Mathematical Techniques in the Approximation Theory that are Rooted in Neural Networks - Hieber's Theorem

Suh, Ko, Huo

Georgia Tech

Summer of 2020

Outline

Preliminaries

Mathematical Problem

- Given a function $f \in \mathcal{C}$ where $\mathcal{C}$ is some class of functions, how many weights, nodes, and layers does one need to approximate f with certain accuracy in some predefined metric?

A Neural Network Class to be considered

We first introduce the formal mathematical representation of the FNN model as follows:

- The network architecture $(L, \mathbf{p})$ consists of a positive integer L called the number of hidden layers and a width vector $\mathbf{p} = (p_0, \ldots, p_{L+1})$.
- σ denotes a ReLU activation function, where it is defined as $\sigma(x) = \max(x, 0)$.
- For $\mathbf{v} = (v_1, \ldots, v_r) \in \mathbb{R}^r$, define a shifted activation function $\sigma_{\mathbf{v}} : \mathbb{R}^r \to \mathbb{R}^r$.

$$\sigma_{\mathbf{v}} \begin{pmatrix} y_1 \\ \vdots \\ y_r \end{pmatrix} = \begin{pmatrix} \sigma(y_1 - v_1) \\ \vdots \\ \sigma(y_r - v_r) \end{pmatrix}.$$

A Neural Network Class to be considered

Neural network with network architecture $(L, \mathbf{p})$ is any function of form

$$f : \mathbb{R}^{P_0} \to \mathbb{R}^{P_{L+1}}, f(x) = W_L \sigma_{V_L} W_{L-1} \sigma_{V_{L-1}} \dots W_1 \sigma_{V_1} W_0 x, \tag{1}$$

where W_i is a $p_{i+1} \times p_i$ weight matrix and $v_i \in \mathbb{R}^{p_i}$ is a shift vector.

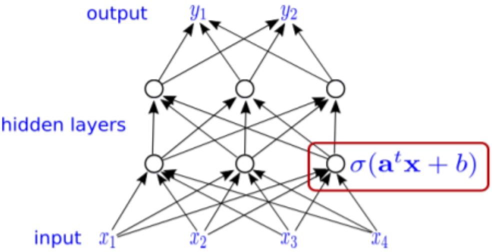


Figure 1: Representation as a direct graph of a network with 2 hidden layers $L = 2$ and width vector $\mathbf{p} = (4, 3, 3, 2)$.

A Neural Network Class to be considered

- All parameter values in the network are bounded by one:

$$\mathcal{F}(L, \mathbf{p}) := \left\{ f \text{ of the form (1)} : \max_{j=0,\ldots,L} \|W_j\|_\infty \vee |v_j|_\infty \leq 1 \right\},$$

  where $\|W_j\|_\infty$ denotes the maximum entry norm of W_j.
- There are only a few non-zero/active network parameters:

$$\sum_{j=0}^{L} \|W_j\|_0 + |v_j|_0 \leq s.$$

  where $\|W_j\|_0$ denotes the number of non-zero entries of W_j.
- Combining all the imposed assumptions, we are going to consider a neural network class whose architecture is constructed as follows:

$$\mathcal{F}(L, \mathbf{p}, s, F) := \left\{ f(x) \in \mathcal{F}(L, \mathbf{p}) : \sum_{j=0}^{L} \|W_j\|_0 + |v_j|_0 \leq s, \|f\|_\infty \leq F \right\}.$$

Assumption on Regression function

In this page, we introduce a multi-index notation which will be frequently used in following slides. For r-dimensional vectors, $a \in [0,1]^r$, $x = (x_1, \ldots, x_r)$ and $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_r)$.

- $$x^{\boldsymbol{\alpha}} := x_1^{\alpha_1} \cdots x_r^{\alpha_r}.$$
- $$|(x-a)^{\boldsymbol{\alpha}}| := \prod_{i=1}^{r} |x_i - a_i|^{\alpha_i}.$$
- $$|\boldsymbol{\alpha}| := |\alpha_1| + \cdots + |\alpha_r|.$$
- $$\partial^{\boldsymbol{\alpha}} f := \frac{\partial^{\alpha}}{\partial^{\alpha_1} \ldots \partial^{\alpha_r}} f.$$

Assumption on Regression function

- Hölder class with β-smoothness index is one of the most commonly studied function classes in literature.
- For $\beta = n + \sigma$ where $n \in \mathbb{N}_0$ and $\sigma \in (0, 1]$, a function has hölder smoothness index β if all partial derivatives up to order n exist and are bounded and the partial derivatives of order n are σ hölder.
- The ball of β-hölder functions with radius K is then defined as

$$\mathcal{C}_r^{\beta}(D, K) = \Big\{ f : D \subset \mathbb{R}^r \to \mathbb{R} : \textstyle\sum_{\boldsymbol{\alpha}:|\boldsymbol{\alpha}|\leq n} \|\partial^{\boldsymbol{\alpha}} f\|_\infty + \sum_{\boldsymbol{\alpha}:|\boldsymbol{\alpha}|=n} \sup_{\substack{\mathbf{x},\mathbf{y}\in D \\ \mathbf{x}\neq\mathbf{y}}} \frac{|\partial^{\boldsymbol{\alpha}} f(\mathbf{x}) - \partial^{\boldsymbol{\alpha}} f(\mathbf{y})|}{|\mathbf{x}-\mathbf{y}|_\infty^{\sigma}} \leq K \Big\}.$$

- In Hieber (2020), $D = [0, 1]^r$. In Petersen and Voigtlaender (2018), $D = [-\frac{1}{2}, \frac{1}{2}]^r$ where r is an input dimension.

Theorem Statement

Theorem 5 of Hieber 2020

For any function $f \in \mathcal{C}_r^\beta([0,1]^r, K)$ and any integers $m \geq 1$ and $N \geq (\beta+1)^r \vee (K+1)e^r$, there exists a network

$$\tilde{f} \in \mathcal{F}\big(L, (r, 6(r+\lceil\beta\rceil)N, \ldots, 6(r+\lceil\beta\rceil)N, 1), s, \infty\big)$$

with depth

$$L = 8 + (m+5)\big(1 + \lceil \log_2(r \vee \beta) \rceil\big)$$

and the number of parameters

$$s \leq 141(r+\beta+1)^{3+r} N(m+6),$$

such that

$$\left\|\tilde{f} - f\right\|_{L^\infty[0,1]^r} \leq (2K+1)(1+r^2+\beta^2)6^r N 2^{-m} + K 3^\beta N^{-\frac{\beta}{r}}.$$

Key Ideas

Key Ideas for proof of Theorem

Key ideas for approximating functions in $\mathcal{C}_r^\beta(D, K)$ with Neural Network are mainly two folded:

- Local Taylor Approximation : We split the input space into small hyper-cubes and construct a network that approximates a local Taylor expansion on each of these hyper-cubes.
- Approximation of multiplication operator : We need to build networks that for given input (x, y) approximately compute the product xy.

Local Taylor Approximation?

- Discretize the input space $[0,1]^r$ with a set of points $D(M) := \{X_\ell = (\ell_j/M)_{j=1,2,\dots,r}, \ell = (\ell_1, \ell_2, \dots, \ell_r) \in \{0,1,2,\dots,M\}^r\}$. The cardinality of this set is $(M+1)^r$.
- Think of Taylor expansion of $f(x)$ at one of the grid points, $x_\ell \in D(M)$, with up to degree n, which can be written as

$$P^\beta_{x_\ell} f(x) := \sum_{\alpha:|\alpha|\le n} (\partial^\alpha f)(x_\ell) \frac{(x - x_\ell)^\alpha}{\alpha!}.$$

- For an arbitrary input $x \in [0,1]^r$, Local Taylor approximation of $f \in \mathcal{C}^\beta_r([0,1]^r, K)$ can be written as follows:

$$P^\beta f(x) := \sum_{x_\ell \in D(M)} P^\beta_{x_\ell} f(x) \prod_{j=1}^{r} \left(1 - M|x_j - x_j^\ell|\right)_+ ,$$

where $x = \{x_1, x_2, \dots, x_r\}$.

Local Taylor Approximation?

- $\forall x \in [0,1]^r$, Local Taylor Approximation of $f(x)$ is written as linear combination of 2^r terms of $P^{\beta}_{x_\ell} f(x)$, for which $x_\ell \in D(M)$ such that $\|x - x_\ell\|_\infty \leq \frac{1}{M}$.

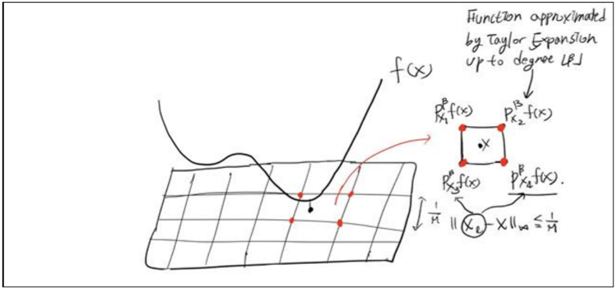


Figure 2: Visualization of intuition on Local Taylor Approximation when $r = 2$.

First Step

First Step

- Neural Network $\tilde{f}$ is not directly used to approximate $f \in \mathcal{C}_r^{\beta}([0,1]^r, K)$, instead it is used to approximate the approximated f through local Taylor expansion, where we denote it as $P^{\beta} f(X)$.
- For $X \in [0,1]^r$, the closeness between functions is measured in a L^{∞} sense. Approximation error can be decomposed with the help of triangular inequality as follows:

$$\left\|\tilde{f} - f\right\|_{L^{\infty}[0,1]^r} \leq \underbrace{\left\|P^{\beta} f(X) - f(X)\right\|_{L^{\infty}[0,1]^r}}_{①} + \underbrace{\left\|\tilde{f} - P^{\beta} f(X)\right\|_{L^{\infty}[0,1]^r}}_{②}.$$

- Controlling ② will be the main focus, whereas a term ① can be easily controlled trough the definition of Hölder class.

Control on ①

Observe $f(x)$ can be written as follows by Multivariate Taylor's Theorem: for any $\xi \in [0,1]$ and any $a \in [0,1]^r$,

$$f(x) = \sum_{\alpha:|\alpha|\leq n-1} \partial^\alpha f(a) \frac{(x-a)^\alpha}{\alpha!} + \sum_{\alpha:|\alpha|=n} \partial^\alpha f(a+\xi(x-a)) \frac{(x-a)^\alpha}{\alpha!}.$$

So for $f \in \mathcal{C}_r^\beta([0,1]^r, K)$,

$$\begin{aligned} |f(x) - P_a^\beta f(x)| &= \sum_{\alpha:|\alpha|=n} |\partial^\alpha f(a+\xi(x-a)) - \partial^\alpha f(a)| \frac{|(x-a)^\alpha|}{\alpha!} \\ &\leq K|x-a|_\infty^\beta. \end{aligned}$$

Control on ①

It is interesting to observe a following fact

$$\sum_{x_\ell \in D(M)} \prod_{j=1}^{r} \left(1 - M|x_j - x_j^\ell|\right)_+ = \prod_{j=1}^{r} \sum_{\ell=1}^{M} \left(1 - M\left|x_j - \frac{\ell}{M}\right|\right)_+ = 1.$$

By using this relation, we can see

$$\begin{aligned}
&\left\|P^\beta(x) - f(x)\right\|_{L^\infty[0,1]^r} \\
&= \left| \sum_{x_\ell \in D(M)} \Big(\underbrace{P^\beta_{x_\ell} f(x) - f(x)}_{\leq K|x - x_\ell|_\infty^\beta} \Big) \prod_{j=1}^{r} \left(1 - M|x_j - x_j^\ell|\right)_+ \right|_\infty \\
&\leq KM^{-\beta}.
\end{aligned}$$

Second Step

Second Step

In order to control the term ②, of course, we first need to build a Neural network which can approximate $P^{\beta}(X)$. This step is involved with several sub-steps:

1. For all $x_\ell \in D(M)$ and for an arbitrary input $x \in [0,1]^r$, we need to build a Neural Network which can approximate $P^{\beta}_{x_\ell}(x)$. Constructed Neural Network has output in $\mathbb{R}^{(M+1)^r}$.
2. For all $x_\ell \in D(M)$ and for an arbitrary input $x \in [0,1]^r$, we need to build a Neural Network which can approximate $\prod_{j=1}^{r}\left(\frac{1}{M} - |x_j - x_j^{\ell}|\right)_+$. Constructed Neural Network has output in $\mathbb{R}^{(M+1)^r}$ as well.

Involved Tools for approximating $P^{\beta}_{x_\ell} f(x)$

Through a r-dimensional Binomial theorem, we can write $P^{\beta}_{x_\ell}(x)$ as linear combination of monomials:

$$P^{\beta}_{x_\ell} f(x) := \sum_{\alpha:|\alpha|\leq n} (\partial^{\alpha} f)(x_\ell) \frac{(x - x_\ell)^{\alpha}}{\alpha!} = \sum_{\gamma:|\gamma|\leq n} C_{\gamma} x^{\gamma}.$$

In order to construct a neural network which can approximate $P^{\beta}_{x_\ell} f(x)$ for a x_ℓ, we need following tools:

1. Multiplication operator $Mult_m(x, y)$ which can approximate the product of two input data x, y.
2. Product operator $Mult^r_m(x_1, \ldots, x_r)$ which can approximate $\Pi^r_{j=1} x_j$ for an input data $x \in [0, 1]^r$.
3. Monomial operator $Mon^r_{m,\gamma}(x_1, \ldots, x_r)$ which can approximate all monomials with up to degree $|\gamma| \leq n$.

Lemma A.1.

In order to construct a neural network which can compute the product of two input data, we first need to have a ReLU neural network which can approximate $x(1-x)$ for an $x \in \mathbb{R}$.

(Lemma A.1.) Let $T^k : [0, 2^{2-2k}] \to [0, 2^k]$,

$$T^k(x) := T_+(x) - T_-^k(x) = (x/2)_+ - (x - 2^{1-2k})_+,$$

and $R^k : [0,1] \to [0, 2^{-2k}]$,

$$R^k(x) := T^k \circ T^{k-1} \circ, \dots, T^1.$$

Then, for any positive integer m,

$$\left| x(1-x) - \sum_{k=1}^{m} R^k(x) \right| \leq 2^{-m}.$$

Detailed proof using induction can be found in the paper.

Visual Illustration of Lemma A.1.

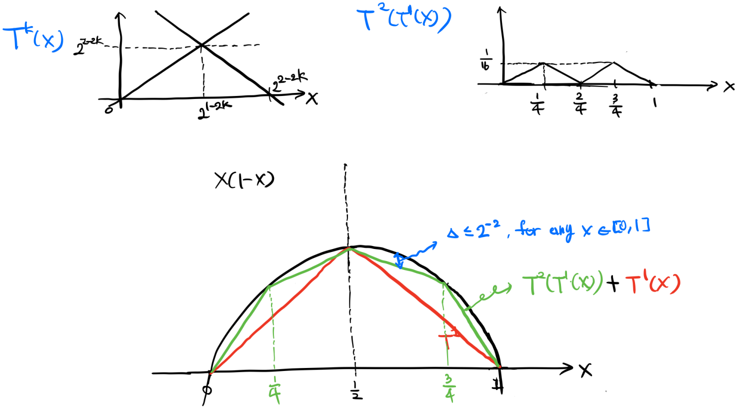


Figure 3: Intuition behind Lemma A.1. can be easily captured by visualization. The Lemma can be proved rigorously via proof by induction on *m*.

Realization of $Mult_m(x, y)$

For given input x, y, we construct a network which returns approximately xy.

> **(Lemma A.2.)** For any positive integer m, there exists a network $Mult_m \in \mathcal{F}(m+4, (2, 6, 6, \ldots, 6, 1))$, such that $Mult_m(x, y) \in [0, 1]$,
>
> $$|Mult_m(x, y) - xy| \leq 2^{-m}, \quad \forall x, y \in [0, 1],$$
>
> and $Mult_m(x, 0) = Mult_m(0, y) = 0$.

1. Let $g(x) = x(1-x)$ and use a following polarization identity :

$$xy = \underbrace{\left(g\left(\frac{x-y+1}{2}\right) + \frac{x+y}{2}\right)}_{①} - \underbrace{\left(g\left(\frac{x+y}{2}\right) + \frac{1}{4}\right)}_{②}$$

2. Our goal is to construct two neural networks which can approximate ① and ②.

Realization of $Mult_m(x, y)$

We can show that there is a network N_m with m hidden layers and width vector $(3, 3, 3, \ldots, 3, 1)$ that computes the function $(T_+(u), T_-^1(u), h(u)) \to \sum_{k=1}^{m+1} R^k(u) + h(u)$.

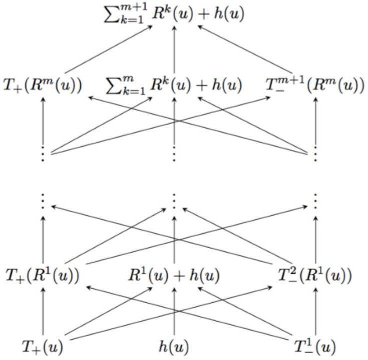


FIG 2. *The network* $(T_+(u), T_-^1(u), h(u)) \mapsto \sum_{k=1}^{m+1} R^k(u) + h(u)$.

Realization of $Mult_m(x, y)$

1. Note that all weight parameters' absolute values are bounded by 1 in network N_m.
2. Set $u = \frac{x-y+1}{2}$, $h(u) = \frac{x+y}{2}$ and apply N_m to approximate ①. Set $u = \frac{x+y}{2}$, $h(u) = \frac{1}{4}$ and apply N_m to approximate ②.
3. Concatenate two constructed networks in parallel, then we have a network with $m+1$ hidden layers and width vector $(2, 6, 6, \ldots, 6, 2)$ that computes:

$$(x, y) \to \Bigg(\underbrace{\sum_{k=1}^{m+1} R^{(k)}\left(\frac{x-y+1}{2}\right) + \frac{x+y}{2}}_{=a}, \underbrace{\sum_{k=1}^{m+1} R^{(k)}\left(\frac{x+y}{2}\right) + \frac{1}{4}}_{=b}\Bigg)$$

Realization of $Mult_m(x, y)$

To ensure the final output to be in $[0, 1]$, we apply to the a, b the two hidden layer network

$$(a, b) \to (1 - (1 - (a - b))_+)_+ = (a - b)_+ \wedge 1.$$

Error bound for approximating xy can be derived as follows:

$$\begin{aligned} |Mult_m(x, y) - xy| &\leq \left| \sum_{k=1}^{m+1} R^{(k)}\left(\frac{x - y + 1}{2}\right) - g\left(\frac{x - y + 1}{2}\right) \right| \\ &+ \left| \sum_{k=1}^{m+1} R^{(k)}\left(\frac{x + y}{2}\right) - g\left(\frac{x + y}{2}\right) \right| \\ &\leq 2^{-m-1} + 2^{-m-1} = 2^{-m}. \end{aligned}$$

Realization of $Mult^r_m(x_1, \dots, x_r)$

Next goal is to construct a product operator which returns approximately $\prod_{i=1}^r x_i$ for $\mathbf{x} \in [0,1]^r$.

(Lemma A.3.) For any positive integer m, there exists a network

$$Mult^r_m \in \mathcal{F}((m+5)\lceil log_2 r \rceil, (r, 6r, 6r, \dots, 6r, 1))$$

such that $Mult^r_m \in [0,1]$ and

$$\left| Mult^r_m(X) - \prod_{i=1}^r x_i \right| \le r^2 2^{-m}, \quad \forall \mathbf{x} = (x_1, x_2, \dots, x_r) \in [0,1]^r.$$

Moreover, $Mult^r_m(x) = 0$ if one of the components of x is zero.

Realization of $Mult^r_m(x_1, \dots, x_r)$

1. For $q = \lceil log_2 r \rceil$, construct a first hidden layer as follows:

$$(x_1, \dots, x_r) \to (x_1, \dots, x_r, \underbrace{1, \dots, 1}_{=2^q - r}).$$

2. Apply the network $Mult_m$ in Lemma A.2. to the pairs (x_1, x_2),(x_3, x_4),...,$(1, 1)$ in order to compute $(Mult_m(x_1, x_2), \dots, Mult_m(1, 1)) \in \mathbb{R}^{2^{q-1}}$.
3. Repeat Step 2. until there is only one entry left.
4. The resulting network is called $Mult^r_m$ and has $q(m + 5)$ hidden layers and all parameters bounded by one.

Error bound on $Mult_m^r(x_1, \ldots, x_r)$

For $q = 1$, the bound holds trivially by Lemma A.2.
Assume that when $q = k - 1$, a following bound holds:

$$\left| Mult_m^r(X) - \prod_{i=1}^{r} x_i \right| \leq 3^{k-2} 2^{-m}.$$

We set $a, b, c, d \in [0, 1]$ as follows:

1. a : Output of network 1, $Mult_m^r(x)$, when $q = k - 1$.
2. b : Output of network 2, $Mult_m^r(x)$, when $q = k - 1$.
3. c : The true value of product network 1 should have.
4. d : The true value of product network 2 should have.

☆ We want to check $|Mult_m(a, b) - cd| \leq 3^{k-1} 2^{-m} \leq r^2 2^{-m}$.

Error bound on $Mult_m^r(x_1, \ldots, x_r)$

☆ We want to check $|Mult_m(a,b) - cd| \leq 3^{k-1}2^{-m} \leq r^2 2^{-m}$.

$$
\begin{aligned}
|Mult_m(a,b) - cd| &= |Mult_m(a,b) - ab + ab - cd| \\
&\leq |Mult_m(a,b) - ab| + |ab - cd| \\
&= 2^{-m} + |ab - bc + bc - cd| \\
&\leq 2^{-m} + |b|\,|a-c| + |c|\,|b-d| \\
&\leq 2^{-m} + 3^{k-2}2^{-m} + 3^{k-2}2^{-m} \\
&\leq 3^{k-1}2^{-m} \leq r^2 2^{-m},
\end{aligned}
$$

where in the last inequality, we use the fact $k = q$ and $(q-1)\log(3) < 2(q-1) < 2\log_2 r = \log_2 r^2$.

Realization of $Mon^r_{m,\gamma}(x_1, \ldots, x_r)$

1. Using the network operator $Mult^r_m$, we are now ready to construct a network which can approximate all monomials of input data $x \in [0,1]^r$ with degree up to $|\alpha| \le n$.
2. Here we use a multi-index notation and $C_{r,\gamma}$ denotes total number of monomials with degree up to $|\alpha| \le n$.

(Lemma A.4.) For $\gamma > 0$ and any positive integer m, there exists a network

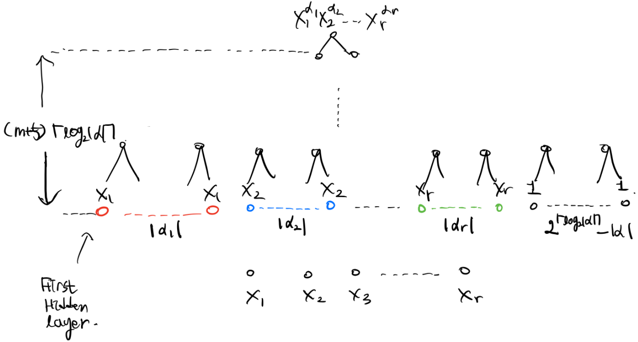
$$Mon^r_{m,\gamma} \in \mathcal{F}(1 + (m+5)\lceil \log_2(\gamma \vee 1)\rceil, \\ (r, 6\lceil\gamma\rceil C_{r,\gamma}, \ldots, 6\lceil\gamma\rceil C_{r,\gamma}, C_{r,\gamma})),$$

such that $Mon^r_{m,\gamma} \in [0,1]^{C_{r,\gamma}}$ and

$$\left| Mon^r_{m,\gamma}(x) - (x^\alpha)_{|\alpha|<\gamma} \right|_\infty \le r^2 2^{-m}, \quad \forall x \in [0,1]^r.$$

Realization of $Mon^r_{m,\gamma}(x_1, \ldots, x_r)$

1. Let's say we want to build a network which can approximate following monomial with degree $|\alpha|$: $x^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_r^{\alpha_r}$.
2. A key idea for building such a network is to construct a first hidden layer with $|\alpha_1|$ x_1s,...,$|\alpha_r|$ x_rs and $2^{\lceil \log_2 |\alpha| \rceil} - |\alpha|$ 1s.



Realization of $Hat^r(x_1, \ldots, x_r)$

(Lemma B.2.) For any positive integer M and m, there exists a network

$$Hat^r \in \mathcal{F}(2 + (m+5)\lceil \log_2 r \rceil, \\ (r, 6r(M+1)^r, \ldots, 6r(M+1)^r, (M+1)^r), s, 1)$$

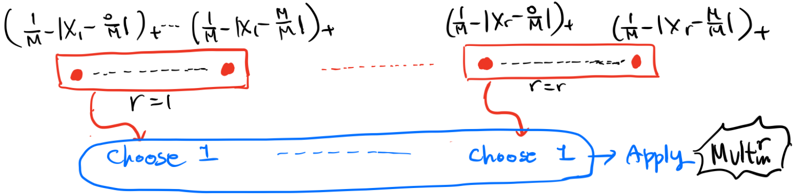
with $s \leq 49r^2(M+1)^r(1 + (m+5)\lceil \log_2 r \rceil)$ such that $Hat^r \in [0,1]^{(M+1)^r}$ and for any $\mathbf{x} = (x_1, x_2, \ldots, x_r) \in [0,1]^r$,

$$\left| Hat^r(x) - \left(\prod_{j=1}^{r} \left(\frac{1}{M} - |x_j - x_j^\ell| \right)_+ \right)_{x_\ell \in D(M)} \right|_\infty \leq r^2 2^{-m}.$$

For any $x_\ell \in D(M)$, the support of the function $x \to (Hat^r(x))_{x_\ell}$ is moreover contained in the support of the function $x \to \prod_{j=1}^r (1/M - |x_j - x_j^\ell|)_+$.

Realization of $Hat^r(x_1, \dots, x_r)$

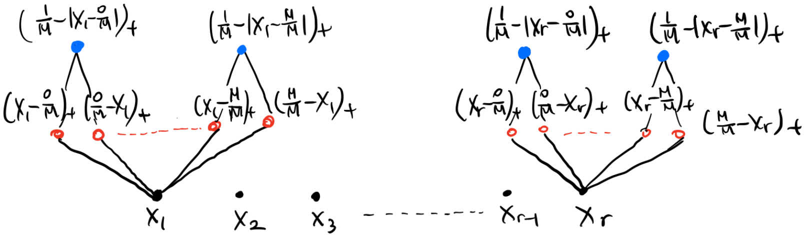
1. Main goal of this Lemma is to construct a neural network which can approximate $\prod_{j=1}^{r}(1/M - |x_j - x_j^\ell|)_+$ for all the points in the grid (i.e. $\forall x_\ell \in D(M)$) with high accuracy.
2. First, for each coordinate of x_j, we need to build a network layer which computes $(1/M - |x_j - \ell/M|)_+$ for all $\ell \in \{0, \dots, M\}$.
3. Second, choose one out of $M + 1$ quantities for each coordinate index and apply $Mult_m^r$ operator for those r chosen values.



Realization of $Hat^r(x_1, \ldots, x_r)$

First two hidden layers of network Hat^r can be constructed as follows:
Note that

$$(1/M - |x_j - \ell/M|)_+ = \big(1/M - (x_j - \ell/M)_+ - (\ell/M - x_j)_+\big)_+.$$



First hidden layer has $2r(M+1)$ nodes and $4r(M+1)$ non-zero weight parameters. Second hiddn layer has $r(M+1)$ nodes and $3r(M+1)$ non-zero weight parameters.

Realization of $Hat^r(x_1, \ldots, x_r)$

1. It remains to apply Mul_m^r operator to r chosen values in second hidden layer of Hat^r.
2. As there are $(M+1)^r$ of these networks, Mul_m^r, this requires $6r(M+1)^r$ units in each hidden layer and $42r^2(M+1)^r\big((m+5)\lceil\log_2 r\rceil+1\big)$ non-zero parameters for the multiplication.
3. Note that the total number of non-zero parameters of the network in $\mathcal{F}(L,(p_1,\ldots,p_L))$ can be bounded by $\sum_{\ell=0}^{L} p_\ell p_{\ell+1} + \sum_{\ell=1}^{L} p_\ell$.
4. Then non-zero parameters of $Mult_m^r$ can be bounded as follows:

$$
\begin{aligned}
&6r^2 + 36r^2((m+5)\lceil\log_2 r\rceil - 1) + 6r + 6r((m+5)\lceil\log_2 r\rceil)\\
&= 36r^2(m+5)\lceil\log_2 r\rceil - 30r^2 + 6r + 6r(m+5)\lceil\log_2 r\rceil\\
&\le 42r^2\big((m+5)\lceil\log_2 r\rceil + 1\big).
\end{aligned}
$$

Realization of $Hat^r(x_1, \ldots, x_r)$

1. Combining all the information elaborated above, we can finally construct a network which can approximate $\prod_{j=1}^{r}(1/M - |x_j - x_j^{\ell}|)_+$ for all $x_\ell \in D(M)$, whose network architecture is as follows:

$$Hat^r \in \mathcal{F}(2 + (m+5)\lceil \log_2 r \rceil, \\ (r, 6r(M+1)^r, \ldots, 6r(M+1)^r, (M+1)^r), s, 1)$$

2. The number of non-zero parameters s can be controlled trivially as follows :

$$\begin{aligned} s &\leq 42r^2(M+1)^r\big((m+5)\lceil \log_2 r \rceil + 1\big) + 7r(M+1) \\ &\leq 49r^2(M+1)^r\big((m+5)\lceil \log_2 r \rceil + 1\big) \end{aligned}$$

Third Step.

Key Idea for the Third Step

Now we are ready to construct a neural network, $\tilde{f}$, which can approximate Local Taylor Approximated $f(x)$ defined as:

$$\sum_{x_\ell \in D(M)} P^\beta_{x_\ell} f(x) \prod_{j=1}^{r} \left(1 - M|x_j - x_j^\ell|\right)_+$$

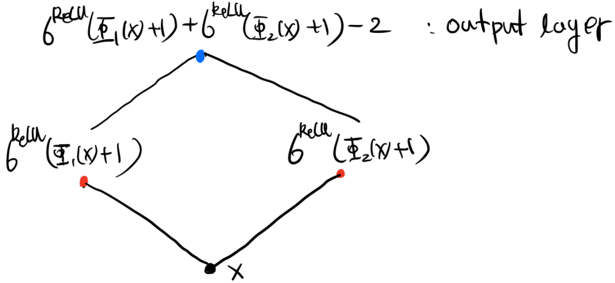
For all points in the grid $x_\ell \in D(M)$, we know how to construct neural networks which approximate :

$$P^\beta_{x_\ell} f(x) \quad \text{and} \quad \prod_{j=1}^{r} \left(1/M - |x_j - x_j^\ell|\right)_+,$$

through $Mon^r_{m,\gamma}$ and Hat^r, respectively. However, in order to realize an inner-product of outputs from these two networks through a fully-connected neural network with ReLU activation function, we need one more trick.

Key Idea for the Third Step

Imagine we have a neural network, $\Phi_1 \in [-1, 1]$, which can approximate $\sin(x)$ function and another neural network, $\Phi_2 \in [-1, 1]$, which approximates $\cos(x)$ function. We want to have a neural network approximating $\sin(x) + \cos(x)$ through Φ_1 and Φ_2. For the construction of network, we apply ReLU activation function. We need to construct a network in a following way :



Bound on $P^{\beta}_{x_\ell} f(x)$

By using the aforementioned idea, we need to scale and shift $P^{\beta}_{x_\ell} f(x)$ so that the modified value is in $[0,1]$. In order to do this, we first need to obtain the maximum of $P^{\beta}_{x_\ell} f(x)$. Recall $P^{\beta}_{x_\ell} f(x)$ is defined as:

$$P^{\beta}_{x_\ell} f(x) := \sum_{\alpha:|\alpha|\leq n} (\partial^\alpha f)(x_\ell) \frac{(x-x_\ell)^\alpha}{\alpha!}$$

By r-dimensional binomial theorem, we can rewrite $(x-x_\ell)^\alpha$ as

$$(x-x_\ell)^\alpha = \sum_{\gamma\leq\alpha} \binom{\alpha}{\gamma} (-x_\ell)^{\alpha-\gamma} x^\gamma, \quad \forall x \in [0,1]^r, \alpha \in \mathbb{N}_0^r.$$

Here for vectors $\gamma, \alpha \in \mathbb{N}_0^r$, $\gamma \leq \alpha$ means that

$$\gamma_1 \leq \alpha_1, \ldots, \gamma_r \leq \alpha_r.$$

Bound on $P_{x_\ell}^{\beta} f(x)$

Then we can write $P_{x_\ell}^{\beta} f(x)$ as follows:

$$\begin{aligned} P_{x_\ell}^{\beta} f(x) &= \sum_{\alpha:|\alpha|\leq n} \frac{(\partial^\alpha f)(x_\ell)}{\alpha!} \sum_{\gamma\leq\alpha} \binom{\alpha}{\gamma} (-x_\ell)^{\alpha-\gamma} x^\gamma \\ &= \sum_{\gamma:|\gamma|\leq n} \underbrace{\left[\sum_{\gamma\leq\alpha \& |\alpha|\leq n} \frac{(\partial^\alpha f)(x_\ell)}{\alpha!} \binom{\alpha}{\gamma} (-x_\ell)^{\alpha-\gamma} \right]}_{:=C_\gamma} x^\gamma. \end{aligned}$$

The absolute value of C_γ can be controlled by using facts $x_\ell \in [0,1]^r$, $f \in \mathcal{C}_r^{\beta}([0,1]^r, K)$ and $(\alpha-\gamma)! \geq 1$: For fixed $\gamma \in \mathbb{N}_0^r$

$$|C_\gamma| \leq \sum_{\gamma\leq\alpha \& |\alpha|\leq n} \frac{|(\partial^\alpha f)(x_\ell)|}{(\alpha-\gamma)!\gamma!} |(-x_\ell)^{\alpha-\gamma}| \leq \frac{K}{\gamma!}.$$

Here, we omit the dependency of x_ℓ when writing C_γ for simplicity.

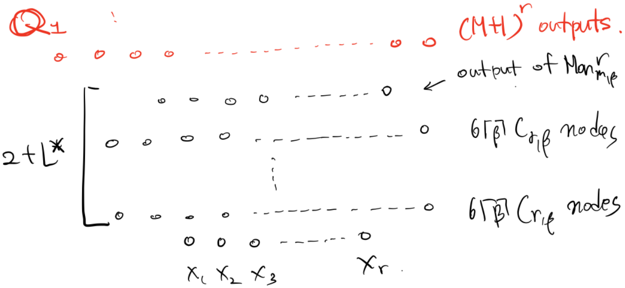
Bound on $P^{\beta}_{x_\ell} f(x)$

Finally, we can bound $P^{\beta}_{x_\ell} f(x), \forall x \in [0,1]^r$ as follows:

$$\begin{aligned} P^{\beta}_{x_\ell} f(x) &= \sum_{\gamma:|\gamma|\leq n} C_\gamma x^\gamma \leq \sum_{\gamma:|\gamma|\leq n} |C_\gamma| \\ &\leq \sum_{\gamma\geq 0} |C_\gamma| \leq \sum_{\gamma\geq 0} \frac{K}{\gamma!} \\ &= K \prod_{j=1}^{r} \sum_{\gamma_j \geq 0} \frac{1}{\gamma_j!} = Ke^r. \end{aligned}$$

Third Step: Substep ①.

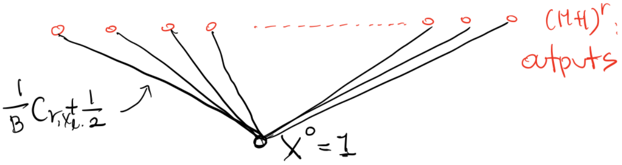
Let $B = \lceil 2Ke^r \rceil$. Then we are ready to construct a network, Q_1, which can approximate $\frac{P^{\beta}_{x_\ell} f(x)}{B} + \frac{1}{2} \in [0,1]^{(M+1)^r}$. We can simply add one hidden layer to the network $Mon^r_{m,\beta}$ as follows :



Note that all the absolute values of weight and bias parameters used for output layer are bounded by 1, since $\frac{1}{B}|C_{\gamma,x_\ell}| \le 1$ for all $\gamma : |\gamma| \le n$, $x_\ell \in D(M)$. We use $L^* := (m+5)\lceil \log_2(\beta \vee r) \rceil$.

Third Step: Substep ①.

Note that $\frac{1}{2}$ can be added by putting $\frac{1}{B}C_{\gamma,x_\ell} + \frac{1}{2}$ weight on monomial $x^0 = 1$ for each $\gamma : |\gamma| \le n$ and $x_\ell \in D(M)$.



Through these constructions, it is obvious that Q_1 has a following network structure:

$$Q_1 \in \mathcal{F}\big(2 + L^*, (r, 6\lceil\beta\rceil C_{r,\beta}, \ldots, 6\lceil\beta\rceil C_{r,\beta}, C_{r,\beta}, (M+1)^r)\big),$$

such that $Q_1 \in [0,1]^{(M+1)^r}$.

Third Step: Substep ①.

The approximation error in a L^∞ sense of the network Q_1 for any $x \in [0,1]^r$ can be calculated trivially as : For any integer $m \geq 1$,

$$\left| Q_1(x) - \left(\frac{P^\beta_{x_\ell} f(x)}{B} + \frac{1}{2} \right)_{x_\ell \in D(M)} \right|_\infty \leq \frac{1}{B} \sum_{\gamma : |\gamma| \leq n} |C_\gamma| \beta^2 2^{-m}$$
$$\leq \frac{Ke^r}{B} \beta^2 2^{-m} \leq \beta^2 2^{-m}.$$

Total number of non-zero parameters can be bounded as :

$$6r(\beta+1)C_{r,\beta} + 42(\beta+1)^2 C^2_{r,\beta}(1+L^*) + C_{r,\beta}(M+1)^r.$$

Third Step: Substep ②.

Consider now a parallel network (Q_1, Hat^r):

1. Observe that $C_{r,\beta} \leq (\beta+1)^r \leq N$ by the definition of $C_{r,\beta}$ and the assumptions on N in the statement of Theorem 5.
2. The parallelized network (Q_1, Hat^r) has a following architecture :

$$(Q_1, Hat^r) \in \mathcal{F}(2 + (m+5)\lceil \log_2(r \vee \beta) \rceil,$$
$$(r, 6(r + \lceil \beta \rceil)N, \ldots, 6(r + \lceil \beta \rceil)N, 2(M+1)^r))$$

3. Note that all network parameters are bounded by 1.

Third Step: Substep ②.

The total number of network parameter of (Q_1, Hat^r) can be bounded as follows: We set M to be the largest integer such that $(M+1)^r \leq N$.

$$
\begin{aligned}
&6r(\beta+1)C_{r,\beta} + 42(\beta+1)^2 C_{r,\beta}^2(L^*+1) + C_{r,\beta}(M+1)^r + 49r^2(M+1)^r(L^*+\\
&\leq 6r(\beta+1)C_{r,\beta}N(1+L^*) + 42(\beta+1)^2 C_{r,\beta}N(1+L^*) + C_{r,\beta}N(1+L^*) + 49\\
&\leq 49(\beta^2+2\beta+1+r\beta+r+r^2+1)C_{r,\beta}N(1+L^*)\\
&\leq 49(\beta+r+1)^2 C_{r,\beta}N(1+L^*)\\
&\leq 49(\beta+r+1)^{2+r}N(1+L^*)\\
&\leq 98(\beta+r+1)^{3+r}N(m+5)
\end{aligned}
$$

where in the last inequality, we use the fact:

$$
\begin{aligned}
1+(m+5)\lceil\log_2(\beta\vee r)\rceil &\leq (m+5)\big(1+\lceil\log_2(\beta\vee r)\rceil\big)\\
&\leq 2(m+5)(\beta\vee r)\\
&\leq 2(m+5)(\beta+r+1).
\end{aligned}
$$

Third Step: Substep ③.

1. Next, we pair the x_ℓth entry of Q_1 and Hat^r and apply to each of the $(M+1)^r$ pairs the $Mult_m$ network described in *Lemma.A*.2.
2. In the last layer, we add up all entries.
3. Finally, we have a network Q_2 which can approximate

$$\sum_{x_\ell \in D(M)} \left(\frac{P_{x_\ell}^{\beta} f(x)}{B} + \frac{1}{2} \right) \prod_{j=1}^{r} \left(\frac{1}{M} - |x_j - x_j^\ell| \right)_+ .$$

4. The network's architecture is as follows:

$$Q_2 \in \mathcal{F}(3 + (m+5)\big(1 + \lceil \log_2(r \vee \beta) \rceil\big),$$
$$(r, 6(r + \lceil \beta \rceil)N, \ldots, 6(r + \lceil \beta \rceil)N, 1)).$$

Third Step: Substep ③.

Note that the required number of parameters for $Mult_m$ is at most :

$$6 + 12 + 36(m+3) + 6(m+4) \leq 42(m+5)$$

We need $(M+1)^r$ $Mult_m$s serially and $(M+1)^r$ parameters for adding up entries in the last hidden layer. This means that at least

$$42(m+5)(M+1)^r + (M+1)^r \leq 43(m+5)N$$

non-zero parameters are required for the steps 1. and 2. in the previous slide. By combining the bound we obtained for the number of non-zero parameters for (Q_1, Hat^r) network, the s for Q_2 is bounded by :

$$141(r+\beta+1)^{3+r}N(m+5).$$

Third Step: Substep ③.

By triangular inequality, we can get the approximation error bound for Q_2:

$$\begin{aligned}&\left| Q_2 - \sum_{x_\ell \in D(M)} \left(\frac{P^\beta_{x_\ell} f(x)}{B} + \frac{1}{2} \right) \prod_{j=1}^{r} \left(\frac{1}{M} - |x_j - x_j^\ell| \right)_+ \right| \\ &\leq \sum_{x_\ell \in D(M): \|x - x_\ell\|_\infty \leq 1/M} (1 + r^2 + \beta^2) 2^{-m} \\ &\leq (1 + r^2 + \beta^2) 2^{r-m}.\end{aligned}$$

In the last inequality, we use the fact for any $x \in [0,1]^r$, there are 2^r points in the grid $D(M)$ whose L^∞ distance between an input data x is within $1/M$.

Third Step: Substep ④.

Finally, we need a network operator which can perform re-scaling and re-shifting as follows:

$$\sum_{x_\ell \in D(M)} \left(\frac{P^{\beta}_{x_\ell} f(x)}{B} + \frac{1}{2} \right) \prod_{j=1}^{r} \left(\frac{1}{M} - |x_j - x_j^{\ell}| \right)_+ \to \sum_{x_\ell \in D(M)} P^{\beta}_{x_\ell} f(x) \prod_{j=1}^{r} \left(1 - M|x_j - x_j^{\ell}| \right)_+$$

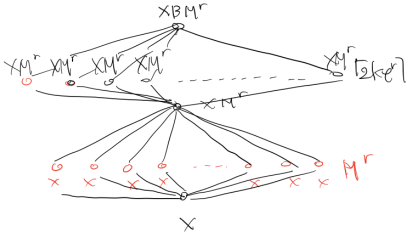
Note that

$$\sum_{x_\ell \in D(M)} \left(\frac{P^{\beta}_{x_\ell} f(x)}{B} + \frac{1}{2} \right) \prod_{j=1}^{r} \left(\frac{1}{M} - |x_j - x_j^{\ell}| \right)_+ = \underbrace{\frac{1}{BM^r} \sum_{x_\ell \in D(M)} P^{\beta}_{x_\ell} f(x) \prod_{j=1}^{r} \left(1 - M|x_j - x_j^{\ell}| \right)_+ + \frac{1}{2M^r}}_{:=a}.$$

Third Step: Substep ④.

We need a network operator with ReLU activation function computing $a \to BM^r(a - 1/2M^r)$.

1. The network $x \to BM^r x$ is in the class $\mathcal{F}(3, (1, M^r, 1, B, 1))$ with shift vectors v_j are all equal to zero and weight matrices W_j having all entries equal to one.



2. Network $a \to BM^r(a - 1/2M^r)$ computes in the first hidden layer $(a - 1/2M^r)_+$ and $(1/2M^r - a)_+$ and then applies the network $x \to BM^r x$ to both units. In the output layer the second value is subtracted from the first one.

Third Step: Substep ④.

It is trivial to check network $a \to BM^r(a - 1/2M^r)$ has at most $12N + 6$ non-zero parameters.

1. For the network $x \to BM^r x$, we need $2M^r + 2\lceil 2Ke^r \rceil$ parameters. Because of the assumption $N \geq (K+1)e^r$, at most $6N$ parameters are required.
2. We need two $x \to BM^r x$ networks and extra 6 parameters for the network $a \to BM^r(a - 1/2M^r)$.

Third Step: Substep ④.

Apply the network $a \to BM^r(a - 1/2M^r)$ to the output of Q_2, then there exists a network Q_3 in

$$Q_3 \in \mathcal{F}(8 + (m+5)\big(1 + \lceil \log_2(r \vee \beta) \rceil\big), \\ (r, 6(r + \lceil \beta \rceil)N, \ldots, 6(r + \lceil \beta \rceil)N, 1)),$$

such that, for all $x \in [0,1]^r$,

$$\left| Q_3 - \sum_{x_\ell \in D(M)} P^\beta_{x_\ell} f(x) \prod_{j=1}^r \left(1 - M|x_j - x_j^\ell|\right)_+ \right| \\ \leq \lceil 2Ke^r \rceil M^r(1 + r^2 + \beta^2)2^{r-m} \\ \leq (2K+1)(2e)^r M^r(1 + r^2 + \beta^2)2^{-m} \\ \leq (2K+1)(1 + r^2 + \beta^2)6^r N 2^{-m}.$$

Third Step: Substep ④.

The number of non-zero parameters of Q_3 is bounded by

$$141(r+\beta+1)^{3+r}N(m+5)+(12N+6)\leq 141(r+\beta+1)^{3+r}N(m+6).$$

We are ready to obtain an approximation error bound :

$$\begin{aligned}\left\|\tilde{f}-f\right\|_{L^\infty[0,1]^r} &\leq \left\|P^\beta f(X)-f(X)\right\|_{L^\infty[0,1]^r}+\left\|\tilde{f}-P^\beta f(X)\right\|_{L^\infty[0,1]^r}\\ &\leq KM^{-\beta}+(2K+1)(1+r^2+\beta^2)6^rN2^{-m}\\ &\leq 3^\beta N^{-\frac{\beta}{r}}+(2K+1)(1+r^2+\beta^2)6^rN2^{-m}.\end{aligned}$$

where in the last inequality, we use

$$(M+1)^r\leq N\leq (M+2)^r\leq (3M)^r.$$

We finally conclude the proof.