Function approximation with one-bit Bernstein and neural networks

Weilin Li

Courant Institute of Mathematical Sciences

weilinli@cims.nyu.edu

Faraway Fourier Talks
December 20, 2021

References

Figure: Sinan Güntürk

References:

- Approximation with one-bit polynomials in Bernstein form.
  arXiv:2112.09183
- Approximation of functions with one-bit neural networks.
  arXiv:2112.09181

Can also be found on my webpage:

weilinli@cims.nyu.edu

Overview

Table of Contents

Main question

Can we approximate any reasonable function with a coarsely quantized neural network?

Main question

Can we approximate any reasonable function with a coarsely quantized neural network?

- A feed forward neural network is a function represented as a composition,

$$x \mapsto \rho_L(A_L(\cdots \rho_1(A_1 x + b_1)) + b_L),$$

where each weight matrix A_ℓ and bias vector b_ℓ are real.

Main question

Can we approximate any reasonable function with a coarsely quantized neural network?

- A feed forward neural network is a function represented as a composition,

$$x \mapsto \rho_L(A_L(\cdots \rho_1(A_1 x + b_1)) + b_L),$$

  where each weight matrix A_ℓ and bias vector b_ℓ are real.

- A quantized NN is one where all non-zero parameters (weights and biases) must be selected from a set of discrete values ("alphabet").

  For example, a "high-resolution" arithmetic progression

$$\mathcal{A} := \{-M\delta, (-M+1)\delta, \ldots, M\delta\}.$$

Main question

Can we approximate any reasonable function with a coarsely quantized neural network?

- A feed forward neural network is a function represented as a composition,

$$x \mapsto \rho_L(A_L(\cdots \rho_1(A_1 x + b_1)) + b_L),$$

  where each weight matrix A_ℓ and bias vector b_ℓ are real.

- A quantized NN is one where all non-zero parameters (weights and biases) must be selected from a set of discrete values ("alphabet").

  For example, a "high-resolution" arithmetic progression

$$\mathcal{A} := \{-M\delta, (-M+1)\delta, \ldots, M\delta\}.$$

- A coarsely quantized NN uses an alphabet with a small collection of values.

  For example, a one-bit alphabet,

$$\mathcal{A}_1 := \{\pm 1\}.$$

Motivation 1: Practical issue

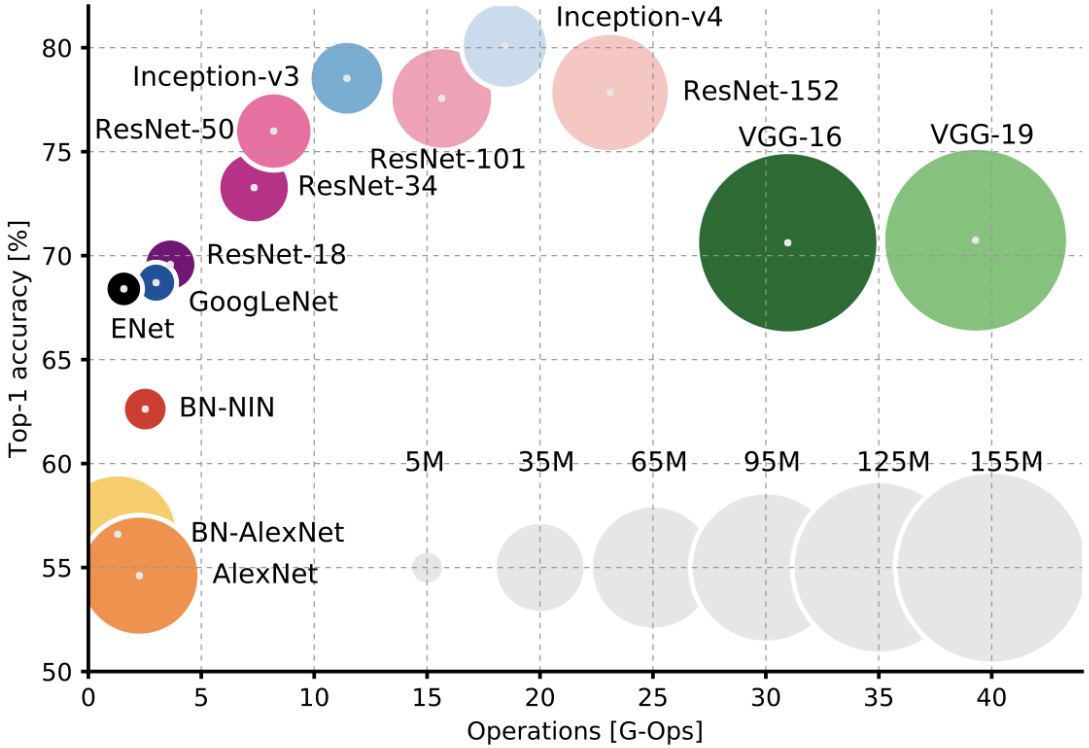


Figure: Memory footprint of some networks [Canziani, Paszke, Culurciello 16]

How to transfer huge networks onto low power devices (phones, cars, tablets, etc.)?

Survey article [Yunhui Guo 18]

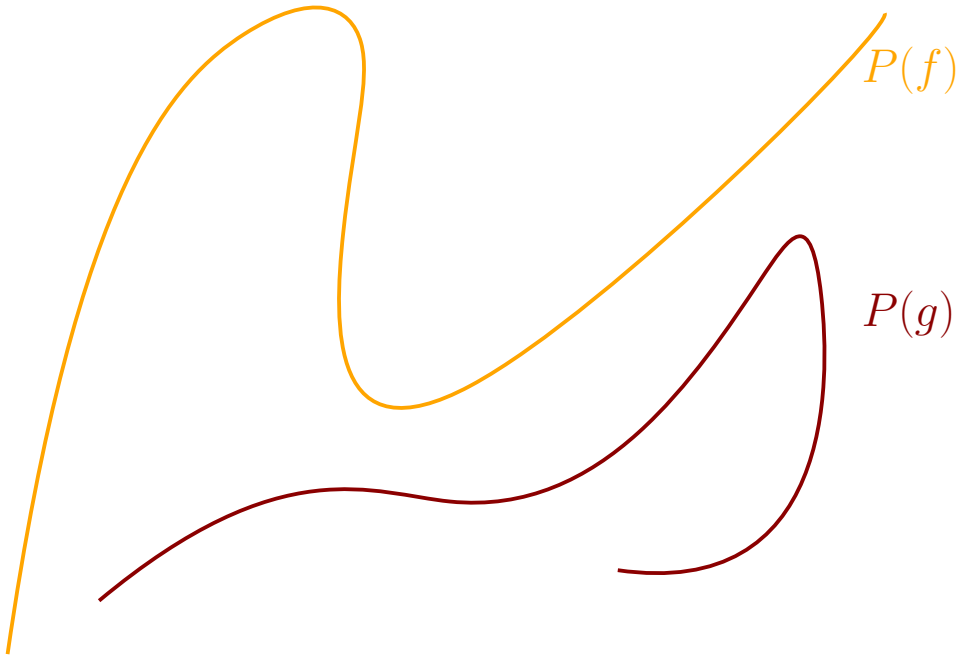
$$\begin{aligned} \text{Quantize the network} &\longrightarrow \text{low-bit representation} \\ &\longrightarrow \text{cheaper storage cost and faster forward pass} \\ &\longrightarrow \text{less energy consumption} \end{aligned}$$

Use of sophisticated quantization techniques: [Lybrand, Saab 21], [Ashbrock, Powell 21]

Motivation 2: Over-parameterization

For each f, look at the parameters that represent f,

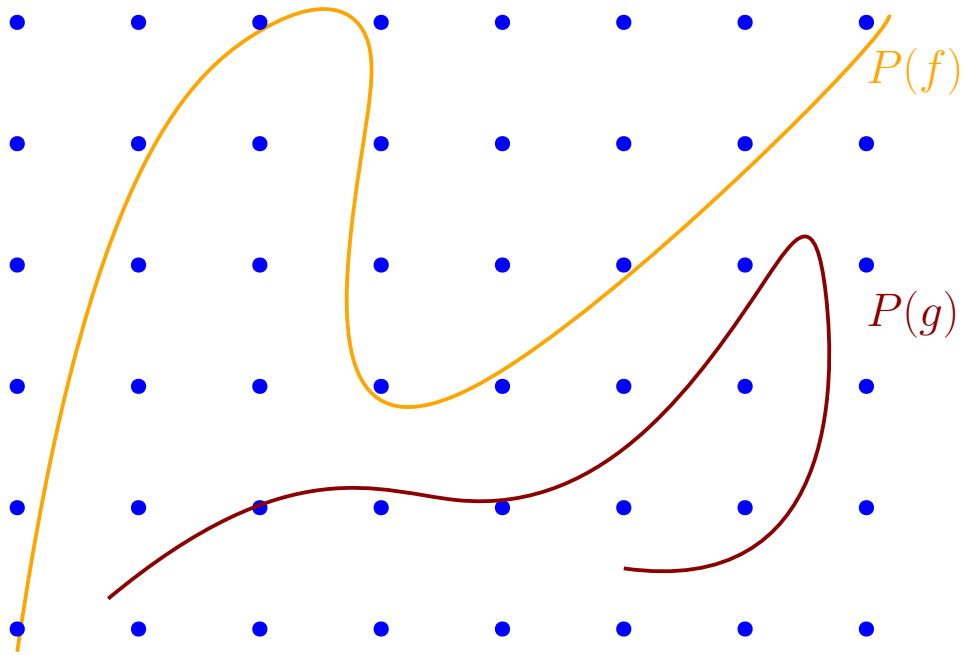
$$P(f) = \{\theta \in \mathbb{R}^D : NN_\theta = f\}$$



Motivation 2: Over-parameterization

For each f, look at the parameters that represent

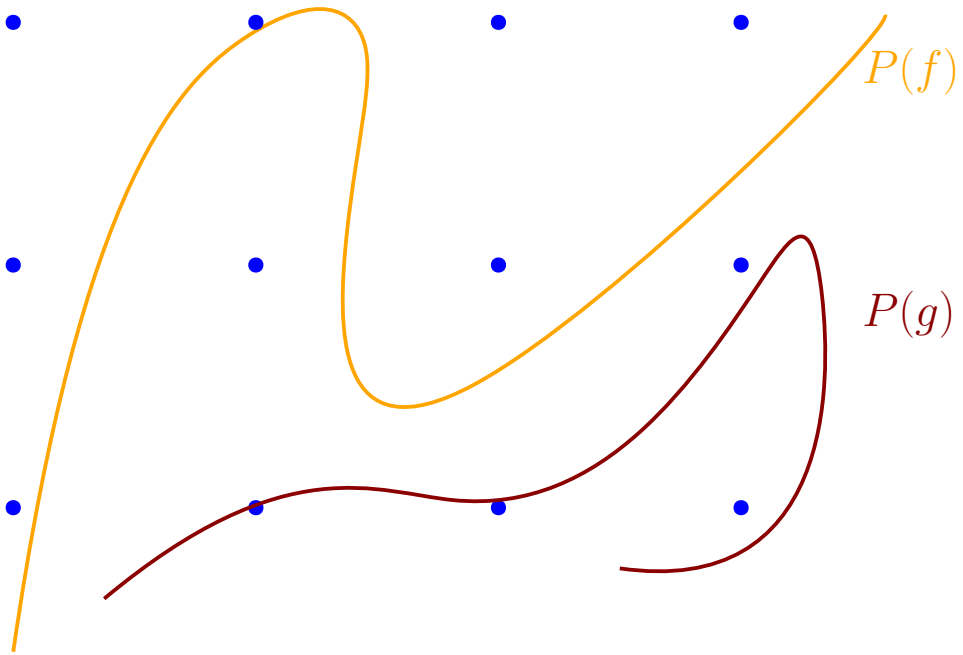
$$P(f) = \{\theta \in \mathbb{R}^D : NN_\theta = f\}$$



Motivation 2: Over-parameterization

For each f, look at the parameters that represent

$$P(f) = \{\theta \in \mathbb{R}^D : NN_\theta = f\}$$



A more technical viewpoint from approximation theory

What makes the main question difficult to answer?

Numerous results for approximation by neural networks (incomplete list of course):

[Cybenko 89], [Barron 94], [Yarotsky 17], [Shaham, Cloninger, Coifman 18], [Bolcskei, Grohs, Kutyniok, Petersen 19], [Daubechies, DeVore, Foucart, Hanin 21], [Lu, Shen, Yang, Zhang 21]

A more technical viewpoint from approximation theory

What makes the main question difficult to answer?

Numerous results for approximation by neural networks (incomplete list of course):

[Cybenko 89], [Barron 94], [Yarotsky 17], [Shaham, Cloninger, Coifman 18], [Bolcskei, Grohs, Kutyniok, Petersen 19], [Daubechies, DeVore, Foucart, Hanin 21], [Lu, Shen, Yang, Zhang 21]

Main strategy of the above approximation strategies:
For fixed f,

$$f \approx \sum_k a_k \phi_k$$

- $\{\phi_k\}_k$ are implementable by a neural network with carefully chosen parameters, e.g., local polynomials, ridge functions, wavelets, Fourier
- $\{a_k\}_k \subseteq \mathbb{R}$ depend on f, e.g., basis or frame coefficients

A more technical viewpoint from approximation theory

What makes the main question difficult to answer?

Numerous results for approximation by neural networks (incomplete list of course):

[Cybenko 89], [Barron 94], [Yarotsky 17], [Shaham, Cloninger, Coifman 18], [Bolcskei, Grohs, Kutyniok, Petersen 19], [Daubechies, DeVore, Foucart, Hanin 21], [Lu, Shen, Yang, Zhang 21]

Main strategy of the above approximation strategies:
For fixed f,

$$f \approx \sum_k a_k \phi_k$$

- $\{\phi_k\}_k$ are implementable by a neural network with carefully chosen parameters, e.g., local polynomials, ridge functions, wavelets, Fourier
- $\{a_k\}_k \subseteq \mathbb{R}$ depend on f, e.g., basis or frame coefficients

Approach needs to be modified when we consider coarsely quantized networks!

Table of Contents

Our wish list

Find a suitable set of functions $\{\phi_k\}_k$ such that:

- **Approximation.** Linear combinations of $\{\phi_k\}_k$ can efficiently approximate large function classes.
- **Implementation.** The $\{\phi_k\}_k$ can be implemented by a coarsely quantized neural network.

Our wish list

Find a suitable set of functions $\{\phi_k\}_k$ such that:

- **Approximation.** Linear combinations of $\{\phi_k\}_k$ can efficiently approximate large function classes.
- **Implementation.** The $\{\phi_k\}_k$ can be implemented by a coarsely quantized neural network.
- **Quantization.** Coefficients in the $\{\phi_k\}_k$ basis can be quantized in an effective way. For each $\{a_k\}_k$ we can find $\{\sigma_k\} \subseteq \mathcal{A}$ such that

$$\sum_k a_k \phi_k - \sum_k \sigma_k \phi_k \quad \text{is small.}$$

Our wish list

Find a suitable set of functions $\{\phi_k\}_k$ such that:

- **Approximation.** Linear combinations of $\{\phi_k\}_k$ can efficiently approximate large function classes.
- **Implementation.** The $\{\phi_k\}_k$ can be implemented by a coarsely quantized neural network.
- **Quantization.** Coefficients in the $\{\phi_k\}_k$ basis can be quantized in an effective way. For each $\{a_k\}_k$ we can find $\{\sigma_k\} \subseteq \mathcal{A}$ such that

$$\sum_k a_k \phi_k - \sum_k \sigma_k \phi_k \quad \text{is small.}$$

Three term decomposition:

$$f - f_{NN} = \underbrace{\left(f - \sum_k a_k \phi_k\right)}_{\text{approximation by } \phi \text{ error}} + \underbrace{\left(\sum_k a_k \phi_k - \sum_k \sigma_k \phi_k\right)}_{\text{quantization error}} + \underbrace{\left(\sum_k \sigma_k \phi_k - f_{NN}\right)}_{\text{implementation error}}.$$

Function approximation with integer Bernstein

Kantorovich: If $f\colon [0,1] \to \mathbb{R}$ is continuous and $f(0)$ and $f(1)$ are integers, then the function

$$B_n^*(f)(x) := \sum_{k=0}^{n} \left[f\left(\frac{k}{n}\right) \binom{n}{k} \right] x^k (1-x)^{n-k},$$

converges uniformly to f as $n \to \infty$. Here, $[t]$ is rounding t to the nearest integer.

Proof can be found in Chapter 2.4 of *Constructive Approximation: Advanced Problems* by Lorentz, v Golitschek, and Makovoz.

Function approximation with integer Bernstein

Kantorovich: If $f\colon [0,1] \to \mathbb{R}$ is continuous and $f(0)$ and $f(1)$ are integers, then the function

$$B_n^*(f)(x) := \sum_{k=0}^{n} \left[f\left(\frac{k}{n}\right) \binom{n}{k} \right] x^k (1-x)^{n-k},$$

converges uniformly to f as $n \to \infty$. Here, $[t]$ is rounding t to the nearest integer.

Proof can be found in Chapter 2.4 of *Constructive Approximation: Advanced Problems* by Lorentz, v Golitschek, and Makovoz.

- Implies that polynomials with integer coefficients are dense in the space of continuous functions.
- Main drawback is that if $\|f\|_\infty \leq 1$, then

$$\left| \left[f\left(\frac{k}{n}\right) \binom{n}{k} \right] \right| \quad \text{could be as big as} \quad 2^n,$$

so not helpful for coarse quantization.

One-bit Bernstein?

Bernstein polynomials of degree n: for each $0 \leq k \leq n$,

$$p_{n,k}(x) := \binom{n}{k} x^k (1-x)^{n-k}.$$

Berstein: For any $f \in C([0,1])$,

$$\|f - B_n(f)\| = \left\| f - \sum_{k=0}^{n} f\left(\frac{k}{n}\right) p_{n,k} \right\|_\infty \to 0.$$

One-bit Bernstein?

Bernstein polynomials of degree n: for each $0 \leq k \leq n$,

$$p_{n,k}(x) := \binom{n}{k} x^k (1-x)^{n-k}.$$

Berstein: For any $f \in C([0,1])$,

$$\|f - B_n(f)\| = \left\| f - \sum_{k=0}^{n} f\left(\frac{k}{n}\right) p_{n,k} \right\|_\infty \to 0.$$

Question: For any $f \in C([0,1])$ and $\epsilon > 0$, is there an integer n and $\{\sigma_k\}_{k=0}^n$ such that $\sigma_k \in \{\pm 1\}$ and

$$\left\| f - \sum_{k=0}^{n} \sigma_k p_{n,k} \right\|_\infty \leq \epsilon?$$

One-bit Bernstein?

Bernstein polynomials of degree n: for each $0 \leq k \leq n$,

$$p_{n,k}(x) := \binom{n}{k} x^k (1-x)^{n-k}.$$

Question: For any $f \in C([0,1])$ such that $\|f\|_\infty \leq 1$ and $\epsilon > 0$, is there an integer n and $\{\sigma_k\}_{k=0}^n$ such that $\sigma_k \in \{\pm 1\}$ and for all x away from the endpoints,

$$\left| f(x) - \sum_{k=0}^{n} \sigma_k p_{n,k}(x) \right| < \epsilon?$$

A first approximation result

Theorem

For any Lipschitz continuous f with $\|f\|_\infty \le 1$ and integer n, there exist $\{\sigma_k\}_{k=0}^n \subset \{\pm 1\}$, such that

$$\left| f(x) - \sum_{k=0}^{n} \sigma_k p_{n,k}(x) \right| \lesssim \frac{1 + |f|_{Lip}}{1 + \sqrt{nx(1-x)}}.$$

A first approximation result

Theorem

For any Lipschitz continuous f with $\|f\|_\infty \leq 1$ and integer n, there exist $\{\sigma_k\}_{k=0}^n \subset \{\pm 1\}$, such that

$$\left| f(x) - \sum_{k=0}^{n} \sigma_k p_{n,k}(x) \right| \lesssim \frac{1 + |f|_{Lip}}{1 + \sqrt{nx(1-x)}}.$$

- By density, can extend to continuous functions.
- In comparison to Kantorovich, our construction uses significantly fewer bits.
- Implies a L^p estimate: for all $p \in [1, 2)$,

$$\left\| f - \sum_{k=0}^{n} \sigma_k p_{n,k} \right\|_{L^p([0,1])} \lesssim \frac{1 + |f|_{Lip}}{\sqrt{n}}.$$

A constructive proof

The sequence of ± 1 are chosen through an algorithm called first order $\Sigma\Delta$.
Set $a_k = f(k/n)$ and we find σ such that it satisfies a finite difference equation

$$a_k - \sigma_k = (\Delta u)_k := u_k - u_{k-1}.$$

Can always be done with a $\|u\|_\infty \lesssim 1$ due to the assumption that $\|f\|_\infty \leq 1$. Explicitly,

$$u_k := u_{k-1} + a_k - \sigma_k, \quad \sigma_k := \text{sign}(a_k + u_{k-1}).$$

A constructive proof

The sequence of ± 1 are chosen through an algorithm called first order $\Sigma\Delta$.
Set $a_k = f(k/n)$ and we find σ such that it satisfies a finite difference equation

$$a_k - \sigma_k = (\Delta u)_k := u_k - u_{k-1}.$$

Can always be done with a $\|u\|_\infty \lesssim 1$ due to the assumption that $\|f\|_\infty \leq 1$. Explicitly,

$$u_k := u_{k-1} + a_k - \sigma_k, \quad \sigma_k := \text{sign}(a_k + u_{k-1}).$$

Bound the total error:

$$\begin{aligned}
&\left| f(x) - \sum_{k=0}^{n} \sigma_k p_{n,k}(x) \right| \\
&\leq \left| f(x) - \sum_{k=0}^{n} a_k p_{n,k}(x) \right| + \left| \sum_{k=0}^{n} (a_k - \sigma_k) p_{n,k}(x) \right| \\
&\lesssim \frac{|f|_{Lip}}{\sqrt{n}} + \left| \sum_{k=0}^{n} (\Delta u)_k \, p_{n,k}(x) \right|
\end{aligned}$$

Defining $p_{n,k+2} := 0$,

$$\begin{aligned}
\left| \sum_{k=0}^{n} (\Delta u)_k \, p_{n,k}(x) \right| &= \left| \sum_{k=0}^{n} u_k \left(p_{n,k}(x) - p_{n,k+1}(x) \right) \right| \\
&\leq \|u\|_\infty \sum_{k=0}^{n} |p_{n,k}(x) - p_{n,k+1}(x)| \quad \text{(Frame variation)} \\
&= \frac{\|u\|_\infty}{(n+1)x(1-x)} \sum_{k=0}^{n} |(k+1) - (n+1)x| \, p_{n+1,k+1}(x) \\
&= \frac{\|u\|_\infty}{(n+1)x(1-x)} \left(\sum_{k=0}^{n} |(k+1) - (n+1)x|^2 \, p_{n+1,k+1}(x) \right)^{1/2} \\
&\lesssim \frac{1}{\sqrt{nx(1-x)}}.
\end{aligned}$$

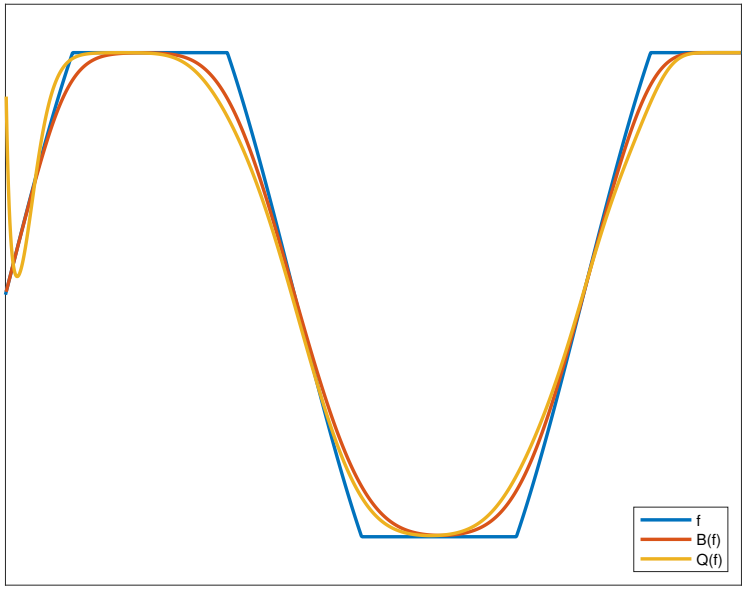
Frame variation 2 papers, [Benedetto, Powell, Yilmaz, 06]

Simple demonstration

Consider the continuous function,

$$f(x) = \max\left(\min\left(\frac{3}{2}\sin(8x), 1\right), -1\right).$$

Plot of $B_n(f)$ and constructed quantized approximation, for $n = 100$.



Coefficients of quantized approximation (where $-1 \leftrightarrow 0$):

$$10101111111111111111111111111111011110101001000000$$

$$00000000000000000000100010011011011111111111111$$

Table of Contents

Technical difficulties when extending to higher dimensions

The previous results give us hope that Bernstein polynomials satisfy our “wish list”, but there is still a main concern when applying these results to networks:

Technical difficulties when extending to higher dimensions

The previous results give us hope that Bernstein polynomials satisfy our "wish list", but there is still a main concern when applying these results to networks:

high dimensionality $\longrightarrow$ spaces of smooth functions.

Multivariate Bernstein of order n: For each $k \in \mathbb{N}^d$ with $0 \leq k_j \leq n$,

$$p_{n,k}(x) := p_{n,k_1}(x_1) \cdots p_{n,k_d}(x_d).$$

Technical difficulties when extending to higher dimensions

The previous results give us hope that Bernstein polynomials satisfy our "wish list", but there is still a main concern when applying these results to networks:

high dimensionality $\longrightarrow$ spaces of smooth functions.

Multivariate Bernstein of order n: For each $k \in \mathbb{N}^d$ with $0 \leq k_j \leq n$,

$$p_{n,k}(x) := p_{n,k_1}(x_1) \cdots p_{n,k_d}(x_d).$$

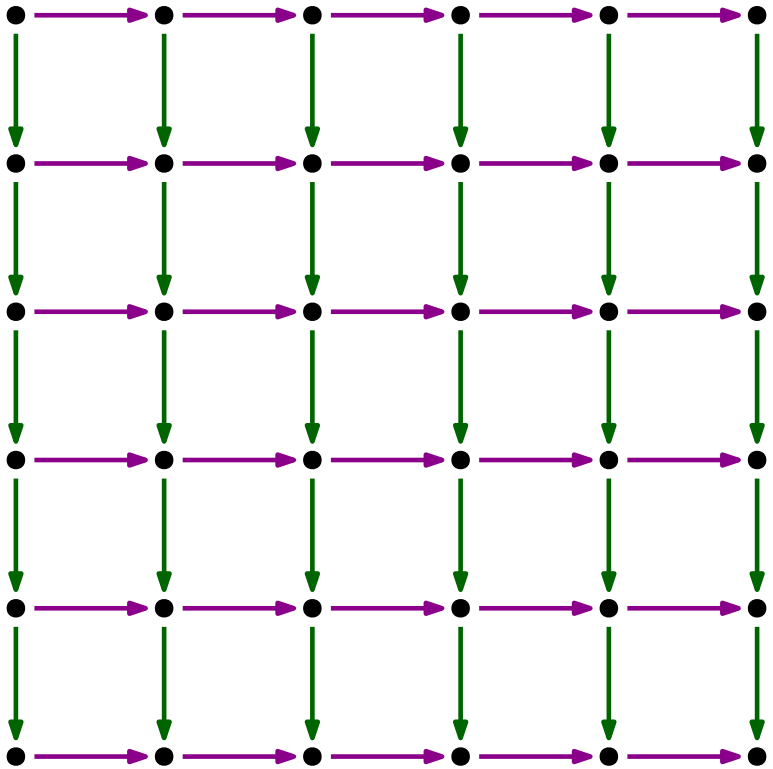
- High dimensionality is a major issue for $\Sigma\Delta$. No available $O(1)$-alphabet result for stable d dimensional $\Sigma\Delta$.
- Higher order smoothness is an issue do to this saturation result:
  If $f \in C^2([0,1])$, then

$$\lim_{n\to\infty} n(B_n(f)(x) - f(x)) = \frac{f''(x)x(1-x)}{2}.$$

  Bernstein polynomial of f is unable to exploit additional smoothness.

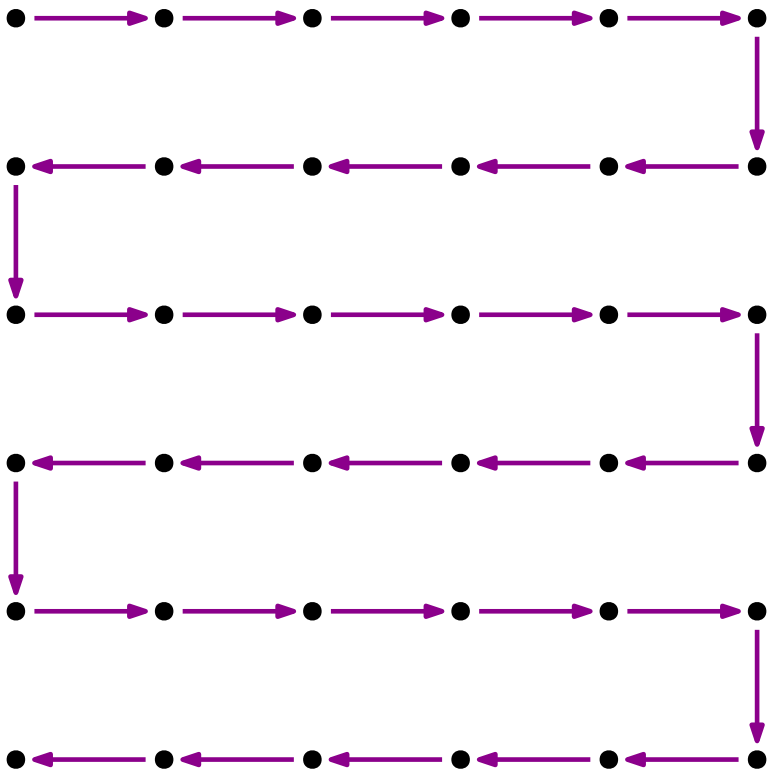
$\Sigma\Delta$ in higher dimensions

Say $d = 2$. How to order $\{p_{n,k}\}_{k_1=0,k_2=0}^{n}$?



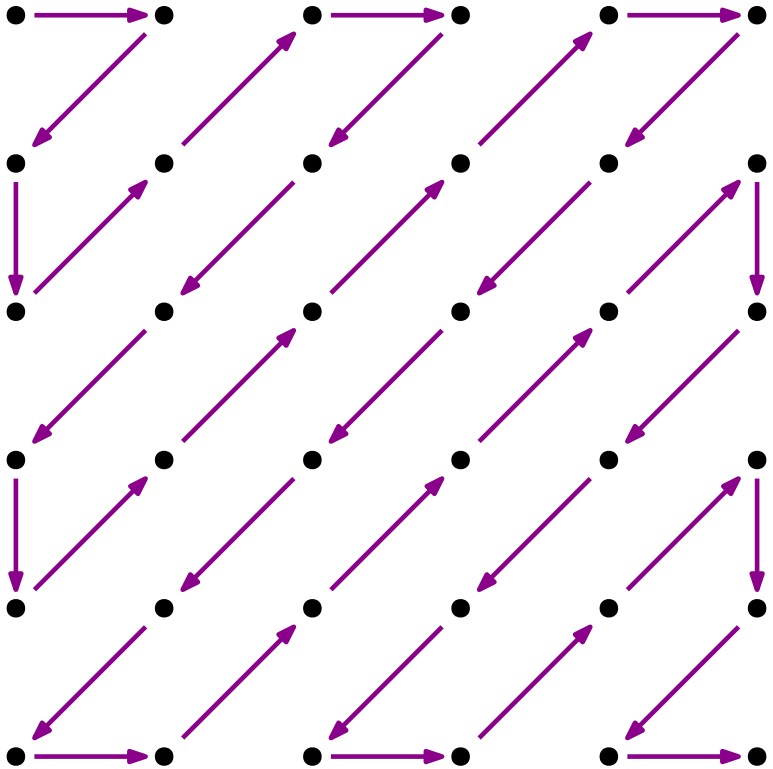
$\Sigma\Delta$ in higher dimensions

Say $d = 2$. How to order $\{p_{n,k}\}_{k_1=0,k_2=0}^{n}$?



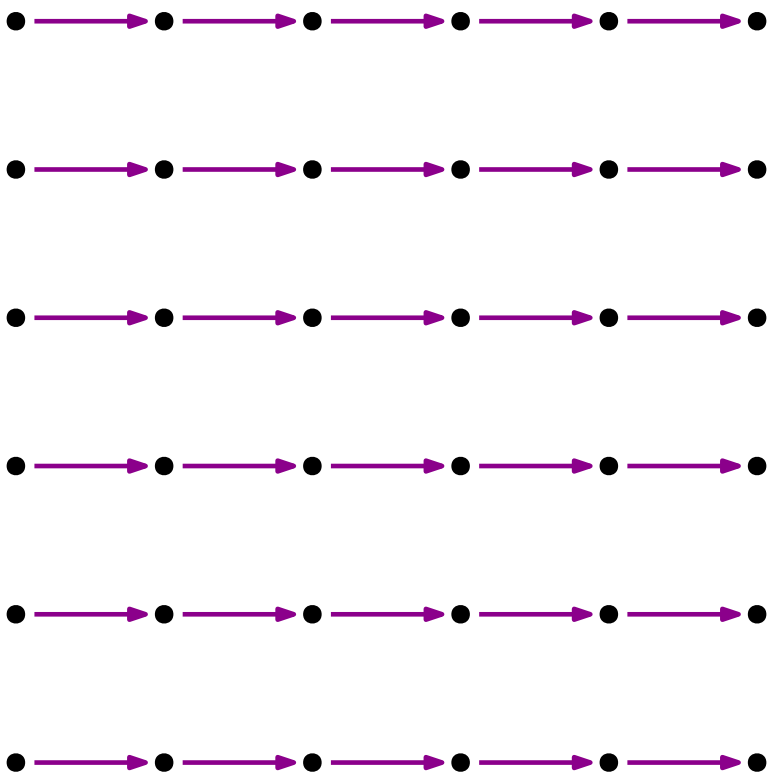
$\Sigma\Delta$ in higher dimensions

Say $d = 2$. How to order $\{p_{n,k}\}_{k_1=0,k_2=0}^{n}$?



$\Sigma\Delta$ in higher dimensions

Say $d = 2$. How to order $\{p_{n,k}\}_{k_1=0,k_2=0}^{n}$?



Quantization error

Theorem

Pick any $1 \leq \ell \leq d$ *and for any* $\{a_k\}_{0\leq k\leq n}$ *with* $\|a\|_\infty \leq \mu < 1$, *let* $\{\sigma_k\}_{0\leq k\leq d}$ *be a solution to a* r*-th order* $\Sigma\Delta$ *scheme applied only in the* ℓ*-th direction,*

$$a - \sigma = (\Delta_\ell)^r u.$$

Then we have

$$\left| \sum_{0\leq k\leq n} a_k p_{n,k}(x) - \sum_{0\leq k\leq n} \sigma_k p_{n,k}(x) \right| \lesssim_{r,\mu} \min\left(1, n^{-r/2} x_\ell^{-r} (1-x_\ell)^{-r}\right).$$

Relies on stability of one-dimension r-th order $\Sigma\Delta$ schemes

[Daubechies, Devore 03], [Güntürk 03].

Iterated Bernstein

Saturation: If $f \in C^2([0,1])$, then

$$\lim_{n\to\infty} n(B_n(f)(x) - f(x)) = \frac{f''(x)x(1-x)}{2}.$$

How to get around the saturation problem?

Iterated Bernstein

Saturation: If $f \in C^2([0,1])$, then

$$\lim_{n\to\infty} n(B_n(f)(x) - f(x)) = \frac{f''(x)x(1-x)}{2}.$$

How to get around the saturation problem?

Classical work: [Micchelli 73], [Felbecker 79] Define the iterated Bernstein operator

$$U_{n,m}(f) := \left(I - (I - B_n)^m\right)(f).$$

The for every $f \in C^s([0,1])$,

$$\|f - U_{n,\lceil s/2\rceil}(f)\|_\infty \lesssim_s \|f\|_{C^s} n^{-s/2}.$$

Takes advantage of smoothness, but is $U_{m,n}(f)$ a linear combination of Bernstein polynomials with small enough coefficients (necessary for quantization)?

Approximation error by linear combinations of Bernstein

Theorem

Fix any integers $d, s \geq 1$, $\delta \in (0,1)$, *and* $f \in C^s([0,1]^d)$. *Then for all integers*

$$n \geq \frac{sd^2 \|f\|_{C^2}}{4\delta},$$

there exist $\{a_k\}_{0 \leq k \leq n}$ *such that*

$$\left\| f - \sum_{0 \leq k \leq n} a_k p_{n,k} \right\|_\infty \lesssim_{s,d} \|f\|_{C^s} n^{-s/2},$$

and

$$\|a\|_\infty \leq \|f\|_\infty + \delta.$$

How to implement Bernstein polynomials?

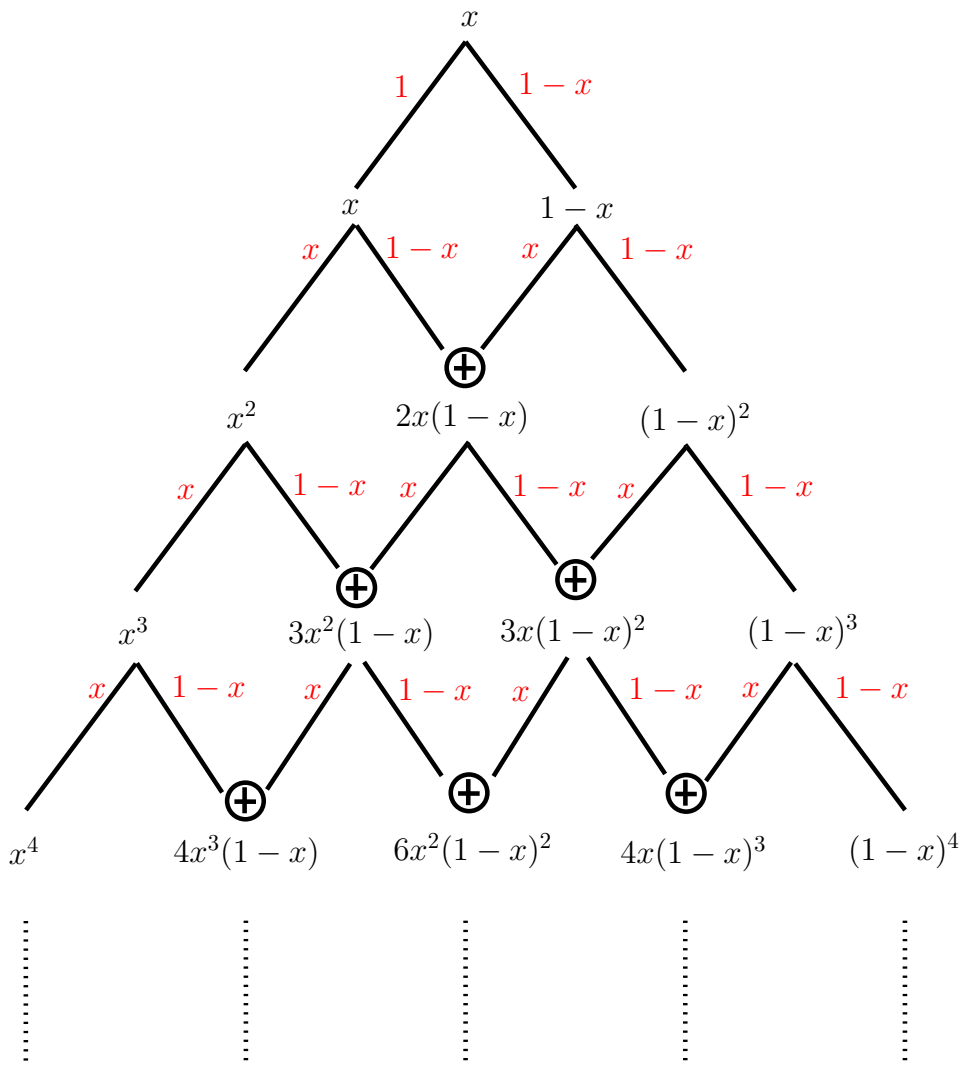


Figure: Pascal triangle implementation of Bernstein

How to multiply with coarse quantization

Key formula:

$$ab = \frac{1}{2}(a+b)^2 - \frac{1}{2}a^2 - \frac{1}{2}b^2$$

Need to find a coarsely quantized neural network that can implement squaring. Suffices for only non-negative a, b.

How to multiply with coarse quantization

Key formula:

$$ab = \frac{1}{2}(a+b)^2 - \frac{1}{2}a^2 - \frac{1}{2}b^2$$

Need to find a coarsely quantized neural network that can implement squaring. Suffices for only non-negative a, b.

- Easy if we use the quadratic non-linearity, $\rho(t) = t^2/2$, so

$$ab = \rho(a+b) - \rho(a) - \rho(b).$$

  This is a $\{\pm 1\}$-quantized quadratic network.

How to multiply with coarse quantization

Key formula:

$$ab = \frac{1}{2}(a+b)^2 - \frac{1}{2}a^2 - \frac{1}{2}b^2$$

Need to find a coarsely quantized neural network that can implement squaring. Suffices for only non-negative a, b.

- Easy if we use the quadratic non-linearity, $\rho(t) = t^2/2$, so

  $$ab = \rho(a+b) - \rho(a) - \rho(b).$$

  This is a $\{\pm 1\}$-quantized quadratic network.

- Squaring can be efficiently approximated by a ReLU network [Yarotsky 17], by exploiting the formula

  $$x(1-x) = \sum_{m=1}^{\infty} \frac{\phi^{\circ m}}{4^m}, \quad \phi(x) = \begin{cases} 2x & \text{if } 0 \le x \le \frac{1}{2}, \\ 2-2x & \text{if } \frac{1}{2} \le x \le 1. \end{cases}$$

  Can do approximate multiplication with a $\{\pm 1/2, \pm 1, \pm 2\}$-quantized ReLU network.

Table of Contents

Putting everything together

$$f - f_{NN} = \underbrace{f - f_B}_{\text{Bern. approx. error}} + \underbrace{f_B - f_Q}_{\text{Bern. quan. error}} + \underbrace{f_Q - f_{NN}}_{\text{Bern. implementation error}}.$$

- $f_B =$ iterated Bernstein of f, then covered to linear Bernstein.
- $f_Q =$ function obtained from r-th order $\Sigma\Delta$ applied to coefficients of f_B.
- $f_{NN} =$ coarsely-quantized neural network approximation of f_Q.

Theorem

Fix any integers $s, d \geq 1$, $\mu \in (0,1)$, *and any* $f \in C^s([0,1]^d)$ *with* $\|f\|_\infty \leq \mu$.

Theorem

Fix any integers $s, d \geq 1$, $\mu \in (0,1)$, *and any* $f \in C^s([0,1]^d)$ *with* $\|f\|_\infty \leq \mu$. *For any* $1 \leq \ell \leq d$ *and all integers*

$$n \geq \frac{sd^2 \|f\|_{C^2}}{2(1-\mu)},$$

there exists a function f_{NN} *that is implementable by a coarsely quantized neural network*

Theorem

Fix any integers $s, d \geq 1$, $\mu \in (0,1)$, *and any* $f \in C^s([0,1]^d)$ *with* $\|f\|_\infty \leq \mu$. *For any* $1 \leq \ell \leq d$ *and all integers*

$$n \geq \frac{sd^2 \|f\|_{C^2}}{2(1-\mu)},$$

there exists a function f_{NN} *that is implementable by a coarsely quantized neural network such that for all* $x \in [0,1]^d$,

$$|f(x) - f_{NN}(x)| \lesssim_{s,d,\mu} \|f\|_{C^s} \min\left(1, n^{-s/2} x_\ell^{-s} (1-x_\ell)^{-s}\right).$$

Theorem

Fix any integers $s, d \geq 1$, $\mu \in (0,1)$, and any $f \in C^s([0,1]^d)$ with $\|f\|_\infty \leq \mu$. For any $1 \leq \ell \leq d$ and all integers

$$n \geq \frac{sd^2 \|f\|_{C^2}}{2(1-\mu)},$$

there exists a function f_{NN} that is implementable by a coarsely quantized neural network such that for all $x \in [0,1]^d$,

$$|f(x) - f_{NN}(x)| \lesssim_{s,d,\mu} \|f\|_{C^s} \min\left(1, n^{-s/2} x_\ell^{-s} (1-x_\ell)^{-s}\right).$$

f_{NN} can be chosen as either

- *a $\{\pm 1\}$-quantized quadratic neural network with*

$$O(n+d) \quad \text{layers},$$
$$O(dn^2 + dn^d) \quad \text{nodes and parameters}.$$

- *a $\{\pm 1/2, \pm 1, \pm 2\}$-quantized ReLU neural network with*

$$O\Big((n+d)\big(d+s/2\big)\log n\Big) \quad \text{layers},$$
$$O\Big((n^2+dn^d)\big(d+s/2\big)\log n\Big) \quad \text{nodes and parameters}.$$

Network in the main theorem

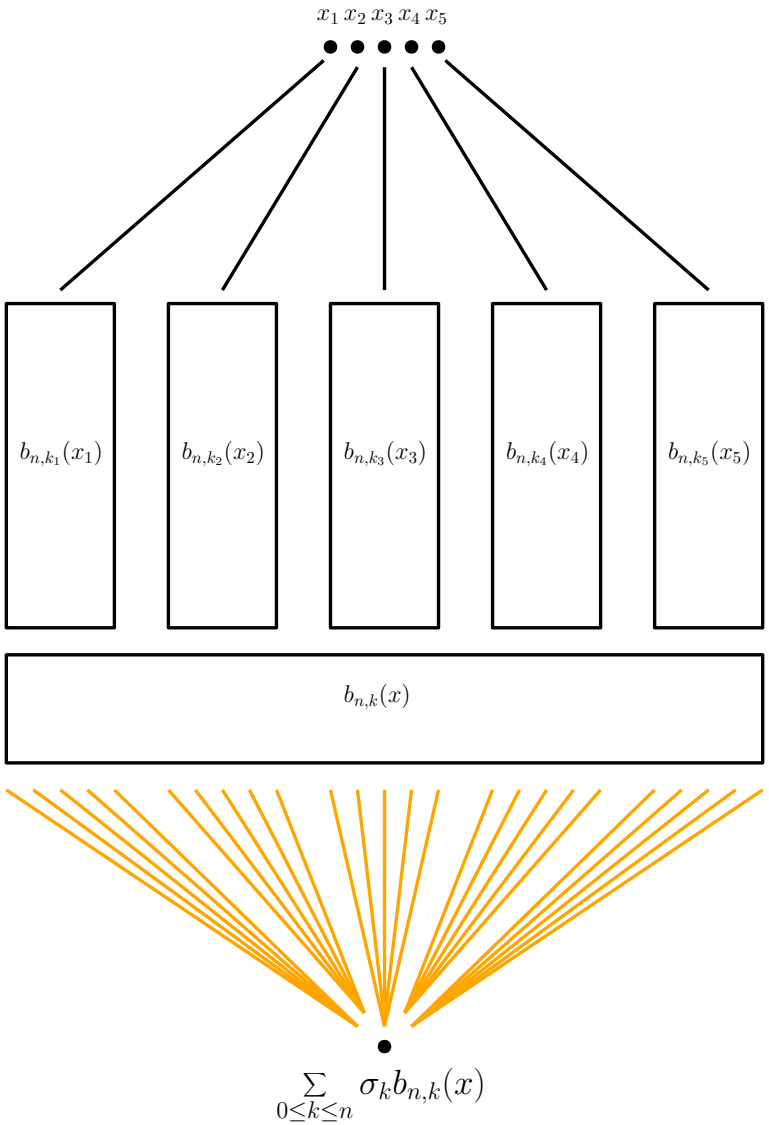


Figure: Coarsely-quantized network

Binary Bernstein Algorithm

System parameters and assumptions:

- d, dimension or number of variables,
- s, smoothness of the target function,
- $\mu \in (0,1)$, upper bound on $\|f\|_\infty$,
- n, any integer at least $sd^2\|f\|_{C^2}/(2(1-\mu))$.

Algorithm:

- Input samples $\{f(k/n)\}_{0\le k\le n}$.
- Calculate $\{a_k\}_{0\le k\le n}$ defined to be

$$a_k := U_{n,\lceil s/2\rceil}(f)\left(\frac{k}{n}\right).$$

- Run s-th order $\Sigma\Delta$ quantization in the ℓ-th direction on $\{a_k\}_{0\le k\le n}$ to obtain

$$\{\sigma_k\}_{0\le k\le n} \subset \{\pm 1\}.$$

- Guaranteed that

$$\left|f(x) - \sum_{0\le k\le n} \sigma_k p_{n,k}(x)\right| \lesssim_{s,d,\mu} \|f\|_{C^s} \min\left(1, n^{-s/2} x_\ell^{-s}(1-x_\ell)^{-s}\right).$$

Open question

Can these results be adapted and generalized to other approximation systems?

Open question

Can these results be adapted and generalized to other approximation systems?

Some properties that we used:

- Bernstein polynomials are non-negative and form a partition of unity.
- Growth conditions on central moments.
- Bernstein polynomials are localized but not too much.
- Explicit formulas for $p_{n,k+1} - p_{n,k}$; in particular, there is significant cancellation.

Binary encoding of a neural network

Given a trained neural network F we can query it at lattice points $\{F(k/n)\}_{0\le k\le n}$ and use our algorithm to produce a one-bit coefficients

$$\{\sigma_k\}_{0\le k\le n} \subseteq \{\pm 1\}.$$

(Can even be computed in parallel.)

References

Figure: Sinan Güntürk

References:

- Approximation with one-bit polynomials in Bernstein form.
  arXiv:2112.09183
- Approximation of functions with one-bit neural networks.
  arXiv:2112.09181

Can also be found on my webpage:

weilinli@cims.nyu.edu

Thank you!!