Course: Mathematics of Deep Learning

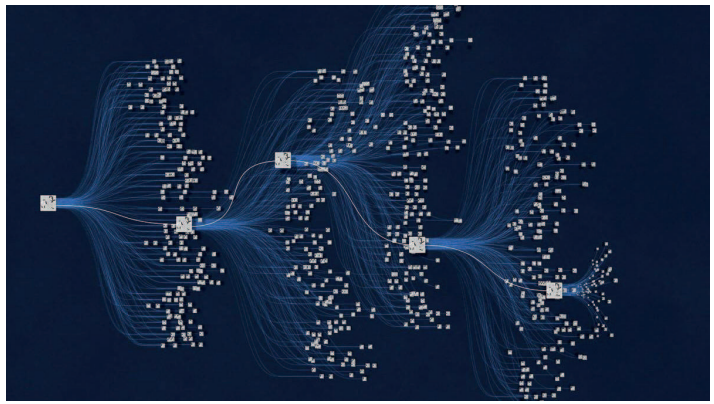
Gitta Kutyniok

(Technische Universität Berlin and University of Tromsø)

October 15, 2019

Goals of the Course

- Introduction into the mathematics of deep learning.
- Understanding how deep neural networks are trained.
- Knowing the possibilities and limitations of deep learning.
- Laying the foundation for delving further into this topic, both theoretically and practically.



Topics of the Course

- Basics of Statistical Learning Theory
- Training of Deep Neural Networks
- Mathematical Theory of Deep Learning:
  - Expressivity
  - Optimization
  - Generalization
  - Interpretability
- Deep Learning in Mathematical Approaches:
  - Inverse Problems/Imaging Science
  - Numerical Analysis of Partial Differential Equations

Organization of the Course

- Lecture:
  - Tu 8-10, MA 042 (Start at 8:30AM?)
- Practical session:
  - Hector Andrade-Loarca
  - October 29, 2019
- Exam:
  - Credit Points: 5 LP.
  - Oral or Written: How many?
- Announcements:
  - Please enroll in ISIS2.
- Literature:
  - Please check the hand out.

Requirements of the Course

Necessary:

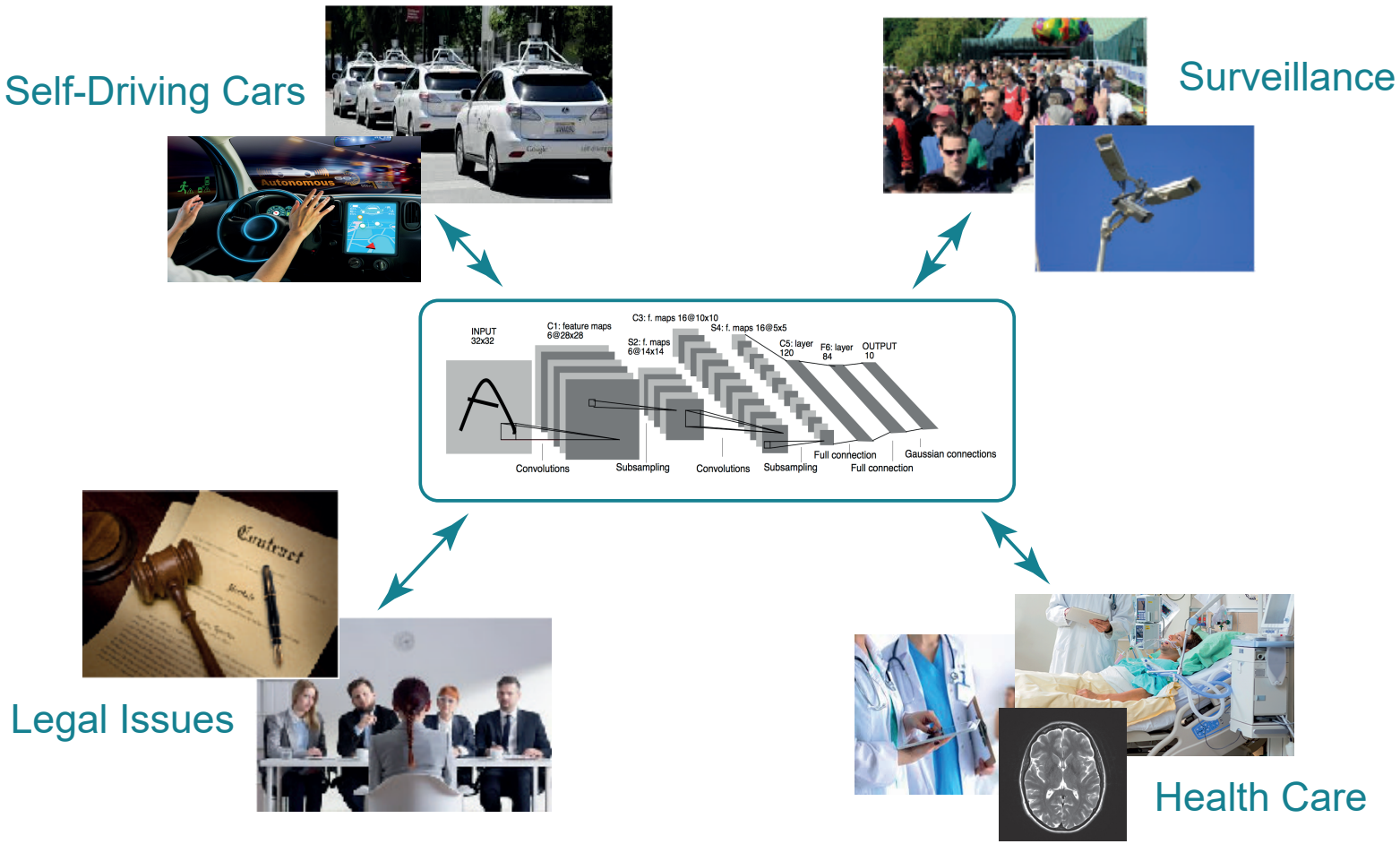
- Analysis I-III
- Linear Algebra
- Probability Theory I

Desirable:

- Functional Analysis I (Who ?)
- Probability Theory II (Who ?)

Let's start with some highlights...

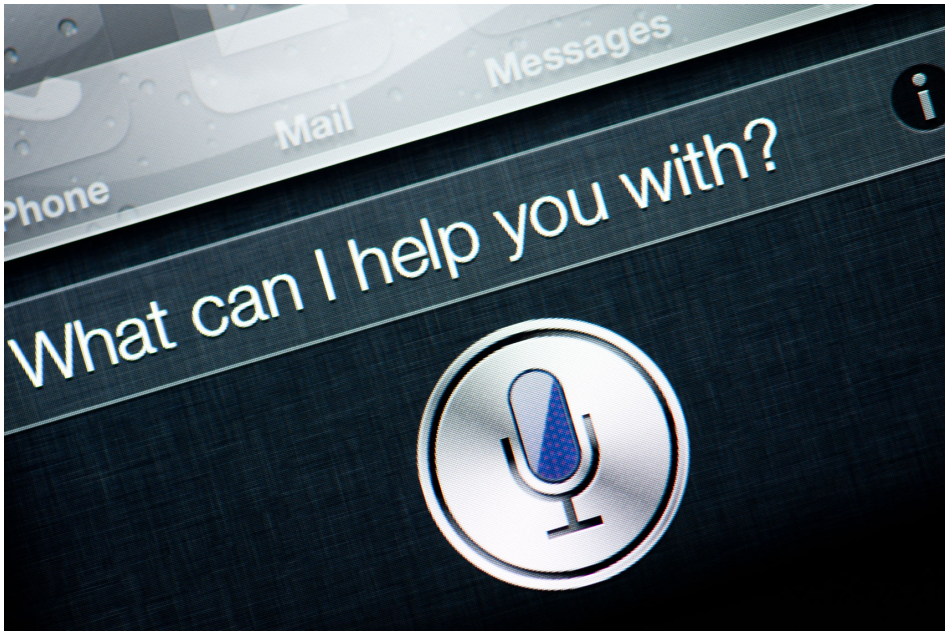
The Dawn of Deep Learning



Speech Recognition

Application of deep learning to cell phones:

Speech recognition systems such as Siri belong to the current standard in numerous cell phones!



Deep Learning = Artificial Intelligence?

AlphaGo Zero Shows Machines Can Become Superhuman Without Any Help

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

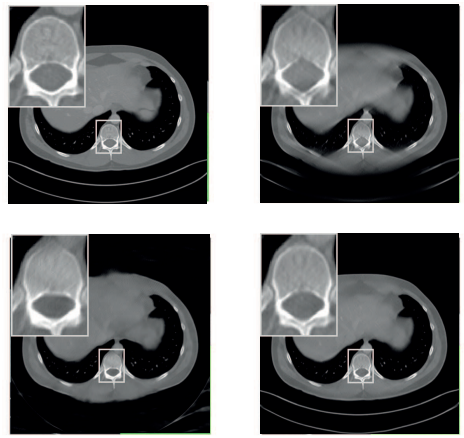
'...AlphaGo Zero...started with nothing but a blank board and the rules of the game. It learned simply by playing millions of games against itself, using what it learned in each game to improve."

MIT Technology Review (Oct. 2017)

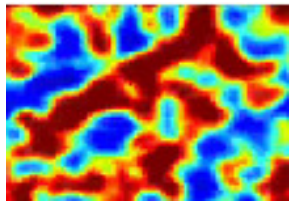
Impact of Deep Learning on Mathematics

Some Examples:

- Inverse Probleme/Imaging Science (2012–)
  - ⇝ *Denoising*
  - ⇝ *Edge Detection*
  - ⇝ *Inpainting*
  - ⇝ *Classification*
  - ⇝ *Superresolution*
  - ⇝ *Limited-Angle Computed Tomography*
  - ⇝ ...



- Numerical Analysis of Partial Differential Equations (2017–)
  - ⇝ *Black-Scholes PDE*
  - ⇝ *Allen-Cahn PDE*
  - ⇝ *Parametric PDEs*
  - ⇝ ...



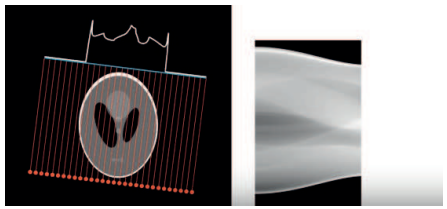
- Modelling (2018–)
  - ⇝ *Learning of equations from data*
  - ⇝ *Learning of PDEs*

Examples: Deep Learning changes Mathematical Methods

Limited-Angle Computed Tomography

A CT scanner samples the *Radon transform*

$$\mathcal{R}f(\phi, s) = \int_{L(\phi,s)} f(x) dS(x),$$



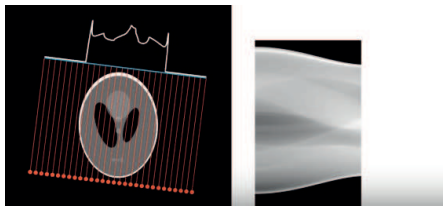
for $L(\phi, s) = \left\{x \in \mathbb{R}^2 : x_1 \cos(\phi) + x_2 \sin(\phi) = s\right\}$, $\phi \in [-\pi/2, \pi/2)$, and $s \in \mathbb{R}$.

Examples: Deep Learning changes Mathematical Methods

Limited-Angle Computed Tomography

A CT scanner samples the *Radon transform*

$$\mathcal{R}f(\phi, s) = \int_{L(\phi,s)} f(x) dS(x),$$



for $L(\phi, s) = \left\{x \in \mathbb{R}^2 : x_1 \cos(\phi) + x_2 \sin(\phi) = s\right\}$, $\phi \in [-\pi/2, \pi/2)$, and $s \in \mathbb{R}$.

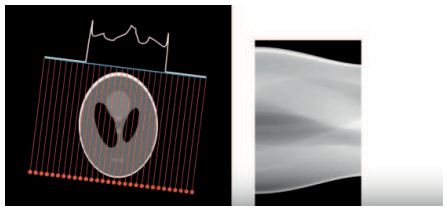
Challenging inverse problem if $\mathcal{R}f(\cdot, s)$ is only sampled on $[-\phi, \phi]$, $\phi < \pi/2$

Examples: Deep Learning changes Mathematical Methods

Limited-Angle Computed Tomography

A CT scanner samples the *Radon transform*

$$\mathcal{R}f(\phi, s) = \int_{L(\phi,s)} f(x) dS(x),$$



for $L(\phi, s) = \left\{x \in \mathbb{R}^2 : x_1 \cos(\phi) + x_2 \sin(\phi) = s\right\}$, $\phi \in [-\pi/2, \pi/2)$, and $s \in \mathbb{R}$.

Challenging inverse problem if $\mathcal{R}f(\cdot, s)$ is only sampled on $[-\phi, \phi]$, $\phi < \pi/2$

Learn the Invisible (Bubba, K, Lassas, März, Samek, Siltanen, Srinivan; 2018):

Step 1: Use model-based methods as far as possible

- Solve with sparse regularization using shearlets.

Step 2: Use data-driven methods where it is necessary

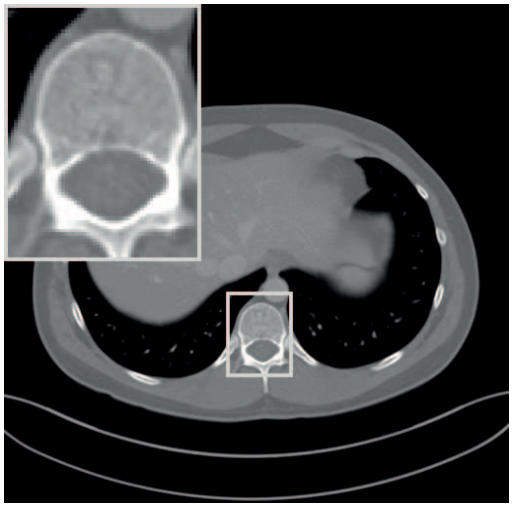
- Use a deep neural network to recover the missing components.

Step 3: Carefully combine both worlds

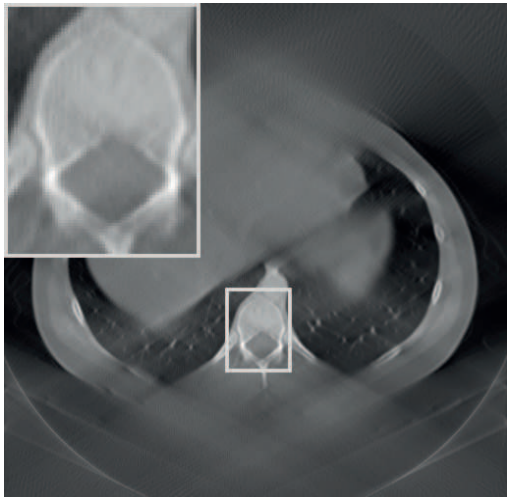
- Combine outcome of Step 1 and 2.

Learn the Invisible (LtI)

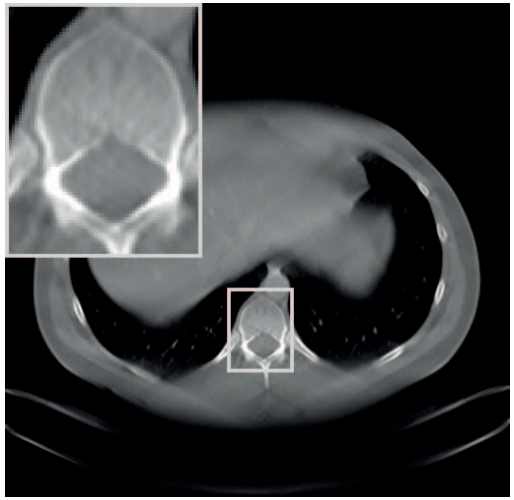
(Bubba, K, Lassas, März, Samek, Siltanen, Srinivan; 2018)



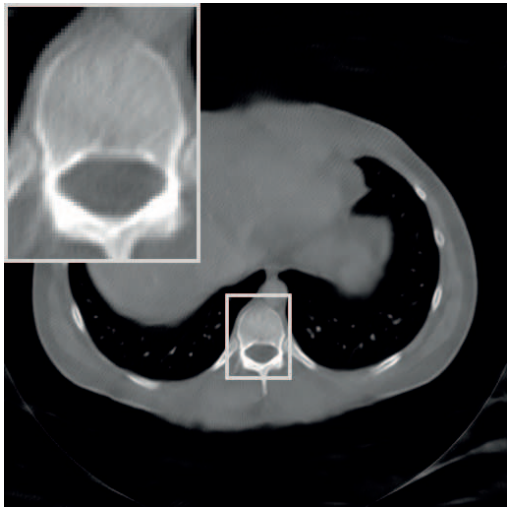
Original



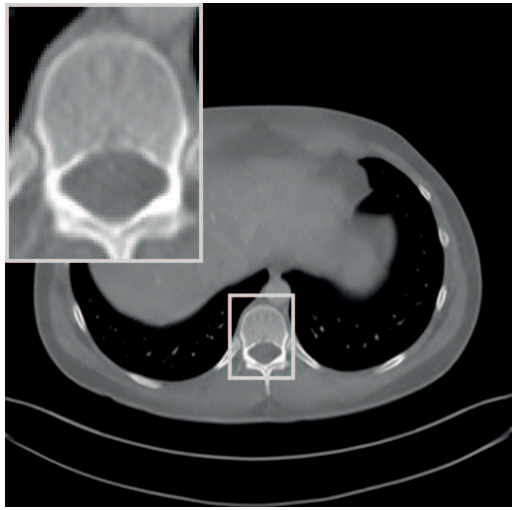
Filtered Backprojection



Sparse Regularization with Shearlets



[Gu & Ye, 2017]

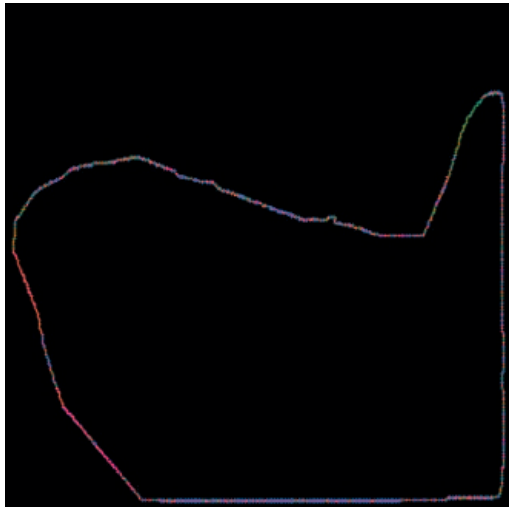


Learn the Invisible (LtI)

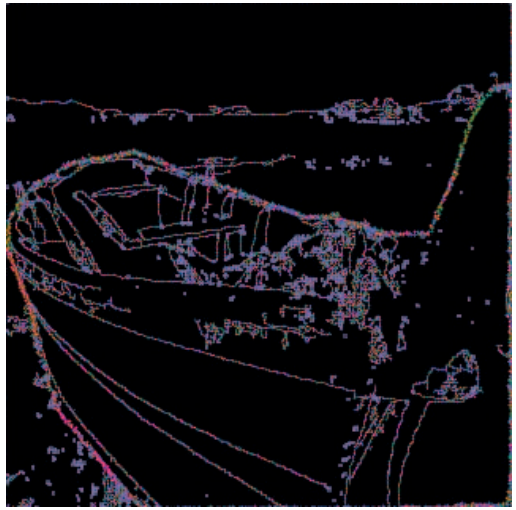
Deep Network Shearlet Edge Extractor (DeNSE)

(Andrade-Loarca, K, Öktem, Petersen; 2019)

Original

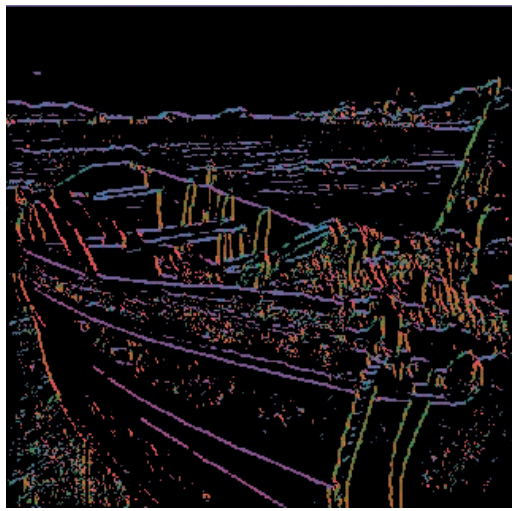


Human Annotation



SEAL [Yu et al; 2018]

CoShREM [Reisenhofer et al.; 2015]



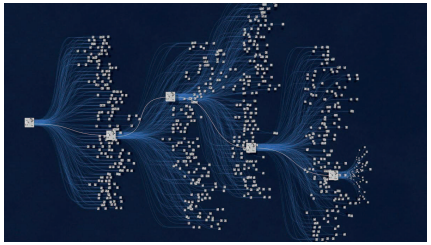
DeNSE

Deep Learning changes Mathematical Methods

Optimal Balancing of
Classical Methods and Deep Learning!

Deep Neural Networks

Deep neural networks have recently shown impressive results in a variety of real-world applications and outformed already various mathematical based approaches.



Deep Neural Networks

Deep neural networks have recently shown impressive results in a variety of real-world applications and outformed already various mathematical based approaches.

Problem:

Very few theoretical results explaining their success!

Deep Neural Networks

Deep neural networks have recently shown impressive results in a variety of real-world applications and outformed already various mathematical based approaches.

Problem:

Very few theoretical results explaining their success!

Nice article:

Deep, Deep Trouble:

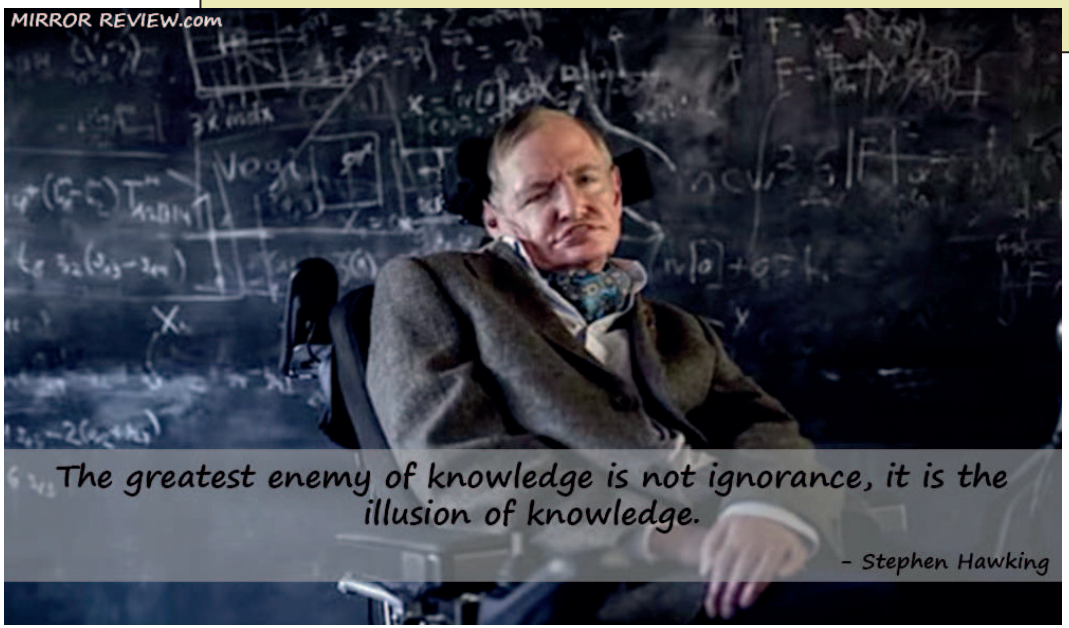
Deep Learnings Impact on Image Processing, Mathematics, and Humanity

Michael Elad (CS, Technion), SIAM News

Deep Learning = Alchemy?

„Ali Rahimi, a researcher in artificial intelligence (AI) at Google in San Francisco, California, took a swipe at his field last December—and received a 40-second ovation for it. Speaking at an AI conference, Rahimi charged that **machine learning algorithms, in which computers learn through trial and error, have become a form of „alchemy."** Researchers, he said, **do not know why some algorithms work and others don't, nor do they have rigorous criteria for choosing one AI architecture over another**...."

Science, May 2018



A Bit of History

First Appearance of Neural Networks

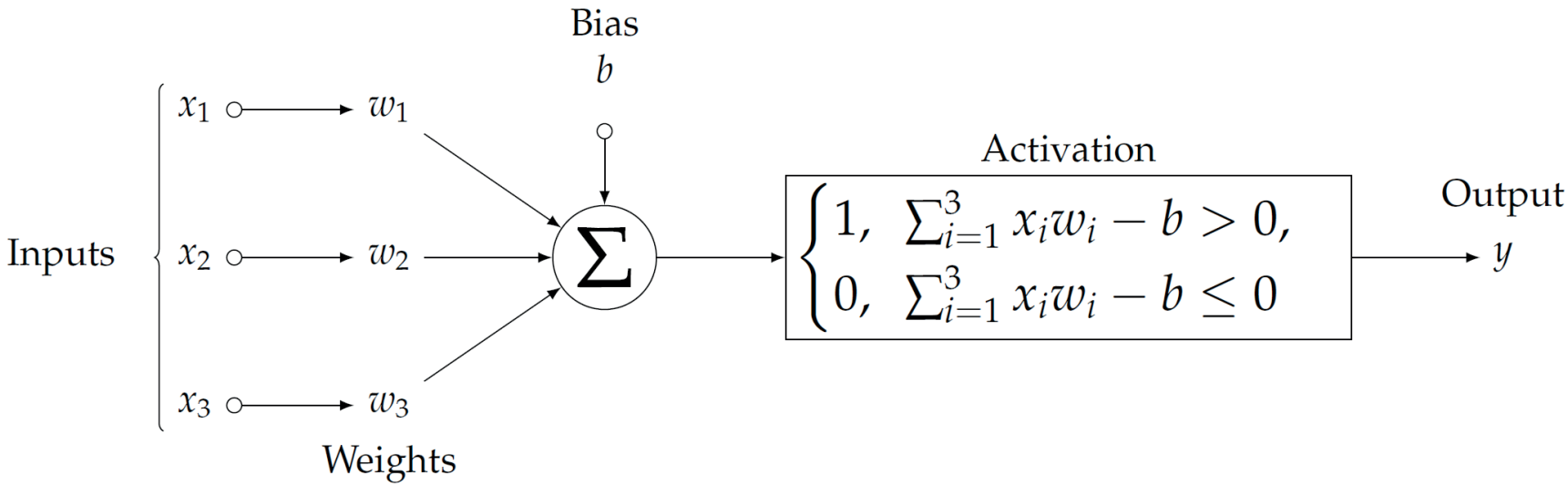
Key Task of McCulloch and Pitts (1943):

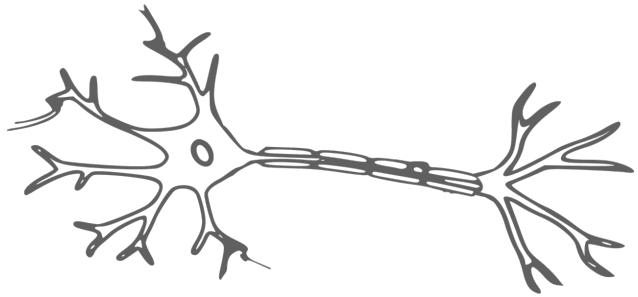
- Develop an algorithmic approach to learning.
- Mimicking the functionality of the human brain.

Goal: Artifical Intelligence!

Artificial Neurons

Mimic the human brain!





Artificial Neurons

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

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

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

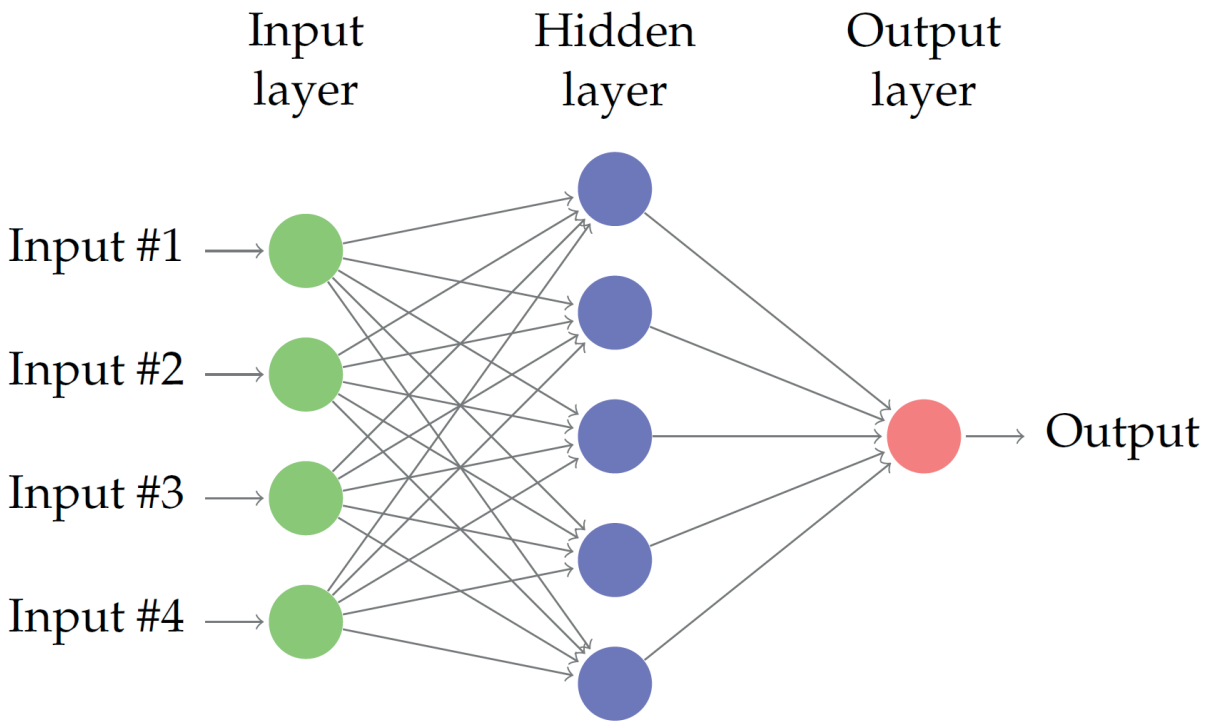
Examples of Activation Functions:

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

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*.



Second Appearance of Neural Networks

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

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

Second Appearance of Neural Networks

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

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

Current Situation:

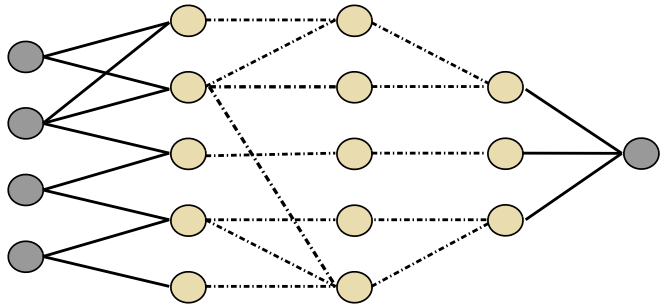
- Tremendous impact on science and public life.
- Development of theoretical foundation accelerating.
- Joint effort of mathematics and theoretical computer science.
- Paradigm change in mathematics.

Mathematics of Deep Neural Networks

The Mathematics of Deep Neural Networks

"Rough" Definition:

Assume the following notions:



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

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

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

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

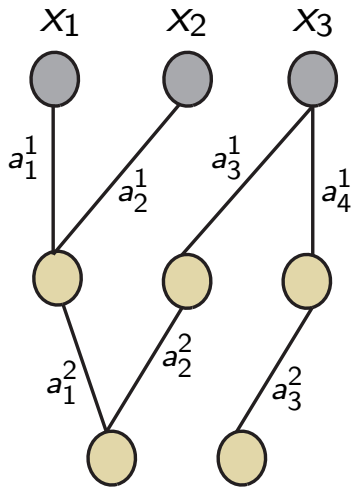
Affine Linear Maps and Weights

Remark: The affine linear map T_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

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

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

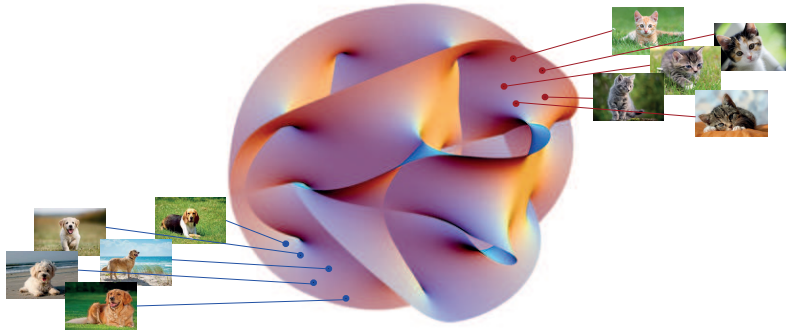
$$A_2 = \begin{pmatrix} a_1^2 & a_2^2 & 0 \\ 0 & 0 & a_3^2 \end{pmatrix}$$



Training of Deep Neural Networks

High-Level Set Up:

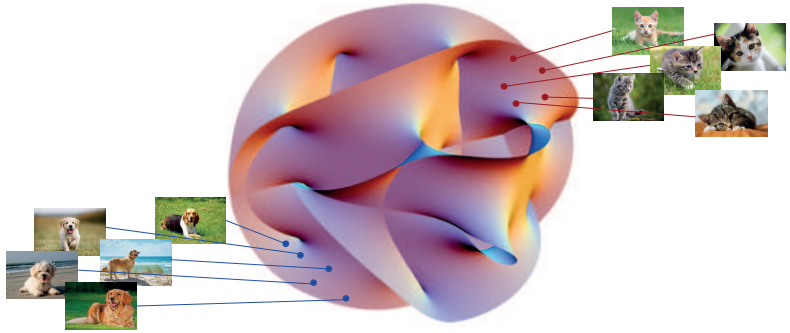
- Samples $(x_i, f(x_i))_{i=1}^m$ of a function such as $f : \mathcal{M} \to \{1, 2, \dots, K\}$.



Training of Deep Neural Networks

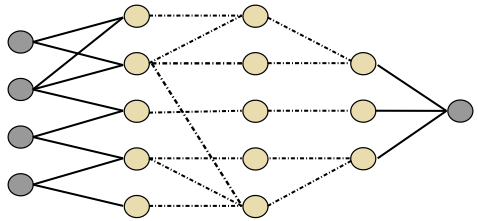
High-Level Set Up:

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



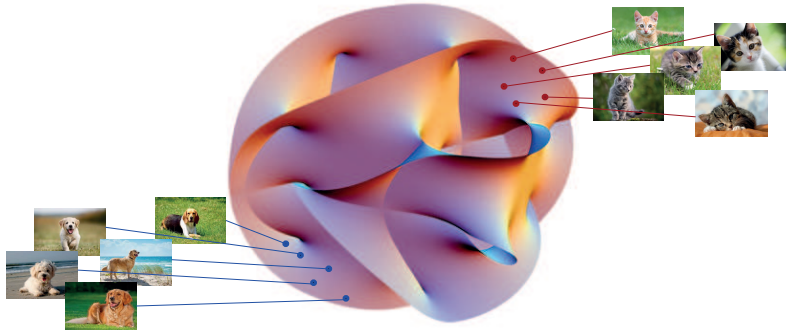
- Select an architecture of a deep neural network, i.e., a choice of d, L, $(N_\ell)_{\ell=1}^L$, and ϱ.

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

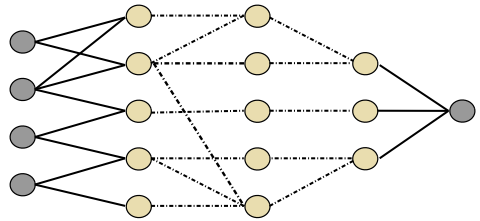


Training of Deep Neural Networks

High-Level Set Up:



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



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

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

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

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

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

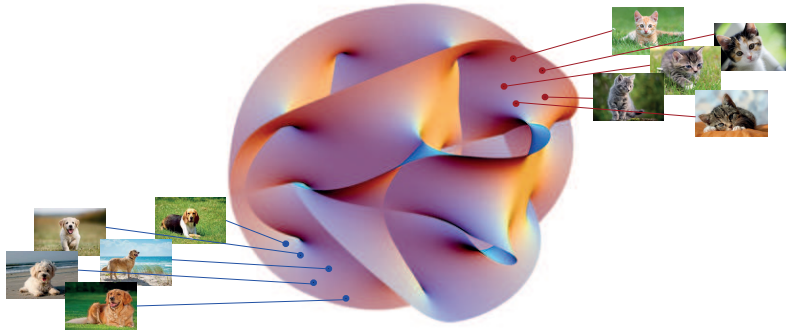
$$\Phi_{(A_\ell, b_\ell)_\ell}(x) = T_L \varrho(T_{L-1} \varrho(\dots \varrho(T_1(x))).$$

  This is often done by stochastic gradient descent.

Training of Deep Neural Networks

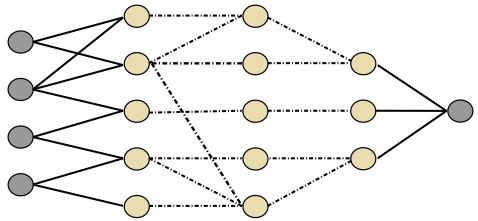
High-Level Set Up:

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



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

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



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

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

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

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

This is often done by stochastic gradient descent.

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

Fundamental Questions concerning Deep Neural Networks

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

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

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, ...*

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, ...*

Fundamental Questions concerning Deep Neural Networks

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

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

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

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

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

Topics of the Course

- Basics of Statistical Learning Theory
- Training of Deep Neural Networks
- Mathematical Theory of Deep Learning:
  - Expressivity
  - Optimization
  - Generalization
  - Interpretability
- Deep Learning in Mathematical Approaches:
  - Inverse Problems/Imaging Science
  - Numerical Analysis of Partial Differential Equations

Mathematical Learning Theory

What is Learning?

Definition by T. Mitchell (1997):

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

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

Examples for Task T

Classification Task:

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

$\rightsquigarrow$ Handwritten Digits, ...

Regression Task:

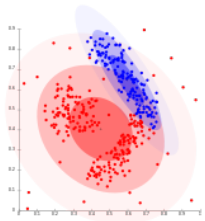
Compute a function $f : \mathbb{R}^n \to \mathbb{R}$, which hence predicts a numerical value.

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

Density Estimation Task:

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

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



Example for Experience E

Experience as a Data Set:

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

Two Cases:

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

Example for Performance Measure P

Accuracy as Performance Measure:

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

Cross-Validation:

The data set is often split into two sets:

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

Linear Regression as one Example, I

Task T:

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

Experience E:

We split our data set into

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

Performance Measure P:

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

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

Linear Regression as one Example, II

Learning Algorithm:

- Define a *hypothesis space*

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

- Given training data

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

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

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

Find the *empirical target function*

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

Linear Regression as one Example, III

Computing the Empirical Target Function:

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

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

- Let

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

- With this notation, we obtain

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

- Let

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

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

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

The General Statistical Learning Problem

A Mathematician's Definition of Learning

Definition (Statistical Learning Problem): Let

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

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

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

Then the corresponding *learning problem* is to find

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

A Mathematician's Definition of Learning

Definition (Statistical Learning Problem): Let

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

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

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

Then the corresponding *learning problem* is to find

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

Simplification:

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

The Regression Function

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

$$\sigma_{\mathcal{X}}(S) = \sigma(\pi_{\mathcal{X}}^{-1}(S)),$$

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

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

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

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

The Regression Function

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

$$\sigma_{\mathcal{X}}(S) = \sigma(\pi_{\mathcal{X}}^{-1}(S)),$$

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

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

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

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

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

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

The Target Function

Definition:

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

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

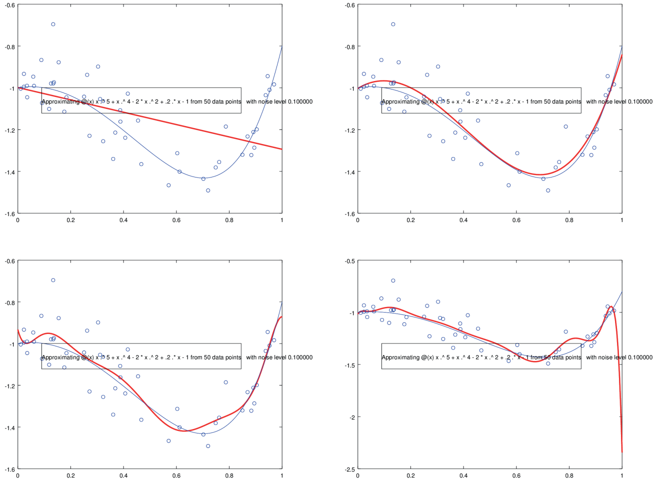
Remark:

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

Examples of Hypotheses Spaces:

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

Underfitting versus Overfitting



The Empirical Target Function

The True Situation:

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

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

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

Definition:

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

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

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

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

The Bias-Variance Decomposition

The error of the empirical target function can be decomposed:

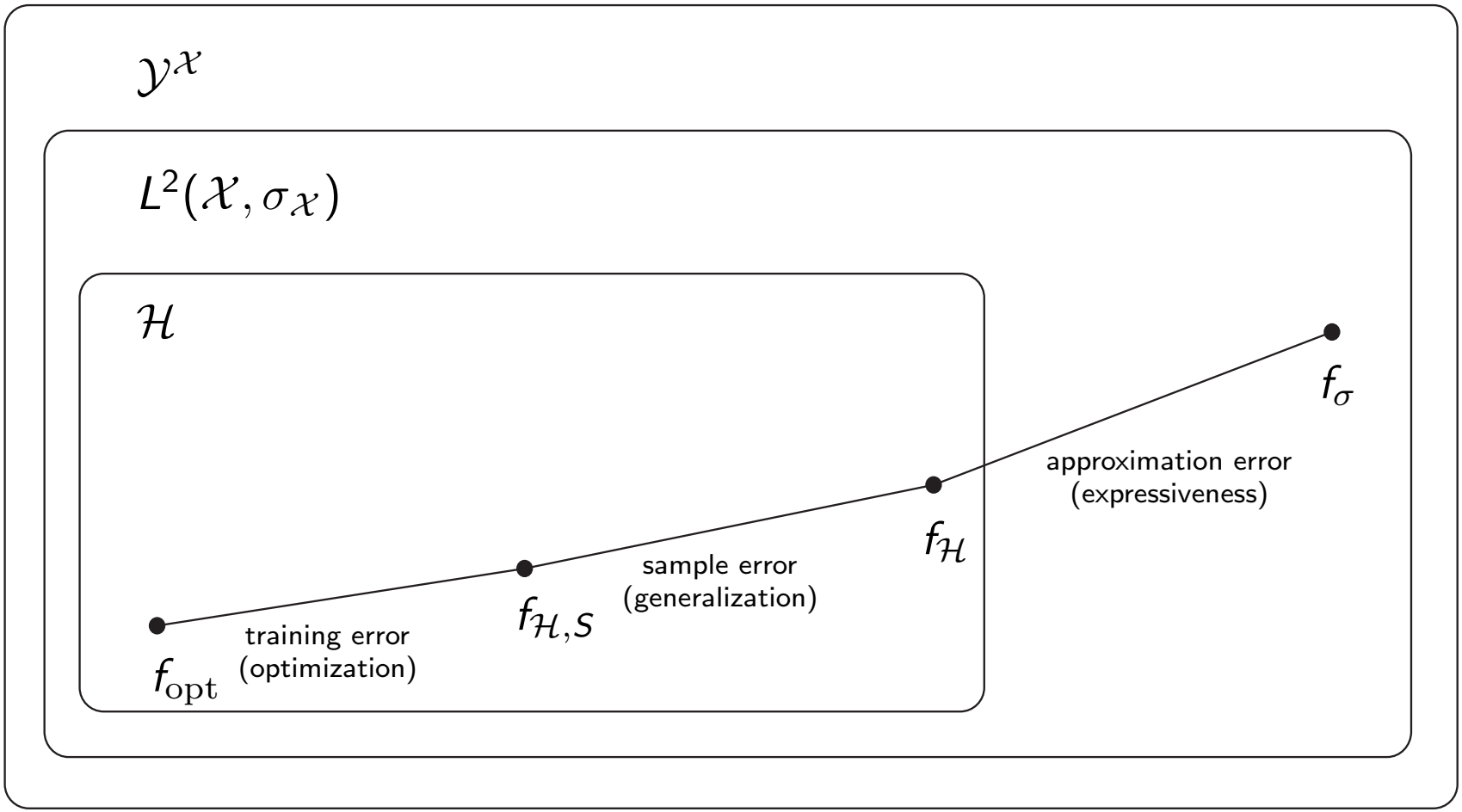
$$\mathcal{E}(f_{\mathcal{H},S}) = \underbrace{\mathcal{E}(f_{\mathcal{H},S}) - \mathcal{E}(f_{\mathcal{H}})}_{\text{sample error}} + \underbrace{\mathcal{E}(f_{\mathcal{H}})}_{\text{approximation error}}$$

Remark:

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

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

Three Types of Errors in Learning Problems



Aim for a universally best method!!!

Fundamental Questions concerning Deep Neural Networks

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

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

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

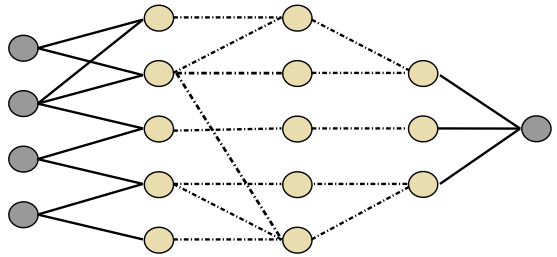
  $\rightsquigarrow$ *Learning Theory, Optimization, Statistics, ...*
- *Interpretability:*
  - Why did a trained deep neural network reach a certain decision?
  - Which components of the input do contribute most?

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

Deep Neural Networks Enter the Stage:

Let's be more concrete....

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 $N_0 := d$,
- $A_\ell \in \mathbb{R}^{N_\ell \times N_{\ell-1}}, \ell = 1, ..., L$ be the weights of the edges
- $b_\ell \in \mathbb{R}^{N_\ell}, \ell = 1, ..., L$ be the biases, and
- $\varrho : \mathbb{R} \to \mathbb{R}$ be the (non-linear) activation function.

Then

$$\Phi = ((A_\ell, b_\ell))_{\ell=1}^{L}$$

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

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

where $x_0 := x$, $x_\ell := \varrho(A_\ell x_{\ell-1} - b_\ell)$, $\ell = 1, ...L-1$, and $x_L := A_L x_{L-1} - b_L$ is called the *realization of Φ with activation function ϱ*.

Key Notions, II

Definition (continued):

We further call

- $N(\Phi) := d + \sum_{\ell=1}^{L} N_L$ the total number of neurons,
- $L(\Phi) := L$ the number of layers,
- $M(\Phi) := \sum_{\ell=1}^{L} \|A_\ell\|_0 + \|b_\ell\|_0$ 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.

Key Notions, II

Definition (continued):

We further call

- $N(\Phi) := d + \sum_{\ell=1}^{L} N_L$ the total number of neurons,
- $L(\Phi) := L$ the number of layers,
- $M(\Phi) := \sum_{\ell=1}^{L} \|A_\ell\|_0 + \|b_\ell\|_0$ 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 ϱ 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

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

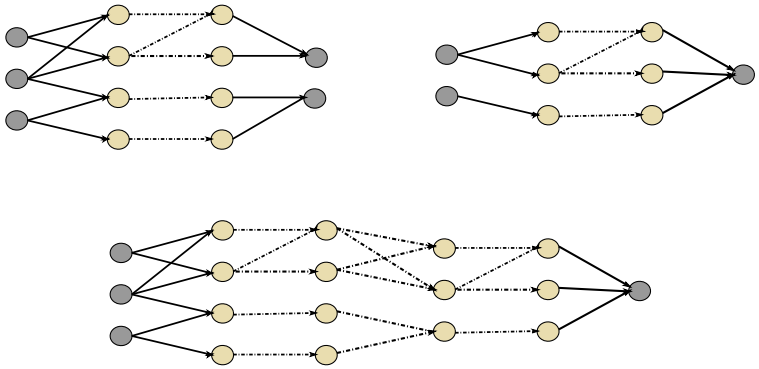
Then $\Phi \in \mathcal{NN}_{d,M,N,L}$ with

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

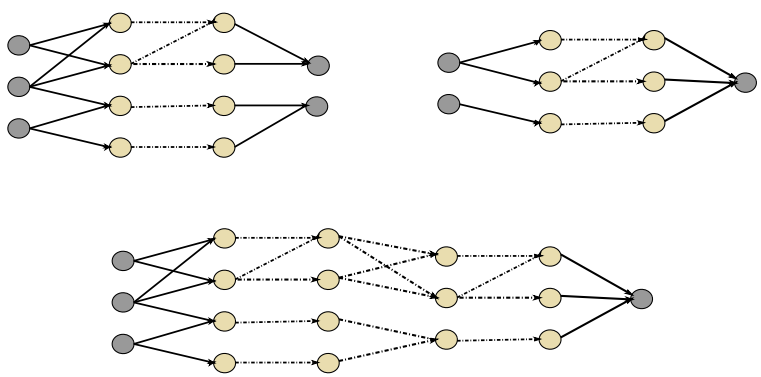
(2) An artificial neuron is a realization with activation function ϱ of a neural network $\Phi \in \mathcal{NN}_{d,M,N,L}$ with

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

Basic Operations: Concatenation



Basic Operations: Concatenation



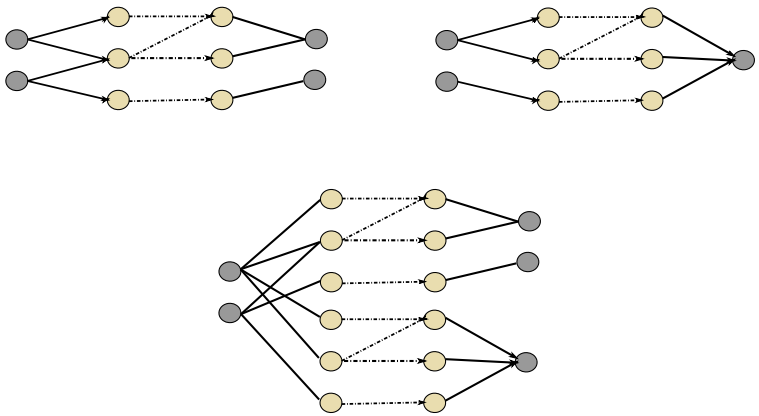
Lemma: Let $L_1, L_2 \in \mathbb{N}$ and $\Phi^i = ((A_1^i, b_1^i), ..., (A_{L_i}^i, b_{L_i}^i)), i = 1, 2$, be neural networks such that the input layer of Φ^1 has the same dimension as the output layer of Φ^2. Then, for any activation function ϱ,

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

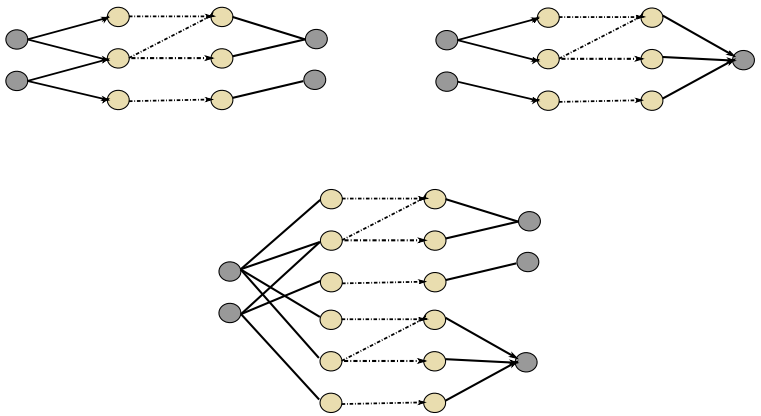
where $\Phi^1 \circ \Phi^2$ denotes the *concatenation of Φ^1 and Φ^2*:

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

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 activation function ϱ and any $x \in \mathbb{R}^d$,

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

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

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

$$\hat{A}_1 = \begin{pmatrix} A_1^1 \\ A_1^2 \end{pmatrix}, \hat{b}_1 = \begin{pmatrix} b_1^1 \\ b_1^2 \end{pmatrix}, \hat{A}_\ell = \begin{pmatrix} A_\ell^1 & 0 \\ 0 & A_\ell^2 \end{pmatrix}, \hat{b}_\ell = \begin{pmatrix} b_\ell^1 \\ b_\ell^2 \end{pmatrix}, 1 \leq \ell \leq L.$$

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

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

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

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

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

Deep Neural Networks: Training

Problem Setting

Let $d = N_0 \in \mathbb{N}$ and $N_1, ..., N_L, L \in \mathbb{N}$ and let ϱ be an activation function. Then consider the hypothesis space

$$\mathcal{H} := \{R_\varrho(\Phi) : \Phi = ((A_1, b_1), ..., (A_L, b_L)),\ A_\ell \in \mathbb{R}^{N_{\ell-1}, N_\ell},\ b_\ell \in \mathbb{R}^{N_\ell}\}.$$

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

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

Problem Setting

Let $d = N_0 \in \mathbb{N}$ and $N_1, ..., N_L, L \in \mathbb{N}$ and let ϱ be an activation function. Then consider the hypothesis space

$$\mathcal{H} := \{R_\varrho(\Phi) : \Phi = ((A_1, b_1), ..., (A_L, b_L)),\ A_\ell \in \mathbb{R}^{N_{\ell-1}, N_\ell},\ b_\ell \in \mathbb{R}^{N_\ell}\}.$$

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

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

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

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

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

Gradient Descent

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

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

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

Gradient Descent

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

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

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

In our problem... we have

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

Since

$$\nabla_{((A_\ell, b_\ell))_{\ell=1}^{L}} F = \sum_{i=1}^{m} \nabla_{((A_\ell, b_\ell))_{\ell=1}^{L}} \mathcal{L}(f, x_i, y_i),$$

we need to compute

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

Backpropagation

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

Result: The matrices $\nabla_{(A_\ell, b_\ell)_{\ell=1}^L} \mathcal{L}(f, x, y)$.

Algorithm:

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

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

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

for $\ell = L - 1$ to 1 **do**

$$\delta_\ell := \operatorname{diag}(\varrho'(z_\ell)) A_{\ell+1}^T \cdot \delta_{\ell+1};$$

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

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

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!*