# Approximation of structured functions and simple neural networks

## Approximation, sampling and compression in data science Isaac Newton Institute

**February 2019, Cambridge, UK**

**Jan Vybíral**
Czech Technical University
Prague, Czech Republic

# Outline

- (In)tractability of approximation in high dimensions
- Sparse Additive Models
- Sums of ridge functions and neural networks

# Approximation of multivariate functions

- $K$ - class of objects of interest: vectors, functions, operators,...
- ...approximated by (same or simpler) objects ...
- ...usually with only limited information about the unknown object...
- with the similarity (approximation error) measured by some sort of distance
- worst case - supremum of the approximation error over $K$
- average case - mean value of the (square of the) approximation error over $K$ w.r.t. some probability measure on $K$

# Approximation of multivariate functions

Let $f : \Omega \subset \mathbb{R}^d \to \mathbb{R}$ be a function of many ($d \gg 1$) variables

We want to approximate $f$ using only (a small number of) function values $f(x_1), \ldots, f(x_n)$

Questions:

- Which functions (assumptions on $f$)?
- How to measure the error?
- How to choose the sampling points?
- Decay of the error with growing number of sampling points?
- Algorithms and optimality?
- Dependence on $d$?

# Sampling numbers

$\mathcal{F}_d$ - a class of $d$-variate functions on $\Omega \subset \mathbb{R}^d, \mathcal{F}_d \subset C(\Omega)$

**Information map:** $N : \mathcal{F}_d \to \mathbb{R}^n$, $N(f) = (f(x_1), \ldots, f(x_n)) \in \mathbb{R}^n$

Continuous **recovery map:** (linear or not) $\phi : \mathbb{R}^n \to L_\infty(\Omega)$

**Sampling operator:** $S_n = \phi \circ N$

**Approximation error:** $e(S_n) := \sup_{f \in \mathcal{F}_d} \|f - S_n(f)\|_\infty$

. . . consider the worst function for your algorithm. . .

**Sampling numbers:** $g_{n,d}(\mathcal{F}_d, L_\infty) := \inf_{S_n} e(S_n)$

. . . the error of the "best algorithm" when using only $n$ function values. . .

**Its inverse function:** $n(\varepsilon, d) = \min\{n \in \mathbb{N} : g_{n,d}(\mathcal{F}_d, L_\infty) \leq \varepsilon\}$
. . . minimal number of sampling points needed to achieve an approximation of the error at most $\varepsilon > 0$. . .

If $\mathcal{F}_d$ is a unit ball of a function space $X(\Omega) \subset C(\Omega)$ and the error is measured in $Y(\Omega)$, then

$$g_{n,d}(\mathcal{F}_d, Y(\Omega)) = \inf_{S_n} \sup_{\|f\|_{X(\Omega)} \le 1} \|f - S_n(f)\|_{Y(\Omega)}$$
$$= \inf_{S_n} \|I - S_n : \mathcal{L}(X(\Omega), Y(\Omega))\|$$

Well known for many classical function spaces, like Sobolev spaces, Besov spaces, Triebel-Lizorkin spaces, etc.

**Typical decay:** $n^{-s/d}$

Birman, Solomyak, Temlyakov, Kudryavtsev, Kashin, DeVore, Maiorov, Kruglyak, Heinrich, Novak, Triebel, and many others ...

Novak & Woźniakowski: Tractability of Multivariate Problems, Volumes II & III

# Curse of dimension

Many classical problems suffer from exponential dependence of the results on $d$!

**Example: Approximation of smooth functions**
Let $\mathcal{F}_d := \{f : [0,1]^d \to \mathbb{R}, \|D^\alpha f\|_\infty \leq 1, \alpha \in \mathbb{N}_0^d\}$
**Smoothness does not help!**
Infinitely differentiable functions on $\Omega = [0,1]^d$:

Novak, Woźniakowski (2009): Initial error is the same as error of uniform approximation for $n \leq 2^{\lfloor d/2 \rfloor} - 1$
. . . curse of dimension!
. . . the number of sampling points must grow exponentially in $d$

V. '14: Uniform approximation on the cube $[-1/2, 1/2]^d$ of functions from the class

$$F_d^1 = \left\{ f \in C^\infty([-1/2, 1/2]^d) : \sup_{k \in \mathbb{N}_0} \sum_{|\beta|=k} \frac{\|D^\beta f\|_\infty}{\beta!} \leq 1 \right\}$$

is weakly tractable, ie.

$$\lim_{\varepsilon^{-1}+d \to \infty} \frac{\ln n(\varepsilon, d)}{\varepsilon^{-1} + d} = 0.$$

. . . similar results for other norms, domains, . . . based on Taylor's theorem or other "constructive" techniques

. . . A. Hinrichs, E. Novak, M. Ullrich, H. Woźniakowski, P. Kritzer, F. Pillichshammer, J. Dick, . . .

# Sparse additive models: Notation

H. Tyagi and J.V., *Learning general sparse additive models from point queries in high dimensions*, to appear in Constr. Approx.

$r_0 = 1$, $f : [-1,1]^d \to \mathbb{R}$

$$f(x) = \sum_{j \in \mathcal{S}_1} \phi_j(x_j),$$

where $x = (x_1, \ldots, x_d)$ and $\mathcal{S}_1 \subset \{1, \ldots, d\}$ with $|\mathcal{S}_1| \ll d$

# Sparse additive models: Notation

H. Tyagi and J.V., *Learning general sparse additive models from point queries in high dimensions*, to appear in Constr. Approx.

$r_0 = 1$, $f : [-1,1]^d \to \mathbb{R}$

$$f(x) = \sum_{j \in \mathcal{S}_1} \phi_j(x_j),$$

where $x = (x_1, \ldots, x_d)$ and $\mathcal{S}_1 \subset \{1, \ldots, d\}$ with $|\mathcal{S}_1| \ll d$

$r_0 = 2$, $f : [-1,1]^d \to \mathbb{R}$

$$f(x) = \sum_{j \in \mathcal{S}_1} \phi_j(x_j) + \sum_{(j_1,j_2) \in \mathcal{S}_2} \phi_{(j_1,j_2)}(x_{j_1}, x_{j_2}),$$

with $\mathcal{S}_2 \subset \binom{\{1,\ldots,d\}}{2}$ and $|\mathcal{S}_2| \ll \binom{d}{2}$

# Sparse additive models: Notation

General $1 \leq r_0 \leq d$, $f : [-1,1]^d \to \mathbb{R}$

$$f(x) = \sum_{j \in \mathcal{S}_1} \phi_j(x_j) + \sum_{(j_1,j_2) \in \mathcal{S}_2} \phi_{(j_1,j_2)}(x_{j_1}, x_{j_2}) + \cdots + \sum_{(j_1,\ldots,j_{r_0}) \subset \mathcal{S}_{r_0}} \phi_{(j_1,\ldots,j_{r_0})}(x_{j_1}, \ldots, x_{j_{r_0}}).$$

- $\mathcal{S}_{r_0} \subset \binom{\{1,\ldots,d\}}{r_0}$
- $|\mathcal{S}_{r_0}| \ll \binom{d}{r_0}$

# Anchored-ANOVA

The above decompositions are not unique: In

$$f(x) = \sum_{j \in \mathcal{S}_1} \phi_j(x_j),$$

one may add constants to $\phi_j$, which sum up to zero, etc.

We assume that $f$ is given in the Anchored-ANOVA decomposition:

$$\begin{aligned} f(x) &= f_\emptyset + \sum_{i=1}^{d} f_i(x_i) + \sum_{i=1}^{d-1} \sum_{j=i+1}^{d} f_{i,j}(x_i, x_j) + \cdots + f_{1,\ldots,d}(x_1, \ldots, x_d) \\ &= \sum_{U \subseteq [d]} f_U(x_U), \end{aligned}$$

$f_U(x_U) = 0$ if $x_j = 0$ for some $j \in U$

# Univariate case: Setting

$r_0 = 1$ :

$$f(x) = \mu + \sum_{j \in \mathcal{S}_1} \phi_j(x_j).$$

1. *Smoothness.* $\phi_j$'s are smooth: $|\phi_j(x) - \phi_j(y)| \leq L|x - y|^\alpha$
2. *Identifiability.* There is an $x_j^* \in [-1, 1]$ with $|\phi_j(x_j^*)| > D_1$.

# Univariate case: Sampling scheme

For $m \in \mathbb{N}$ and every $x^l = \left(\frac{l}{m}, \ldots, \frac{l}{m}\right)$, $l = -m, \ldots, m$:

# Univariate case: Sampling scheme

For $m \in \mathbb{N}$ and every $x^l = \left(\frac{l}{m}, \ldots, \frac{l}{m}\right)$, $l = -m, \ldots, m$:

- Generate $\beta \in \{-1, +1\}^d$ at random
- $\beta$ splits $\{1, \ldots, d\}$ into two disjoint subsets:

$$B_- := \{j : \beta_j = -1\} \quad \text{and} \quad B_+ := \{j : \beta_j = +1\}$$

# Univariate case: Sampling scheme

For $m \in \mathbb{N}$ and every $x^l = \left(\frac{l}{m}, \ldots, \frac{l}{m}\right)$, $l = -m, \ldots, m$:

- Generate $\beta \in \{-1, +1\}^d$ at random
- $\beta$ splits $\{1, \ldots, d\}$ into two disjoint subsets:

$$B_- := \{j : \beta_j = -1\} \quad \text{and} \quad B_+ := \{j : \beta_j = +1\}$$

- Define $x^{+,l}$, $x^{-,l}$

$$x_i^{+,l} = \begin{cases} x_i^l \ ; & \beta_i = +1, \\ 0 \ ; & \beta_i = -1 \end{cases} \quad \text{and} \quad x_i^{-,l} = \begin{cases} 0 \ ; & \beta_i = +1, \\ x_i^l \ ; & \beta_i = -1, \end{cases}$$

i.e.

$$x^{+,l} = x^l \chi_{B_+} = x^l|_{B_+} \quad \text{and} \quad x^{-,l} = x^l \chi_{B_-} = x^l|_{B_-}$$

# Univariate case: Sampling scheme

- Observe that

$$\begin{aligned} f(x^{+,l}) - f(x^{-,l}) &= \sum_{j\in\mathcal{S}_1} \phi_j(x_j^{+,l}) - \sum_{j\in\mathcal{S}_1} \phi_j(x_j^{-,l}) \\ &= \sum_{j\in\mathcal{S}_1\cap B_+} \phi_j(x_j^{+,l}) - \sum_{j\in\mathcal{S}_1\cap B_-} \phi_j(x_j^{-,l}) \\ &= \sum_{j\in\mathcal{S}_1} \beta_j \phi_j(x_j^l) = \langle \beta, z^l \rangle \end{aligned}$$

  is a compressed sensing measurement of the $\mathcal{S}_1$-sparse vector $z_j^l := \phi_j(x_j^l)$, $j = 1, \ldots, d$.
- If the sampling of $f$ is noisy, we get noisy CS-measurements.

# Univariate case: The algorithm

- For each $x^l = \left(\frac{l}{m}, \ldots, \frac{l}{m}\right)$, generate $\beta^1, \ldots, \beta^n \in \{-1, +1\}^d$
- Define $x^{+,l}, x^{-,l}$, sample $y_j = \langle \beta^j, z^l \rangle + \eta_j$, $1 \leq j \leq n$
- Recover a CS-solution $\widehat{z^l}$
- Update $\widehat{\mathcal{S}_1} := \widehat{\mathcal{S}_1} \cup \{p : |\widehat{z_p^l}| \geq \varepsilon\}$ for some threshold $\varepsilon > 0$

Recovery guarantee: For $m \geq (3L/D_1)^{1/\alpha}$, $\varepsilon < D_1/3$ and $n \geq c|\mathcal{S}_1| \log(d)$, $\mathcal{S}_1$ is recovered exactly with high probability. The number of sampling points is $2(2m+1)n$.

# Bivariate case: Setting

$r_0 = 2$ :

$$f(x) = \mu + \sum_{j \in \mathcal{S}_1} \phi_j(x_j) + \sum_{(j_1,j_2) \in \mathcal{S}_2} \phi_{(j_1,j_2)}(x_{j_1}, x_{j_2}).$$

1. *Smoothness:* $|\phi_{(j_1,j_2)}(x_1, x_2) - \phi_{(j_1,j_2)}(y_1, y_2)| \leq L\|x - y\|_2^{\alpha}$
2. *Identifiability.* There is an $(x_1, x_2) \in [-1, 1]^2$ with $|\phi_{(j_1,j_2)}(x_1, x_2)| > D_2$.

# Bivariate case: Sampling scheme

For $l_1, l_2 \in \{-m, \ldots, m\}$ and disj. decomp. $\{1, \ldots, d\} = \mathcal{A}_1 \cup \mathcal{A}_2$:

$$x_j^{(l_1,l_2)} = \begin{cases} l_1/m, & \text{if } j \in \mathcal{A}_1, \\ l_2/m, & \text{if } j \in \mathcal{A}_2, \end{cases}$$

# Bivariate case: Sampling scheme

For $l_1, l_2 \in \{-m, \ldots, m\}$ and disj. decomp. $\{1, \ldots, d\} = \mathcal{A}_1 \cup \mathcal{A}_2$:

$$x_j^{(l_1,l_2)} = \begin{cases} l_1/m, & \text{if } j \in \mathcal{A}_1, \\ l_2/m, & \text{if } j \in \mathcal{A}_2, \end{cases}$$

Again $\beta \in \{-1, +1\}^d$, $B_- := \{j : \beta_j = -1\}$, $B_+ := \{j : \beta_j = +1\}$

# Bivariate case: Sampling scheme

For $l_1, l_2 \in \{-m, \ldots, m\}$ and disj. decomp. $\{1, \ldots, d\} = \mathcal{A}_1 \cup \mathcal{A}_2$:

$$x_j^{(l_1,l_2)} = \begin{cases} l_1/m, & \text{if } j \in \mathcal{A}_1, \\ l_2/m, & \text{if } j \in \mathcal{A}_2, \end{cases}$$

Again $\beta \in \{-1, +1\}^d$, $B_- := \{j : \beta_j = -1\}$, $B_+ := \{j : \beta_j = +1\}$

Generate *four* points:

$$x_{1,j}^{(l_1,l_2)} = x_j^{(l_1,l_2)} \quad \text{if} \begin{cases} \beta_j = 1 \text{ and } j \in \mathcal{A}_1 \text{ or} \\ \beta_j = 1 \text{ and } j \in \mathcal{A}_2, \end{cases}$$

$$x_{2,j}^{(l_1,l_2)} = x_j^{(l_1,l_2)} \quad \text{if} \begin{cases} \beta_j = -1 \text{ and } j \in \mathcal{A}_1 \text{ or} \\ \beta_j = 1 \text{ and } j \in \mathcal{A}_2, \end{cases}$$

. . . same for $x_{3,j}^{(l_1,l_2)}$ and $x_{4,j}^{(l_1,l_2)}$

# Bivariate case: Sampling scheme

Sampling scheme:

$$f(x_1^{(l_1,l_2)}) - f(x_2^{(l_1,l_2)}) - f(x_3^{(l_1,l_2)}) + f(x_4^{(l_1,l_2)}) = \sum_{(j_1,j_2)\in\mathcal{A}\cap\mathcal{S}_2} \beta_{j_1}\beta_{j_2}\phi_{(j_1,j_2)}(x_{j_1}^{(l_1,l_2)}, x_{j_2}^{(l_1,l_2)}),$$

where $\mathcal{A} = \{(j_1, j_2) : j_1 \text{ and } j_2 \text{ are in different } \mathcal{A}_1, \mathcal{A}_2\}$

We get bilinear CS-measurements of (at most $|\mathcal{S}_2|$-sparse) vector

$$\left(z^{(l_1,l_2)}\right)_{(j_1,j_2)} := \phi_{(j_1,j_2)}(x_{j_1}^{(l_1,l_2)}, x_{j_2}^{(l_1,l_2)}), \quad (j_1,j_2) \in \binom{\{1,\ldots,d\}}{2}$$

# Bivariate case: The algorithm

- Disjoint decompositions $\{1, \ldots, d\} = \mathcal{A}_1 \cup \mathcal{A}_2$ are generated by hash-functions
- A family $\mathcal{H} = \{h_1, \ldots, h_N : h_j : \{1, \ldots, d\} \to \{1, 2\}\}$
- $\mathcal{A}_1^j = \{l : h_j(l) = 1\} = h_j^{-1}(\{1\})$,
  $\mathcal{A}_2^j = \{l : h_j(l) = 2\} = h_j^{-1}(\{2\})$
- for every $(j_1, j_2)$ there is $h \in \mathcal{H}$, which is injection on $\{j_1, j_2\}$
- Hash-families are known to exist with only $\Omega(\log(d))$ elements
- Bilinear CS-measurements can be handled via Hanson-Wright inequality on tail probabilities for quadratic forms of independent random variables
- Recovery guarantees for number of sampling points *linear* in $|\mathcal{S}_2|$ and logarithmic in $d$

# Multivariate case

- General $r_0 \geq 1$ is possible
- Hash-families of $h : \{1, \ldots, d\} \to \{1, \ldots, r_0\}$
- Sampling scheme uses $2^{r_0}$ sampling points combined with $\pm 1$
- Recovery guarantees quadratic in $|\mathcal{S}_{r_0}|$ and logarithmic in $d$

# Neural networks

Motivated by biological research on human brain and neurons
W. McCulloch, W. Pitts (1943); M. Minsky, S. Papert (1969)
**Artificial Neuron:**
. . . gets activated if a linear combination of its inputs grows over a certain threshold. . .

- Inputs $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$
- Weights $w = (w_1, \ldots, w_n) \in \mathbb{R}^n$
- Comparing $\langle w, x \rangle$ with a threshold $b \in \mathbb{R}$
- Plugging the result into the "activation function" - jump (or smoothed jump) function $\sigma$

**Artificial neuron** is a function

$$x \to \sigma(\langle x, w \rangle - b),$$

where $\sigma : \mathbb{R} \to \mathbb{R}$ might be $\sigma(x) = \text{sign}(x)$ or $\sigma(x) = e^x/(1 + e^x)$, etc.

# Neural networks

Artificial neural network is a directed, acyclic graph of artificial neurons

- Input: $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$
- First layer of neurons: $y_1 = \sigma(\langle x, w_1^1 \rangle - b_1^1), \ldots, y_{n_1} = \sigma(\langle x, w_{n_1}^1 \rangle - b_{n_1}^1)$
- The outputs $y = (y_1, \ldots, y_{n_1})$ become inputs for the next layer ...; last layer outputs $y \in \mathbb{R}$
- "Deep Learning" relies on an artificial neural network with $\sim 100 - 1000$ layers
- Training the network: given inputs $x^1, \ldots, x^N$ and outputs $y^1, \ldots, y^N$ and optimize over weights $w$'s and $b$'s
- Non-convex minimization over a huge space...???

M. Fornasier, J.V., I. Daubechies, *Identification of shallow neural networks by fewest samples*, submitted: **Sums of ridge functions**

Recovery of

$$f(x) = \sum_{j=1}^{k} g_j(\langle a_j, x \rangle)$$

- We would like to identify $a_1, \dots, a_k$, then $g_1, \dots, g_k$
- **Step 1.:** Sampling of

$$\nabla f(x) = \sum_{j=1}^{k} g_j'(\langle a_j, x \rangle) a_j$$

  at different points gives elements of $\mathrm{span}\{a_1, \dots, a_k\} \subset \mathbb{R}^d$
- Afterwards, we can reduce the dimension to $d = k$

  ... one $k$-dimensional problem...

Recovery of individual $a_i$'s for $d = k$?

- **Step 2.:** Second order derivatives:

$$\nabla^2 f(x) = \sum_{j=1}^{k} g_j''(\langle a_j, x \rangle) a_j \otimes a_j$$

- We can recover $\tilde{L}$ - an approximation of

$$L = \text{span}\{a_i \otimes a_i, i = 1, \ldots, k\} \subset \mathbb{R}^{k \times k}$$

- **Step 3.:** We try to find $a_i \otimes a_i$ in $\tilde{L}$
- We look for matrices in $\tilde{L}$ with the smallest rank
- We analyze the non-convex problem

$$\arg\max \ \|M\|, \quad \text{s.t.} \quad M \in \tilde{L}, \|M\|_F \leq 1$$

- Every algorithm, which is able to find $a_1 \otimes a_1$ can also find $a_j \otimes a_j$, $j = 2, \ldots, k$, hence **it must be non-convex**

# Thank you for your attention!