On the statistical theory of deep learning

Lecture 2

Sophie Langer

Nantes, 19 May 2022

UNIVERSITY
OF TWENTE.

Advantages of multiple layers

- Localization
- Approximation of polynomials with deep neural networks

Localization

Chui et al. (1994) define localized approximation as the ability to approximate $[-1,1]^d$ hypercubes by a neural networks with fixed number of neurons

Local approximation property: There exists a sequence of neural networks $(f_r)_r$ with activation function σ, K neurons und L hidden layers, such that for any $A > 0$

$$\lim_{r\to\infty} \int_{[-A,A]} |\mathbf{1}_{[-1,1]^d}(\mathbf{x}) - f_r(\mathbf{x})| d\mathbf{x} \to 0$$

$\hookrightarrow$ Property does not hold for shallow networks with Heaviside activation function (Theorem 2.2 in Chui et al. (1994))

Localization with shallow networks

- Shallow ReLU networks localize in one dimension:

$$f_r(x) = \sigma(rx + r) - \sigma(rx + r - 1) - \sigma(rx + 1 - r) + \sigma(rx - r)$$

$\hookrightarrow$ It seems that for higher dimension, one can only localize in one direction

- Conjecture: Shallow networks with some activation function σ do not provide local approximation

Localization with multilayer networks

- Taking two hidden layers allows us to localize in arbitrary dimensions
- For Heaviside activation function $\sigma_0 = \mathbf{1}_{\{\cdot \geq 0\}}$:

$$\mathbf{1}_{[-1,1]^d}(\mathbf{x}) = \sigma_0 \left(\sum_{i=1}^{d} \sigma_0(x_i + 1) + \sigma_0(-x_i + 1) - 2d + \frac{1}{2} \right)$$

$\hookrightarrow$ Outer neurons are only activated **iff** all inner neurons output one. This is the case **iff** $i \in \{1, \dots, d\}$, $-1 \leq x_i \leq 1$

Localization with multilayer networks

For Sigmoid activation function $\sigma(x) = 1/(1+\exp(-x))$:

$$\sigma(\alpha x) \approx \sigma_0(x) \quad \text{for large } \alpha.$$

For ReLU activation function $\sigma(x) = \max\{x, 0\}$

$$\sigma(\alpha x) - \sigma(\alpha x - 1) \approx \sigma_0(x), \quad \text{for large } \alpha$$

$\rightsquigarrow$ Approximation quality depends on α

Approximation of x^{2^k} with shallow and deep networks

- The function $x \to x^{2^k}$ lies in the closure of a shallow network with $2^k + 1$ neurons
- For multilayer networks we only need k layers with 3 neurons resp.
- Rescaled finite second order differences

$$\frac{\sigma(t + 2xh) - 2\sigma(t + xh) + \sigma(t)}{\sigma''(t)h^2} \approx x^2$$

Shallow vs. deep

For the approximation with deep networks, we only need a three times differentiable activation function

What do we learn from the example?

- x^{2^k} can be written as

$$\underbrace{x^2 \circ x^2 \circ \cdots \circ x^2}_{k-\text{times}}$$

- Thus: Functions of the form

$$f = g_q \circ \cdots \circ g_0$$

can be better approximated by deep networks

Analysis of deep ReLU networks

$\mathcal{F}(L, r) := \mathcal{F}_\sigma(L, r)$ with $\sigma(x) = \max\{x, 0\}$

We talk about

- Properties of deep ReLU networks
- Approximating different functions by ReLU networks
- Convergence results based on ReLU networks
- Comparison to another statistical method
- Image classification with convolutional neural networks

Properties of deep ReLU networks

1. *Identity network:* Identities can be passed through the network without an error

$$f_{id} : \mathbb{R} \to \mathbb{R}, \quad f_{id}(z) = \sigma(z) - \sigma(-z) = z, \quad z \in \mathbb{R}$$

and

$$f_{id} : \mathbb{R}^d \to \mathbb{R}, \quad f_{id}(\mathbf{x}) = (f_{id}(x_1), \ldots, f_{id}(x_d)) = (x_1, \ldots, x_d), \quad \mathbf{x} \in \mathbb{R}^d$$

Passing on identities via several hidden layers:

$$\begin{aligned} f_{id}^0(\mathbf{x}) &= \mathbf{x}, \quad \mathbf{x} \in \mathbb{R}^d \\ f_{id}^{t+1}(\mathbf{x}) &= f_{id}(f_{id}^t(\mathbf{x})) = \mathbf{x}, \quad t \in \mathbb{N}_0, \mathbf{x} \in \mathbb{R}^d \end{aligned}$$

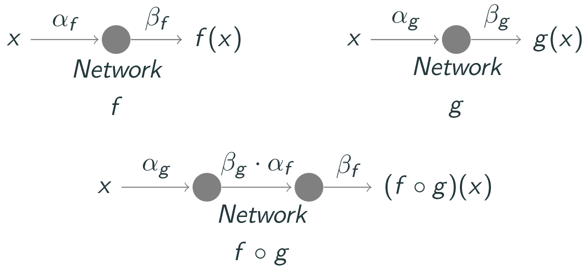
Properties of deep ReLU networks

2. *Combined network*: Two networks can be combined by making the output of one network the input of the other network:

   For $f \in \mathcal{F}(L_f, r_f)$ and $g \in \mathcal{F}(L_g, r_g)$ with $L_f, L_g, r_f, r_g \in \mathbb{N}$ is

$$(f \circ g) \in \mathcal{F}(L_f + L_g, \max\{r_f, r_g\})$$

   the *combined network*.



Properties for deep ReLU networks

3. *Parallelized network:* Two networks with the same number of layers can be computed in a joint network:

   For $f \in \mathcal{F}(L, r_f)$ and $g \in \mathcal{F}(L, r_g)$ is

   $$(f, g)$$

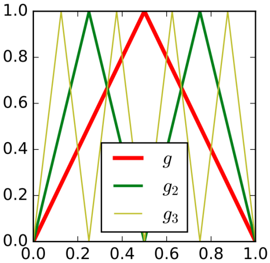
   the parallelised network with L hidden layers and $r_f + r_g$ neurons per layer.
4. *Enlarged network:* We have $\mathcal{F}(L, r) \subseteq \mathcal{F}(L, r')$ with $r \leq r'$.

ReLU approximation of the square function

We start with approximating the square function. Here we use the following result.

Let $g : [0,1] \to [0,1]$ with

$$g(x) = \begin{cases} 2x \quad , & x \leq \frac{1}{2} \\ 2 \cdot (1-x) \quad , & x > \frac{1}{2} \end{cases}$$



and

$$g_s = \underbrace{g \circ g \circ \cdots \circ g}_{s}.$$

Lemma: For $x \in [0,1]$ we have

$$\left| x(1-x) - \sum_{s=1}^{R} \frac{g_s(x)}{2^{2s}} \right| \leq 2^{-2R-2}.$$

ReLU approximation of the square function

Approximating the square function by deep ReLU networks:

Lemma: For each $R \in \mathbb{N}$ and each $a \geq 1$ there exists a network

$$f_{sq} \in \mathcal{F}(R, 9)$$

with

$$|f_{sq}(x) - x^2| \leq a^2 \cdot 4^{-R}$$

for $x \in [-a, a]$.

Multiplication with ReLU networks

We use

$$xy = \frac{1}{4} \cdot ((x+y)^2 - (x-y)^2)$$

and can show:

Lemma: For each $R \in \mathbb{N}$ and each $a \geq 1$ there exist a network

$$f_{mult} \in \mathcal{F}(R, 18)$$

with

$$|f_{mult}(x,y) - xy| \leq 2 \cdot a^2 \cdot 4^{-R}$$

for $x, y \in [-a, a]$.

Approximating a product of d components with ReLU networks

Lemma: For each $R \in \mathbb{N}$ and each $a \geq 1$ there exists a network

$$f_{mult,d} \in \mathcal{F}(R \cdot \lceil \log_2(d) \rceil, 18d)$$

with

$$\left| f_{mult,d}(\mathbf{x}) - \prod_{i=1}^{d} x_i \right| \leq 4^{4d+1} \cdot a^{4d} \cdot d \cdot 4^{-R}$$

for $\mathbf{x} \in [-a, a]^d$.

Approximating polynomials with ReLU networks

Let $\mathcal{P}_N$ be the linear span of all monomials of the form

$$\prod_{k=1}^{d} (x_k)^{r_k}$$

for $r_1, \ldots, r_d \in \mathbb{N}_0$ and $r_1 + \cdots + r_d \leq N$. Then $\mathcal{P}_N$ is a linear vector space with

$$dim\ \mathcal{P}_N = \left| \left\{ (r_0, \ldots, r_d) \in \mathbb{N}_0^{d+1} : r_0 + \cdots + r_d = N \right\} \right| = \binom{d+N}{d}.$$

Lemma: Let $m_1, \ldots, m_{\binom{d+N}{d}}$ be all monomials of the space $\mathcal{P}_N$ for $N \in \mathbb{N}$. For $r_1, \ldots, r_{\binom{d+N}{d}} \in \mathbb{R}$ let

$$p\left(\mathbf{x}, y_1, \ldots, y_{\binom{d+N}{d}}\right) = \sum_{i=1}^{\binom{d+N}{d}} r_i \cdot y_i \cdot m_i(\mathbf{x}), \quad \mathbf{x} \in [-a, a]^d, y_i \in [-a, a]$$

and let $\bar{r}(p) = \max_{i \in \{1, \ldots, \binom{d+N}{d}\}} |r_i|$.

Approximating polynomials with ReLU networks

Then for every $a \geq 1$ and every $R \in \mathbb{N}$ the network

$$f_p \in \mathcal{F}\left(R \cdot \lceil \log_2(N+1) \rceil, 18 \cdot (N+1) \cdot \binom{d+N}{d}\right)$$

satsifes

$$\left| f_p(\mathbf{x}, y_1, \dots, y_{\binom{d+N}{d}}) - p(\mathbf{x}, y_1, \dots, y_{\binom{d+N}{d}}) \right| \leq c(d,N) \cdot \bar{r}(p) \cdot a^{4(N+1)} \cdot 4^{-R}$$

for all $\mathbf{x} \in [-a,a]^d$, $y_1, \dots, y_{\binom{d+N}{d}} \in [-a,a]$ and a constant $c(d,N) > 0$, only depending on d and N.

Approximating (p, C)-smooth functions by ReLU networks

In the following we approximate smooth functions with ReLU networks. In particular, we consider functions of the following definition:

Definition: Let $p = q + s$ for $q \in \mathbb{N}_0$ and $0 < s \leq 1$. Let $C > 0$. A function $f : \mathbb{R}^d \to \mathbb{R}$ is *(p,C)-smooth*, if for every $\boldsymbol{\alpha} \in \mathbb{N}_0^d$ with $\sum_{j=1}^d \alpha_j = q$ the partial derivative $\partial^q f / (\partial x_1^{\alpha_1} \ldots \partial x_d^{\alpha_d})$ exists and satisfies

$$\left| \frac{\partial^q f}{\partial x_1^{\alpha_1} \ldots \partial x_d^{\alpha_d}}(\mathbf{x}) - \frac{\partial^q f}{\partial x_1^{\alpha_1} \ldots \partial x_d^{\alpha_d}}(\mathbf{z}) \right| \leq C \cdot \|\mathbf{x} - \mathbf{z}\|^s$$

for all $\mathbf{x}, \mathbf{z} \in \mathbb{R}^d$.

Approximating (p, C)-smooth functions by ReLU networks

The following result shows a Taylor approximation of (p, C)-smooth functions.

Lemma: Let $p = q + s$ for $q \in \mathbb{N}_0$ and $s \in (0, 1]$. Let $C > 0$. Let $f : \mathbb{R}^d \to \mathbb{R}$ a (p, C)-smooth function, let $\mathbf{x}_0 \in \mathbb{R}^d$ and $T_{f,q,\mathbf{x}_0}$ a Taylor polynomial of order q around $\mathbf{x}_0$ defined by

$$T_{f,q,\mathbf{x}_0}(\mathbf{x}) = \sum_{\mathbf{j} \in \mathbb{N}_0 : \|\mathbf{j}\|_1 \leq q} (\partial^{\mathbf{j}} f)(\mathbf{x}_0) \cdot \frac{(\mathbf{x} - \mathbf{x}_0)^{\mathbf{j}}}{\mathbf{j}!}.$$

Then we have

$$|f(\mathbf{x}) - T_{f,q,\mathbf{x}_0}(\mathbf{x})| \leq c(q, d) \cdot C \cdot \|\mathbf{x} - \mathbf{x}_0\|^p$$

for $\mathbf{x} \in \mathbb{R}^d$ and with $c(q, d) > 0$ only depending on q and d.

Proof: Lemma 1 in Kohler (2014).

Approximating (p, C)-smooth functions with ReLU networks

Idea of the proof:

- We partition $[-a, a)^d$ $(a \geq 1)$ in M^d and M^{2d} half-open equivolume cubes

$$[\boldsymbol{\alpha}, \boldsymbol{\beta}) = [\alpha_1, \beta_1) \times \cdots \times [\alpha_d, \beta_d), \quad \boldsymbol{\alpha}, \boldsymbol{\beta} \in \mathbb{R}^d.$$

- And denote the corresponding partition by

$$\mathcal{P}_1 = \{C_{k,1}\}_{k \in \{1,\dots,M^d\}} \text{ und } \mathcal{P}_2 = \{C_{j,2}\}_{j \in \{1,\dots,M^{2d}\}}$$

- For each $i \in \{1, \dots, M^d\}$ we denote with $\tilde{C}_{1,i}, \dots, \tilde{C}_{M^d,i}$ the cubes of $\mathcal{P}_2$ contained in $C_{i,1}$
- We order the cubes in such a way that for $k, i \in \{1, \dots, M^d\}$

$$(\tilde{C}_{k,i})_{left} = (C_{i,1})_{left} + \mathbf{v}_k,$$

where $\mathbf{v}_k \in \{0, 2a/M^2, \dots, (M-1) \cdot 2a/M^2\}^d$.

Approximating (p, C)-smooth functions with ReLU networks

$$(\tilde{C}_{k,i})_{left} = (C_{i,1})_{left} + \mathbf{v}_k$$

- $\mathbf{v}_k$ denotes the position of $(\tilde{C}_{k,i})_{left}$ relatively to $(C_{i,1})_{left}$ and we order the cubes such that this position is independent of i
- Then we have

$$\mathcal{P}_2 = \{\tilde{C}_{k,i}\}_{k,i \in \{1,\dots,M^d\}}$$

- The Taylor expansion $T_{f,q,(C_{\mathcal{P}_2}(\mathbf{x}))_{left}}(\mathbf{x})$ can then be computed by the piecewise Taylor polynomial defined on $\mathcal{P}_2$:

$$T_{f,q,(C_{\mathcal{P}_2}(\mathbf{x}))_{left}}(\mathbf{x}) = \sum_{k,i \in \{1,\dots,M^d\}} T_{f,q,(\tilde{C}_{k,i})_{left}}(\mathbf{x}) \cdot \mathbf{1}_{\tilde{C}_{k,i}}(\mathbf{x})$$

Approximating (p, C)-smooth functions with ReLU networks

Using the Lemma from above leads to

$$\left\| f(\mathbf{x}) - T_{f,q,(C_{\mathcal{P}_2}(\mathbf{x}))_{left}}(\mathbf{x}) \right\|_{\infty,[-a,a)^d} \leq c(q,d) \cdot (2 \cdot a \cdot d)^p \cdot C \cdot \frac{1}{M^{2p}}.$$

Theorem: Let

- $f : \mathbb{R}^d \to \mathbb{R}$ be a (p, C)-smooth function
- $a \geq 1, \ M \geq 2$
- $L \gtrsim \log_4(M)$
- $r \gtrsim M^d$

Then there exists a network $\hat{f}_{wide} \in \mathcal{F}(L, r)$, such that

$$\|f - \hat{f}_{wide}\|_{\infty,[-a,a]^d} \lesssim M^{-2p}$$

Approximating (p, C)-smooth functions with deep ReLU networks

A similar result holds for very *deep* ReLU networks:

Theorem: Let

- $f : \mathbb{R}^d \to \mathbb{R}$ be a (p, C)-smooth function
- $a \geq 1,\ M \geq 2$
- $L \gtrsim M^d$
- $r = const.$

Then there exists a network $\hat{f}_{deep} \in \mathcal{F}(L, r)$ with

$$\|f - \hat{f}_{deep}\|_{\infty,[-a,a]^d} \lesssim M^{-2p}.$$

Proof: See Theorem 2 in Kohler und Langer (2021).

Mathematical problem

$X = \{Images\}$

$$\xrightarrow{f : X \to Y} Y = \{Muffin, Chiwawa\}$$

The data are used to fit a network, i.e. estimate the weights in the network

How fast does the estimated network converge to the truth function f as sample size increases?

Nonparametric regression

Prediction problem

- Given a $\mathbb{R}^d \times \mathbb{R}$-valued random vector $(\mathbf{X}, Y)$ with $\mathbf{E}\{Y^2\} < \infty$

  Functional relation between $\mathbf{X}$ *and* Y*?*

- Choose $f^* : \mathbb{R}^d \to \mathbb{R}$ such that

$$\mathbf{E}\left\{|f^*(\mathbf{X}) - Y|^2\right\} = \min_{f:\mathbb{R}^d \to \mathbb{R}} \mathbf{E}\left\{|f(\mathbf{X}) - Y|^2\right\}.$$

- One can show that $f^*(\mathbf{x}) = m(\mathbf{x}) = \mathbf{E}\{Y|\mathbf{X} = \mathbf{x}\}$ holds.
- $m : \mathbb{R}^d \to \mathbb{R}$ is the so-called **regression function**

Nonparametric regression

- **Problem:** Distribution of $(\mathbf{X}, Y)$ is unknown
- But: We have given n copies of $(\mathbf{X}, Y)$
  $\rightsquigarrow \mathcal{D}_n = \{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)\}$ (i.i.d.)
- Aim: Construct an estimator

$$m_n(\cdot) = m_n(\cdot, \mathcal{D}_n) : \mathbb{R}^d \to \mathbb{R},$$

such that the L_2 risk

$$\int |m_n(\mathbf{x}) - m(\mathbf{x})|^2 d\mathbf{x}$$

is *small*.

Regression estimator

Neural network estimator:

$$\tilde{m}_n(\cdot) = \text{argmin}_{f \in \mathcal{F}(L_n, r_n)} \frac{1}{n} \sum_{i=1}^{n} |f(\mathbf{X}_i) - Y_i|^2$$

and set $m_n(\mathbf{x}) = T_{c \cdot \log(n)} \tilde{m}_n(\mathbf{x}) = \max\{-c \cdot \log(n), \min\{\mathbf{x}, c \cdot \log(n)\}\}$

Analyse the expected L_2 error

$$\mathbf{E} \int |m_n(\mathbf{x}) - m(\mathbf{x})|^2 \, \mathbf{P}_{\mathbf{X}}(d\mathbf{x})$$

$\hookrightarrow$ Study the dependence of n (convergence rate)

The choice of the function class

- Classical approach: Regression function is (p, C)-smooth
- Optimal rate: $n^{-\frac{2p}{2p+d}}$ (Stone (1982))

The choice of the function class

- Classical approach: Regression function is (p, C)-smooth
- Optimal rate: $n^{-\frac{2p}{2p+d}}$ (Stone (1982))
  $\hookrightarrow$ suffers from the *curse of dimensionality*
- For a better understanding of deep learning, this setting is useless
- Aim: Find a proper structural assumption on m, such that neural network estimators can achieve good convergence results even in high dimensions

The choice of the function class

Additive models

- $m(\mathbf{x}) = \sum_{k=1}^{K} g_k(x_k)$ with $g_k : \mathbb{R} \to \mathbb{R}$ (p, C)-smooth
  Optimal rate $n^{-\frac{2p}{2p+1}}$ (Stone (1985))

The choice of the function class

Additive models

- $m(\mathbf{x}) = \sum_{k=1}^{K} g_k(x_k)$ with $g_k : \mathbb{R} \to \mathbb{R}$ (p, C)-smooth
  Optimal rate $n^{-\frac{2p}{2p+1}}$ (Stone (1985))
- Interactionmodels

$$m(\mathbf{x}) = \sum_{I \subset \{1,\ldots,d\}, |I| \le d^*} g_I(x_I)$$

  with $g_I(x_I) : \mathbb{R}^{|I|} \to \mathbb{R}$ (p, C)-smooth
  Optimal rate $n^{-\frac{2p}{2p+d^*}}$ (Stone (1995))

$\rightsquigarrow$ For both models the rate does not depend on d anymore

The choice of the function class

Single index model

$$m(\mathbf{x}) = g(\mathbf{a}^T \mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^d$$

with $g : \mathbb{R} \to \mathbb{R}$ univariate and $\mathbf{a} \in \mathbb{R}^d$ being a d-dimensional vector.

The choice of the function class

Single index model

$$m(\mathbf{x}) = g(\mathbf{a}^T\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^d$$

with $g : \mathbb{R} \to \mathbb{R}$ univariate and $\mathbf{a} \in \mathbb{R}^d$ being a d-dimensional vector.

Projection pursuit model

$$m(\mathbf{x}) = \sum_{k=1}^{K} g_k(\mathbf{a}_k^T\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^d$$

for $K \in \mathbb{N}$, $g_k : \mathbb{R} \to \mathbb{R}$ and $\mathbf{a}_k \in \mathbb{R}^d$
$\hookrightarrow$ Optimal rate $n^{-\frac{2p}{2p+1}}$ (Györfi et al. (2002))

The choice of the function class

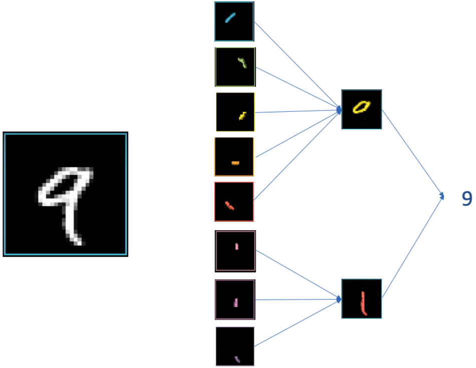
- With all models one can circumvent the curse of dimensionality
- But: Rates can only be obtained in practice if the true (then unknown) regression function corresponds to this structure

$\hookrightarrow$ Goal: Low assumptions on the regression function that allow good rate of convergence results

The choice of the function class

Im many applications the corresponding functions show some sort of a **hierarchical structure**:

- Image processing: Pixel $\rightarrow$ Edges $\rightarrow$ Local patterns $\rightarrow$ object



The choice of the function class

Hierarchical composition model:

a) We say that m satisfies a *hierarchical composition model of level 0*, if there exists a $K \in \{1, \ldots, d\}$ such that

$$m(\mathbf{x}) = x_K \quad \text{for all } \mathbf{x} \in \mathbb{R}^d.$$

The choice of the function class

Hierarchical composition model:

a) We say that m satisfies a *hierarchical composition model of level 0*, if there exists a $K \in \{1, \dots, d\}$ such that

$$m(\mathbf{x}) = x_K \quad \text{for all } \mathbf{x} \in \mathbb{R}^d.$$

b) We say that m satisfies a *hierarchical composition model of level* $l+1$, if there exist a $K \in \mathbb{N}$, $g : \mathbb{R}^K \to \mathbb{R}$ and $f_1, \dots, f_K : \mathbb{R}^d \to \mathbb{R}$ such that $f_1, \dots, f_K$ satisfy a hierarchical composition model of level l and

$$m(\mathbf{x}) = g(f_1(\mathbf{x}), \dots, f_K(\mathbf{x})) \quad \text{for all } \mathbf{x} \in \mathbb{R}^d.$$

Hierarchical composition model - Example

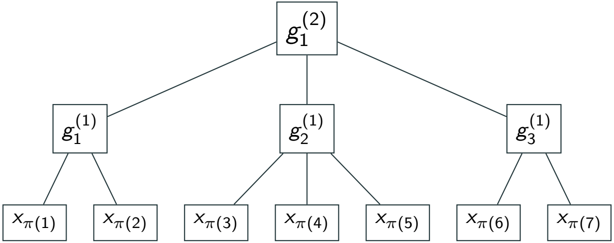


Illustration of a hierarchical composition model of level 2

Hierarchical composition models

The hierarchical composition model satisfies the *smoothness and order constraint* $\mathcal{P}$, if

- $\mathcal{P} \subseteq [1, \infty) \times \mathbb{N}$
- all functions g satisfy $g : \mathbb{R}^K \to \mathbb{R}$ and g is (p, C)-smooth for some $(p, K) \in \mathcal{P}$

Hierarchical composition models

The hierarchical composition model satisfies the *smoothness and order constraint* $\mathcal{P}$, if

- $\mathcal{P} \subseteq [1, \infty) \times \mathbb{N}$
- all functions g satisfy $g : \mathbb{R}^K \to \mathbb{R}$ and g is (p, C)-smooth for some $(p, K) \in \mathcal{P}$

Further assumptions

- all functions g are Lipschitz continuous
- $\mathbf{E}(\exp(c \cdot Y^2)) < \infty$ and $\mathrm{supp}(\mathbf{X})$ is bounded

Results for sparse neural network estimators

Theorem(Schmidt-Hieber (2020)): If

- $L \asymp \log(n)$
- $r \asymp n^C$, with $C \geq 1$
- network sparsity $\asymp \max_{(p,K)\in\mathcal{P}} n^{\frac{K}{2p+K}} \cdot \log(n)$.

the neural network estimator with ReLU activation function achieves the rate of convergence

$$\max_{(p,K)\in\mathcal{P}} n^{-\frac{2p}{2p+K}}.$$

Results for sparse neural network estimators

Result of Bauer and Kohler (2019): For a generalized hierarchical interaction model a sparse neural network estimator with sigmoidal activation function achieves a rate of convergence

$$n^{-\frac{2p}{2p+d^*}}.$$

Is sparsity really necessary?

Remark

Sparse neural network estimators are able circumvent the curse of dimensionality

Is sparsity really necessary?

Remark

Sparse neural network estimators are able circumvent the curse of dimensionality

Conjecture

In order to achieve good rate of convergence results, one should use neural networks, which are not fully connected.

Is sparsity really necessary?

Remark

Sparse neural network estimators are able circumvent the curse of dimensionality

Conjecture

In order to achieve good rate of convergence results, one should use neural networks, which are not fully connected. $\rightsquigarrow$ This is **not true**!

Result for fully connected neural network estimators

Theorem: If

- number of hidden layer $L_n \asymp \max_{(p,K)\in\mathcal{P}} n^{\frac{K}{2\cdot(2p+K)}}$
- number of neurons $r_n = \lceil \tilde{c} \rceil$

or

- number of hidden layer $L_n \asymp \log(n)$
- number of neurons $r_n \asymp \max_{(p,K)\in\mathcal{P}} n^{\frac{K}{2\cdot(2p+K)}}$.

Result for fully connected neural network estimators

Theorem: If

- number of hidden layer $L_n \asymp \max_{(p,K)\in\mathcal{P}} n^{\frac{K}{2\cdot(2p+K)}}$
- number of neurons $r_n = \lceil \tilde{c} \rceil$

or

- number of hidden layer $L_n \asymp \log(n)$
- number of neurons $r_n \asymp \max_{(p,K)\in\mathcal{P}} n^{\frac{K}{2\cdot(2p+K)}}$.

Then

$$\mathbf{E} \int |m_n(\mathbf{x}) - m(\mathbf{x})|^2 \mathbf{P}_{\mathbf{X}}(d\mathbf{x}) \le c \cdot (\log(n))^6 \cdot \max_{(p,K)\in\mathcal{P}} n^{-\frac{2p}{2p+K}}.$$

Advantage of full connectivity

Topology of the network is much easier in view of an implementation of a corresponding estimator:

Listing 1: Python code for fitting of fully connected neural networks to data x_{learn} and y_{learn}

```
model = Sequential()
model.add(Dense(d, activation="relu", input_shape=(d,)))
for i in np.arange(L):
    model.add(Dense(K, activation="relu"))
model.add(Dense(1))
model.compile(optimizer="adam",
              loss="mean_squared_error")
model.fit(x=x_learn,y=y_learn)
```

Excursion

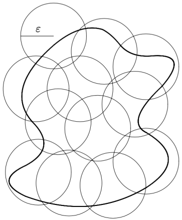
Let

- $\epsilon > 0$
- $\mathcal{G}$ a set of functions $f : \mathbb{R}^d \to \mathbb{R}$
- $\mathbf{z}_1^n = (\mathbf{z}_1, \dots, \mathbf{z}_n)$ n fixed points in $\mathbb{R}^d$.

Then we denote by

(a) $\mathcal{N}_1(\epsilon, \mathcal{G}, \mathbf{z}_1^n)$ the minimal $N \in \mathbb{N}$ such that there exist functions $g_1, \dots, g_N : \mathbb{R}^d \to \mathbb{R}$ with the property that for every $g \in \mathcal{G}$ there is a $j = j(g) \in \{1, \dots, N\}$ such that

$$\frac{1}{n} \sum_{i=1}^{n} |g(\mathbf{z}_i) - g_j(\mathbf{z}_i)| < \epsilon.$$

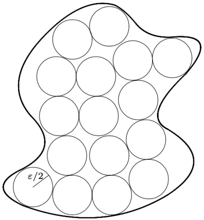


Excursion

(b) $\mathcal{M}_1(\epsilon, \mathcal{G}, \mathbf{z}_1^n)$ the maximal $M \in \mathbb{N}$ such that there exist function $g_1, \ldots, g_M \in \mathcal{G}$ with

$$\frac{1}{n}\sum_{i=1}^{n} |g_j(\mathbf{z}_i) - g_k(\mathbf{z}_i)| \geq \epsilon$$

for all $1 \leq j < k \leq M$.



Source: Györfi et al. (2002)

Excursion

Let $\mathcal{A}$ be a class of subsets of $\mathbb{R}^d$ with $\mathcal{A} \neq \emptyset$ and $n \in \mathbb{N}$. Then

- $s(\mathcal{A}, \{z_1, \ldots, z_n\}) = |\{A \cap \{z_1, \ldots, z_n\} : A \in \mathcal{A}\}|$ denotes the number of different subsets of $\{z_1, \ldots, z_n\}$ of the form $\{A \cap \{z_1, \ldots, z_n\}, A \in \mathcal{A}\}$

Excursion

Let $\mathcal{A}$ be a class of subsets of $\mathbb{R}^d$ with $\mathcal{A} \neq \emptyset$ and $n \in \mathbb{N}$. Then

- $s(\mathcal{A}, \{z_1, \ldots, z_n\}) = |\{A \cap \{z_1, \ldots, z_n\} : A \in \mathcal{A}\}|$ denotes the number of different subsets of $\{z_1, \ldots, z_n\}$ of the form $\{A \cap \{z_1, \ldots, z_n\}, A \in \mathcal{A}\}$
- $S(\mathcal{A}, n) = \max_{\{z_1,\ldots,z_n\} \subset \mathbb{R}^d} s(\mathcal{A}, \{z_1, \ldots, z_n\})$ denotes the maximal number of different subsets of n points that can be picked out by sets from $\mathcal{A}$

Excursion

Let $\mathcal{A}$ be a class of subsets of $\mathbb{R}^d$ with $\mathcal{A} \neq \emptyset$ and $n \in \mathbb{N}$. Then

- $s(\mathcal{A}, \{z_1, \ldots, z_n\}) = |\{A \cap \{z_1, \ldots, z_n\} : A \in \mathcal{A}\}|$ denotes the number of different subsets of $\{z_1, \ldots, z_n\}$ of the form $\{A \cap \{z_1, \ldots, z_n\}, A \in \mathcal{A}\}$
- $S(\mathcal{A}, n) = \max_{\{z_1,\ldots,z_n\} \subset \mathbb{R}^d} s(\mathcal{A}, \{z_1, \ldots, z_n\})$ denotes the maximal number of different subsets of n points that can be picked out by sets from $\mathcal{A}$
- $V_{\mathcal{A}} = \sup\{n \in \mathbb{N} : S(\mathcal{A}, n) = 2^n\}$ is the *VC dimension*, that denotes the largest integer n such that there exists a set of n points in $\mathbb{R}^d$ such that each of its subsets can be represented in the form $A \cap \{z_1, \ldots, z_n\}$ for some $A \in \mathcal{A}$.

Excursion

Let $\mathcal{A}$ be a class of subsets of $\mathbb{R}^d$ with $\mathcal{A} \neq \emptyset$ and $n \in \mathbb{N}$. Then

- $s(\mathcal{A}, \{z_1, \ldots, z_n\}) = |\{A \cap \{z_1, \ldots, z_n\} : A \in \mathcal{A}\}|$ denotes the number of different subsets of $\{z_1, \ldots, z_n\}$ of the form $\{A \cap \{z_1, \ldots, z_n\}, A \in \mathcal{A}\}$
- $S(\mathcal{A}, n) = \max_{\{z_1,\ldots,z_n\} \subset \mathbb{R}^d} s(\mathcal{A}, \{z_1, \ldots, z_n\})$ denotes the maximal number of different subsets of n points that can be picked out by sets from $\mathcal{A}$
- $V_{\mathcal{A}} = \sup\{n \in \mathbb{N} : S(\mathcal{A}, n) = 2^n\}$ is the *VC dimension*, that denotes the largest integer n such that there exists a set of n points in $\mathbb{R}^d$ such that each of its subsets can be represented in the form $A \cap \{z_1, \ldots, z_n\}$ for some $A \in \mathcal{A}$.

For some function class $\mathcal{G}$ we denote by

$$\mathcal{G}^+ := \left\{ \left\{ (z, t) \in \mathbb{R}^d \times \mathbb{R} : t \leq g(z) \right\} ; g \in \mathcal{G} \right\}$$

the set of all subgraphs of functions of $\mathcal{G}$.

On the proof

Lemma: Let

- $\mathbf{E}\{\exp(c \cdot Y^2)\} < \infty$ for a constant $c > 0$
- $|m| < \infty$
- $\tilde{m}_n$ be a least squares estimator on the function space $\mathcal{F}_n$
- $m_n(\cdot) = T_{\tilde{c}\cdot\log(n)}\tilde{m}_n$ for a constant $\tilde{c} > 0$.

Then we have for $n > 1$ and a constant $c > 0$ (independent of n and the parameters of the estimator)

$$\mathbf{E}\int |m_n(\mathbf{x}) - m(\mathbf{x})|^2 \mathbf{P}_{\mathbf{X}}(d\mathbf{x})$$

$$\leq \frac{c \cdot (\log n)^2 \cdot \sup_{\mathbf{x}_1^n \in (\mathbb{R}^d)^n} \left(\log\left(\mathcal{N}_1\left(\frac{1}{n\cdot\tilde{c}\log(n)}, T_{\tilde{c}\log(n)}\mathcal{F}_n, \mathbf{x}_1^n\right)\right) + 1\right)}{n}$$

$$+ 2 \cdot \inf_{f \in \mathcal{F}_n} \int |f(\mathbf{x}) - m(\mathbf{x})|^2 \mathbf{P}_{\mathbf{X}}(d\mathbf{x}).$$

On the proof

Lemma: Let

- $1/n^c \leq \epsilon < \tilde{c} \cdot \log(n)/8$
- $L, r \in \mathbb{N}$.

Then we have for sufficiently large n, $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ and a constant c independent of n, L und r

$$\log\left(\mathcal{N}_1\left(\frac{1}{n \cdot \tilde{c}\log(n)}, T_{\tilde{c}\log(n)}\mathcal{F}_n, \mathbf{x}_1^n\right)\right) \leq c \cdot \log(n) \cdot \log(L \cdot r^2) \cdot L^2 \cdot r^2.$$

On the proof

To proof this we need the following results:

Lemma 1: Let $\mathcal{G}$ a class of functions on $\mathbb{R}^d$ and $\epsilon > 0$. Then we have

$$\mathcal{N}_1(\epsilon, \mathcal{G}, \mathbf{z}_1^n) \leq \mathcal{M}_1(\epsilon, \mathcal{G}, \mathbf{z}_1^n)$$

für all $\mathbf{z}_1, \ldots, \mathbf{z}_n \in \mathbb{R}^d$.

Proof: See Lemma 9.2 in Györfi et al. (2002).

On the proof

To proof this we need the following results:

Lemma 1: Let $\mathcal{G}$ a class of functions on $\mathbb{R}^d$ and $\epsilon > 0$. Then we have

$$\mathcal{N}_1(\epsilon, \mathcal{G}, \mathbf{z}_1^n) \leq \mathcal{M}_1(\epsilon, \mathcal{G}, \mathbf{z}_1^n)$$

für all $\mathbf{z}_1, \ldots, \mathbf{z}_n \in \mathbb{R}^d$.

Proof: See Lemma 9.2 in Györfi et al. (2002).

Lemma 2: Let $\mathcal{G}$ be a class of functions $g : \mathbb{R}^d \to [-B, B]$ with $V_{\mathcal{G}^+} \geq 2$ and let $0 < \epsilon < B/8$. Then we have

$$\mathcal{M}_1(\epsilon, \mathcal{G}, \mathbf{z}_1^n) \leq 3 \left(\frac{4eB}{\epsilon} \log \left(\frac{6eB}{\epsilon} \right) \right)^{V_{\mathcal{G}^+}}$$

for all $\mathbf{z}_1, \ldots, \mathbf{z}_n \in \mathbb{R}^d$.

Proof: See Theorem 9.4 in Györfi et al. (2002).

On the proof

Lemma 3: Let $L, r \in \mathbb{N}$ und $\mathcal{F}(L, r)$ be the corresponding class of neural networks. Then we have

$$V_{\mathcal{F}(L,r)^+} \leq c \cdot L^2 \cdot r^2 \cdot \log(L^2 \cdot r^2)$$

for a constant $c > 0$ sufficiently large.

On the proof

Lemma 3: Let $L, r \in \mathbb{N}$ und $\mathcal{F}(L, r)$ be the corresponding class of neural networks. Then we have

$$V_{\mathcal{F}(L,r)^+} \leq c \cdot L^2 \cdot r^2 \cdot \log(L^2 \cdot r^2)$$

for a constant $c > 0$ sufficiently large.

Proof: Follows from Theorem 6 in Bartlett et al. (2017) and the fact, that a fully connected network with L hidden layers and r neurons per layer has

$$\begin{aligned} W &= (d+1) \cdot r + (L-1) \cdot (r-1) \cdot r + r + 1 \\ &= (d+1) \cdot r + L \cdot (r^2 + r) - r^2 + 1 \end{aligned}$$

weights.

Summary

- **Deep neural networks ars able to circumvent the curse of dimensionality** under structural assumptions on the regression function
- **Sparsety is not necessary** to derive good rate of convergence

Regression functions with low local dimensionality

Observation

Highdimensional data follow locally a low dimensional distribution

Example

Bike sharing data

Assumption

Regressionfunction is locally low dimensional

$\rightsquigarrow$ m depends locally only on a small number of input components

Regression functions with low local dimensionality

A mathematical formulation

Let $A_1, \ldots, A_K \subset \mathbb{R}^d$, $f_1, \ldots, f_K : \mathbb{R}^d \to \mathbb{R}$ and $J_1, \ldots, J_K \subset \{1, \ldots, d\}$ be index sets with maximal cardinality d^*. Then the function m is of the form

$$m(\mathbf{x}) = \sum_{k=1}^{K} f_k(\mathbf{x}_{J_k}) \cdot \mathbf{1}_{A_k}(\mathbf{x}).$$

Problem: Function is globally neither (p, C)-smooth nor continuous $\hookrightarrow$ unrealistic!

Regression functions with low local dimensionality

Let $A_1, \ldots, A_K$ be d-dimensional polytopes. Let $\mathbf{a}_{i,k} \in \mathbb{R}^d$ with $\|\mathbf{a}_{i,k}\| \leq 1$, $b_{i,k} \in \mathbb{R}$ $\delta_{i,k} > \epsilon > 0$, $K_1 \in \mathbb{N}$

$$(P_k)_{\delta_k} = \left\{ \mathbf{x} \in \mathbb{R}^d : \mathbf{a}_{i,k}^T \mathbf{x} \leq b_{i,k} - \delta_{i,k} \text{ for } i \in \{1, \ldots, K_1\} \right\}$$

and

$$(P_k)^{\delta_k} = \left\{ \mathbf{x} \in \mathbb{R}^d : \mathbf{a}_{i,k}^T \mathbf{x} \leq b_{i,k} + \delta_{i,k} \text{ for } i \in \{1, \ldots, K_1\} \right\}$$

with $\delta_k = (\delta_{1,k}, \ldots, \delta_{K,k})$.

Regression functions with low local dimensionality

Definition (Kohler, Krzyżak and L. (2022))

A function $f : \mathbb{R}^d \to \mathbb{R}$ has local dimensionality $d^* \in \{1, \dots, d\}$ on $[-A, A]^d$ for $A > 0$ with order (K_1, K_2), $\mathbf{P_X}$-border $\epsilon > 0$ and borders $\delta_{i,k} > 0$ for $i = 1, \dots, K_1$, $k = 1, \dots, K_2$, if there exist functions

$$f_k : \mathbb{R}^{d^*} \to \mathbb{R}$$

and $\delta_k = (\delta_{1,k}, \dots, \delta_{K_1,k})$ such that

$$\sum_{k=1}^{K_2} f_k(\mathbf{x}_{J_k}) \cdot 1_{(P_k)_{\delta_k}}(\mathbf{x}) \le f(\mathbf{x}) \le \sum_{k=1}^{K_2} f_k(\mathbf{x}_{J_k}) \cdot 1_{(P_k)^{\delta_k}}(\mathbf{x}) \quad (\mathbf{x} \in A)$$

and

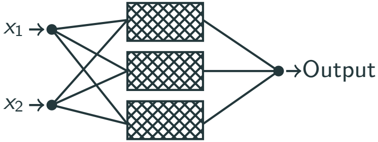
$$\mathbf{P_X}\left(\left(\bigcup_{k=1}^{K_2} (P_k)^{\delta_k} \setminus (P_k)_{\delta_k}\right) \cap A\right) \le \epsilon$$

A corresponding neural network regression estimator

Let $\mathcal{F}^{(sparse)}_{M^*,L,r,\alpha}$ be the class of stacked neural networks, i.e., functions of the form

$$f(\mathbf{x}) = \sum_{i=1}^{M^*} \mu_i \cdot f_i(\mathbf{x}) \quad (\mathbf{x} \in \mathbb{R}^d)$$

with $|\mu_i| \leq \alpha$ and $f_i \in \mathcal{F}(L, r, \alpha)$.



A corresponding neural network regression estimator

Stacked neural network estimator:

$$\tilde{m}_n \in \arg \min_{f \in \mathcal{F}^{(sparse)}_{M^*, L_n, r_n, \alpha_n}} \frac{1}{n} \sum_{i=1}^{n} |Y_i - f(\mathbf{X}_i)|^2$$

Choose Parameter M^* with the splitting of the sample procedure

- learning sample of size $n_l = \lceil n/2 \rceil$
- test sample of size $n_t = n - n_l = \lfloor n/2 \rfloor$
- $M^* \in \mathcal{P}_n = \{2^l : l \in \{1, \ldots, \lceil \log(n) \rceil\}\}$

Truncated estimator: $m_n(\mathbf{x}) = T_{\beta_n} \tilde{m}_n(\mathbf{x}) \quad (\mathbf{x} \in \mathbb{R}^d)$

Rate of convergence of the estimator

Assumptions

- Regression function m has local dimensionality d^* with order (K_1, K_2), $\mathbf{P}_X$-border $1/n$ and $\delta_{i,k} \geq c_1/n^{c_2}$ for $c_1, c_2 > 0$
- All functions f_k in the definition are bounded and (p, C)-smooth
- $\mathbf{E}(\exp(c_3 \cdot Y^2)) < \infty$ and $\mathrm{supp}(\mathbf{X})$ is bounded

Rate of convergence of the estimator

Theorem: If

- number of hidden layers $L_n \asymp \log(n)$
- number of neurons $r_n = \lceil c_1 \rceil$
- bound on the weights $\alpha_n = c_2 \cdot n^{c_3}$.

Then

$$\mathbf{E} \int |m_n(\mathbf{x}) - m(\mathbf{x})|^2 \mathbf{P}_{\mathbf{X}}(d\mathbf{x}) \leq c_4 \cdot (\log(n))^5 \cdot n^{-\frac{2p}{2p+d^*}}.$$

Remarks

- With stacked neural network estimators we are able to circumvent the curse of dimensionality for regression functions with low local dimensionality
- The rate is optimal up to some logarithmic factor
- The proof is based on a result that analyzes the connection between neural networks and MARS

Deep Learning and MARS: A connection

MARS

- Adaptive procedure for regression estimation based on splines
- Model uses product of piecewise linear functions of the form

$$B_{J,t}(x_1, \ldots, x_d) = \prod_{j \in J} (\pm(x_j - t_j))_+$$

- MARS (Multivariate Adaptive Regression Splines) fits linear combination of such functions to data
- Adaptive construction of the functions B_k by forward/backward selection
  $\rightsquigarrow$ Greedy algorithm

MARS

- As soon as a subbasis $B_1, \ldots, B_K$ is chosen, the principle of least squares is used to construct an estimator

$$m_n(\mathbf{x}) = \sum_{k=1}^{K} \hat{a}_k \cdot B_k(\mathbf{x}),$$

where

$$(\hat{a}_k)_{k=1,\ldots,K} = \arg \min_{(a_k)_{k=1,\ldots,K} \in \mathbb{R}^K} \frac{1}{n} \sum_{i=1}^{n} \left| Y_i - \sum_{k=1}^{K} a_k \cdot B_k(\mathbf{X}_i) \right|^2 .$$

MARS

- If we have an oracle which produces the optimal subset of basis functions, the expected L_2-error of the estimator would satisfy

$$\inf_{K\in\mathbb{N}, B_1,\dots,B_K\in\mathcal{B}} \left(\frac{K}{n} + \min_{(a_k)_{k\in\{1,\dots,K\}}} \int \left| \sum_{k=1}^{K} a_k \cdot B_k(\mathbf{x}) - m(\mathbf{x}) \right|^2 \mathbf{P}_{\mathbf{X}}(dx) \right)$$

$\hookrightarrow$ Does not hold for MARS, as there is no guarantee that the optimal basis can be found with a hierarchical forward/backward stepwise subset selection procedure

Deep Learning and MARS: A connection

Theorem: If

- number of hidden layers $L_n \asymp \log(n)$
- number of neurons $r_n = 2d + 38$
- bound on the weights $\alpha_n = c_1 \cdot n^{c_2}$
- learning sample size $n_l = \lceil n/2 \rceil$

we have for $n > 7$

$$\mathbf{E} \int |m_n(\mathbf{x}) - m(\mathbf{x})|^2 \mathbf{P_X}(d\mathbf{x}) \leq (\log(n))^5 \cdot \inf_{I \in \mathbb{N},\ B_1,\dots,B_I \in \mathcal{B}} \left(c_3 \cdot \frac{I}{n} + \min_{(a_i)_{i \in \{1,\dots,I\}} \in [-c_4 \cdot n, c_4 \cdot n]^I} \int |\sum_{i=1}^{I} a_i \cdot B_i(\mathbf{x}) - m(\mathbf{x})|^2 \mathbf{P_X}(d\mathbf{x}) \right).$$

Intrinsic dimensionality

- Results mainly focus on the structure of the underlying regression function
- Less results explore the geometric properties of the data

  Are estimators based on networks able to exploit the structure of the input data?

- Assumption: $\mathbf{X}$ is concentrated on some d^*-dimensional Lipschitz-manifold

d^*-dimensional Lipschitz-manifold

Formal definition: Let $\mathcal{M} \subseteq \mathbb{R}^d$ be compact and let $d^* \in \{1, \ldots, d\}$.

a) We say that $U_1, \ldots, U_r$ is an *open covering of* $\mathcal{M}$, if $U_1, \ldots, U_r \subset \mathbb{R}^d$ are open (with respect to the Euclidean topology on $\mathbb{R}^d$) and satisfy

$$\mathcal{M} \subseteq \bigcup_{l=1}^{r} U_l.$$

d^*-dimensional Lipschitz-manifold

Formal definition: Let $\mathcal{M} \subseteq \mathbb{R}^d$ be compact and let $d^* \in \{1, \ldots, d\}$.

a) We say that $U_1, \ldots, U_r$ is an *open covering of* $\mathcal{M}$, if $U_1, \ldots, U_r \subset \mathbb{R}^d$ are open (with respect to the Euclidean topology on $\mathbb{R}^d$) and satisfy

$$\mathcal{M} \subseteq \bigcup_{l=1}^{r} U_l.$$

b) We say that

$$\psi_1, \ldots, \psi_r : [0,1]^{d^*} \to \mathbb{R}^d$$

are *bi-Lipschitz functions*, if there exists $0 < C_{\psi,1} \leq C_{\psi,2} < \infty$ such that

$$C_{\psi,1} \cdot \|\mathbf{x}_1 - \mathbf{x}_2\| \leq \|\psi_l(\mathbf{x}_1) - \psi_l(\mathbf{x}_2)\| \leq C_{\psi,2} \cdot \|\mathbf{x}_1 - \mathbf{x}_2\| \quad (1)$$

holds for any $\mathbf{x}_1, \mathbf{x}_2 \in [0,1]^{d^*}$ and any $l \in \{1, \ldots, r\}$.

d^*-dimensional Lipschitz-manifold

c) We say that $\mathcal{M}$ is a *d^*-dimensional Lipschitz-manifold* if there exist bi-Lipschitz functions $\psi_i : [0,1]^{d^*} \to \mathbb{R}^d$ $(i \in \{1,\dots,r\})$, and an open covering $U_1,\dots,U_r$ of $\mathcal{M}$ such that

$$\psi_l((0,1)^{d^*}) = \mathcal{M} \cap U_l$$

holds for all $l \in \{1,\dots,r\}$. Here we call $\psi_1,\dots,\psi_r$ the *parametrizations* of the manifold.

Main result

Theorem: If

- $\mathbf{X}$ is concentrated on a d^*-dimensional Lipschitz manifold $\mathcal{M}$
- $L_n \asymp \log(n)$
- $r_n \asymp n^{d^*/(2(2p+d^*))}$

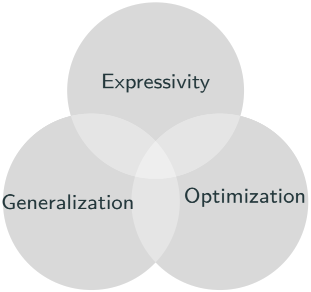
Then

$$\mathbf{E} \int |m_n(\mathbf{x}) - m(\mathbf{x})|^2 \, \mathbf{P}_{\mathbf{X}}(d\mathbf{x}) \leq c_1 \cdot (\log n)^6 \cdot n^{-\frac{2p}{2p+d^*}}.$$

Summary

- Under structural assumptions on the regression function, neural networks are able to circumvent the curse of dimensionality
- Networks are also able to exploit the structure of the input data
- Sparsity is not the answer

What we have learned



Fundamental research topics of Deep Learning

- Approximation properties of DNNs
- Generalization results of DNNs
- **But**: Results did not take into account the optimization, i.e., the training of the networks

$\rightsquigarrow$ Cannot be used to improve estimators in practice

Should it not be the aim of statistical theory to not only understand but also improve estimators in practice?

Barron's result

Define

$$\mathcal{F}_n = \left\{ \sum_{k=1}^{\lceil \sqrt{n} \rceil} \alpha_k \cdot \sigma(\beta_k \cdot \mathbf{x} + \gamma_k) : \alpha_k, \gamma_k \in \mathbb{R}, \beta_k \in \mathbb{R}^d, \sum_{k=0}^{K_n} |\alpha_k| \leq L_n \right\},$$

where $\sigma(u) = 1/(1 + \exp(-u))$ $(u \in \mathbb{R})$ and let

$$m_n(\cdot) = \text{argmin}_{f \in \mathcal{F}_n} \frac{1}{n} \sum_{i=1}^{n} |Y_i - f(\mathbf{X}_i)|^2$$

be the corresponding least squares estimator.

Barron's result

Define

$$\mathcal{F}_n = \left\{ \sum_{k=1}^{\lceil \sqrt{n} \rceil} \alpha_k \cdot \sigma(\beta_k \cdot \mathbf{x} + \gamma_k) : \alpha_k, \gamma_k \in \mathbb{R}, \beta_k \in \mathbb{R}^d, \sum_{k=0}^{K_n} |\alpha_k| \leq L_n \right\},$$

where $\sigma(u) = 1/(1 + \exp(-u))$ $(u \in \mathbb{R})$ and let

$$m_n(\cdot) = \operatorname{argmin}_{f \in \mathcal{F}_n} \frac{1}{n} \sum_{i=1}^{n} |Y_i - f(\mathbf{X}_i)|^2$$

be the corresponding least squares estimator. Then

$$\mathbf{E} \int |m_n(\mathbf{x}) - m(\mathbf{x})|^2 \, \mathbf{P}_{\mathbf{X}}(d\mathbf{x}) \leq c_1 \cdot (\log n)^5 \cdot \frac{1}{\sqrt{n}}$$

holds whenever the Fourier transform of the regression function has a finite first moment.

An estimator learned by gradient descent

We study the rate of convergence of a neural network estimators learned by gradient descent

An estimator learned by gradient descent

We study the rate of convergence of a neural network estimators learned by gradient descent

We need the following definitions:

$$\sigma(u) = 1/(1 + \exp(-u)) \quad (u \in \mathbb{R}),$$

An estimator learned by gradient descent

We study the rate of convergence of a neural network estimators learned by gradient descent

We need the following definitions:

$$\sigma(u) = 1/(1 + \exp(-u)) \quad (u \in \mathbb{R}),$$

$$f_{net,\mathbf{w}}(\mathbf{x}) = \alpha_0 + \sum_{j=1}^{K_n} \alpha_j \cdot \sigma(\beta_j^T \cdot \mathbf{x} + \gamma_j)$$

where

$$\mathbf{w} = (\alpha_0, \alpha_1, \ldots, \alpha_{K_n}, \beta_1, \ldots, \beta_{K_n}, \gamma_1, \ldots, \gamma_{K_n}),$$

An estimator learned by gradient descent

We study the rate of convergence of a neural network estimators learned by gradient descent

We need the following definitions:

$$\sigma(u) = 1/(1 + \exp(-u)) \quad (u \in \mathbb{R}),$$

$$f_{net,\mathbf{w}}(\mathbf{x}) = \alpha_0 + \sum_{j=1}^{K_n} \alpha_j \cdot \sigma(\beta_j^T \cdot \mathbf{x} + \gamma_j)$$

where

$$\mathbf{w} = (\alpha_0, \alpha_1, \ldots, \alpha_{K_n}, \beta_1, \ldots, \beta_{K_n}, \gamma_1, \ldots, \gamma_{K_n}),$$

and

$$F(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} |Y_i - f_{net,\mathbf{w}}(\mathbf{X}_i)|^2 + \frac{c_2}{K_n} \cdot \sum_{k=0}^{K_n} \alpha_k^2.$$

An estimator learned by gradient descent

- Initial weights:

$$\mathbf{w}(0) = (\alpha_0(0), \dots, \alpha_{K_n}(0), \beta_1(0), \dots, \beta_{K_n}(0), \gamma_1(0), \dots, \gamma_{K_n}(0))$$

such that

$$\alpha_0(0) = \alpha_1(0) = \cdots = \alpha_{K_n}(0) = 0$$

An estimator learned by gradient descent

- Initial weights:

$$\mathbf{w}(0) = (\alpha_0(0), \dots, \alpha_{K_n}(0), \beta_1(0), \dots, \beta_{K_n}(0), \gamma_1(0), \dots, \gamma_{K_n}(0))$$

such that

$$\alpha_0(0) = \alpha_1(0) = \cdots = \alpha_{K_n}(0) = 0$$

and $\beta_1(0), \dots, \beta_{K_n}(0), \gamma_1(0), \dots, \gamma_{K_n}(0)$ independently randomly chosen such that

  - $\beta_k(0)$ are uniformly distributed on a sphere with radius B_N
  - $\gamma_j(0)$ are uniformly distributed on $[-B_n \cdot \sqrt{d}, B_n \cdot \sqrt{d}]$.

An estimator learned by gradient descent

- Initial weights:

$$\mathbf{w}(0) = (\alpha_0(0), \dots, \alpha_{K_n}(0), \beta_1(0), \dots, \beta_{K_n}(0), \gamma_1(0), \dots, \gamma_{K_n}(0))$$

  such that

$$\alpha_0(0) = \alpha_1(0) = \cdots = \alpha_{K_n}(0) = 0$$

  and $\beta_1(0), \dots, \beta_{K_n}(0), \gamma_1(0), \dots, \gamma_{K_n}(0)$ independently randomly chosen such that
  - $\beta_k(0)$ are uniformly distributed on a sphere with radius B_N
  - $\gamma_j(0)$ are uniformly distributed on $[-B_n \cdot \sqrt{d}, B_n \cdot \sqrt{d}]$.
- t_n gradient descent steps:

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \lambda_n \cdot \nabla_{\mathbf{w}} F(\mathbf{w}(t)) \quad (t = 0, \dots, t_n - 1).$$

An estimator learned by gradient descent

- The estimator:

$$\tilde{m}_n(\cdot) = f_{net,\mathbf{w}(t_n)}(\cdot) \quad \text{and} \quad m_n(\mathbf{x}) = T_{c_1 \cdot \log n} \tilde{m}_n(\mathbf{x})$$

where $T_L z = \max\{\min\{z, L\}, -L\}$ for $z \in \mathbb{R}$ and $L \geq 0$.

An estimator learned by gradient descent

- The estimator:

$$\tilde{m}_n(\cdot) = f_{net,\mathbf{w}(t_n)}(\cdot) \quad \text{and} \quad m_n(\mathbf{x}) = T_{c_1 \cdot \log n} \tilde{m}_n(\mathbf{x})$$

where $T_L z = \max\{\min\{z, L\}, -L\}$ for $z \in \mathbb{R}$ and $L \geq 0$.

- Main assumption: Fourier transform

$$\mathcal{F}m(\omega) = \frac{1}{(2\pi)^{d/2}} \cdot \int_{\mathbb{R}^d} e^{-i \cdot \omega^T x} \cdot m(x)\, dx$$

of the regression function satisfies

$$|\mathcal{F}m(\omega)| \leq \frac{c_2}{\|\omega\|^{d+1+\epsilon}} \quad (\omega \in \mathbb{R}^d \setminus \{0\}) \tag{2}$$

for some $\epsilon \in (0, 1]$ and some $c_2 > 0$.

An estimator learned by gradient descent

Theorem: If

- Fourier transform $\mathcal{F}m$ satisfies (2)
- number of neurons $K_n \approx \sqrt{n}$
- $B_n \approx n^{5/2}$
- learning rate $\lambda_n \approx n^{-1.25}$
- gradient descent steps $t_n \approx n^{1.75}$

Then

$$\mathbf{E} \int |m_n(x) - m(x)|^2 \mathbf{P}_X(dx) \leq c_2 \cdot (\log n)^4 \cdot \frac{1}{\sqrt{n}}.$$

On the proof

Set $\tilde{K}_n = \lceil K_n/(\log n)^4 \rceil$. **In the proof** we show that with high probability

$$\mathbf{w}(0) = (\alpha_0(0), \dots, \alpha_{K_n}(0), \beta_1(0), \dots, \beta_{K_n}(0), \gamma_1(0), \dots, \gamma_{K_n}(0))$$

is chosen such that

$$\int \left| \sum_{k=1}^{\tilde{K}_n} \bar{\alpha}_{i_k} \cdot \sigma(\beta_{i_k}(0)^T \cdot \mathbf{x} + \gamma_{i_k}(0)) - m(\mathbf{x}) \right|^2 \mathbf{P_X}(d\mathbf{x})$$

is small for some (random) $1 \leq i_1 < \cdots < i_{\tilde{K}_n}$ and some (random) $\bar{\alpha}_{i_1}, \dots, \bar{\alpha}_{i_{\tilde{K}_n}} \in \mathbb{R}$,

On the proof

Set $\tilde{K}_n = \lceil K_n/(\log n)^4 \rceil$. **In the proof** we show that with high probability

$$\mathbf{w}(0) = (\alpha_0(0), \dots, \alpha_{K_n}(0), \beta_1(0), \dots, \beta_{K_n}(0), \gamma_1(0), \dots, \gamma_{K_n}(0))$$

is chosen such that

$$\int \left| \sum_{k=1}^{\tilde{K}_n} \bar{\alpha}_{i_k} \cdot \sigma(\beta_{i_k}(0)^T \cdot \mathbf{x} + \gamma_{i_k}(0)) - m(\mathbf{x}) \right|^2 \mathbf{P}_{\mathbf{X}}(d\mathbf{x})$$

is small for some (random) $1 \leq i_1 < \dots < i_{\tilde{K}_n}$ and some (random) $\bar{\alpha}_{i_1}, \dots, \bar{\alpha}_{i_{\tilde{K}_n}} \in \mathbb{R}$, and that during the gradient descent the inner weights

$$\beta_{i_1}(0), \gamma_{i_1}(0), \dots, \beta_{i_{\tilde{K}_n}}(0), \gamma_{i_{\tilde{K}_n}}(0)$$

change only slightly.

A lower bound

Under the above assumption a much better rate of convergence than $1/\sqrt{n}$ is not possible:

A lower bound

Under the above assumption a much better rate of convergence than $1/\sqrt{n}$ is not possible:

Theorem: Let $\mathcal{D}$ be the class of all distributions of $(\mathbf{X}, Y)$ which satisfy the assumptions of the above Theorem. Then

$$\inf_{\hat{m}_n} \sup_{(X,Y)\in\mathcal{D}} \mathbf{E} \int |\hat{m}_n(\mathbf{x}) - m(\mathbf{x})|^2 \mathbf{P}_{\mathbf{X}}(d\mathbf{x}) \geq c_1 \cdot n^{-\frac{1}{2}-\frac{1}{d+1}},$$

where the infimum is taken with respect to all estimates $\hat{m}_n$, i.e., all measurable functions of the data.

A simplified estimator

Insights in our statistical analysis help us simplify our estimate as follows:

Choose

- $\beta_1, \ldots, \beta_{K_n}, \gamma_1, \ldots, \gamma_{K_n}$ i.i.d.
- $\beta_1, \ldots, \beta_{K_n}$ uniformly distributed on $\{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\| = B_n\}$
- $\gamma_1, \ldots, \gamma_{K_n}$ uniformly distributed on $[-B_n \cdot \sqrt{d}, B_n \cdot \sqrt{d}]$

Denote the linear function space by

$$\mathcal{F}_n = \left\{ f : \mathbb{R}^d \to \mathbb{R} : f(\mathbf{x}) = \alpha_0 + \sum_{j=1}^{K_n} \alpha_j \cdot \sigma\left(\beta_j^T \cdot \mathbf{x} + \gamma_j\right) \right.$$

$$\left. \text{for some } \alpha_0, \ldots, \alpha_{K_n} \in \mathbb{R} \right\}$$

A simplified estimator

Choose the estimate according to the principle of least squares

$$\tilde{m}_n = \arg\min_{f \in \mathcal{F}_n} \frac{1}{n} \sum_{i=1}^{n} |Y_i - f(\mathbf{X}_i)|^2.$$

A simplified estimator

Choose the estimate according to the principle of least squares

$$\tilde{m}_n = \arg\min_{f \in \mathcal{F}_n} \frac{1}{n} \sum_{i=1}^{n} |Y_i - f(\mathbf{X}_i)|^2.$$

Truncate it on some level $\beta_n = c_1 \cdot \log n$

$$m_n = T_{\beta_n} \tilde{m}_n,$$

where $T_L z = \max\{\min\{z, L\}, -L\}$ for $z \in \mathbb{R}$ and $L \geq 0$.

A simplified estimator

Theorem: If

- the Fourier transform $\mathcal{F}m$ satisfies (2)
- number of summands $K_n \approx \sqrt{n}$
- $B_n = \frac{1}{\sqrt{d}} \cdot (\log n)^2 \cdot K_n \cdot n^2$.

Then

$$\mathbf{E} \int |m_n(\mathbf{x}) - m(\mathbf{x})|^2 \mathbf{P}_{\mathbf{X}}(d\mathbf{x}) \leq c_1 \cdot (\log n)^4 \cdot \frac{1}{\sqrt{n}}.$$

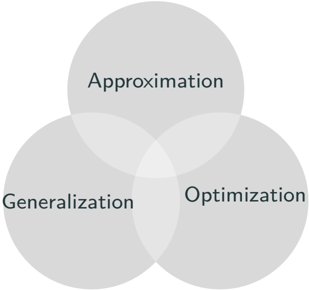
A simplified estimator

- Same rate as for the neural network estimate learned by gradient descent, **but** much faster in computation
- Ability to learn a good hierarchical representation of the data is considered as a key factor of Deep Learning
  $\rightsquigarrow$ So-called representation learning (see Goodfellow et al. (2016))
  Suprisingly: In our estimate it is much more a representation **guessing**

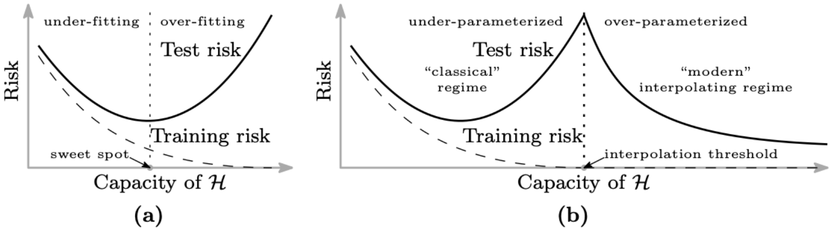
Summary

- In the analysis all three aspects of Deep Learning, namely approximation, generalization and optimization, were considered simultaneously
- Statistical insights helped us to construct a simplified estimate, which can be much faster computed in applications

$\rightsquigarrow$ Much faster in applications



Three competing aspects – or maybe not?



$\rightsquigarrow$ Not covered by classical statistical learning theory

Why do **overparametrized** networks learn?

Photos everywhere

Grzegorz Czapski/Alamy

https://www.businessinsider.com/most-surveilled-cities-in-the-world-china-london-atlanta-2019-8

Videos on Youtube

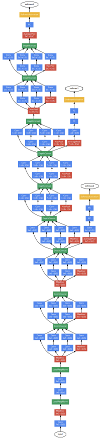
https://everysecond.io/youtube

Every second of clock time $>$ ***8 hours*** *of videos are uploaded on Youtube* $\Leftrightarrow$ 720.000 ***hours*** *(*$\approx$ 82.2 *years) of videos every day*

Deep Learning in image classification

Enable machines to view the world as humans do

- Majority of bits flying around the internet are visual data
- Human beings have no chance to filter/understand/watch this
- Important: Find algorithms that utilize and understand this data
- Deep convolutional neural networks (CNNs) have achieved a huge breakthrough in image recognition
  - Facebook's photo tagging
  - Self-driving cars
  - ...
- Famous networks based on CNNs: LeNet, AlexNet, GoogLeNet, ...



A challenging image for computers to recognize

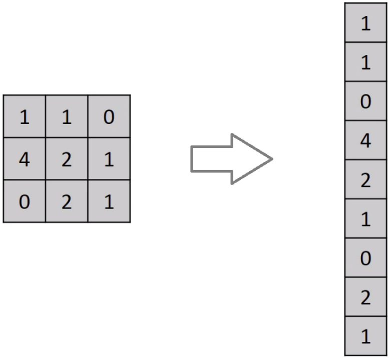
Source: Mumford (1996)

Why CNNs over feedforward networks?

- Image ⇔ Matrix of pixels
- Why not just flatten the image and feed it into a feedforward network?

↪ Not able to capture spatial and temporal dependencies

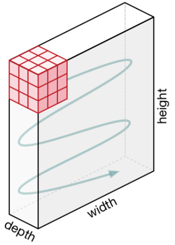
↪ Solution: Application of filters/convolutional layers to detect features, reduce parameters and reuse the weight matrix



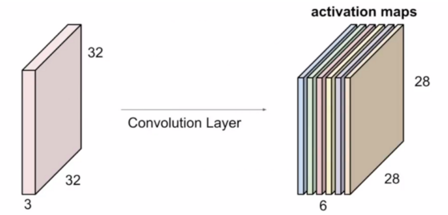
https://rubikscode.net/2018/02/26/introduction-to-convolutional-neural-networks/

Convolutional layer

- **Convolution**: Slide over the image spatially, computing dot products
- Objective: Extract high-level features
- Each convolutional layer contains a series of filters
- Finally an activation function is applied to these filters



Source:https://towardsdatascience.com/an-introduction-to-convolutional-neural-networks-eb0b60b58fd7



Source:http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture6.pdf

Convolutional layer

More mathematically:

- Convolutional layer $\ell \in \{1, \ldots, L\}$ consists of $k_\ell \in \mathbb{N}$ feature maps
- Convolution in layer ℓ is performed by using a window of values of layer $\ell - 1$ of size $M_\ell \in \{1, \ldots, d\}$
- Each neuron of a feature map is connected to a region of neighboring neurons in the previous layer

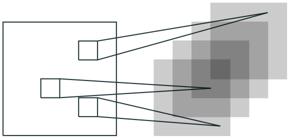


Illustration of a convolutional layer

The s-th feature map ($s \in \{1, \ldots, k_\ell\}$) of the ℓ-th hidden layer ($\ell \in \{1, \ldots, L\}$) can be described by

$$\mathbf{o}_s^\ell = \sigma(\mathbf{w}_s^\ell \star \mathbf{o}_s^{\ell-1}) \quad \text{with} \quad \mathbf{o}_s^0 = \mathbf{x}.$$

Pooling layers

- Here: Only in the last step a max-pooling layer is applied

$$f_{\mathbf{w}}(\mathbf{x}) = (|\mathbf{o}_1^L|_\infty, \ldots, |\mathbf{o}_{k_L}^L|_\infty).$$

$\hookrightarrow$ class of convolutional neural network is defined by $\mathcal{F}^{CNN}_{\sigma,L,\mathbf{k},\mathbf{M}}$

Convolutional neural networks (CNNs)

Final network class:

Combination of convolutional and fully-connected network:

$$\mathcal{F}_n = \left\{ g \circ f : f \in \mathcal{F}^{CNN}_{\sigma, L^{(1)}, \mathbf{k}^{(1)}, \mathbf{M}}, g \in \mathcal{F}_\sigma(L^{(2)}, \mathbf{k}^{(2)}), \right\}$$

with parameters

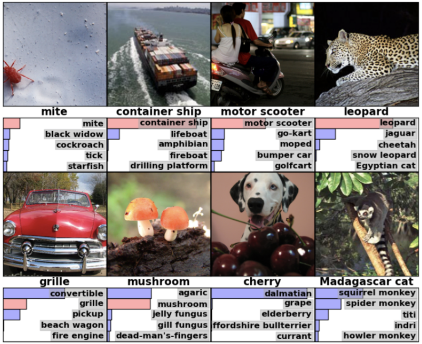
$$\mathbf{L} = (L^{(1)}, L^{(2)}),\ \mathbf{k}^{(1)} = \left(k_1^{(1)}, \ldots, k_{L^{(1)}}^{(1)}\right),$$

$$\mathbf{k}^{(2)} = \left(k_1^{(2)}, \ldots, k_{L^{(2)}}^{(2)}\right),\ \mathbf{M} = (M_1, \ldots, M_{L^{(1)}})$$

.

Convolutional neural networks in image classification

Why is Deep Learning so successful in image classification?



Source: Krizhevsky et al. (2012)

Image classification

- Task of categorizing images into one of several predefined classes
- Let

$$\mathcal{D}_n = \{(\mathbf{X}, Y), (\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)\}$$

  i.i.d. with values in $[0,1]^{d \times d} \times \{-1, 1\}$
- $\mathbf{X}$ is image from class Y, which contains at position (i,j) the grey scale value of the pixel of the image at the corresponding position
- Aim: Predict Y given $\mathbf{X}$
- Classifier: Function $f : [0,1]^{d \times d} \to \mathbb{R}$, where we predict $+1$ for $f(\mathbf{x}) \geq 0$ and -1 when $f(\mathbf{x}) < 0$
- $\mathbf{P}$ is distribution of $(\mathbf{X}, Y)$ and

$$\eta(\mathbf{x}) = \mathbf{P}\{Y = 1 | \mathbf{X} = \mathbf{x}\} \quad (\mathbf{x} \in [0,1]^{d \times d})$$

  the so-called aposteriori probability

Image classification

- Prediction error: $\mathbf{P}(Yf(\mathbf{X}) \leq 0)$
- Bayes' rule

$$f^*(\mathbf{x}) = \begin{cases} 1, & \text{if } \eta(\mathbf{x}) > \frac{1}{2} \\ -1, & \text{elsewhere} \end{cases}$$

  minimizes the prediction error
- But: Distribution of $(\mathbf{X}, Y)$ is unknown
- Estimate a classifier $\hat{C}_n$ such that its misclassification risk

$$\mathbf{P}\{\hat{C}_n(\mathbf{X}) \neq Y | \mathcal{D}_n\}$$

  is *small*

The CNN-classifier

- Let

$$\mathcal{F}_n = \left\{ g \circ f : f \in \mathcal{F}^{CNN}_{L_n^{(1)}, r^{(1)}, \mathbf{M}}, g \in \mathcal{F}(L_n^{(2)}, r^{(2)}), \|g \circ f\|_\infty \leq \beta_n \right\}$$

- Use $\hat{C}_n(\mathbf{x}) = sgn(\hat{f}_n(\mathbf{x}))$ with

$$\hat{f}_n = \arg\min_{f \in \mathcal{F}_n} \frac{1}{n} \sum_{i=1}^{n} \log(1 + \exp(-Y_i \cdot f(\mathbf{X}_i)))$$

  as classifier

- Analyze its performance by

$$\mathbf{E}\left\{ \mathbf{P}\{\hat{C}_n(\mathbf{X}) \neq Y | \mathcal{D}_n\} - \min_{f : [0,1]^{d \times d} \to \{-1,1\}} \mathbf{P}\{f(\mathbf{X}) \neq Y\} \right\}$$

$$= \mathbf{P}\{\hat{C}_n(\mathbf{X}) \neq Y\} - \mathbf{P}\{f^*(\mathbf{X}) \neq Y\}$$

Assumption on the aposteriori probability

- For nontrivial results: Restrict the class of distributions
- Here: Assume that $\eta(\mathbf{x}) = \mathbf{P}\{Y = 1|\mathbf{X} = \mathbf{x}\}$ satisfies a (p, C)-smooth hierarchical max-pooling model
- Based on the following observation:
  - Human beings decide if an object is on an image by scanning subparts of the image
  - For each subpart human estimates a probability, that the searched object is on it
  - Probability that the object is on the image $\Leftrightarrow$ Maximum of probabilities for each subpart of the image

  $\hookrightarrow$ Max-pooling model
  - Probability that a subpart contains object $\Leftrightarrow$ Parts of the object are identifiable

  $\hookrightarrow$ Hierarchical structure

Main result

Theorem: If

- η satisfies a (p, C)-smooth hierarchical max-pooling model of level l
- number of hidden layers $L_n^{(1)} \asymp n^{2/(2p+4)}$ and $L_n^{(2)} \asymp n^{1/4}$
- size of the filters $M_s = 2^{\pi(s)}$ with $\pi(s) = \sum_{i=1}^{l} \mathbf{1}_{\{s \geq i + \sum_{r=l-i+1}^{l-1} 4^r \cdot \lceil c_1 \cdot n^{2p/(2p+4)} \rceil\}}$
- number of neurons/feature maps is constant.

We have

$$\mathbf{P}\{Y \neq \hat{C}_n(\mathbf{X})\} - \mathbf{P}\{Y \neq f^*(\mathbf{X})\} \leq c_2 \cdot (\log n) \cdot n^{-\min\{p/(4p+8), 1/8\}}.$$

Main result

Theorem: If, in addition,

$$\mathbf{P}\left\{\mathbf{X} : \left|\log \frac{\eta(\mathbf{X})}{1-\eta(\mathbf{X})}\right| > \frac{1}{2} \cdot \log n\right\} \geq 1 - \frac{1}{\sqrt{n}}$$

holds, the rate improves to

$$\mathbf{P}\{Y \neq \hat{C}_n(\mathbf{X})\} - \mathbf{P}\{Y \neq f^*(\mathbf{X})\} \leq c_3 \cdot (\log n)^2 \cdot n^{-\min\{p/(2p+4),1/4\}}.$$

Remarks

- The rates does not depend on the input dimension d of the image and CNNs are able to circumvent the curse of dimensionality under proper assumptions on the aposteriori probabilities
- The second assumption requires that with high probability the aposteriori probability is very close to zero or one
  $\hookrightarrow$ Realistic as human beings have often not much doubt about the class of objects

Another perspective on image classification

In our setting: Each pixel is considered as a variable and we learn a d-dimensional function $\rightsquigarrow$ Problem is considerably harder if d increases

Another perspective: View image as a two-dimensional object
$\rightsquigarrow$ Increasing the number of pixels leads to higher image resolution and therefore a better performance

Another perspective on image classification

In our setting: Each pixel is considered as a variable and we learn a d-dimensional function $\rightsquigarrow$ Problem is considerably harder if d increases

Another perspective: View image as a two-dimensional object
$\rightsquigarrow$ Increasing the number of pixels leads to higher image resolution and therefore a better performance
$\rightsquigarrow$ Stay tuned: New article to follow shortly (joint work with Johannes Schmidt-Hieber)

Outlook

Many open problems remain...

- Multi-class classification
- Properties of energy landscapes $\hookrightarrow$ Relation between local and global minima, saddlepoints...
- Complex network structures: CNNs, RNNs,...
- Analysis of approximation, generalization and optimization, simultaneously for all kind of network structures

Thank you for your attention!

Some references

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press, Cambridge, Massachusetts.

Bartlett, P., Montanari, A., and Rakhlin, A. (2021). Deep learning: A statistical viewpoint. *Acta Numerica* **30**, pp. 87-201.

Kohler, M., and Langer, S. (2021). On the rate of convergence of fully connected very deep neural network regression estimates using ReLU activation functions. *Annals of Statistics* **49**, pp. 2231-2249 .

Kohler, M., Krzyżak, A., and Langer, S. (2022). Estimation of a function of low local dimensionality by deep neural networks. To appear in *IEEE Transactions on Information Theory*.

Braun, A., Kohler, M., Langer, S. and Walk, H. (2021). The Smoking Gun: Statistical Theory Improves Neural Network Estimates, *arXiv:2107.09550*