On the statistical theory of deep learning

Lecture 1

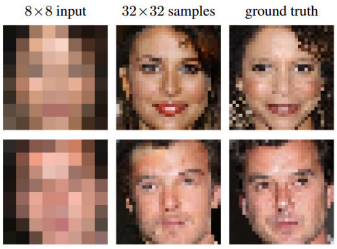
Sophie Langer

Nantes, 19 May 2022

UNIVERSITY
OF TWENTE.

Deep learning - A big hype...?

Google Brain's image super-resolution



Autonomous driving

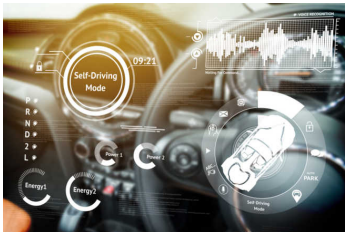
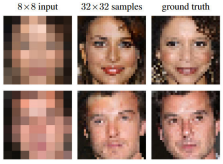


Image and speech recognition • Medical imaging • Web-searches • Weather forecasting
Credit scores • Fraud detection • Language translation
Email/Spam filtering • Cyber defense
...

Deep learning - A big hype...?

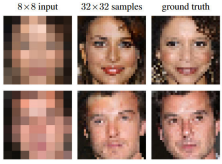


...and sometimes pretty easy to fool

"A young boy is holding a baseball bat"

"Snowpow"

Deep learning - A big hype...?



$\rightsquigarrow$ There is a need of good theory!

"A young boy is holding a baseball bat"

"Snowpow"

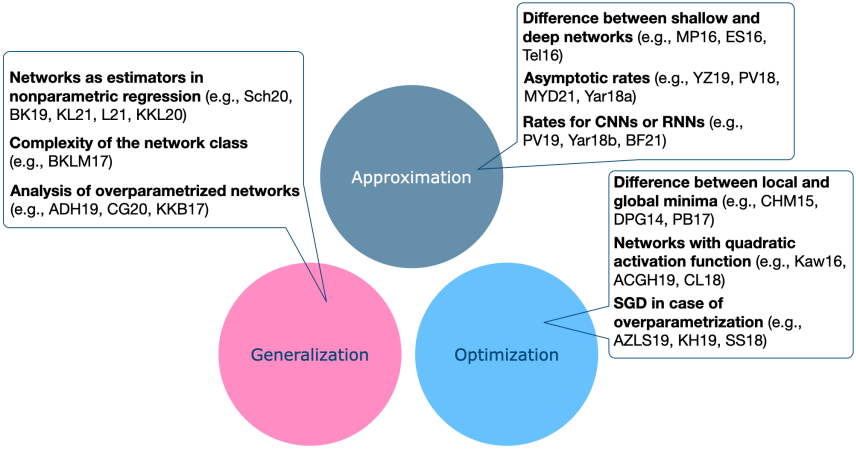
Why theory?

By far, the greatest danger for artificial intelligence is that people conclude too early that they understand it.

Eliezer Yudkowsky, AI theorist

The problem

Explaining the procedure is a highly complex task



Theory, but how?

Successful applications

...**but** lack of mathematical understanding

Problem

- Complex data structures $\hookrightarrow$ no available statistical models
- Combination of different network architectures with different regularization methods in applications
- Fitting a network to data is a non-linear problem in the network parameters
- Non-convex function class

But

- there are some interesting theoretical results

Aim of this course

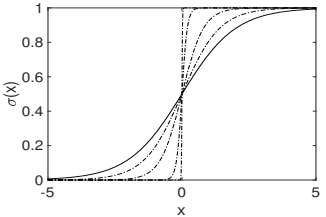
Summarize current state-of-the-art of deep learning theory and discuss open problem

Organization of the course

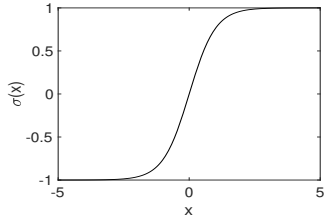
- Survey on neural networks structures and deep learning
- Theory for shallow networks
- Advantages of additional layers
- Approximation results of deep ReLU networks
- Statistical theory for deep ReLU networks
- Improving neural networks by statistical theory
- Convolutional neural networks in image classification

Multilayer feedforward neural networks

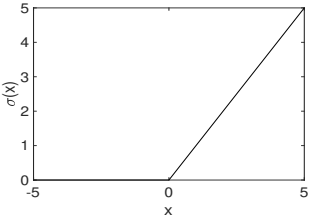
Activation function $\sigma : \mathbb{R} \to \mathbb{R}$



(a) Sigmoidal function
$\sigma(x) = 1/(1 + e^{-x/T})$
with different parameters T



(b) Tangens hyperbolicus
$\sigma(x) = \tanh(x)$



(c) Rectifier linear unit (ReLU)
$\sigma(x) = \max\{x, 0\}$

Famous activation functions

Multilayer feedforward neural networks

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

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

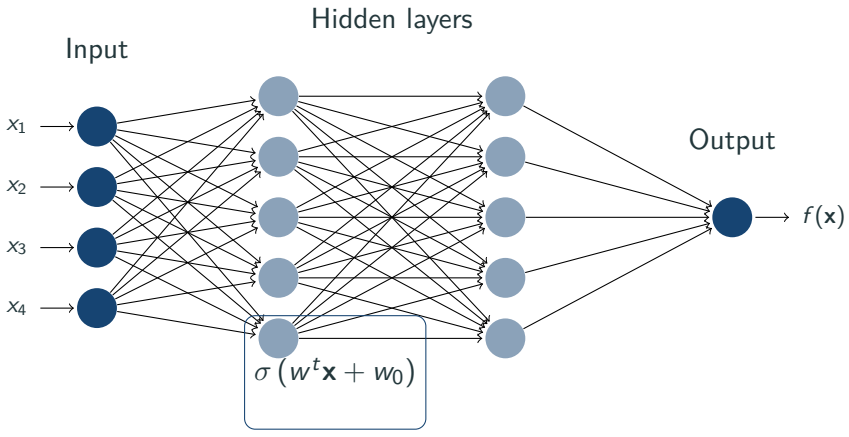
Neural network with network architecture $(L, \mathbf{k})$

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

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))

Multilayer feedforward neural networks

- Feedforward $\hookrightarrow$ Information is passed through the network in one direction
- Each neuron and the whole network are functions, resp.
- Each neuron receives signal from previous neurons, weights it using the weight vector, shifts the value through the bias and then applies the activation function to it
- Network architecture is given
- Weights and bias are chosen during the training process

Multilayer feedforward neural networks

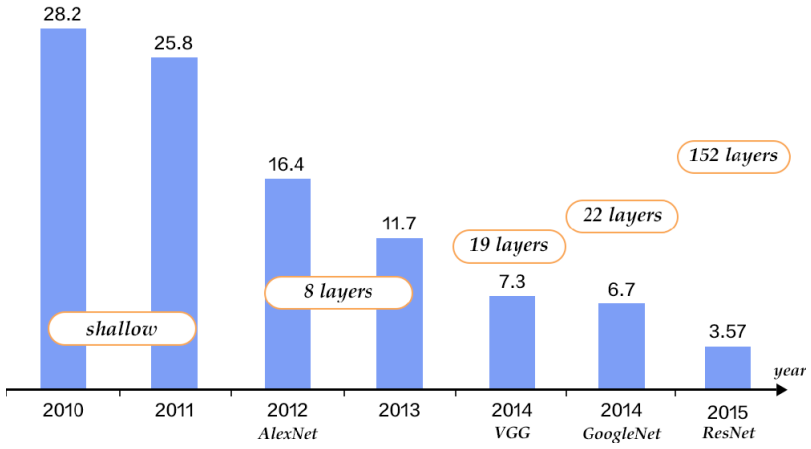
Special cases:

- $L = 1 \hookrightarrow$ Shallow neural networks
- $L \geq 2 \hookrightarrow$ Deep neural networks
- If parameter $s \in \mathbb{N}$ bounds the number of weights in the network
  $\hookrightarrow$ Sparse neural networks
- No restriction by a parameter $s \in \mathbb{N}$
  $\hookrightarrow$ Fully connected neural networks

Other famous network structures:

- Convolutional neural networks
- Recurrent neural networks

Revolution of depth



Source: http://paddlepaddle.org/

Network training

- Training $\Leftrightarrow$ Optimization of the weights in the network
- Depending on the task: Choose a proper loss function
- Algorithms: (Stochastic) gradient descent

Loss function

Given

- Data set $(\mathbf{X}_i, Y_i) \in \mathcal{X} \times \mathcal{Y}$, $i \in \{1, \ldots, n\}$
- Function class $f_{\mathbf{c}} : \mathcal{X} \to \mathcal{Y}, \quad \mathbf{c} \in \mathcal{C}$

Loss function calculates the quality of the function fit $f_{\mathbf{c}}$ via

$$L(\mathbf{c}) = \sum_{i=1}^{n} \ell(Y_i, f_{\mathbf{c}}(\mathbf{X}_i))$$

Loss function

Choice of the loss function: Distinguish between

- Classification: Neural network is asked which of the k categories an input belongs to $\hookrightarrow$ i.e., image classification
- Regression: Neural network is asked to predict a numerical value given an input $\hookrightarrow$ Prediction tasks like the expected claim amount an insured person will make

Loss functions

Regression:

Standard choice is empirical L_2-risk

$$L(\mathbf{c}) = \frac{1}{n} \sum_{i=1}^{n} (Y_i - f_{\mathbf{c}}(\mathbf{X}_i))^2.$$

Classification in k classes:

- i-th output is written as a k-dimensional 0/1-vector $\mathbf{Y}_i$
- $f_{\mathbf{c}}(\mathbf{X}_i) \in \mathbb{R}^k$ is vector of probabilities $\hookrightarrow$ Returns the probability for each class to be the correct one
- Standard choice: Loss induced by log-likelihood (also known as cross entropy)

$$L(\mathbf{c}) = -\sum_{i=1}^{n} \mathbf{Y}_i^T \log(f_{\mathbf{c}}(\mathbf{X}_i))$$

Gradient descent

Aim: Minimize loss function

$$L(\mathbf{c}) = \sum_{i=1}^{n} \ell(Y_i, f_{\mathbf{c}}(\mathbf{X}_i)).$$

Gradient descent: Choose initial parameter $\mathbf{c}_0 \in \mathbb{R}^N$ and learning rate $\eta > 0$, move along the sequence given by

$$\mathbf{c}_{t+1} = \mathbf{c}_t - \eta \cdot \nabla_{\mathbf{c}} L(\mathbf{c}_t)$$

Stochastic gradient descent

- For large datasets, gradient descent is computational not feasible
- SGD uses for each update only one observation $(\mathbf{X}_{t(i)}, Y_{t(i)})$,

$$\mathbf{c}_{t+1} = \mathbf{c}_t - \eta \cdot \nabla_{\mathbf{c}} \ell(Y_{t(i)}, f_{\mathbf{c}}(\mathbf{X}_{t(i)}))$$

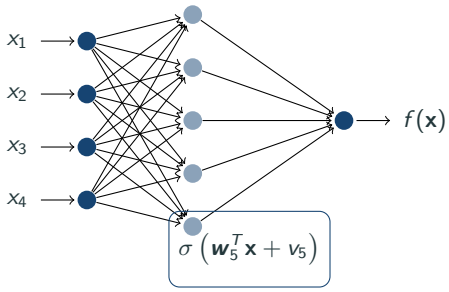
- Computationally much faster
- But: Can introduce a lot of noise
- Compromise: Compute the gradient descent based on a small subset
  $\hookrightarrow$ mini-batch

Function class of shallow neural networks

Function $f : \mathbb{R}^d \to \mathbb{R}$ of the form

$$f(\mathbf{x}) = \sum_{k=1}^{K} \alpha_k \cdot \sigma(\mathbf{w}_k^T \mathbf{x} + v_k)$$

with $\mathbf{w}_k \in \mathbb{R}^d, \alpha_k, v_k \in \mathbb{R}$.



Shallow network with $K = 5$

Questions

- How large is this class? What functions can we generate?
- How well can we approximate functions of a specific smoothness?

Universal approximation property

$$\mathcal{F}_{K,\sigma} = \left\{ f(\cdot) = \sum_{k=1}^{K} \alpha_k \cdot \sigma(\mathbf{w}_k^T \cdot + v_k) : \mathbf{w}_k \in \mathbb{R}^d, v_k, \alpha_k \in \mathbb{R} \right\}$$

- Functions in the class $\mathcal{F}_{K,\sigma}$ have $K(d+2)$ parameters
- Nested space: $\mathcal{F}_{K,\sigma} \subseteq \mathcal{F}_{K',\sigma}$ for $K \leq K'$

Universal approximation property: Shallow networks with activation function σ have the universal approximation property if for any $\epsilon > 0$ and any continuous function f on $[0,1]^d$, there exists an integer $K = K(f, \epsilon)$, such that

$$\inf_{g \in \mathcal{F}_{K,\sigma}} \|f - g\|_{L^\infty([0,1]^d)} \leq \epsilon.$$

Universal approximation property

- One-dimensional case is much easier (e.g. for ReLU)
- Many different proofs for the universal approximation exist using
  - Fourier transform
  - Radon transform
  - Hahn-Banach theorem

Discussion of different approaches gives us insights into how neural networks approximate functions.

How to show the universal approximation property

Many proofs first show universal approximation for the one-dimensional case

- Show, that univariate functions $\{\sigma(w \cdot + v) : w, v \in \mathbb{R}\}$ span the space of continuous functions
- Scalar product of these functions is not considered here

Afterwards

- Show that the function space spanned by so-called *ridge* functions

$$f(\cdot) = \sum_{j=1}^{K} g_j(\mathbf{w}_j^T \cdot)$$

with g_j univariate and continuous has the universal approximation property

Why the detour via ridge functions?

- More flexible function class than shallow neural networks
- Fitting a ridge function to data is known as projection pursuit

$\hookrightarrow$ Then we can also use shallow neural networks for projection pursuit

Universal approximation of polynomial activation functions

Activation function is a polynomial of degree r

- Span of $\{\sigma(w \cdot + v), w, v \in \mathbb{R}\}$ is contained in the space of polynomials with degree at most r
- Strongly oscillating functions cannot be approximated by polynomials
- Universal approximation does not hold
- Hidden layers help in this case $\hookrightarrow$ Nesting helps to generate polynomials of a higher degree

Universal approximation for univariate functions

Theorem: Let $\sigma \in \mathcal{C}^\infty$ and assume σ is not a polynomial. Then the corresponding shallow neural network fulfills the univariate universal approximation property.

Proof:

- $\Delta_h^1 \sigma(t) := (\sigma(t + xh) - \sigma(t))/h$
- $\Delta_h^k \sigma(t) := \Delta_h^1(\Delta_h^{k-1}\sigma)(t)$
- definition of the k-th derivative $\rightsquigarrow$

$$\left| \frac{\Delta_h^k \sigma(t)}{x^k} - \sigma^{(k)}(t) \right| \to 0, \quad \text{as } h \to 0$$

Universal approximation for univariate functions

- σ is not a polynomial $\rightsquigarrow$ there exists for each k a real number t_k with $\sigma^{(k)}(t_k) \neq 0$
- multiplying with x^k and division $\sigma^{(k)}(t_k)$ yields

$$\left| \frac{\Delta_h^k \sigma(t_k)}{\sigma^{(k)}(t_k)} - x^k \right| \to 0, \quad \text{as } h \to 0$$

- for any $h > 0$, $(\sigma^{(k)}(t_k))^{-1} \Delta_h^k \sigma(t_k)$ can be realized by a shallow network with $k+1$ units
- $\rightsquigarrow$ build networks approximating the function $x \mapsto x^k$ arbitrarily well in sup-norm
- apply Weierstrass approximation theorem

Universal approximation for univariate functions

Remarks

- Proof provides explicit construction of networks that closely resemble polynomials
- Weights in the network are of different sizes (some are extremely large, others very small)
- Conditions on the activation function are low: Only $\sigma^{(k)}(t_k) \neq 0$ must hold

$\hookrightarrow$ Small perturbations of the activation function can lead to completely different properties

$\hookrightarrow$ Linear activation function $\sigma(x) = x$ resembles the space of linear functions

$\hookrightarrow$ Small perturbations span the space of all continuous functions

$\hookrightarrow$ Networks can use the local features of activation functions

Universal approximation for univariate functions

Transfer to continuous activation functions

Theorem: Let σ be a continuous activation function, but no polynomial. Then the corresponding shallow neural network fulfills the univariate universal property.

Proof:

Proof by contradiction. Assume that $x \to x^k$ is not in $\overline{span} \bigcup_K \mathcal{F}_{K,\sigma}$.

Universal approximation in L^2

Lemma: Let $f \in L^2([0,1]^d)$. Then, there exist $\tilde{\mathbf{w}}_j \in \mathbb{R}^d$ and $c_j \in \mathbb{R}$, $j \geq 1$, such that

$$f(\mathbf{x}) = \sum_{j=1}^{\infty} \tilde{c}_j \cos(\tilde{\mathbf{w}}_j^T \mathbf{x})$$

(convergence in L^2).

Proof of universal approximation in L^2

Proof:

- $\{\phi_j : j \geq 0\}$ ONB for $L^2[0,1] \rightsquigarrow \{\bigotimes_{k=1}^d \phi_{j_k} : j_1, \ldots, j_d \geq 0\}$ ONB in $L^2([0,1]^d)$
- any $f \in L^2([0,1]^d)$ can be expanded in the cosine basis:

$$f(x_1, \ldots, x_d) = \sum_{(i_1,\ldots,i_d)\in\mathbb{N}^d} a_{i_1,\ldots,i_d} \prod_{j=1}^{d} \cos(i_j \cdot \pi \cdot x_j)$$

- Addition theorem

$$\cos(u)\cos(v) = \frac{1}{2}(\cos(u+v) + \cos(u-v))$$

- $\rightsquigarrow$ Applying this a lot of time

$$f(x_1, \ldots, x_d) = \sum_{(i_1,\ldots,i_d)\in\mathbb{N}^d} a_{i_1,\ldots,i_d} \prod_{j=1}^{d} \cos(i_j \cdot \pi \cdot x_j) = \sum_j \tilde{c}_j \cdot \cos(\tilde{w}_j{}^T \mathbf{x})$$

for suitable $\tilde{w}_j \in \mathbb{R}^d$ and $\tilde{c}_j \in \mathbb{R}$.

Universal approximation via Fourier transform

- Fourier transform $\mathcal{F}f(\boldsymbol{\xi}) = \int e^{-i\boldsymbol{\xi}^T \mathbf{x}} f(\mathbf{x}) d\mathbf{x}$
- Inverse Fourier transform $\mathcal{F}^{-1} f(\mathbf{x}) = (2\pi)^{-d} \int e^{i\mathbf{x}^T \boldsymbol{\xi}} f(\boldsymbol{\xi}) d\boldsymbol{\xi}$

$\hookrightarrow$ $f = \mathcal{F}^{-1}\mathcal{F}f$

- For any complex number z, $z = |z|e^{i\phi}$ for some real number $\phi = \phi(z)$

$\hookrightarrow$ There exists a real valued function $\phi(\mathbf{w})$ such that

$$\mathcal{F}f(\mathbf{w}) = e^{i\phi(\mathbf{w})} |\mathcal{F}f(\mathbf{w})|$$

- Fourier inversion

$$\begin{aligned} f(\mathbf{x}) &= \frac{1}{(2\pi)^d} Re \int e^{i\mathbf{w}^T \mathbf{x}} e^{i\phi(\mathbf{w})} |\mathcal{F}f(\mathbf{w})| d\mathbf{w} \\ &= \frac{1}{(2\pi)^d} \int \cos(\mathbf{w}^T \mathbf{x} + \phi(\mathbf{w})) |\mathcal{F}f(\mathbf{w})| d\mathbf{w} \end{aligned}$$

Universal approximation via Fourier transform

$$f(\mathbf{x}) = \frac{1}{(2\pi)^d} \int \cos(\mathbf{w}^T \mathbf{x} + \phi(\mathbf{w})) |\mathcal{F} f(\mathbf{w})| d\mathbf{w}$$

- Discretization of the integral gives the structure of a shallow neural network with activation function cos()
- Barron (1993) used this to show convergence rates of shallow neural networks

Universal approximation via Fourier transform

Similar approach shows the universal approximation for a broader class of activation functions. Here the following identity is used.

Lemma: If $f \in L^2(\mathbb{R}^d)$ and $\phi \in L^1(\mathbb{R})$ with $\mathcal{F}\phi(1) \neq 0$, then

$$f(\mathbf{x}) = \frac{1}{(2\pi)^d \mathcal{F}\phi(1)} \int_{\mathbb{R}^{d+1}} \phi(\mathbf{w}^T\mathbf{x} + v)\mathcal{F}f(\mathbf{w})e^{-iv}dvd\mathbf{w}.$$

Proof: Follows by Fourier inversion

$$f(\mathbf{x}) = \frac{1}{(2\pi)^d} \int e^{i\mathbf{w}^T\mathbf{x}}(\mathcal{F}f)(\mathbf{w})d\mathbf{w}$$

and

$$\int_{\mathbb{R}} \phi(\mathbf{w}^T\mathbf{x} + v)e^{-iv}dv = e^{i\mathbf{w}^T\mathbf{x}} \int_{\mathbb{R}} \phi(\mathbf{w}^T\mathbf{x} + v)e^{-i(\mathbf{w}^T\mathbf{x}+v)}dv = \mathcal{F}\phi(1)e^{i\mathbf{w}^T\mathbf{x}}.$$

Universal approximation via Fourier transform

Problem: Most of the activation functions are not in $L^1(\mathbb{R})$.

Remedy

- Sigmoidal activation function $\sigma - \sigma(\cdot + \Delta)$ is in L^1 for $\Delta > 0$
- ReLU activation function $\sigma(x) - 2\sigma(x-1) + \sigma(x+2)$ is in L^1

Lemma shows that fast approximation rates can be obtained if ϕ are $\mathcal{F}f$ smooth and $\mathcal{F}f$ decreases quickly.

Universal approximation via Radon transform

Radon transform returns all line integrals of a function

$$Rf(s, \mathbf{w}) = \int_{\langle \mathbf{x}, \mathbf{w} \rangle = s} f(\mathbf{x}) d\mu_{d-1}(\mathbf{x})$$

Inverse Radon transform (on an appropriate space)

$$f(\mathbf{x}) = \int_{\|\mathbf{w}\|=1} k(\mathbf{w}^T \mathbf{x}, \mathbf{w}) d\tilde{\mu}_{d-1}(\mathbf{w}).$$

$\hookrightarrow$ Universal approximation property: Show that the integral can be approximated by a Riemann sum (see Carroll und Dickinson (1989)).

Approximation rates for shallow neural networks

How well can we approximate functions with a special smoothness by shallow neural networks?

Approximation rates for shallow neural networks

How well can we approximate functions with a special smoothness by shallow neural networks?

Result of Mhaskar (1996): Let

- σ be a smooth activation function
- the true function be β-smooth in the L^2-Sobolev sense

then a shallow network with m hidden neurons achieves an rate of approximation

$$m^{-\beta/d},$$

where d is the dimension of the target function.

Sketch of the proof: Approximate polynomials of ridge functions first $\hookrightarrow$ standard results of polynomials lead to the rate

Approximation rates for shallow neural networks

Result of Petrushev (1999): For (in the Sobolev-sense) s-smooth activation functions, optimal approximation rates are obtained for $s + (d-1)/2$-smooth functions.

$\hookrightarrow$ Good approximation rates for functions that are smoother than the activation function

$\hookrightarrow$ Effect becomes better as input dimension increases

Sketch of the proof:

- Univariate universal approximation and reduction to ridge functions
- Approach via Radon transform $\hookrightarrow$ Has polynomial eigenbasis
- Standard results for approximation via multivariate polynomials

Criticism

- Proofs always relate shallow networks to polynomials
- Would obtain same approximation rates by directly using polynomials
- Does not help to identify problems where neural networks perform better than other methods

$\hookrightarrow$ Barron's result

Barron's result

Approach uses Maurey's theorem:

Theorem Let $(\mathbb{H}, \|\cdot\|)$ be a separable Hilbert space and $\mathcal{G} \subset \mathbb{H}$. Denote by $\overline{conv}(\mathcal{G})$ the closure of the convex hull of $\mathcal{G}$. For any $f \in \overline{conv}(\mathcal{G})$ there exist $g_1, \ldots, g_m \in \mathcal{G}$ such that

$$\left\| f - \frac{1}{m} \sum_{j=1}^{m} g_j \right\| \leq \frac{sup_{g \in \mathcal{G}} \|g\|}{\sqrt{m}}.$$

Barron's result

Fourier representation:

$$f(\mathbf{x}) - f(\mathbf{0}) = \frac{1}{(2\pi)^d} \int \Big(\cos(\mathbf{w}^T\mathbf{x} + \phi(\mathbf{w})) - \cos(\phi(\mathbf{w}))\Big) |\mathcal{F}f(\mathbf{w})| d\mathbf{w}$$
$$= \int g(\mathbf{x}, \mathbf{w}, \phi(\mathbf{w})) d\Lambda(\mathbf{w}), \tag{1}$$

with

- $\|\mathbf{w}\|_1 = \sum_{j=1}^d |w_j|$
- $C_f = \int \|\mathbf{w}\|_1 |\mathcal{F}f(\mathbf{w})| d\mathbf{w}$
- $d\Lambda(\mathbf{w}) = C_f^{-1} \|\mathbf{w}\|_1 |\mathcal{F}f(\mathbf{w})| d\mathbf{w}$
- $g(\mathbf{x}, \mathbf{w}, v) = C_f (2\pi)^{-d} (\cos(\mathbf{w}^T\mathbf{x} + v) - \cos(v)) / \|\mathbf{w}\|_1$

Barron's result

$$f(\mathbf{x}) - f(\mathbf{0}) = \int g(\mathbf{x}, \mathbf{w}, \phi(\mathbf{w}))d\Lambda(\mathbf{w})$$

- Λ is a probability measure
- $f(\cdot) - f(\mathbf{0})$ lies in the closure of the convex hull of the class

$$\mathcal{G}_{\cos} := \{g(\cdot, \mathbf{w}, v) : \mathbf{w} \in \mathbb{R}^d, v \in [0, 2\pi]\}$$

- It can be shown that $\overline{\text{conv}}\ \mathcal{G}_{\cos}$ lies in

$$\overline{\text{conv}}\ \{\gamma\sigma(\mathbf{w}^T \cdot + v) : |\gamma| \leq 2C_f/(2\pi)^d, \mathbf{w} \in \mathbb{R}^d, v \in \mathbb{R}\}$$

for any sigmoidal activation function σ

Barron's approximation theorem

Application of Maurey's theorem $\hookrightarrow$

- for any sigmoidal activation function,
- any $m \geq 1$,
- any function f

exist a shallow neural network with

$$\left\| f(\cdot) - f(\mathbf{0}) - \sum_{j=1}^{m} c_j \sigma(\mathbf{w}_j^T \cdot + v_j) \right\| \leq \frac{2C_f}{(2\pi)^d \sqrt{m}}.$$

$\hookrightarrow$ Rate is independent of the dimension d

On Barron's rate

- $C_f = \int \|\mathbf{w}\|_1 |\mathcal{F}f(\mathbf{w})| d\mathbf{w}$
- Candes (2002) shows, that truncated Fourier series achieve an even faster approximation rate

$$m^{-1/2-1/d}$$

for the same function class $\{f : C_f < \infty\}$

↪ Shallow networks do not outperform

↪ Summarizing: Approximation properties for shallow neural networks can also be shown for Fourier series or polynomial approximation

Statistical model

How perform shallow neural networks on unknown new data sets?

Nonparametric regression problem

- $(\mathbf{X}, Y)$ is $\mathbb{R}^d \times \mathbb{R}$-valued random variable with $\mathbf{E}\{Y^2\} < \infty$
- i.i.d. copies of $(\mathbf{X}, Y)$ of size $n \hookrightarrow \mathcal{D}_n = \{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)\}$
- Aim: Construct an estimator $m_n(\cdot) = m_n(\cdot, \mathcal{D}_n) : \mathbb{R}^d \to \mathbb{R}$ for the regression function $m : \mathbb{R}^d \to \mathbb{R}$, $m(\mathbf{x}) = \mathbf{E}\{Y|\mathbf{X} = \mathbf{x}\}$ such that

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

is *small*

$\hookrightarrow$ Covers binary classification problem: Choose $Y \in \{0, 1\}$ and $m(\mathbf{x}) = \mathbf{P}\{Y|\mathbf{X} = \mathbf{x}\}$

Oracle inequality

- m_n minimizes empirical risk:

$$m_n \in \operatorname{argmin}_{\theta\in\Theta} \sum_{i=1}^{n} (Y_i - m_\theta(\mathbf{X}_i))^2$$

- Standard results of empirical process theory: In case that Θ is a discrete set with cardinality $|\Theta|$ we have

$$\mathbf{E} \int |m_n(\mathbf{x}) - m(\mathbf{x})|^2 \mathbf{P}_\mathbf{X}(d\mathbf{x}) \leq C \cdot \inf_{\theta\in\Theta} |m - m_\theta|^2 + C \cdot \frac{\log|\Theta|}{n}$$

Statistical bound for shallow networks

Barron's result (1994):

- Neural networks of $\mathcal{F}_{K,\sigma}$ have $K(d+2)$ parameters

$\hookrightarrow$ Discretize network parameters

$$\log|\Theta| \leq K(d+2)\log n$$

- Oracle inequality + approximation theory:

$$\mathbf{E}\int |m_n(\mathbf{x}) - m(\mathbf{x})|^2 \mathbf{P}_\mathbf{X}(d\mathbf{x}) \leq const \cdot \left(\frac{1}{K} + \frac{K \cdot \log n}{n}\right)$$

  falls $C_f = \int \|\mathbf{w}\|_1 |\mathcal{F}f(\mathbf{w})| d\mathbf{w} < \infty$.
- Bias variance trade-off: $K = \sqrt{n/\log n}$
- Yields the rate

$$\sqrt{\frac{\log n}{n}}$$

Summary

Shallow neural networks:

- Universal approximation property
- Approximation rates
- Convergence rates

No advantages over Fourier series or polynomials

Next topic: Advantages of several hidden layers