Robust and efficient identification of neural networks

Sampling recovery and related problems
Lomonosov Moscow State University

May 2021

joint work with M. Fornasier (TU Munich)
and I. Daubechies (Duke)

Jan Vybíral
Czech Technical University
Prague, Czech Republic

Outline

- Approximation of multivariate functions
- Structural assumptions: active variables, SPAM's, ...
- Ridge functions - single neurons:
  - Recovery algorithms
  - Lower bounds
- Sums of ridge functions - one layer neural networks:
  - Non-linear optimization
  - Whitening
  - ...

Classical task of approximation/sampling:
Given a function f with some properties (i.e. from some set) and few its function values $y_1 = f(x_1), \dots, y_n = f(x_n)$ generate a function g, which is close to f in some sense.

Typically, the error (i.e. the distance of f and g) decays with n. 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, Cohen, Kruglyak, Heinrich, Novak, Triebel, Sickel, Ullrich and many many others ...

Curse of dimension

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

Example: Approximation of smooth functions
Let $\mathcal{F}_d := \{f : [0,1]^d \to \mathbb{R}, \|D^\alpha f\|_\infty \le 1, \alpha \in \mathbb{N}_0^d\}$
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 $n \le 2^{\lfloor d/2 \rfloor} - 1$
. . . curse of dimension!
. . . the number of sampling points must grow exponentially in d

Structural assumptions

- **Active variables:**

R. DeVore, G. Petrova, and P. Wojtaszczyk: Approximation of functions of few variables in high dimensions, Constr. Appr. 2011:

$$f(x_1,\dots,x_d) := g(x_{i_1},\dots,x_{i_\ell}),\quad \ell \ll N.$$

1-Lipschitz function f can be recovered uniformly with accuracy ε from $C(\ell)\varepsilon^{-\ell}\log_2 d$ sampling points.

Use of low-rank matrix recovery:
H. Tyagi, V. Cevher, Learning non-parametric basis independent models from point queries via low-rank methods, ACHA 2014

Revisited also in
K. Schnass, J.V., Compressed learning of high-dimensional sparse functions, Proceedings of ICASSP '11
S. Foucart, Sampling schemes and recovery algorithms for functions of few coordinate variables, J. Compl. 2020

Structural assumptions

• **Sparse additive models:** H. Tyagi and J. Vybiral: Learning non-smooth sparse additive models from point queries in high dimensions, Constr. Appr. 2019:

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

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

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 $|\mathcal{S}_2| \ll \binom{d}{2}$

Ridge functions

Ridge functions

Let $g : \mathbb{R} \to \mathbb{R}$ and $a \in \mathbb{R}^d \setminus \{0\}$.
Ridge function with *ridge profile* g and *ridge vector* a is the function

$$f(x) := g(\langle a, x \rangle).$$

Constant along the hyperplane $a^\perp = \{y \in \mathbb{R}^d : \langle y, a \rangle = 0\}$ and its translates.

More general, if $g : \mathbb{R}^k \to \mathbb{R}$ and $A \in \mathbb{R}^{k \times d}$ with $k \ll d$ then

$$f(x) := g(Ax)$$

is a $k-$*ridge function*.

Ridge functions in approximation theory

Approximation of a function by functions from the dictionary

$$D_{\mathrm{ridge}} = \{\varrho(\langle k, x\rangle - b) : k \in \mathbb{R}^d, b \in \mathbb{R}\}$$

- Fundamentality
- Greedy algorithms

Lin & Pinkus, Fundamentality of ridge functions, J. Approx. Theory 75 (1993), no. 3, 295–311

Cybenko, Approximation by superpositions of a sigmoidal function, Math. Control Signals Systems 2 (1989), 303–314

Leshno, Lin, Pinkus & Schocken, Multilayer feedforward networks with a nonpolynomial activation function can approximate any function, Neural Networks 6 (1993), 861–867

Ridge functions: Approximation algorithms

$k=1$: $f(x)=g(\langle a,x\rangle)$, $\quad \|a\|_2=1$, $\quad g$ smooth

Approximation has two parts: approximation of g and of a

Recovery of a - from $\nabla f(x)$:

$$\nabla f(x)=g'(\langle a,x\rangle)a, \nabla f(0)=g'(0)a.$$

After recovering a, the problem becomes essentially one-dimensional and one can use arbitrary sampling method to approximate g.

$g'(0)\neq 0\ldots g'(0)=1$

A.Cohen, I.Daubechies, R.DeVore, G.Kerkyacharian, D.Picard, '12
Capturing ridge functions in high dimensions from point queries

- $\boldsymbol{k = 1 : f(x) = g(\langle a, x\rangle)}$
- $f : [0,1]^d \to \mathbb{R}$
- $g \in C^s([0,1])$, $1 < s$
- $\|g\|_{C^s} \leq M_0$
- $\boldsymbol{\|a\|_{\ell_q^d} \leq M_1, 0 < q \leq 1}$
- $\boldsymbol{a \geq 0}$

Then

$$\|f - \hat{f}\|_\infty \leq CM_0 \left\{ L^{-s} + M_1 \left(\frac{1 + \log(d/L)}{L} \right)^{1/q-1} \right\}$$

using $3L + 2$ sampling points

- First sampling along the diagonal $\frac{i}{L}\mathbf{1} = \frac{i}{L}(1, \ldots, 1), i = 0, \ldots, L$:

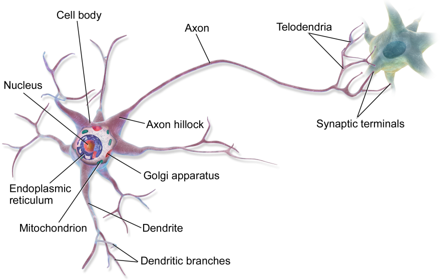
$$f\left(\frac{i}{L}\mathbf{1}\right) = g\left(\left\langle \frac{i}{L}\mathbf{1}, a\right\rangle\right) = g(i\|a\|_1/L)$$

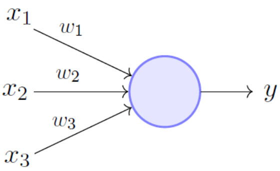
- Recovery of g on a grid of $[0, \|a\|_1]$
- Finding i_0 with largest $g((i_0 + 1)\|a\|_1/L) - g(i_0\|a\|_1/L)$
- Approximating $D_{\varphi_j} f(i_0/L \cdot \mathbf{1}) = g'(i_0\|a\|_1/L)\langle a, \varphi_j\rangle$ by first order differences
- Then recovery of a from $\langle a, \varphi_1\rangle, \ldots, \langle a, \varphi_m\rangle$ by methods of compressed sensing (CS)

Further results:

- M. Fornasier, K. Schnass, and J.V., Learning functions of few arbitrary linear parameters in high dimensions, Found. Comput. Math. (2012)
  $\rightsquigarrow$ Recovery algorithm for ridge functions on a ball
- A. Kolleck and J.V., On some aspects of approximation of ridge functions, J. Appr. Theory (2015)
  $\rightsquigarrow$ Ridge functions on cubes and other topics
- S. Mayer, T. Ullrich, and J.V., Entropy and sampling numbers of classes of ridge functions, Constr. Appr. (2015)
  $\rightsquigarrow$ Lower bounds, entropy numbers
- B.Doerr and S. Mayer, The recovery of ridge functions on the hypercube suffers from the curse of dimensionality, J. Compl. (2021)
  $\rightsquigarrow$ Ridge functions on cubes, lower bounds

Motivated by biological research on human brain and neurons
W. McCulloch, W. Pitts (1943); M. Minsky, S. Papert (1969)





Perceptron Model (Minsky-Papert in 1969)

Artificial Neuron

...gets activated if a linear combination of its inputs grows over a certain threshold...

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

Artificial neuron is a function

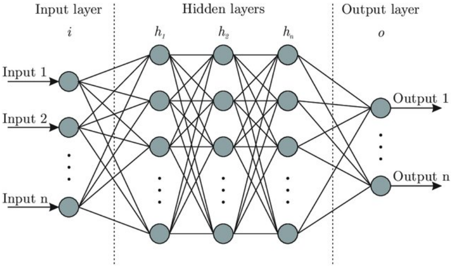
$$x \to \sigma(\langle x, w \rangle - b),$$

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

Artificial 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 $y \in \mathbb{R}$



M. Fornasier, J.V., I. Daubechies,
Robust and resource efficient identification of shallow neural networks by fewest samples
to appear in IMA: Information and Inference

Recovery of

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

- one layer of a neural network

- Special case of $f(x) = g(Ax)$, but we would like to have m onedimensional problems, and not one m-dimensional
- We would like to identify $a_1, \dots, a_m$, then $g_1, \dots, g_m$

- **Step 1.:** Sampling of

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

at different points gives elements of

$$A = \operatorname{span}\{a_1, \dots, a_m\} \subset \mathbb{R}^d$$

- We sample (only) differences and obtain an approximation $\tilde{A} \sim A$, upper bound on $\|P_A - P_{\tilde{A}}\|_F$
- $\bar{A}$ - matrix, columns are a basis of $\tilde{A}$
- We consider

$$\tilde{f}(y) = \sum_{i=1}^{m} g_i(\langle \bar{A}^T a_i, y\rangle) = \sum_{i=1}^{m} g_i(\langle a_i, \bar{A}y\rangle)$$

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

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

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

- Put

$$\mathcal{A} = \operatorname{span}\{a_i \otimes a_i : i = 1, \dots, m\} \subset \mathbb{R}^{m \times m}$$

- We can recover $\tilde{\mathcal{A}} \sim \mathcal{A}$ - an approximation of $\mathcal{A}$

- **Step 3.:** We try to find matrices in $\tilde{\mathcal{A}}$, which are close to $a_i \otimes a_i \in \mathcal{A}$

- We look for matrices in $\tilde{\mathcal{A}}$ with the "smallest" rank

- We analyze the non-convex problem

$$(\star) \qquad \arg\max \ \|M\|, \quad \text{s.t.} \quad M \in \tilde{\mathcal{A}}, \|M\|_F \leq 1$$

- Every algorithm, which is able to find an approximation of $a_1 \otimes a_1$ can also land close to $a_j \otimes a_j$, $j = 2, \ldots, m$, hence **it must be non-convex**

Analysis of $(\star)$:
- Let M be a local maximizer of $(\star)$
- Eigenvalues $\lambda_1, \ldots, \lambda_m$, eigenvectors $u_1, \ldots, u_m$
- Then

$$u_j^T X u_j = \lambda_j \langle X, M \rangle \quad \text{for every } X \in \tilde{\mathcal{A}}$$

and every j with $|\lambda_j| = \|M\|$
- If $a_1, \ldots, a_m$ are nearly orthonormal, then $|\lambda_1| = \|M\|$ is unique and

$$2 \sum_{k=2}^{m} \frac{(u_1^T X u_k)^2}{|\lambda_1 - \lambda_k|} \leq \|M\| \cdot \|X - \langle X, M \rangle_F M\|_F^2 \quad \text{for all } X \in \tilde{\mathcal{A}}$$

second Hadamard variation formula

The Algorithm leads to $\hat{a}$ with $\|\hat{a} - a_{j_0}\|_2$ small.

Whitening: If a_i's are not orthonormal, we consider (any) positive definite

$$G = \sum_{i=1}^{m} \xi_i a_i \otimes a_i \in \mathcal{A}$$

and its singular value decomposition

$$G = UDU^T,$$

then $W := D^{-1/2} U^T$ is the so-called *whitening matrix* and $\{\sqrt{\xi_i} W a_i : i = 1, \ldots, m\}$ is an orthonormal basis.

Passive sampling: sampling points preselected at random from a distribution with known density

Identification of g_i's:
- $(\hat{a}_j)_{j=1}^m$ an approximation of $(a_j)_{j=1}^m$
- $(\hat{b}_j)_{j=1}^m$ the dual basis to $(\hat{a}_j)_{j=1}^m$
- $\hat{g}_j(t) := f(t\hat{b}_j)$.

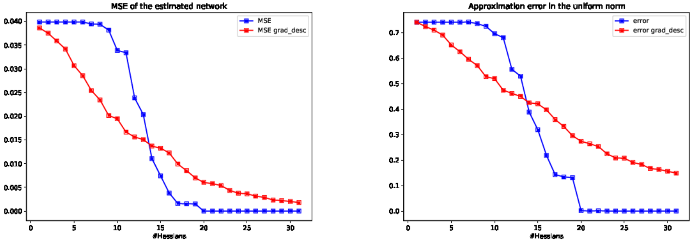


Figure: Average approximation error of 10 random networks with $m = 20, \varepsilon = 1$ in terms of MSE (left), uniform norm (right). The errors were measured over 10^5 datapoints generated uniform at random on the ball B_1^d.

Thank you for your attention!