Overcoming the curse of dimensionality with deep neural networks

Sophie Langer

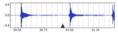
Paris, 03 October 2022

Based on joint work with Alina Braun, Adam Krzyżak, Michael Kohler, Harro Walk

UNIVERSITY
OF TWENTE.

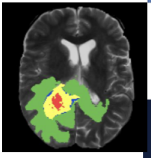
Deep learning is amazing...

Silent video

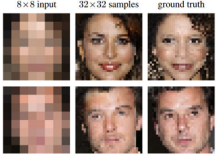


Predicted soundtrack

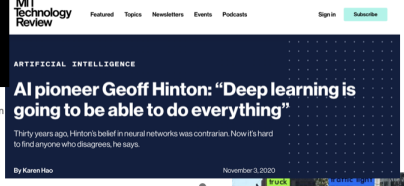
https://news.mit.edu/2016/artificial-intelligence-produces-realistic-sounds-0613



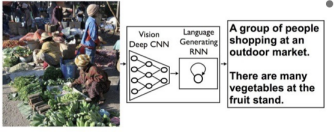
https://paperswithcode.com/task/brain-tumor-segmentation



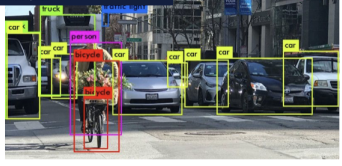
https://arxiv.org/pdf/1702.00783.pdf?xtor=AL-32280680



Caption Generation



http://arxiv.org/abs/1411.4555 "Show and Tell: A Neural Image Caption Generator"



https://azati.ai/image-detection-recognition-and-classification-with-machine-learning/

...and sometimes pretty easy to fool

Granny Smith	85.6%
iPod	0.4%
library	0.0%
pizza	0.0%
toaster	0.0%
dough	0.1%

Granny Smith	0.1%
iPod	99.7%
library	0.0%
pizza	0.0%
toaster	0.0%
dough	0.0%

https://openai.com/blog/multimodal-neurons/

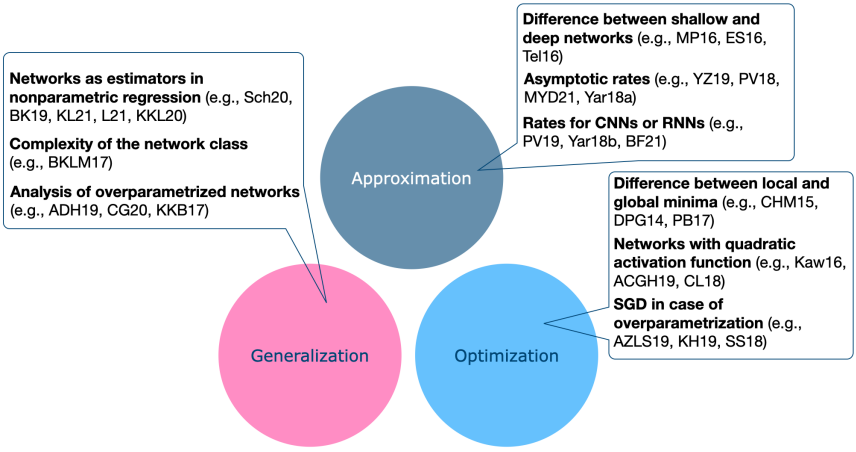
Theory, but why?

Computers are now able to learn on their own, but why?

- Lack of explainability and transparency leads to an unreliable method
- Decisions potentially made by deep learning methods often result in legal and ethical implications
  - Autonomous driving
  - Diagnostic imaging
- Theoretical analysis could improve methods in practice
- Better structure for existing results

The problem

Explaining the procedure is a highly complex task



Three aspects

How rich is the function class?

How well performs the empirical risk minimizer relative to the true one?

How well performs the optimization algorithm relative to the empirical risk minimizer?

Mathematical problem

$X = \{Images\}$

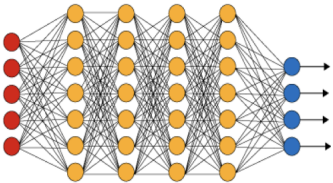
$$\xrightarrow{f : X \to Y} Y = \{Muffin, Chiwawa\}$$

Mathematical problem

$X = \{Images\}$

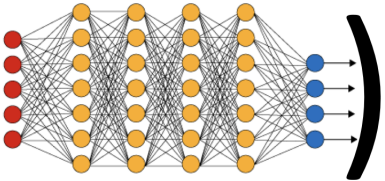
$$\xrightarrow{f : X \to Y} Y = \{Muffin, Chiwawa\}$$

,



Model

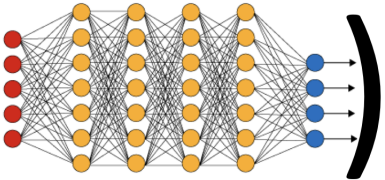
Mathematical problem

$$L($$  $$,$$  $$)$$

X={Images}

Model

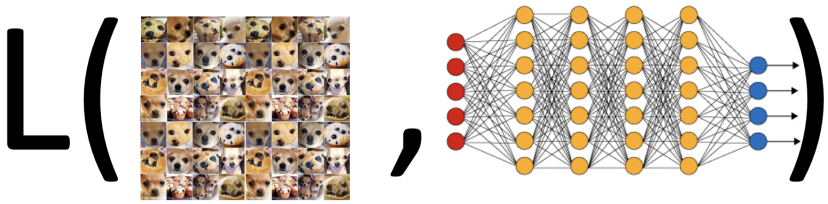
Mathematical problem

L( , )

X={Images} *Model*

In practice, the loss is minimized using (variants of) gradient descent.

Mathematical problem



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

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 $f^*(\mathbf{x}) = m(\mathbf{x}) = \mathbf{E}\{Y|\mathbf{X} = \mathbf{x}\} \rightsquigarrow$ **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*.

Simple feedforward neural networks

Activation function $\sigma : \mathbb{R} \to \mathbb{R}$

- We study the ReLU activation function $\sigma(x) = \max\{x, 0\}$

Network architecture $(L, \mathbf{k})$

- Positive integer L denoting the *number of hidden layers*
- *width vector* $\mathbf{k} = (k_1, \ldots, k_L) \in \mathbb{N}^L$

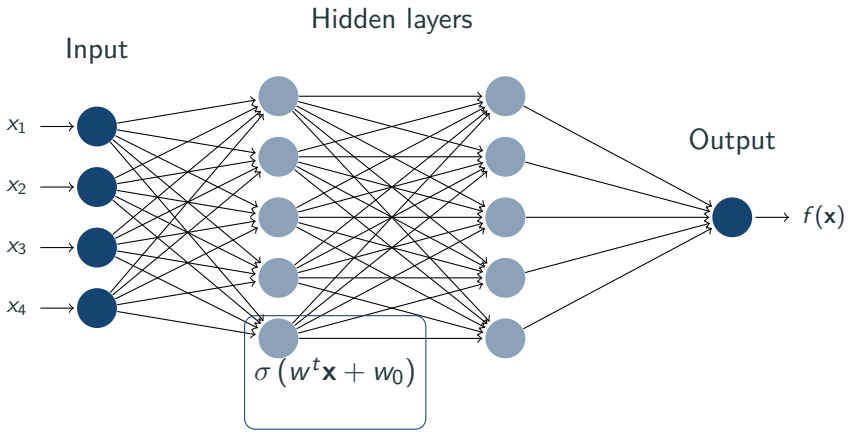
Neural network with network architecture $(L, \mathbf{k})$

$$f : \mathbb{R}^d \to \mathbb{R}, \quad \mathbf{x} \mapsto W_{L+1}\sigma_{\mathbf{v}_L} W_L \sigma_{\mathbf{v}_{L-1}} \cdots W_2 \sigma_{\mathbf{v}_1} W_1 \mathbf{x}$$

Network parameters

- W_i is a $k_i \times k_{i-1}$ matrix
- $\mathbf{v}_i \in \mathbb{R}^{k_i}$

Graphical equivalence



Neural network with network architecture (2,(5,5))

Neural network estimator

Empirical risk minimization

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

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

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

$\rightsquigarrow$ 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))
  $\rightsquigarrow$ suffers from the curse of dimensionality

The choice of the function class

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

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

  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 $g : \mathbb{R} \to \mathbb{R}$ (p, C)-smooth 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 $g : \mathbb{R} \to \mathbb{R}$ (p, C)-smooth 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 $K \in \mathbb{N}$, $g_k : \mathbb{R} \to \mathbb{R}$ (p, C)-smooth and $\mathbf{a}_k \in \mathbb{R}^d$

$\rightsquigarrow$ Optimal rate $n^{-\frac{2p}{2p+1}}$ (Györfi et al. (2002))

The choice of the function class

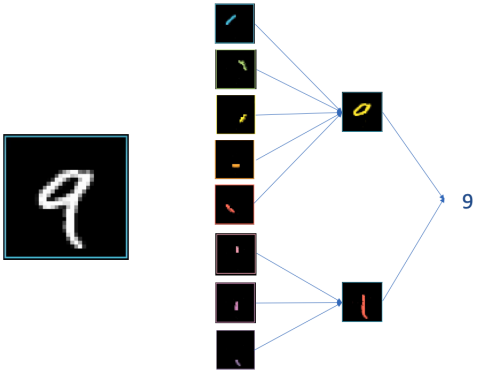
- With all models one can circumvent the curse of dimensionality
- But: Rates can only be obtained in practice if the true (then unknown) regression function corresponds to this structure

$\rightsquigarrow$ 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 $K \in \{1, \dots, d\}$ 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

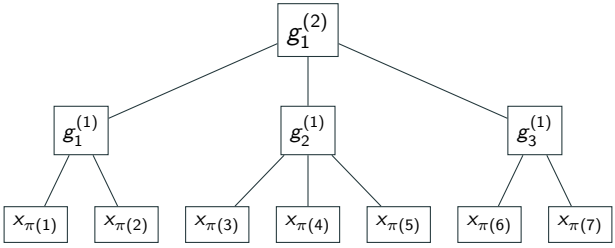


Illustration of a hierarchical composition model of level 2

A more general function class

Example: Additive model

$$m(\mathbf{x}) = \sum_{k=1}^{3} g_k(x_k)$$

If we choose

$$g_1^{(1)}(\mathbf{x}) = g_1(x_1), \quad g_2^{(1)}(\mathbf{x}) = g_2(x_2), \quad g_3^{(1)}(\mathbf{x}) = g_3^{(1)} = g_3(x_3)$$

$$\text{and} \quad g_1^{(2)}(\mathbf{y}) = \sum_{i=1}^{3} y_i$$

$\rightsquigarrow$ Additive models can be written as hierarchical composition models 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 $\operatorname{supp}(\mathbf{X})$ 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

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

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)
```

Other possible assumptions

- Kohler, Langer and Krzyżak(2022) assume regression functions with low local dimensionality d^* and show a rate $n^{-2p/(2p+d^*)}$ for NN estimators
- Barron(1993,1994) assumes regression functions with

$$\int \|\mathbf{w}\|_1 |\mathcal{F}m(\mathbf{w})| d\mathbf{w} < \infty,$$

where

$$\mathcal{F}m(\boldsymbol{\xi}) = \int e^{-i\boldsymbol{\xi}^T \mathbf{x}} m(\mathbf{x}) d\mathbf{x}$$

and shows a rate $1/\sqrt{n}$ for shallow NN estimators

- In case of a mixed smooth Besov space, Suzuki (2018) shows a dimension-free rate for NN estimators

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 $d^* \in \{1, \dots, d\}$.

a) We say that $U_1, \dots, U_r$ is an *open covering of* $\mathcal{M}$, if $U_1, \dots, U_r \subset \mathbb{R}^d$ 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 $d^* \in \{1, \dots, d\}$.

a) We say that $U_1, \dots, U_r$ is an *open covering of* $\mathcal{M}$, if $U_1, \dots, U_r \subset \mathbb{R}^d$ 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

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

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

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

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

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

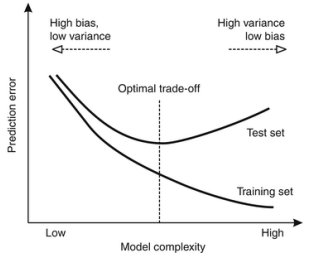
$$\mathbf{E} \int |m_n(\mathbf{x}) - m(\mathbf{x})|^2 \, \mathbf{P}_{\mathbf{X}}(d\mathbf{x}) \lesssim (\log n)^6 \cdot n^{-\frac{2p}{2p+d^*}}.$$

On the proof

Oracle inequality + Bound on the covering number

$$\mathbf{E} \int |m_n(\mathbf{x}) - m(\mathbf{x})|^2 \mathbf{P}_\mathbf{X}(d\mathbf{x}) \lesssim \inf_{f \in \mathcal{F}(L_n, r_n)} \|m - f\|_\infty^2 + \frac{(\log n)^3 \log(L_n \cdot r_n^2) \cdot L_n^2 \cdot r_n^2}{n}$$

- shows the trade-off between approximation power and complexity of the network class



Berrar, Dubitzky, **Overfitting**, 2013

On the approximation error

Due to its compositional structure, functions of the form

$$f = g_K \circ \cdots \circ g_0$$

can be approximated by deep networks

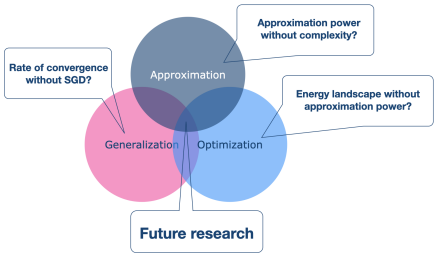
Rough idea:

- Use results of Telgarsky (2016), Yarotsky (2017) to approximate $f(x) = x^2$
- Approximate polynomials with neural networks
- Use that smooth functions can be approximated by Taylor polynomials

Summary

- Neural networks are able to circumvent the curse of dimensionality if
  - structural assumptions on the regression function hold
  - geometric properties on the input components hold
- All these results hold without any sparsity constraint
- At least from a theoretical point of view, sparsity is not the answer for the success of neural networks

The chicken or the egg dilemma



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

$\rightsquigarrow$ Hard to implement a corresponding estimator

For an all-encompassing understanding we need to analyze all three aspects simultaneously!

A promising starting point

- Braun et al. (2022): Analysis of shallow networks learned by gradient descent
  - Good rate of convergence results
  - Improved performance on simulated data

Barron's result

Activation function

$$\sigma(u) = 1/(1 + \exp(-u))$$

Function class

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

Least squares estimator

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

Barron's result

Activation function

$$\sigma(u) = 1/(1+\exp(-u))$$

Function class

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

Least squares estimator

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

If $\int \|\mathbf{w}\|_1 |\mathcal{F}m(\mathbf{w})| d\mathbf{w} < \infty$,

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

An estimator learned by gradient descent

Is that also true for NN estimators trained by gradient descent?

An estimator learned by gradient descent

Is that also true for NN estimators trained by gradient descent?

Shallow neural networks

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

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

Is that also true for NN estimators trained by gradient descent?

Shallow neural networks

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

where

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

Loss function

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

An estimator learned by gradient descent

Initial weights

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

with

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

An estimator learned by gradient descent

Initial weights

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

with

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

and $\beta_1(0), \ldots, \beta_{K_n}(0), \gamma_1(0), \ldots, \gamma_{K_n}(0)$ 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

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

with

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

and $\beta_1(0), \ldots, \beta_{K_n}(0), \gamma_1(0), \ldots, \gamma_{K_n}(0)$ 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

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

An estimator learned by gradient descent

The estimator

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

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

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

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

Main assumption

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

for some $\tilde{c} > 0$.

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

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

On the proof

Set $\tilde{K}_n = \lceil K_n/(\log n)^4 \rceil$. It can be shown that with high probability

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

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

On the proof

Set $\tilde{K}_n = \lceil K_n/(\log n)^4 \rceil$. It can be shown that with high probability

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

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

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

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 Theorem before. Then

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

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

A simplified estimator

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

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

A simplified estimator

The estimator:

$$\tilde{m}_n = \operatorname{argmin}_{f \in \mathcal{F}_n} \frac{1}{n} \sum_{i=1}^{n} |Y_i - f(\mathbf{X}_i)|^2 \quad \text{and} \quad m_n = T_{c \cdot \log n} \tilde{m}_n,$$

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

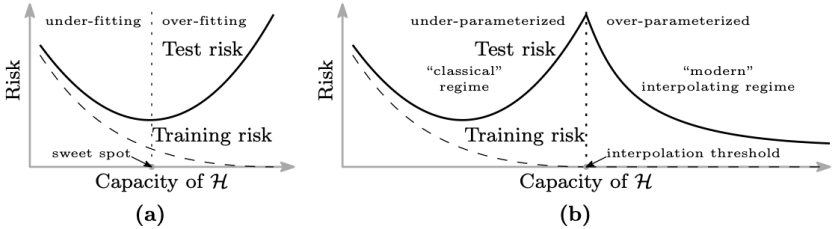
$$\mathbf{E} \int |m_n(\mathbf{x}) - m(\mathbf{x})|^2 \mathbf{P}_{\mathbf{X}}(d\mathbf{x}) \lesssim (\log n)^4 \cdot \frac{1}{\sqrt{n}}.$$

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

Generalization to multiple layers

Three competing aspects – or maybe not?

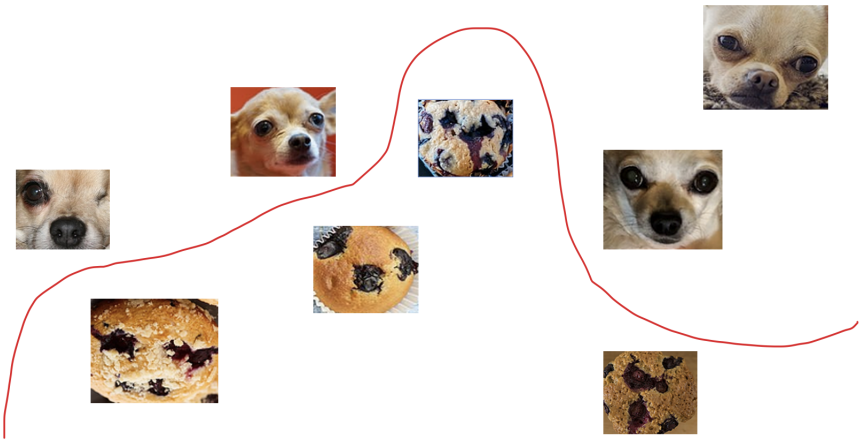


$\rightsquigarrow$ Not covered by classical statistical learning theory

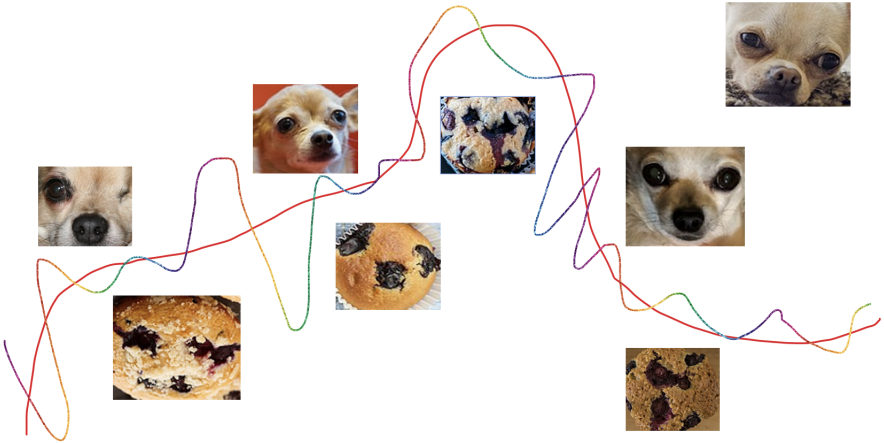
Why do **overparametrized** networks learn?

Overparametrized neural networks can generalize well

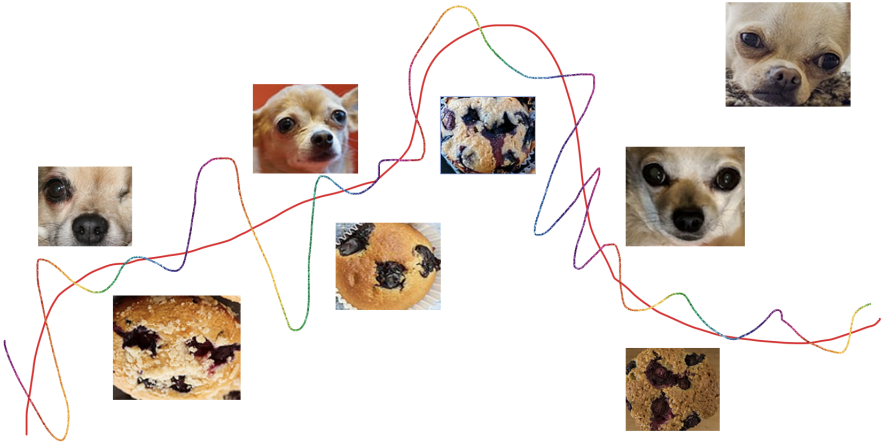
Overparametrized neural networks can generalize well



Overparametrized neural networks can generalize well

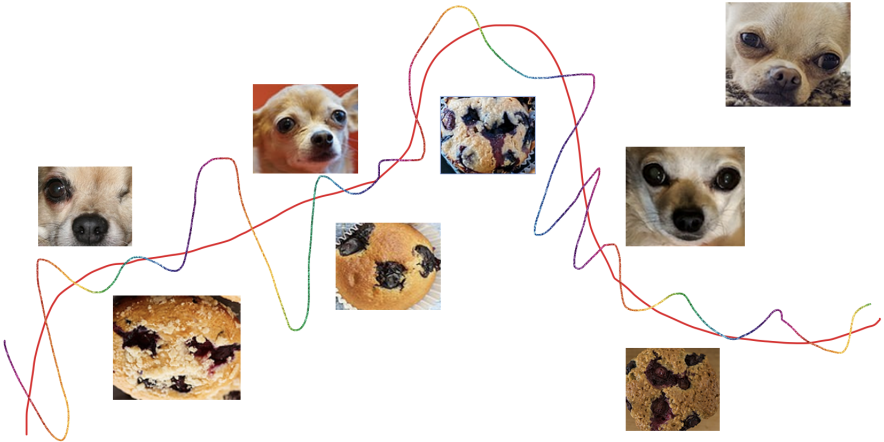


Overparametrized neural networks can generalize well



Small training error $\not\Rightarrow$ small test error

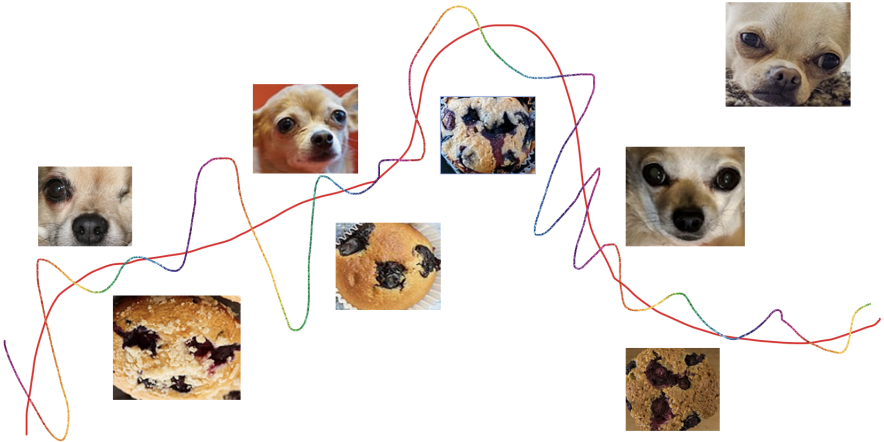
Overparametrized neural networks can generalize well



Small training error $\not\Rightarrow$ small test error

But: Gradient descent algorithms find solutions that generalize well

Overparametrized neural networks can generalize well



Is there an implicit regularization effect?

Outlook

Simultaneous analysis of approximation, generalization and optimization should yield a better deep learning

1. a better understanding of overparametrized neural networks
2. new rate of convergence results for networks trained by (S)GD
3. explainable estimators for practical applications

Thank you for your attention!