varepsilon > 0$. Then there exist $C_\varepsilon > 0$ such that the following holds:
For all $f \in \mathcal{E}^{\frac{1}{\alpha}}(\mathbb{R}^2)$ and $N \in \mathbb{N}$, we can construct a neural network $\Phi \in \mathcal{NN}_{3,O(N),\sigma,O(\text{polylog}(N))}$ satisfying

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

This is the optimal approximation rate:

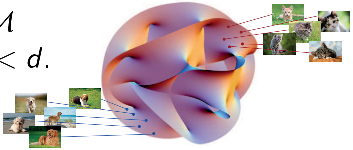
Theorem (Grohs, Keiper, K, Schäfer; 2016): We have

$$\gamma^*(\mathcal{E}^{\frac{1}{\alpha}}(\mathbb{R}^2)) = 2\alpha.$$

Functions on Manifolds

Situation:

We now consider $f : \mathcal{M} \subseteq \mathbb{R}^d \to \mathbb{R}$, where $\mathcal{M}$ is an immersed submanifold of dimension $m < d$.



Road Map for Extension of Results:

- Construct atlas for $\mathcal{M}$ by covering it with open balls.
- Obtain a smooth partition of unity of $\mathcal{M}$.
- Represent any function on $\mathcal{M}$ as a sum of functions on $\mathbb{R}^m$.

$\rightsquigarrow$ *We require function classes $\mathcal{C}$ which are invariant with respect to diffeomorphisms and multiplications by smooth functions such as $\mathcal{E}^{\frac{1}{\alpha}}(\mathbb{R}^2)$.*

Theorem (Bah, Keiper, and K; 2017):

"Deep neural networks are optimal for the approximation of piecewise smooth functions on manifolds."

Finally some Numerics...

Approximation by Learned Networks

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

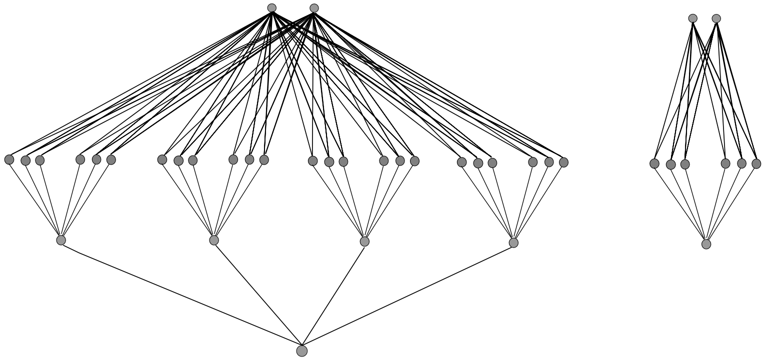
Does this lead to the optimal sparse connectivity?

Our setup:

- Fixed network topology with ReLUs.
- Specific functions to learn.
- Learning through backpropagation.
- Analysis of the connection between approximation error and number of edges.

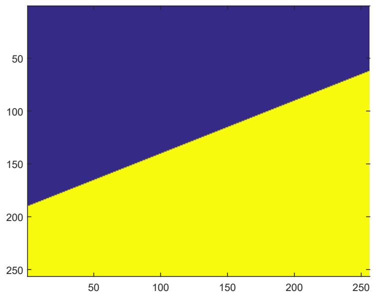
Chosen Network Topology

Topology inspired by previous construction of functions:



Example: Function 1

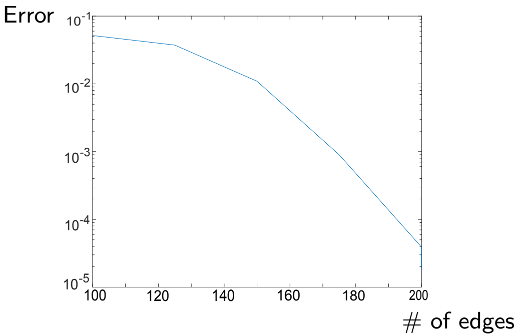
We train the network using the following function:



Function 1: Linear Singularity

Training with Function 1

Approximation Error:

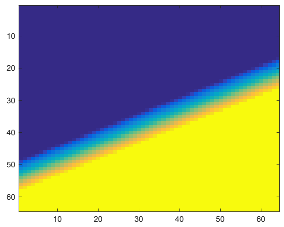


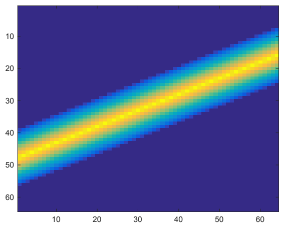
Observation: The decay is exponential. This is expected if the network is a sum of 0-shearlets, which are ridgelets.

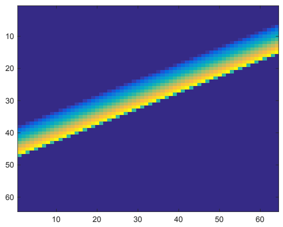
Training with Function 1

The network with fixed topology naturally admits subnetworks.

Examples of Subnetworks:



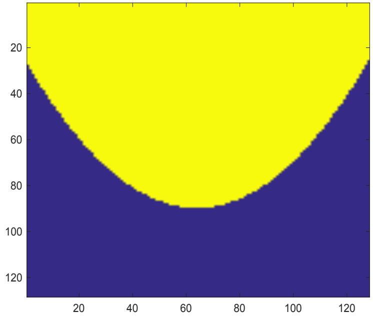




These have indeed the shape of ridgelets!

Example: Function 2

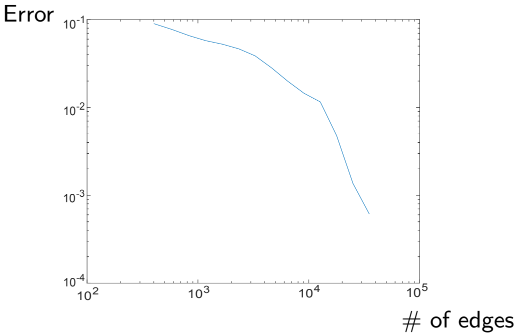
We train the network using the following function:



Function 2: Curvilinear Singularity

Training with Function 2

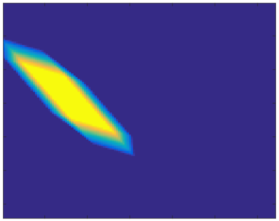
Approximation Error:

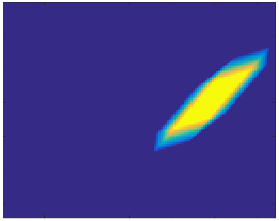


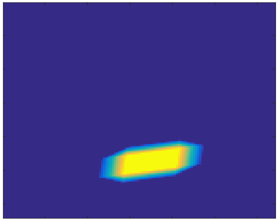
Observation: The decay is of the order M^{-1}. This is expected if the network is a sum of $\frac{1}{2}$-shearlets.

Training with Function 2

Examples of Subnetworks:



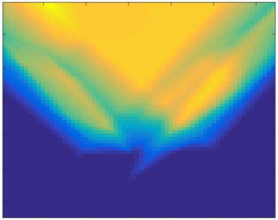


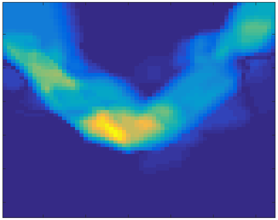


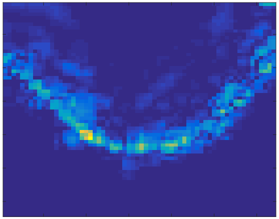
These seem to be indeed anisotropic!

Training with Function 2

Form of Approximation:


Largest Elements

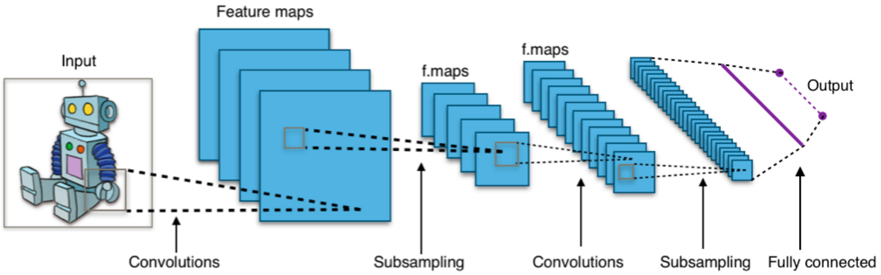

Medium Sized Elements


Smallest Elements

The learnt neural network has a multiscale behavior!

Convolutional Neural Networks

Typical Convolutional Neural Network



Convolutional Layer

Definition:

- A *convolutional node* has a stack of images $X \in \mathbb{R}^{n_1, n_2, S}$ as input. Given a filter $W \in \mathbb{R}^{F,F,S}$, where F is the spatial extend of the filter, and a bias $b \in \mathbb{R}$, it computes

$$\begin{aligned} Z[i,j] &= (W *_{12} X)[i,j] + b \\ &= \sum_{k=1}^{S} \underbrace{W[\cdot,\cdot,k] * [\cdot,\cdot,k]}_{\text{convolution for each image}} + b \end{aligned}$$

- A *convolutional layer* consists of K convolutional nodes $((W_k, b_k))_{k=1}^{K} \subseteq \mathbb{R}^{F,F,S} \times \mathbb{R}$ and produces as output a stack

$$Z[i,j,k] := W_k *_{12} X + b_k$$

Pooling Layer

Definition:

A *pooling operator* R acts layerwise on each $Z \in \mathbb{R}^{n_1,n_2,S}$ and results in

$$R(Z) \in \mathbb{R}^{m_1,m_2,S}, m_1 < n_1, m_2 < n_2.$$

Examples:

1. *Sub-Sampling*:
$$R(Z)[i,j,k] = Z[s_1 i, s_2 j, k]$$
with $\frac{s_1}{n_1}, \frac{s_2}{n_2}$. Then $m_1 = \frac{n_1}{s_1}, m = \frac{n_2}{s_2}$.
2. *Averaging*:
$$R(Z)[i,j,k] = \left(\sum_{l=s_1 \cdot i}^{s_1 i + s_1 - 1} \sum_{t=s_2 \cdot j}^{s_2 j + s_2 - 1} Z[l,t,k] \right) \frac{1}{s_1 + s_2}$$
3. *Max-Pooling*:
$$R(Z)[i,j,k] = \max_{l=s_1 \cdot i, \ldots s_1 \cdot i + s_1 - 1, t = s_2 \cdot i, \ldots s_2 \cdot j + s_2 - 1} |Z[l,t,k]|$$

Overall Architecture

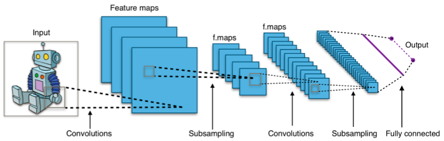
Definition:

A *convolutional neural network (CNN)* with L layers consists of L iterated applications of a convolutional layer followed by an activation layer and possibly a pooling layer.

Remark:

A typical architecture consists of the following components (e.g., LeNet (LeCun et. al 1998)):

- First a CNN as feature-extractor.
- Second a fully connected NN as classifier.



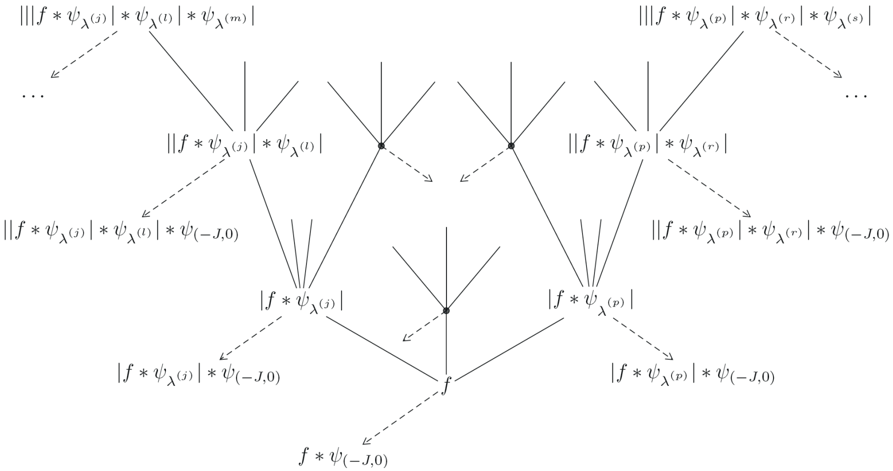
A Mathematical Approach

A Very Nice Idea...

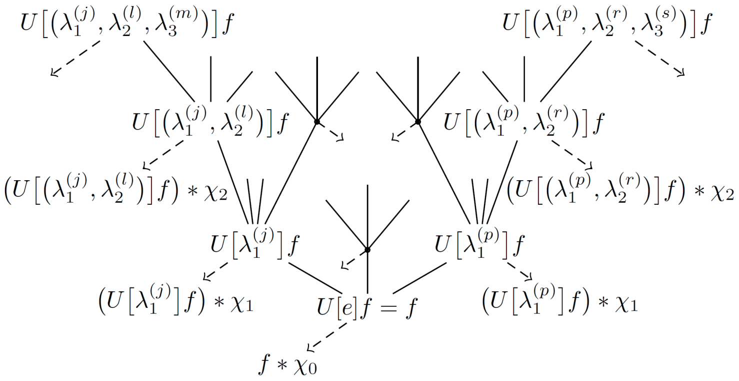
The *scattering transform* (Mallat, 2014) is a special convolutional neural network:

- It uses fixed predefined (wavelet) filters.
- It performs almost as good as a trained neural network in some applications.
- It is more accessible to theoretical analysis.
- There exists a continuous as well as discrete theory.

Scattering Transform


$|||f * \psi_{\lambda^{(j)}}| * \psi_{\lambda^{(l)}}| * \psi_{\lambda^{(m)}}|$
$|||f * \psi_{\lambda^{(p)}}| * \psi_{\lambda^{(r)}}| * \psi_{\lambda^{(s)}}|$
$\dots$
$\dots$
$||f * \psi_{\lambda^{(j)}}| * \psi_{\lambda^{(l)}}|$
$||f * \psi_{\lambda^{(p)}}| * \psi_{\lambda^{(r)}}|$
$||f * \psi_{\lambda^{(j)}}| * \psi_{\lambda^{(l)}}| * \psi_{(-J,0)}$
$||f * \psi_{\lambda^{(p)}}| * \psi_{\lambda^{(r)}}| * \psi_{(-J,0)}$
$|f * \psi_{\lambda^{(j)}}|$
$|f * \psi_{\lambda^{(p)}}|$
$|f * \psi_{\lambda^{(j)}}| * \psi_{(-J,0)}$
$|f * \psi_{\lambda^{(p)}}| * \psi_{(-J,0)}$
f
$f * \psi_{(-J,0)}$

Scattering Transform


$U[(\lambda_1^{(j)}, \lambda_2^{(l)}, \lambda_3^{(m)})]f$
$U[(\lambda_1^{(p)}, \lambda_2^{(r)}, \lambda_3^{(s)})]f$
$U[(\lambda_1^{(j)}, \lambda_2^{(l)})]f$
$U[(\lambda_1^{(p)}, \lambda_2^{(r)})]f$
$(U[(\lambda_1^{(j)}, \lambda_2^{(l)})]f) * \chi_2$
$(U[(\lambda_1^{(p)}, \lambda_2^{(r)})]f) * \chi_2$
$U[\lambda_1^{(j)}]f$
$U[\lambda_1^{(p)}]f$
$(U[\lambda_1^{(j)}]f) * \chi_1$
$U[e]f = f$
$(U[\lambda_1^{(p)}]f) * \chi_1$
$f * \chi_0$

Scattering Transform

Definition: Let

- $\Psi_n = \{\psi_{\lambda_n}\}_{\lambda_n \in \Lambda_n}, \psi_{\lambda_n} \in \mathcal{L}^1(\mathbb{R}^d) \cap \mathcal{L}^2(\mathbb{R}^d)$ with

$$\sum_{\lambda_n \in \Lambda_n} \|f * \psi_{\lambda_n}\|_2^2 \leq B_n \|f\|_2^2 \quad \text{for all } f \in \mathcal{L}^2(\mathbb{R}),$$

- For $R_n \geq 1$ the subsampling factor, let

$$(U_n[\lambda_n]f)(x) = R_n^2 |f * \psi_{\lambda_n}|(R_n x), \quad \lambda \in \mathbb{R}^d,$$

- For a path of index sets $q = (\lambda_1, \dots \lambda_n)$, $\lambda_i \in \Lambda_i$ let

$$U[q]f = U_n[\lambda_n](U_{n-1}[\lambda_{n-1}] - (U_1[\lambda_1]f)),$$

- $\chi_{n-1} := \psi_{\lambda_n}$ for every $n \in \mathbb{N}$.

The associated *scattering transformation* Φ_Ω is defined by

$$f \mapsto \Phi_\Omega(f) := \underbrace{\bigcup_{n=0}^{\infty} \{U[q]f * \chi_{n-1}\}_{q=(\lambda_1, \dots \lambda_n)}}_{\textit{Interpretation}: \text{"feature vector"}}.$$

Translation Invariance of the Scattering Transform

Theorem (Mallat; 2014)(Wiatowski at al.; 2016):
Let Φ_Ω be a scattering transformation with $R_n := 1$ for all n. Then Φ_n is *translation invariant*, i.e.

$$\Phi_\Omega(T_t f) = T_t \Phi_\Omega(f)$$

for all $t \in \mathbb{R}^d$ with $(T_t f)(x) = f(x - t), x \in \mathbb{R}^d$, in particular

$$U[q](T_t f) * \chi_{n-1} = T_t(U[q]f * \chi_{n-1})$$

for all $t \in \mathbb{R}^d$.

Deformation Stability of the Scattering Transform

Theorem (Mallat; 2014)(Wiatowski at al.; 2016):
Let Φ_Ω be a scattering transformation with $\max_{n \in \mathbb{N}} \max\{B_n, B_n L_n^2\} \leq 1$. Then for any $K > 0$, the scattering transformation Φ_Ω is *stable on $\mathcal{E}_s^2(\mathbb{R}^d)$ with respect to deformations*.
This means that for every $K > 0$, there exists $C_K > 0$ such that for all $f \in \mathcal{E}_s^2(\mathbb{R}^d)$ and $\tau \in \mathcal{C}^1(\mathbb{R}^d, \mathbb{R}^d)$ with

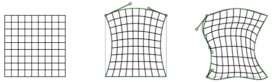
$$\|\tau\|_\infty \leq \frac{1}{2} \quad \text{and} \quad \|D\tau\|_\infty \leq \frac{1}{2d},$$

we have

$$|||\Phi_\Omega(F_\tau f) - \Phi_\Omega(f)||| \leq C_K \|\tau\|_\infty^{\frac{1}{2}},$$

where

$$(F_\tau f)(x) = f(x - \tau(x)).$$



Explainability

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 $x = (x_1, ...x_d)$ 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*.

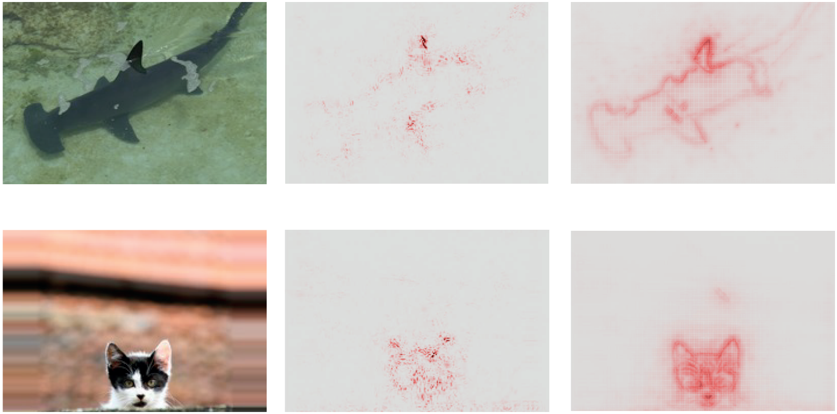
Remarks:

- 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.

Deep Taylor Decomposition



Let's conclude...

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?

$\rightsquigarrow$ *We discussed architectures and several approximation results!*

- *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?

$\rightsquigarrow$ *We discussed architectures and several approximation results!*

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

$\rightsquigarrow$ *We discussed stochastic gradient descent, but the theory is just starting!*

- *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?

$\rightsquigarrow$ *We discussed architectures and several approximation results!*

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

$\rightsquigarrow$ *We discussed stochastic gradient descent, but the theory is just starting!*

- *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?

$\rightsquigarrow$ *There are so far no theoretical results known!*

- *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?

$\rightsquigarrow$ *We discussed architectures and several approximation results!*

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

$\rightsquigarrow$ *We discussed stochastic gradient descent, but the theory is just starting!*

- *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?

$\rightsquigarrow$ *There are so far no theoretical results known!*

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

$\rightsquigarrow$ *We briefly discussed relevance maps!*

THANK YOU!

References available at:

`www.math.tu-berlin.de/~kutyniok`

Our Blog/Database for the Mathematical Theory for Deep Learning:

`www.DeepMath.org`