Mathematics of Deep Learning
Lecture 4: Error Rates for Shallow Nets, Expressive Power of Depth, Kernels & Neural Nets

Prof. Predrag R. Jelenković

Time: Tuesday 4:10-6:40pm

Dept. of Electrical Engineering
Columbia University , NY 10027, USA
Office: 812 Schapiro Research Bldg.
Phone: (212) 854-8174
Email: predrag@ee.columbia.edu
URL: http://www.ee.columbia.edu/∼predrag

General Error Rates: Smooth Function Classes

Recall the usual notation:

- For $\boldsymbol{s} \in \mathbb{Z}_+^d, \boldsymbol{s} = (s_1, \ldots, s_d) \geq \mathbf{0}, |\boldsymbol{s}| = s_1 + \cdots + s_d$, we can define a differential operator

$$D^{\boldsymbol{s}} := \frac{\partial^{|\boldsymbol{s}|}}{\partial x_1^{s_1} \cdots \partial x_d^{s_d}}$$

and a class of functions with up to $s \geq 1$ partial derivatives

$$C^s(\mathbb{R}^d) := \left\{ f : D^{\boldsymbol{k}} f \in C(\mathbb{R}^d), \text{ for all } |\boldsymbol{k}| \leq s \right\}$$

- For these functions, it make sense to measure the distance between the functions and all of their derivatives, motivating the following norm

$$\|f\|_{s,p} := \begin{cases} \left(\sum_{0 \leq |\boldsymbol{k}| \leq s} \|D^{\boldsymbol{k}} f\|_p^p\right)^{1/p}, & 1 \leq p < \infty \\ \max_{0 \leq |\boldsymbol{k}| \leq s} \|D^{\boldsymbol{k}} f\|_\infty, & p = \infty \end{cases}$$

where

$$\|g\|_p := \begin{cases} \left(\int_K |g(\boldsymbol{x})|^p d\boldsymbol{x}\right)^{1/p}, & 1 \leq p < \infty \\ \operatorname{ess\,sup}_{\boldsymbol{x} \in K} |g(\boldsymbol{x})|, & p = \infty \end{cases}$$

over some domain $K \in \mathbb{R}^d$.

Sobolev Spaces

- We will pick K to be a d-ball (or d-cube)

$$B^d = \{\boldsymbol{x} : \|\boldsymbol{x}\|_2^2 = x_1^2 + \cdots + x_d^2 \leq 1\}$$

- Sobolev space $\mathcal{W}_p^s = \mathcal{W}_p^s(B^d)$ is a completion of $C^s(B^d)$ with the respect to the p norm. In addition we assume that all $f \in \mathcal{W}_p^s$ have a bounded norm $\|f\|_{s,p}$.
- Also, we define a space of all NN approximations with n neurons

$$\mathcal{N}_n := \left\{ \sum_{i=1}^{n} a_i \sigma(\boldsymbol{w}_i \cdot \boldsymbol{x} + b_i), \boldsymbol{w}_i \in \mathbb{R}^d, a_i, b_i \in \mathbb{R} \right\}$$

- and (d-width)

$$E_{s,p}(\mathcal{W}_p^s, \mathcal{N}_n) := \sup_{f \in \mathcal{W}_p^s} \inf_{\hat{f} \in \mathcal{N}_n} \|f - \hat{f}\|_{s,p}$$

Uniform Upper and Lower Bounds

For one layer NNs this is well understood:

Theorem Assume that σ is infinitely differentiable and not a polynomial. Also, assume that the NN approximation $\hat{f}_n$, as a mapping $\mathcal{W}_p^s \to \mathcal{N}_n$, is continuous in parameters $(\boldsymbol{w}, a, b)$, than, for each $1 \leq p \leq \infty, m \geq 1$, there is a finite constant C, such that

$$C^{-1} n^{-s/d} \leq E_{s,p}(\mathcal{W}_p^s, \mathcal{N}_n) \leq C n^{-s/d}$$

Remarks:

- We'll give some general idea on the proof of the upper bound. Lower bound is too involved; we saw a version of it from Barrons (1993).
- References: See Theorem 1 and the notes after it in the recent survey by Poggio et al. (2017). For more rigorous presentation see Section 6 in Pinkus (1999), Theorems 6.6 & 6.8.
- An immediate consequence of the preceding theorem is that if we want a uniform ϵ approximation, then the number of neurons n needed is

$$n^{-s/d} = \epsilon \;\Rightarrow\; n = O(\epsilon^{-d/s}) \quad \text{(curse of dimensionality)}$$

We will use this as a reference point for later results.

Upper Bound: General Outline

- Let $\mathcal{W}_p^s \equiv \mathcal{W}_p^s(K)$ be Sobolev space on a compact domain, say cube $K = [0,1]^d$, or a d-dimensional ball.
- Next $\mathcal{P}_k$ be d-dimensional polynomials, than it is well known that

$$E_{s,p}(\mathcal{W}_p^s, \mathcal{P}_k) \leq Ck^{-s}. \tag{1}$$

- Next, how big shallow neural network, $\mathcal{N}_n$, (n number of neurons) do we need to approximate precisely a d-dimensional polynomial $h(\boldsymbol{x}) \in \mathcal{P}_k, \boldsymbol{x} \in \mathbb{R}^d$?
- It is well known that $h(\boldsymbol{x}) \in \mathcal{P}_k$ can be represented as (see proof of Theorem 4.1 in Pinkus)

$$h(\boldsymbol{x}) = \sum_{i=1}^{r} p_i(\boldsymbol{w}_i \cdot \boldsymbol{x}),$$

where $p_i(x)$ are univariate polynomials and $r = \binom{d-1+k}{k} = \dim \mathcal{H}_k$, and $\mathcal{H}_k$ is a space of homogeneous d-dimensional polynomials of degree k.

Upper Bound: General Outline

- The idea for proving the preceding statement is based on the claim that there exist weight vectors $\{\boldsymbol{w}_i \cdot \boldsymbol{x}\}_{i=1}^r$, such that

$$\mathcal{H}_k = \text{span}\{(\boldsymbol{w}_i \cdot \boldsymbol{x})^k, i = 1, \ldots, r\}$$

- Moreover, it can be shown that $\{(\boldsymbol{w}_i \cdot \boldsymbol{x})^j\}_{i=1}^r$ spans all homogeneous polynomials $\mathcal{H}_j, 0 \leq j \leq k$, implying

$$\mathcal{P}_k = \cup_{j=0}^k \mathcal{H}_j = \text{span}\{(\boldsymbol{w}_i \cdot \boldsymbol{x})^k, i = 1, \ldots, r, 0 \leq j \leq k\}$$

implying that $h(\boldsymbol{x}) \in \mathcal{P}_k$ can be represented as

$$h(\boldsymbol{x}) = \sum_{i=1}^r p_i(\boldsymbol{w}_i \cdot \boldsymbol{x}),$$

where p_i are univariate polynomials of degree k.

Upper Bound: General Outline

- Next, if σ is infinitely differentiable and non-polynomial, then each p_i can be approximated to arbitrary precision with $(k+1)$ neurons.
- This comes from the fact that there exists b_0 such that

$$\frac{d^j}{d\lambda^j}\sigma(\lambda x + b_0)\bigg|_{\lambda=0} = x^j \sigma^{(j)}(b_0) \neq 0, \quad j = 1, \dots k$$

  and the number of finite differences need to approximate these derivatives.
- Hence, the number of neurons needed to approximate $h(\boldsymbol{x}) \in \mathcal{P}_k$ to arbitrary precision is

$$n \approx (k+1)r = (k+1)\binom{d-1+k}{k} \approx kk^{d-1} = k^d \to k \approx n^{1/d},$$

  which, in combination with Equation (1)

$$E_{s,p}(\mathcal{W}_p^s, \mathcal{P}_k) \leq Ck^{-s} \approx Cn^{-s/d}.$$

General Takeaways

- Curse of dimensionality: For uniform ϵ approximation, the number of neurons n needed is

$$n^{-s/d} = \epsilon \;\Rightarrow\; n = O(\epsilon^{-d/s}) \quad (),$$

  i.e., exponential number of neurons.
- In terms of approximating a general class of smooth functions on bounded domain, the lower bound tells us
  - The worst case bounds for NNs with n neurons/parameters are equivalent to any other model with n-parameters, e.g., Fourier, wavelets, polynomials, etc.
- We can do better if we assume more structure, i.e., smaller classes fo functions, like in Barron.
- How can depth help?

Barron Assumed a Smaller Class of Functions

Barron defined a much smaller class of functions, which is much more suitable for approximation with NNs

$$\Gamma_C \subset \mathcal{W}_\infty^1 \subset \mathcal{W}_2^1$$

Let $f : \mathbb{R}^d \to \mathbb{R}$. Barron considers a class of functions Γ_C, $C > 0$, satisfying the following integrability condition

$$C_f = \int_{\mathbb{R}^d} \|\boldsymbol{\omega}\|_2 |\tilde{f}(\boldsymbol{\omega})| d\boldsymbol{\omega} \leq C, \tag{2}$$

where $\|\boldsymbol{\omega}\|_2 = \sqrt{\boldsymbol{\omega} \cdot \boldsymbol{\omega}}$ and $f(\boldsymbol{x}), \tilde{f}(\boldsymbol{\omega})$ is the Fourier transform pair

$$f(\boldsymbol{x}) = \int_{\mathbb{R}^d} e^{i\boldsymbol{\omega}\cdot\boldsymbol{x}} \tilde{f}(\boldsymbol{\omega}) d\boldsymbol{\omega}.$$

Remark: Informally, one could thing of condition (2) as f being "band limited", i.e., not significant high frequency components. Alternatively, it can be shown that functions in Γ_C have bounded variation, see p. 180 in Pinkus (1999).

Comments of Barron

Theorem 1 (Barron) For any $f \in \Gamma_{C,B}$, probability measure μ, there exists a NN approximation $f_n(\boldsymbol{x})$ with n neurons, such that

$$\int_B (f(\boldsymbol{x}) - f_n(\boldsymbol{x}))^2 \mu(dx) \leq \frac{4C^2}{n}. \tag{3}$$

The linear coefficients, c_k, in f_n can be chosen to satisfy $\sum |c_k| \leq 2C$.

Proof in Barron is very long. However, it can be shown that Barron functions can be represented as

$$\Gamma_{C,B} \subset \left\{ f(x) : f(x) = \int_{\|\boldsymbol{w}\|_2=1, |b|\leq 1} a(\boldsymbol{w}, b)\sigma(\boldsymbol{w} \cdot \boldsymbol{x} - b) d\pi(\boldsymbol{w}, b) \right\}$$

where π is the probability measure and $\int |a(\boldsymbol{w}, b)| d\pi(\boldsymbol{w}, b) \leq C$.
Proof of (3): Make n random samples $(\boldsymbol{w}_i, b_i)$ from distribution π and make $\hat{f}(\boldsymbol{x}) = 1/n \sum_{i=1}^n a(\boldsymbol{w}_i, b_i)\sigma(\boldsymbol{w}_i \cdot \boldsymbol{x} - b_i)$. Then, (3) is a Var($\hat{f}$).

Remark: Hence, if we assume that a function has an integral NN representation, then we can approximate such integral well via importance sampling - not surprising. We will come to this representation later in the class.

Impact of Depth

- With two hidden layers it is easy to create a pulse in $\mathbb{R}^d$:
  - Hence, we can obtain an elementary proof of Cybenko, and others, using only calculus. Haven't seen this idea in the literature, so please refer to this lecture notes if you decide to use it. I might write a short note about it.
- Namely, let $\sigma(x) = 1_{\{x \geq 0\}}$ and $\boldsymbol{e}_i$ be the standard ith base in $\mathbb{R}^d$, then for $d \geq 2, \boldsymbol{x} \in \mathbb{R}^d$,

$$\Pi_\delta(\boldsymbol{x}) := \sigma\left(\sum_{i=1}^{d}(\sigma(\boldsymbol{x}\cdot\boldsymbol{e}_i) - \sigma(\boldsymbol{x}\cdot\boldsymbol{e}_i - \delta)) - d + 1/2\right) = \begin{cases} 1, & \text{if } x \in [0,\delta]^d, \\ 0, & \text{otherwise.} \end{cases}$$

- Now, choose large n, set $\delta = 1/n$, and consider a grid $S = [0, 1/n, 2/n, \ldots, (n-1)/n]^d$. With this grid we can obtain an arbitrarily accurate NN approximation for any $f \in C([0,1]^d)$ using

$$\hat{f}(\boldsymbol{x}) = \sum_{s \in S} f(s)\Pi_{1/n}(\boldsymbol{x} - \boldsymbol{s}).$$

- If instead of a step function $\sigma(x) = 1_{\{x \geq 0\}}$, we have a general sigmoid, ReLU, generalized polynomial spline, we can combine scaling and finite differences to obtain an arbitrarily close approximation to a prefect pulse.

Power of Depth: Kolmogorov Superposition Theorem

Solution to Hilbert's 13th problem. See Section 7 in Pinkus (1999).

Theorem There exist d constants $\lambda_j > 0, j = 1, \ldots, d, \sum_{j=1}^{d} \lambda_j \leq 1$, and $2d + 1$ strictly increasing continuous functions $\phi_i : [0, 1] \to [0, 1]$, such that any continuous function f of d variables on $[0, 1]^d$ can be represented as

$$f(\boldsymbol{x}) = \sum_{i=1}^{2d+1} g\left(\sum_{j=1}^{d} \lambda_j \phi_i(x_j)\right)$$

for $g \in C([0, 1])$ depending on f.

General Power of Depth: 2 Hidden Layers are Enough

Kolmogorov's theorem can be easily adopted to NN setting (Maiorov&Pinkus 1999), see Theorem 7.1 in Pinkus (1999)

Theorem There exists and activation function $\sigma \in C^{\infty}(\mathbb{R})$, which is strictly increasing, sigmoidal, and has the following property. For any $f \in C([0,1]^d)$, and $\epsilon > 0$, there exist constants, $\delta_i, a_{ij}, b_{ij}, \gamma_i$, and vectors $\boldsymbol{w}_{ij} \in \mathbb{R}^d$, for which

$$\left| f(\boldsymbol{x}) - \sum_{i=1}^{4d+3} \delta_i \sigma \left(\sum_{j=1}^{2d+1} a_{ij} \sigma(\boldsymbol{w}_{ij} \cdot \boldsymbol{x} + b_{ij}) + \gamma_i \right) \right| < \epsilon,$$

for $\boldsymbol{x} \in [0,1]^d$.

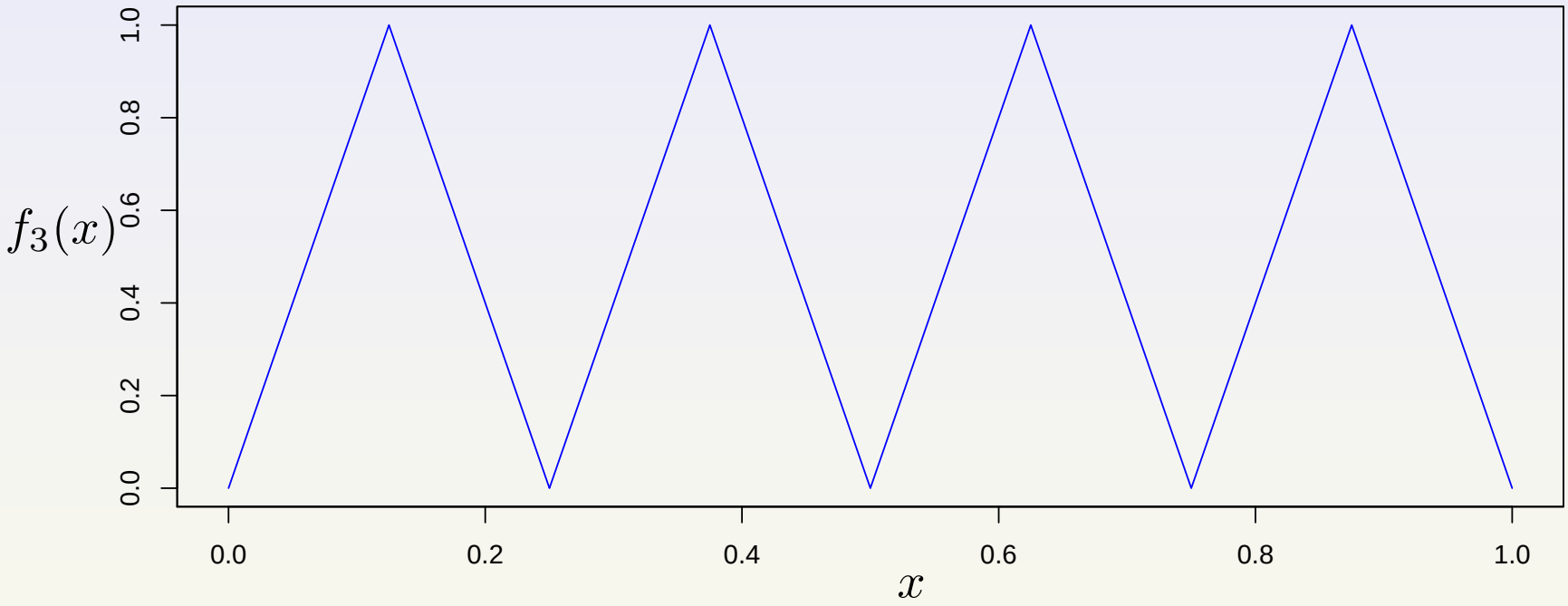
Remarks

- Note, there exists a specific, possibly pathological, activation function, e.g.: discontinuous and dependent on f.
- The first application of Kolmogorov's theorem was by Hecht-Nielsen (1987).
- For a brief historical summary and a ReLU implementation see Montanelli and Yang (2020): Error bounds for deep ReLU networks using the Kolmogorov–Arnold superposition theorem

"High Frequency" Sawtooth Function

Consider sawtooth $f_n(x) : [0,1] \to [0,1]$ with 2^{n-1} equally spaced teeth

Sawtooth Function With 4 Teeth



How many ReLU neurons are needed to represent f_n with 1 hidden layer?

2^n neurons

Depth To The Rescue: Hat Function

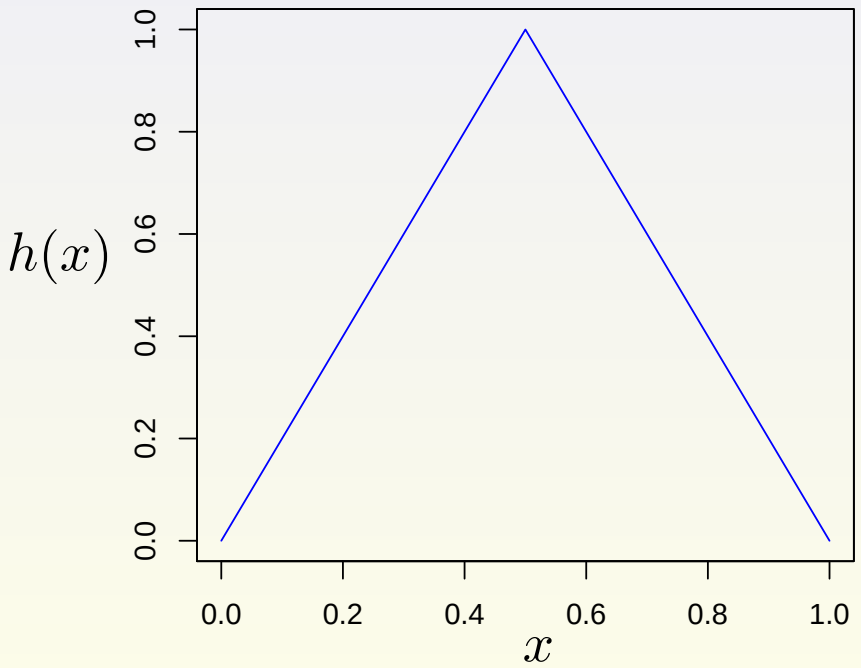
Use 3 ReLUs, $\sigma(x) : \max(0, x)$, to define "hat" function

$$h(x) = 2\sigma(x) - 4\sigma(x - 1/2) + 2\sigma(x - 1)$$

or, equivalently

$$h(x) = \sigma\left(2\sigma(x) - 4\sigma(x - 1/2)\right)$$

Hat Function

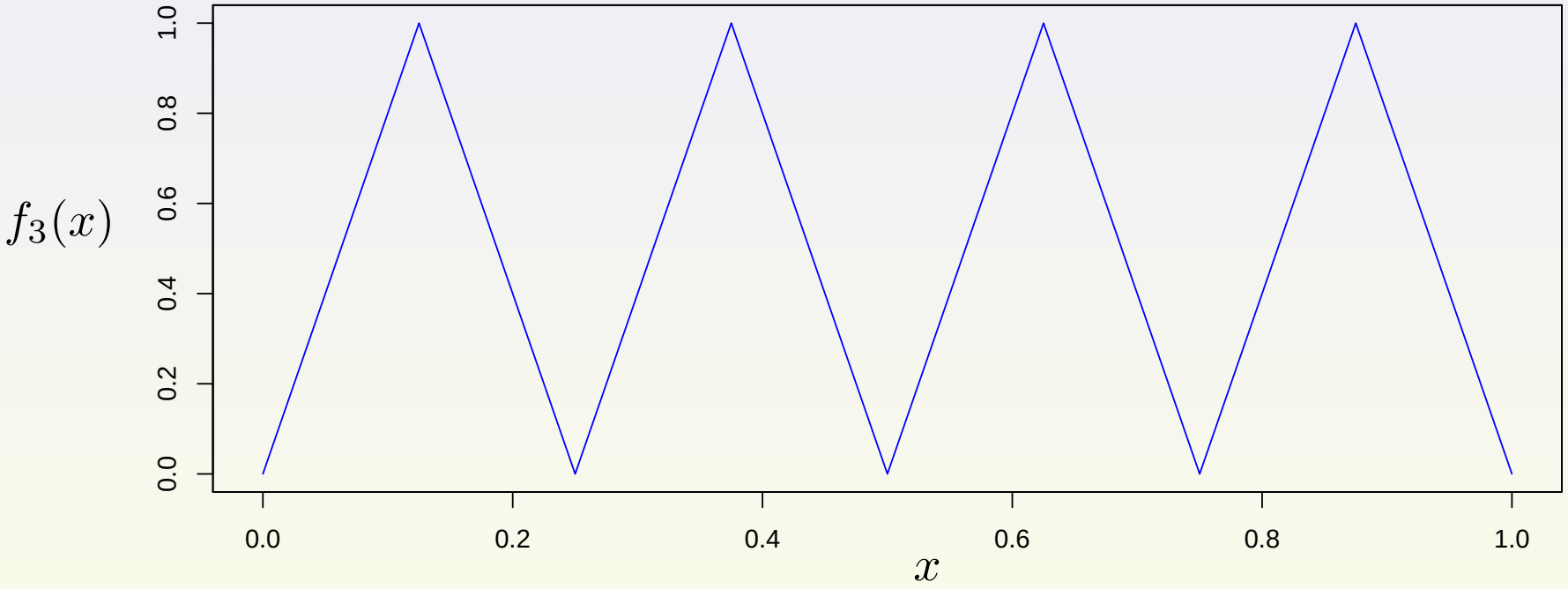


Depth To The Rescue: Hat Function Composition

Define composition of functions $h^{(n)}$ as

$$h^{(n)}(x) = h \circ h \cdots h \circ h(x) = h(h(\cdots h(x) \cdots))$$

$$f_3(x) = h^{(3)}(x) = h(h(h(x)))$$



Claim Function f_n with 2^{n-1} teeth can be represented by a deep network of depth n and $3n$ ReLU neurons.

Depth To The Rescue: Hat Function Composition

Key idea: Hat function composition - exploited in:

- Classification: Telgarsky (2015)
- Function approximation: Yarotsky (2017), Liang and Srikant (2017), Montanelli and Du (2018)

We will cover parts of these papers today.

Hat function composition allows

$$\text{more efficient representation} = \text{smaller } \# \text{ of neurons}$$

Intuition:

- Depth increase the number of oscillations (frequency) exponentially
- While width increases it additively

Hat Function Composition - Classification

Telgarsky (2015) - Classification setup:

- Use ReLU: $\sigma(x) = \max(0, x)$ (for the remainder of today's lecture)
- Classification problem: 2 classes - $\{0, 1\}$
- Classifier: for any function $f : \mathbb{R} \to \mathbb{R}$, define classifier

$$\tilde{f}(x) = 1_{\{f(x) \geq 1/2\}}$$

- Empirical error: for a sequence of points $\{(x_i, y_i\}_{i=1}^n$ with $x_i \in \mathbb{R}$ and $y_i \in \{0, 1\}$, define the empirical error

$$\bar{R}(f) = \frac{1}{n} \sum_{i=1}^{n} 1_{\{f(x_i) \neq y_i\}}$$

- $\mathcal{H}(\sigma; m, l)$: Hypothesis class of functions that can be constructed using NN with l layers, each with width of at most m nodes.

Telgarsky's (2015) Theorem

Theorem Let positive integer k, number of layers l, and number of nodes per layer m be given with $m \leq 2^{(k-3)/l-1}$. Then there exists a collection of $n := 2^k$ points $\{(x_i, y_i)\}_{i=1}^n$ with $x_i \in [0,1]$ and $y_i \in \{0,1\}$ such that

$$\min_{f \in \mathcal{H}(\sigma;2,2k)} \bar{R}(f) = 0 \quad \text{and} \quad \min_{f \in \mathcal{H}(\sigma;m,l)} \bar{R}(f) \geq \frac{1}{6}$$

Proof: Choose $x_i = i2^{-k}$ and set $y_{2i} = 0, y_{2i+1} = 1, 0 \leq i \leq 2^k - 1$

Upper bound: Showing that $2 \times (2k)$ network can produce a perfect classifier: $\bar{R}(f) = 0$.

- Construct k-fold composition of "hat" function, $h^{(k)}(x)$, using $2 \times (2k)$ neural net, i.e., $h^{(k)} \in \mathcal{H}(\sigma; 2, 2k)$
- Note that $h^{(k)}(x)$ perfectly reproduces data

$$h^{(k)}(x_i) = y_i$$

Hence, $\bar{R}(h^{(k)}) = 0$

Telgarsky's (2015) Theorem

Proof: *Lower bound:* Showing that $m \times l$ network cannot produce a perfect classifier, i.e., $\bar{R}(f) \geq 1/6$. This is a bit more involved.

- $f : \mathbb{R} \to \mathbb{R}$ is *t-sawtooth* if it is piecewise linear (affine) with t linear pieces.

Lemma 1 Let $\{(x_i, y_i)\}_{i=1}^n, n = 2^k$, be the set of points defined earlier. Then, every t-sawtooth function $f : \mathbb{R} \to \mathbb{R}$ satisfies

$$\bar{R}(f) \geq \frac{n - 4t}{3n}.$$

Proof lemma. Recall the classifier $\tilde{f}(x) = 1_{\{f(x) \geq 1/2\}}$.

- f crosses $1/2$ at most once in every interval of its linear growth.
- $\tilde{f}(x)$ is piecewise constant (0 or 1) over $2t$ intervals.
  Meaning, n points must fall in $2t$ buckets (intervals).
- At least $n - 4t$ intervals (buckets) must have at least 3 points.
- At least $1/3$ of points in each of the $n - 4t$ are labeled incorrectly (since points alternate). This completes the proof of the lemma.

Telgarsky's (2015) Theorem

Proof: *Lower bound:* (continued)

Lemma 2 If f is p-sawtooth and g is q-sawtooth, then $f+g$ is at most $p+q$-sawtooth and $f \circ g$ is at most pq-sawtooth.

Proof Lemma 2.

- Addition: "joints" of f can fall within linear intervals of g, and adding a linear function does not change the sawtooth degree.
- Composition: each interval on which f is linear has as its domain at most the range of g. Hence, on such intervals, $f(g(x))$ is at most q-sawtooth. Since there are p such intervals, $f \circ g$ is at most pq-sawtooth.

Now, return to the *proof of the theorem.*

- Note: $\sigma(x)$ is 2-sawtooth. Hence, function f, made by $(m \times l)$-NN is at most $(2m)^l$-sawtooth. Thus, by assumption $m \leq 2^{(k-3)/l-1}$,

$$(2m)^l \leq 2^{k-3}$$

- Finally, by Lemma 1, with $n = 2^k$ and $t \leq 2^{k-3}$,

$$\bar{R}(f) \geq \frac{2^k - 4 \cdot 2^{k-3}}{3 \cdot 2^k} = \frac{1}{6},$$

which completes the proof.

General Function Approximation

Use "hat" function composition to efficiently represent general functions, $f : [0,1]^d \to \mathbb{R}$

efficient representation = smaller # of neurons

Papers: Yarotsky (2017), Liang & Srikant (2017), Montanelli & Du (2018)

General approximation plan:

1. Use "hat" function composition to efficiently represent some nice functions, e.g. polynomials, f_P, by neural nets, f_N, so that, in some norm
$$\|f_P - f_N\| \leq \frac{\epsilon}{2}$$
2. Show that polynomials f_P approximate well more general functions, f, (in some larger functional space, e.g., Sobolev space)
$$\|f - f_P\| \leq \frac{\epsilon}{2}$$
3. Put the preceding steps together via triangular inequality

$$\|f - f_N\| = \|(f - f_P) + (f_P - f_N)\| \leq \|f - f_P\| + \|f_P - f_N\| \leq \epsilon$$

Approximating Polynomials: $f(x) = x^2$

First step in approximating multivariate polynomials on $[0,1]^d$ is to construct a quadratic on $[0,1]$:

$$f(x) = x^2$$

Proposition (2, Yarotski, 2017) For any $\epsilon > 0$, the function $f(x) = x^2$ on the $[0,1]$ can be approximated by a ReLU network function, $f_N(x)$, having depth, number of weights and computation units $O(\ln(1/\epsilon))$, such that

$$|f(x) - f_N(x)| < \epsilon$$

Proof: The key observation is that $f(x) = x^2$ can be approximated by a linear combination of "hat" composition functions $h^{(s)}$.

- Let $f_m(x)$ be a piecewise linear interpolation of $f(x)$ with $2^m + 1$ uniformly distributed break points $k2^{-m}, k = 0, 1, \ldots, 2^m$

$$f\left(\frac{k}{2^m}\right) = f_m\left(\frac{k}{2^m}\right) = \left(\frac{k}{2^m}\right)^2$$

Approximating Polynomials: $f(x) = x^2$

Proof: (continued)

- Next,

$$|f_m(x) - f(x)| \leq \frac{1}{2^{2m+2}}$$

- Also, note that

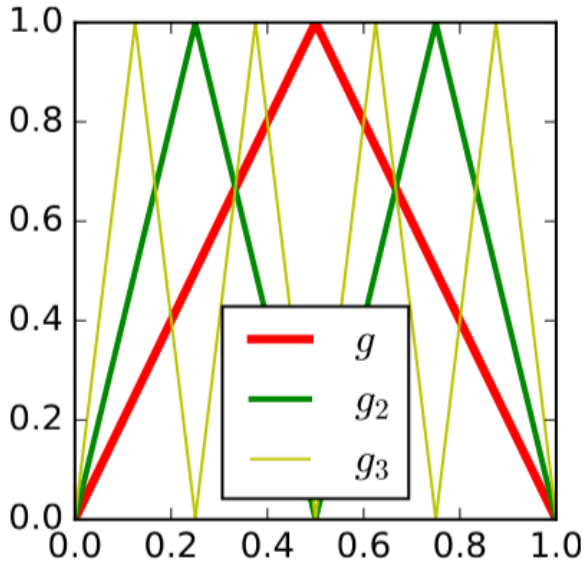
$$f_{m-1}(x) - f_m(x) = \frac{h^{(m)}(x)}{2^{2m}}$$

- From which it follows

$$f_m(x) = x - \sum_{s=1}^{m} \frac{h^{(s)}(x)}{2^{2s}}$$

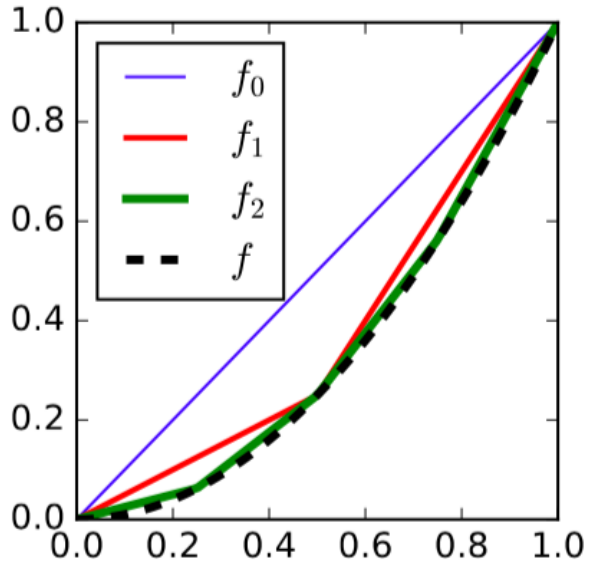
Approximating Polynomials: $f(x) = x^2$

Efficient approximation of $f(x) = x^2$



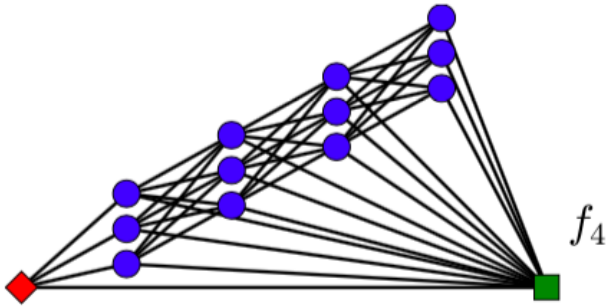
(a)

$g_s \equiv h^{(s)}$



(b)

approximating functions f_m



(c)

network architecture for f_4

Approximating Multivariate Polynomials

Next step in approximating multivariate polynomials on $[0,1]^d$ is to construct a product function on $[0,1]^2$:

$$f(x,y) = xy$$

Proposition (3, Yarotski, 2017) For any $\epsilon > 0$, the function $f(x,y) = xy$ on the $[0,1]^2$ can be approximated by a ReLU network function, $f_N(x,y)$, having depth, number of weights and computation units $O(\ln(1/\epsilon))$, such that

$$|f(x,y) - f_N(x,y)| < \epsilon$$

Proof: Follows immediately from Proposition 2 and

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

Approximating General Smooth Functions

1. Use Propositions 1&2 to approximate polynomials of any degree on $[0,1]^d$
2. Assume that functions, f, are smooth enough (have enough derivatives), to bound the error between f and its polynomial - Taylor expansion

Putting 1. and 2. together yields
Theorem (1, Yarotski, 2017) For any $\epsilon > 0$, a function $f(\boldsymbol{x})$ on the $[0,1]^d$ with up to s partial derivatives can be approximated by a ReLU network function, $f_N(\boldsymbol{x})$, having depth $O(\ln(1/\epsilon))$, and number of weights and computation units $O(\epsilon^{-d/s}\ln(1/\epsilon))$, such that

$$|f(\boldsymbol{x}) - f_N(\boldsymbol{x})| < \epsilon$$

Reducing Curse of Dimensionality: Montanelli & Du

- They use the idea of sparse grids
  for a comprehensive intro to sparse grids, see

[BG04] Sparse Grids, by Bungartz and Griebel, 2004.

[G12] Sparse Grids in a Nutshell by Garcke, 2012.

- Sparse grids have been used earlier in numerical integration to lessen the curse of dimensionality
- Piecewise linear bases (scaled "hat" function) are commonly used in sparse grids
  - As we saw earlier, scaled "hat" function can be constructed easily composition of ReLU units
  - Also we can efficiently approximate as product function, $f(x, y) = xy$, using ReLUs
  - Putting the preceding two facts together, one can obtain multidimensional grids

Reducing Curse of Dimensionality: Montanelli & Du

- One of the key tools from sparse grids (see Lemma 3.13 in [BG04]) is that, given a number N of grid points,

$$N_\epsilon = O(\epsilon^{-1/2} |\log_2 \epsilon|^{3(d-1)}), \quad d \text{ is the dimension}$$

a sufficiently smooth function f, can be approximate by sparse grids function, f_N, with ϵ error

$$\|f - f_N\|_\infty \leq \epsilon$$

- Using the preceding result, and clever construction of sparse grids using ReLUs, in Theorem 1 of [MD18], the epsilon approximation was achieved using

$$N = O(\epsilon^{-1/2} |\log_2 \epsilon|^{3(d-1)/2+1} \log_2 d) \quad \text{ReLUs}$$

Note: $\log_2 \epsilon$ is raised to power instead of $1/\epsilon$ in Yarotsky (18).

Sparse Grids in 1D

- Let $\phi(x)$ be the scaling "hat" function

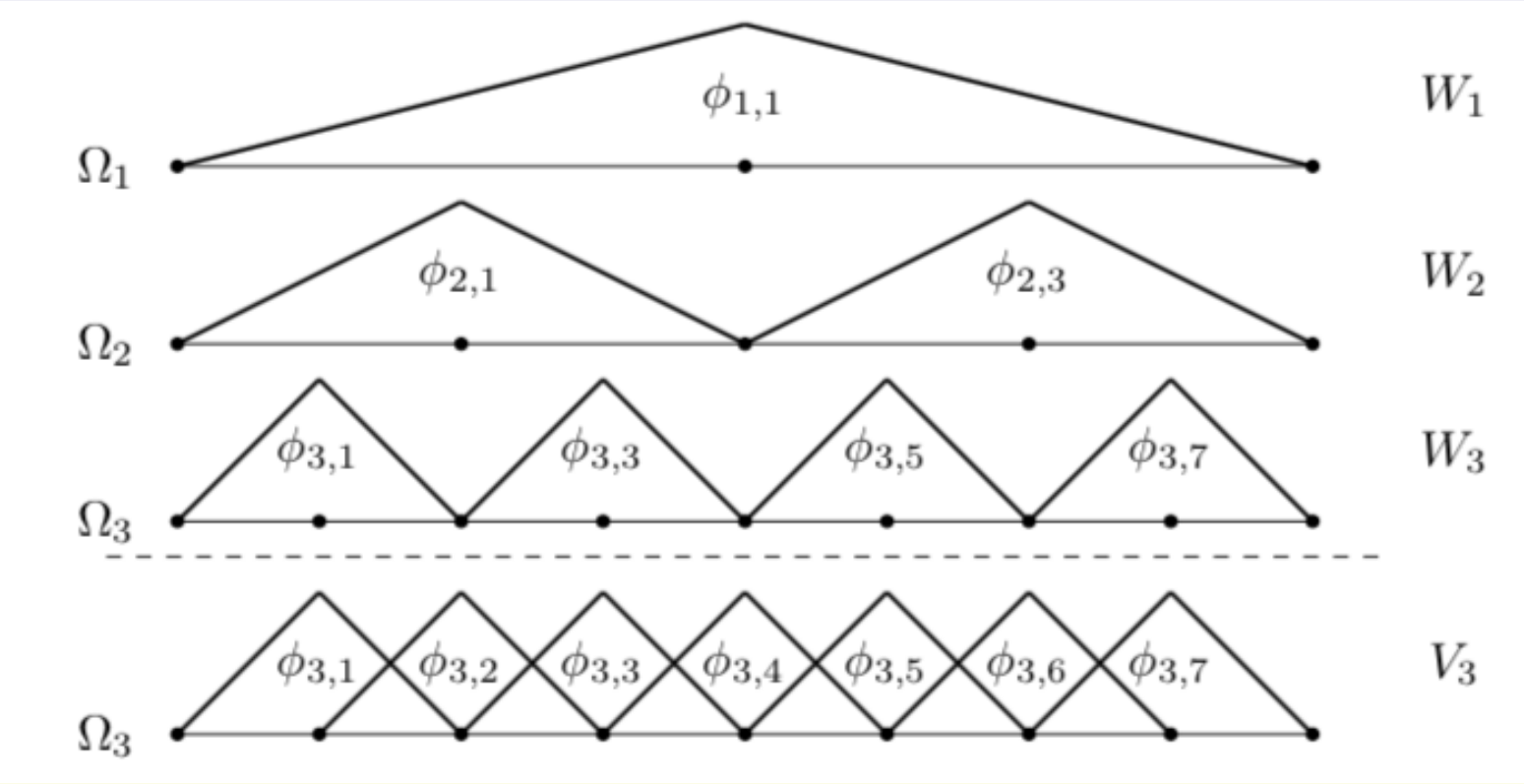
$$\phi(x) = 1 - |x|, \qquad \text{if } \; x \in [-1, 1]$$

and $\phi(x) = 0$, otherwise. Scaling $\phi(x)$, one constructs the basis functions at level l

$$\phi_{l,i}(x) = \phi\left(\frac{x - i2^{-l}}{2^{-l}}\right)$$

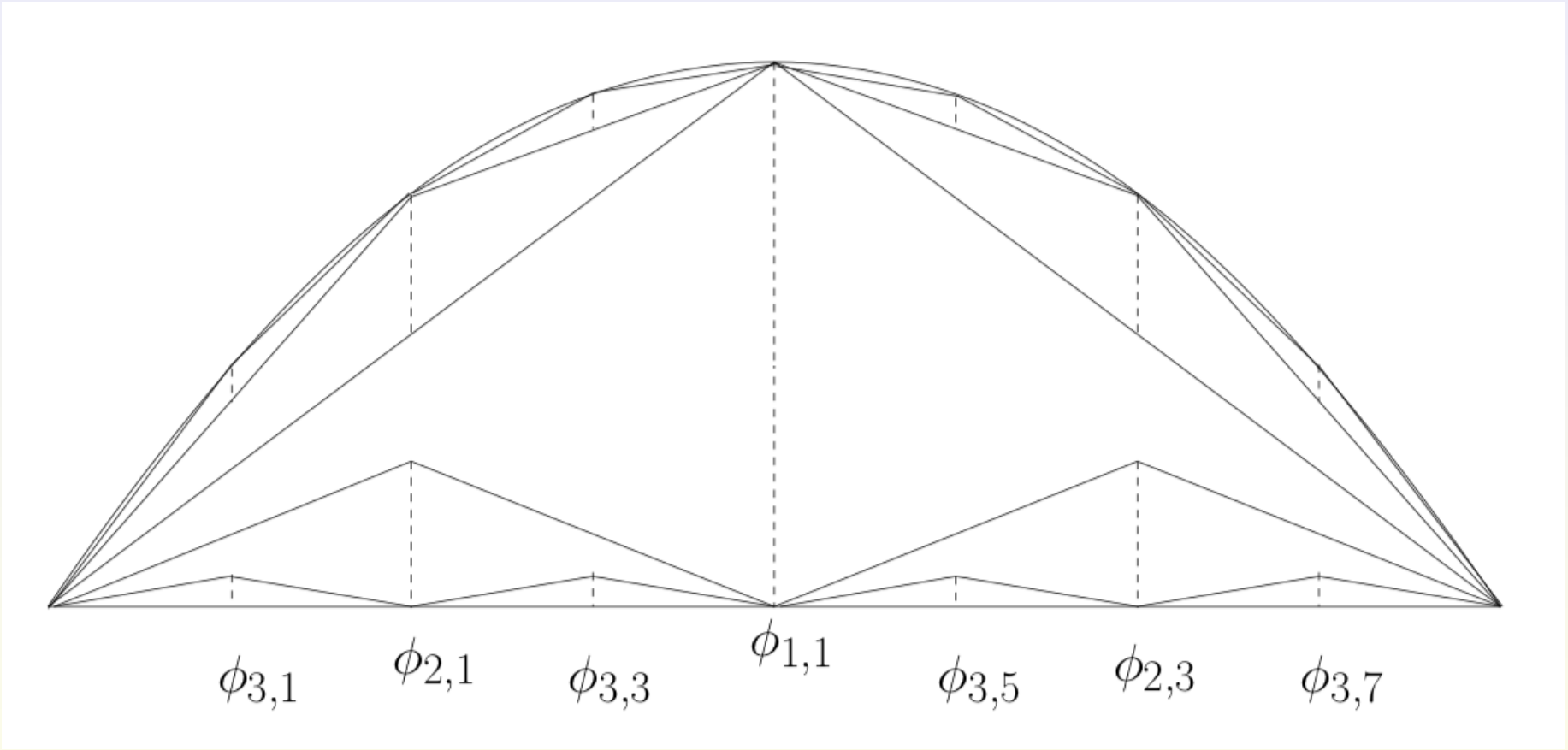
Sparse Grids in 1D: Hierarchical and Nodal Basis

- V_l - nodal bases; W_l - hierarchical basis
- Can be efficiently constructed using compositions of ReLUs



Interpolation of Parabola With Hierarchical Basis

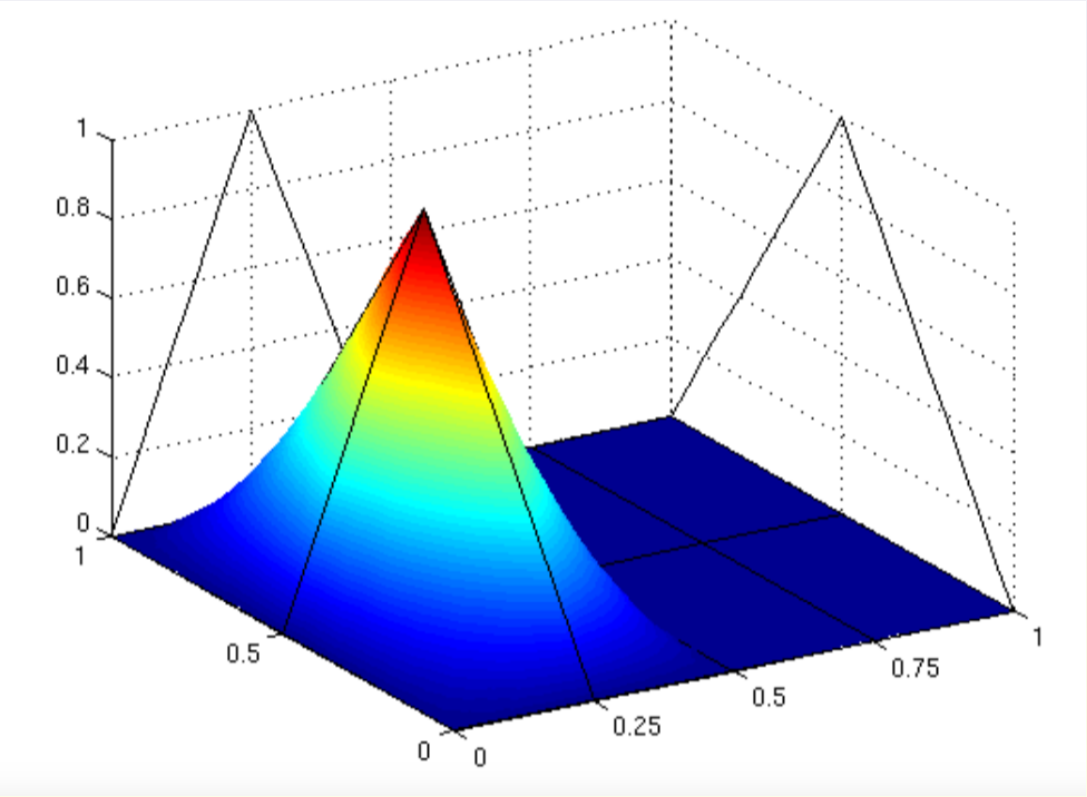
- Interpolation of a parabola with hierarchical basis: level $l = 3$



Sparse Grids in Multiple Dimensions

- Make a cross product (tensor product) of 1D basis
- Can be efficiently constructed using ReLUs: recall we can approximate (Yarotsky (18), Liang and Srikant (17))

$$f(x, y) = xy$$



Results of Montanelli & Du

Hence, in a nutshell, the paper uses the following steps:

- Construct 1D hierarchical basis using the composition property of "hat" function and Deep ReLU networks
- Make product of these to obtain multidimensional hierarchical bases using the efficient approximation of $f(x, y)$ by ReLUs
- Use the preceding construction and Lemma 3.13 in [BG04]) to show that (Theorem 1 of [MD18]) the epsilon approximation was achieved with

$$N = O(\epsilon^{-1/2} |\log_2 \epsilon|^{3(d-1)/2+1} \log_2 d) \quad \text{ReLUs}$$

  Note: $\log_2 \epsilon$ is raised to power instead of $1/\epsilon$ in Yarotsky (17).
- Where is the trick? - Exploit special structure of Korobov spaces, which have nice (infinite) representation in hierarchical (hat function) basis.
- Hence, to get better than exponential bounds, we need to assume a special structure on the space of functions.

"Reducing" Curse of Dimensionality: Poggio et al. (2017)

In general, one has to assume some **special structure** of the space of functions that are being approximated, e.g., smoothness, etc.

- In Why and When Can Deep-but Not Shallow-networks Avoid the Curse of Dimensionality: A Review, by Poggio et al. (2017), the authors deal with compositional functions.
- Compositional functions: This class includes in particular functions that are a composition of two-dimensional functions, e.g., in dimension $d = 8$,

$$f(x_1, \ldots, x_8) = f_3\left(f_{21}(f_{11}(x_1, x_2), f_{12}(x_3, x_4)), f_{22}(f_{13}(x_5, x_6), f_{14}(x_7, x_8))\right)$$

for some bivariate functions $f_{11}, f_{12}, f_{13}, f_{14}, f_{21}, f_{22}$ and f_3.

- Compositional functions can be represented by a binary tree with d inputs variables, $\log_2 d$ levels and $(d-1)$ nodes
  - each of the nodes can be approximated by a subnetwork of size $N = \epsilon^{-2/s}$, where s is the number of derivatives.

Depth Separation: Eldan and Shamir (2016)

Telgarsky doesn't show precisely separation results between networks of depth $l-1$ and l. The first result in this direction is by

- Eldan and Shamir who showed separation between depths $l=2$ and $l=3$ for pretty general activation functions.
- More precisely, assume that σ satisfies, for some C, α,

$$|\sigma(x)| \leq C(1+|x|^{\alpha})$$

- Assume that f is Lipschitz continuous.

Theorem 1 [Eldan and Shamir (16)] There exist universal constants c, C, such that the following holds: For every dimension $d > C$, there is a probability measure μ on $\mathbb{R}^d$ and a function $g : \mathbb{R}^d \to \mathbb{R}$ with the following properties:

1. g is bounded in $[-2, 2]$, supported on $\{x : \|\boldsymbol{x}\|^2 \leq Cd\}$ and expressible by a 3-layer network of width $O(d^5)$
2. Every function f, expressed by a 2-layer network of width at most ce^{cd}, satisfies

$$\mathbb{E}(f-g)^2 \geq c.$$

Depth Separation: Eldan and Shamir (2016)

Informally, 3-layer networks can express certain functions with polynomial number of neurons, while 2-layer networks would require and exponential number of nodes.

Proof sketch. Part 1.

- Choose g to be radial: $g(x) = G(\|\boldsymbol{x}\|^2)$
- Its Fourier transform will be radial
- Representing g with 3 (2 hidden) layers
  - Represent the norm $\|\boldsymbol{x}\|^2$ in one layer: first compute for each coordinate x_i^2 and then sum them up.
  - Compute G with the second hidden layer
  - Sum things up in the third layer

Depth Separation: Eldan and Shamir (2016)

Proof sketch. Part 2 (Lower bound - hard part). Here the idea is to show that expressing function g with 2 (1 hidden) layers would require an exponential number of neurons.

- Consider density function $\phi(\boldsymbol{x})^2$, where ϕ is the inverse Fourier transform of $1_{\{\boldsymbol{x}\in B\}}$, where B is the unit ball at the origin.
- Set $d\mu = \phi(\boldsymbol{x})^2 d\boldsymbol{x}$; then MSE can be written as

$$\mathop{\mathbb{E}}_{\mu}(f-g)^2 = \int (f(\boldsymbol{x})\phi(\boldsymbol{x}) - g(\boldsymbol{x})\phi(\boldsymbol{x}))^2 d\boldsymbol{x} = \|f\phi - g\phi\|^2 \quad (4)$$

- Next, the goal is to prove a lower bound on the preceding equation.
- The idea will be to consider Fourier transforms of $\widehat{f\phi}$ and $\widehat{g\phi}$ and derive the lower bound in the Fourier domain since

$$\mathop{\mathbb{E}}_{\mu}(f-g)^2 = \|f\phi - g\phi\|^2 = \|\widehat{f\phi} - \widehat{g\phi}\|^2$$

Depth Separation: Eldan and Shamir (2016)

Proof sketch. Part 2. Luckily, the Fourier transform of functions expressible by a 2-layer network has a very particular form.

- Consider any 2-layered (1-hidden) function with $f_i(x) = a_i\sigma(x - b_i)$

$$f(\boldsymbol{x}) = \sum_i f_i(\langle \boldsymbol{v}_i, \boldsymbol{x} \rangle)$$

- By examining $\widehat{f\phi}$ as a convolution, $\widehat{f\phi}$ is supported on "tubes"

$$\text{Supp}(\widehat{f\phi}) \subset T = \cup\,(\text{span}(\boldsymbol{v}_i) + B)$$

  where B is a unit-volume ball.

- Next, one shows that the Fourier transform $\widehat{g\phi}$, where g is a 3-layered radial function, has a lot of mass away from the origin.
  - To cover this mass one needs an exponential number of "tubes"

$$\text{span}(\boldsymbol{v}_i) + B$$

  - High dimensionality plays a critical role in this argument.

Boolean Functions

Ona computer, all functions are implemented as

$$f : \{\pm 1\}^d \to \{\pm 1\}^m$$

- Hence, binary classifiers are boolean, $f : \{\pm 1\}^d \to \{\pm 1\}$
- For simplicity consider $\sigma(x) = \text{sign}(x)$
- What boolean functions can be implemented by $\mathcal{H}_{V,E,\text{sign}}$? The following is Theorem 20.2 in [UML] book.
- **Theorem** For every d, let $N(d)$ be the minimal integer such that there exists a graph (V, E) with $|V| = N(d)$ such that the hypothesis class $\mathcal{H}_{V,E,\text{sign}}$ contains all the functions from $\{0,1\}^d \to \{0,1\}$. Then, $N(d)$ is exponential in d.
  Proof based on **Theorem:** The VC dimension of $\mathcal{H}_{V,E,\text{sign}}$ is $O(|E| \log(|E|))$; see Sections 20.3-20.4 in Shai's book.
- We will introduce VC dimension in the future.

Depth Separation: Boolean Functions

The following result follows a long line of work in theoretical computer science, in particular work by Hastad (see the references in Rossman et al. blow).

- Each layer is consists of unary NOT, binary AND or OR gate
- We call such networks Boolean circuits

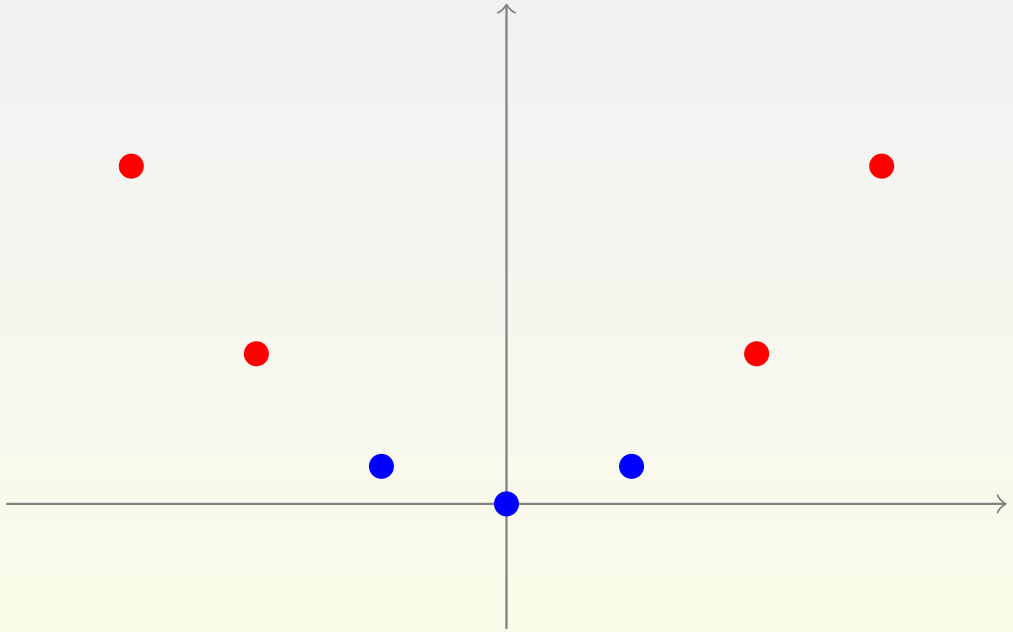
Theorem [Rossman, Servedio, Tan (2015)] For any fixed d, there is a function $f : \{0,1\}^n \to \{0,1\}$ computed by a linear circuit of depth d and size $O(n)$ such that any depth-$(d-1)$ circuit computing a function that agrees with f on a fraction $1/2 + \delta$ of the possible inputs, for $\delta > 0$, has size at most $O(2^{n^{\Omega(1/d)}})$.

Feature Transformations: Kernels

Example: Classifying red and blue point on a real line

- Can't be separated by a single point (hyperplane)

- Nonlinear feature transformation: $x \rightarrow (x, x^2)$
- Now, the points are separable by a single hyperplane (!)



- This is called feature extraction
- Well-developed theory: Reproducing Kernel Hilbert Spaces (RKHS)

Supervised Learning Theory

Supervised learning: Given training data $(\boldsymbol{x}, \boldsymbol{y})$, i.e.,

$$(x_1, x_2, \ldots, x_n) \rightarrow \boxed{\boldsymbol{f}(x)} \rightarrow (y_1, y_2, \ldots, y_n)$$

Problem:

- Don't know $\boldsymbol{f}$
- Find the "best" approximation $\hat{\boldsymbol{f}}$

What have we seen?

- Linear Ridge: $\hat{f}(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j$

$$\min_{\beta_0, \boldsymbol{\beta}} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} \beta_j \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

- Lasso: same as above, just change the penalty to ℓ_1 norm:

$$\lambda \sum_{j=1}^{p} \beta_j^2 \rightarrow \lambda \sum_{j=1}^{p} |\beta_j|$$

Supervised Learning Theory

- Polynomial Ridge or Lasso: Same as before, but more general, polynomial, approximation function

$$\hat{f}(X) = \beta_0 + \sum_{j=1}^{p} \sum_{l=1}^{k} \beta_{jl} (X_j)^l$$

- Splines: Pick $\hat{f}(X)$ to be piecewise polynomial, e.g., piecewise linear. Problem: discontinuities.
- Smoothing Splines: Impose continuity and derivative constrains.

Supervised Learning Theory

Classification: let $y_i \in \{-1, 1\}$

- Support Vector Classifier: can be obtained by solving

$$\min_{\beta_0, \boldsymbol{\beta}} \sum_{i=1}^{n} \max\left[0, 1 - y_i(\beta_0 + x_{i1}\beta_1 + \cdots + x_{ip}\beta_p)\right] + \lambda \sum_{j=1}^{p} \beta_j^2$$

  $\max\left[0, 1 - y_i(\beta_0 + x_{i1}\beta_1 + \cdots + x_{ip}\beta_p)\right]$ is called **hinge loss**
  For general SVM use a kernel generalization of the hyperplane

- Logistic regression with ℓ_2 penalty: replace the hinge loss in the preceding expression with

$$\left\{y_i(\beta_0 + x_{i1}\beta_1 + \cdots + x_{ip}\beta_p) - \ln\left(1 + e^{\beta_0 + x_{i1}\beta_1 + \cdots + x_{ip}\beta_p}\right)\right\}$$

Unified Supervised Learning Theory

In general, all the preceding leaning problems can be written as

$$\hat{f}^* = \arg\min_{\hat{f}\in\mathcal{H}} \sum_{i=1}^{n} L(y_i, \hat{f}(x_i)) + \lambda\Omega(\hat{f}) \tag{5}$$

where L is a general loss function, and $\lambda\Omega(\hat{f})$ is the penalty, and $\mathcal{H}$ is the space of approximation functions that we are considering.

- Example: linear functions $\hat{f}(X) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$
  - $\mathcal{H}$: set off all p-dimensional linear functions
  - Penalty - square of the ℓ_2 norm
    $$\Omega(\hat{f}) = < \hat{f}, \hat{f} > = \|f\|^2 = \sum_{j=1}^{p} \beta_j^2$$
  - Loss: $L(y_i, \hat{f}(x_i)) = (y_i - \hat{f}(x_i))^2$

Q: What is the most general that $\hat{f}, \mathcal{H}, L, \Omega$, can be such that the problem has nice analytical and computational properties?

RKHS: Reproducing Kernel Hilbert Spaces

We have already seen it with SVM: here some more details[1]

- Hilbert space $\{\mathcal{H}\}$: natural generalization of Euclidian spaces.
- It has an inner product: $< f, g >$, for any $f, g \in \mathcal{H}$, which allows computing angles between the elements of $\{\mathcal{H}\}$.
  In particular, $< f, g >= 0$, then f, g are orthogonal.
- Norm/distance: Norm $\|f\| = \sqrt{< f, f >}$, and
  distance $d(f, g) = \|f - g\| = \sqrt{< f - g, f - g >}, \qquad f, g \in \mathcal{H}$
- Euclidian example: if $x, y \in \mathbb{R}^d$, then $< x, y >= x_1 y_1 + \cdots + x_p y_p$

$$\|x\| = \sqrt{x_1^2 + x_2^2 + \cdots + x_p^2}, \quad d(x, y) = \sqrt{(x_1 - y_1)^2 + \cdots + (x_p - y_p)^2}$$

- Example - matrices: $< A, B >= \text{trace}(A^\top B)$
- Example - random variable: $< X, Y >= \text{cov}(X, Y)$

RKHS: Reproducing Kernel Hilbert Spaces

RKHS - $\mathcal{H}_k$: Subset of Hilbert spaces with really nice properties

- Kernel is a positive definite function $k(x, y)$
  $(\sum \alpha_i \alpha_j k(x_i, x_j) \geq 0)$
- **Reproducing property**: if $f \in \mathcal{H}_k$, then

$$< f, k(\cdot, x) >= f(x)$$

- Each kernel, $k(x, y)$, defines uniquely the Hilbert space, $\mathcal{H}_k$ Hence, in this space, $\mathcal{H}_k$, all we need to know if the kernel!
- Functions in this space $f(x) \in \mathcal{H}_k$ can be written as

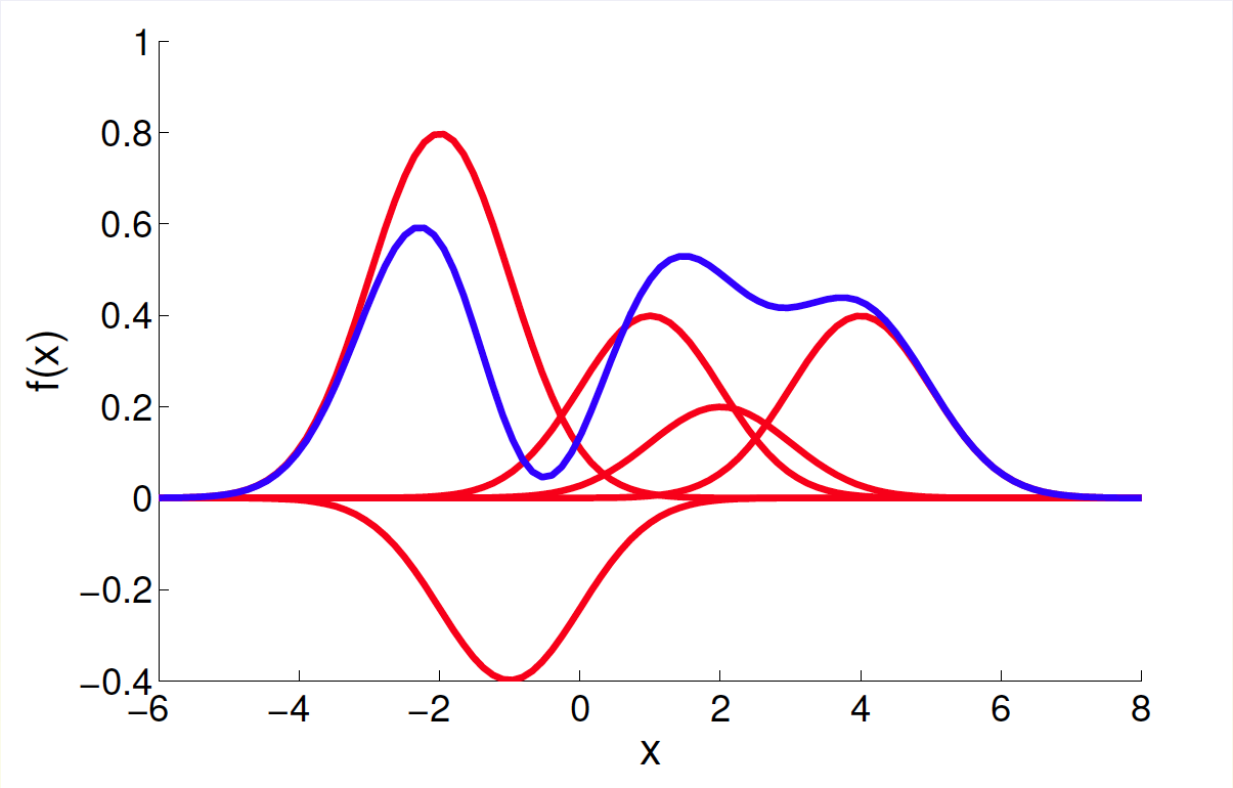
$$f(x) = \sum_i \beta_i k(x, x_i)$$

Example: Gaussian (Radial) Kernel

Gaussian kernel in 1D: $k(x, y) = e^{-\frac{(x-y)^2}{2\sigma}}$

Typical function: sum of weighted Gaussian "blobs" - blue line below

$$f(x) = \sum_i \beta_i k(x, x_i) = \sum_i \beta_i e^{-\frac{(x-x_i)^2}{2\sigma}}$$



$\mathcal{H}_k$ is a linear space, but $f(x)$ is very nonlinear!

Generalized Learning in RKHS

Here we optimize over all approximation function in $\mathcal{H}_k$

$$\hat{f}^* = \arg\min_{\hat{f} \in \mathcal{H}_k} \sum_{i=1}^{n} L(y_i, \hat{f}(x_i)) + \lambda \Omega(\|\hat{f}\|^2_{\mathcal{H}_k}) \tag{6}$$

Representer Theorem For any loss function L and any strictly increasing function $\Omega : \mathbb{R} \to \mathbb{R}$, an optima solution, $\hat{f}^*$, of Equation (6) has a form

$$\hat{f}^*(x) = \sum_{i=1}^{n} \beta_i k(x, x_i)$$

$\hat{f}^*$ has a finite representation (!) even though $\mathcal{H}_k$ can be (is) infinite.
We can put much of the supervised learning under the same umbrella.
Drawback: works only with ℓ_2 norm, $\|\hat{f}\|_{\mathcal{H}_k}$ - doesn't work for Lasso.

Representer Theorem: Proof

Let's drop "hat" from $\hat{f}$ and let

$$f_s(x) = \sum_{i=1}^{n} \beta_i k(x, x_i)$$

Then, any function $f \in \mathcal{H}_k$ can be decomposed as

$$f(x) = f_s(x) + f_\perp(x) \tag{7}$$

where $f_\perp(x)$ is perpendicular to f_s. Hence, (Pythagoras theorem for $\mathcal{H}_k$)

$$\|f\|^2_{\mathcal{H}_k} = \|f_s\|^2_{\mathcal{H}_k} + \|f_\perp\|^2_{\mathcal{H}_k} \geq \|f_s\|^2_{\mathcal{H}_k} \tag{8}$$

Next, by kernel reproducing property and orthogonality

$$f(x_i) =< f, k(\cdot, x_i) >=< f_s, k(\cdot, x_i) >= f_s(x_i) \tag{9}$$

Finally, (8) and (9) imply that an optimizer of (6) must be in the form of $f_s(x)$ from (7) since Ω is increasing and

$$L(y_i, f(x_i)) = L(y_i, f_{\boldsymbol{s}}(x_i)), \qquad \Omega(\|\hat{f}\|^2_{\mathcal{H}_k}) \geq \Omega(\|f_{\boldsymbol{s}}\|^2_{\mathcal{H}_k})$$

Generalized Ridge

For quadratic loss function

$$\hat{f}^* = \arg\min_{\hat{f}\in\mathcal{H}_k} \sum_{i=1}^{n} (y_i - \hat{f}(x_i))^2 + \lambda \|\hat{f}\|^2_{\mathcal{H}_k} \tag{10}$$

The coefficients, β_i^*, of the optimizer

$$\hat{f}^*(x) = \sum_{i=1}^{n} \beta_i^* k(x, x_i)$$

are explicitly given by

$$\boldsymbol{\beta}^* = (\boldsymbol{K} + \lambda \boldsymbol{I})^{-1} \boldsymbol{y}$$

where $\boldsymbol{K}$ is a square matrix with elements $K_{ij} = k(x_i, x_j)$, $\boldsymbol{I}$ is an identity matrix, and $\boldsymbol{y}$ is the column vector $(y_1, \ldots, y_n)^\top$.

Generalized Ridge: Proof

From the Representer Theorem we know that the minimizer has to be of the form

$$f(x) = \sum_{i=1}^{n} \beta_i k(x, x_i)$$

and their quadratic norm is given by

$$\|f\|^2_{\mathcal{H}_k} =< f, f >= f(x) = \sum_{i=1}^{n}\sum_{j=1}^{n} \beta_i \beta_j k(x_j, x_i) = \boldsymbol{\beta}^\top \boldsymbol{K}\boldsymbol{\beta}$$

since, by reproducing property, $< k(\cdot, x_j), k(\cdot, x_i) >= k(x_j, x_i)$. Also, by reproducing property,

$$f(x_i) =< f(\cdot), k(\cdot, x_i) >= \sum_{j=1}^{n} \beta_j k(x_j, x_i)$$

By replacing the preceding 2 equations in the optimization function in Equation (10)

$$\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} (y_i - \sum_{j=1}^{n} \beta_j k(x_j, x_i))^2 + \lambda \boldsymbol{\beta}^\top \boldsymbol{K}\boldsymbol{\beta}$$

which by taking the derivative w.r.t. $\boldsymbol{\beta}$ and setting it to 0 yields

$$\boldsymbol{\beta} = (\boldsymbol{K} + \lambda \boldsymbol{I})^{-1} \boldsymbol{y}$$

Class Reading: Expressive Power Of Depth

We tried to answer the question: Can deep neural networks express certain types of functions much more efficiently in terms of the number of neurons compared to shallow networks?

- Approximation theory of the MLP model in neural networks, by Pinkus, 1999.
- Representation Benefits of Deep Feedforward Networks, by Telgarsky, 2015.
- Benefits of depth in neural networks, by Telgarsky, 2016.
- Error bounds for approximations with deep ReLU networks, by Yarotsky, 2017.
- WHY DEEP NEURAL NETWORKS FOR FUNCTION AP- PROXIMATION?, Liang and Srikant, 2017.
- NEW ERROR BOUNDS FOR DEEP RELU NETWORKS USING SPARSE GRIDS, by Montanelli and Du, 2018.
- Why and When Can Deep-but Not Shallow-networks Avoid the Curse of Dimensionality: A Review, by Poggio et al. (2017)
- Will cover some ideas from: The Power of Depth for Feedforward Neural Networks, by Eldan and Shamir, 2016.
- An average-case depth hierarchy theorem for Boolean circuits, by Rossman, Servedio, Tan, 2015.
- Shallow and Deep Networks are Near-Optimal Approximators of Korobov Functions , by M. Blanchard, M. A. Bennouna, by ICLR 2022.
- Check the citations as well as the references inside these papers for additional reading.

Next Class

- **Optional:** If interested in how to approximate functions that don't always have derivatives, check: weak derivatives, distributions and Sobolev spaces. (E.g., see Functional Anal. by Rudin, or Real Anal. by Folland.) In general, don't worry about these.
- Continue the new topic on connection between Neural Nets and Kernels. Some good books on Kernels: (available online through CU Library)
  1. Chapters 1, 2, 13-16 in: B. Schölkopf and A. J. Smola. *Learning with Kernels.* MIT Press, Cambridge, MA, 2002.
  2. Chapters 1, 5.3, 6, 7 in (this book is mathematically advanced) A. Berlinet and C. Thomas-Agnan. *Reproducing Kernel Hilbert Spaces in Probability and Statistics.* Kluwer, 2004.

Have Fun!