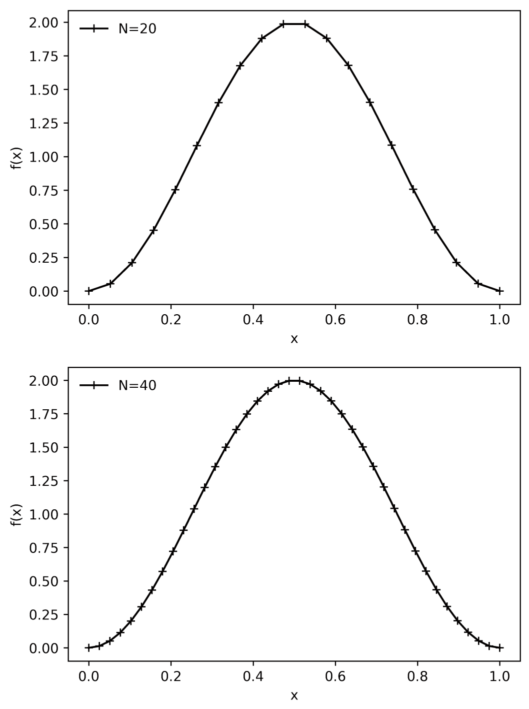

**Fig. 2.5:** Reconstruction of $f^{\text{obj}}(x) = 1 - \cos 2\pi x$ with Algorithm 4. On the left with 20 neurons, on the right with 40 neurons. The results have the left-right symmetry.

### 2.4.2 2D piecewise affine Finite Elements

One starts with a triangular mesh of a closed polygon domain $\Omega \subset \mathbb{R}^2$

$$\Omega = \overline{\bigcup_{r \in \mathbb{Z}} T_r}$$

where all $T_r$'s are disjoint open triangles such that $T_r \bigcap T_s = \emptyset$ for $r \neq s$. The index $0 < h < \infty$ will refer to an upper bound on the diameters of the triangle, that is $\text{diam}(T_r) \leq h$ for all $r$. We consider that the number of triangles $N_h = \#\{T_r\}$ is finite, that is the triangulation is

$$\mathcal{T}_h = \{T_r\}_{r=1}^{N_h}.$$

The nodes $\mathbf{x}_j$'s are the summit of the triangles. A triangle is defined by 3 nodes. The list of all nodes is denoted as

$$\mathcal{N}_h = \{\mathbf{x}_j\}_j.$$

**Definition 2.4.5.** *The $P^1$ Finite Element space is*

$$V_h = \left\{ v \in C^0(\mathbb{R}^2) \mid \text{ for all } T_r \text{ then } v_{|T_r} \text{ is affine on } T_r \right\}.$$

Let $v \in V_h$, the

$$(x, y) \in T_r \Rightarrow v(x, y) = a_r + b_r x + c_r y \text{ for some } (a_r, b_r, c_r) \in \mathbb{R}^3.$$

The basis functions $\varphi_j \in V_h$ satisfy $\varphi_j(\mathbf{x}_i) = \delta_{ji}$.

**Lemma 2.4.6.** *All functions $v_h \in V_h$ have the unique representation*

$$v_h = \sum_j \alpha_j \varphi_j, \qquad \alpha_j = v_h(\mathbf{x}_j). \tag{2.16}$$

*Proof.* Take $v_h \in V_h$ and set $w_h = v_h - \sum_j v_h(\mathbf{x}_j)\varphi_j$. Then $w_h(\mathbf{x}_i) = 0$ for all nodes $\mathbf{x}_i$. Since a affine function in a triangle is uniquely defined by the 3 values at the 3 nodes of the triangles, then $w_h(\mathbf{x}) = 0$ for all $\mathbf{x}$ in all triangles $T_i$. Then $w_h = 0$. The representation is necessarily unique. □

For a given basis function $\varphi_j$, the list of the triangles in its support is

$$N(j) = \{r \mid \mathbf{x}_j \in \overline{T_r}\}.$$

The support of $\varphi_j$ can be written as

$$\operatorname{supp}(\varphi_j) = \overline{\bigcup_{r \in N(j)} T_r}.$$

The maximal number of triangles per node is

$$m_h = \max_j \# N(j).$$

The linear function which is equal to $\varphi_j$ on a triangle $T_r$ is

$$g_r^j(\mathbf{x}) = a_r^j + b_r^j x + c_r^j y = \varphi_j(\mathbf{x}) \text{ on } T_r.$$

An elegant formula is the following.

**Lemma 2.4.7.** *Assume the support of the basis function supp($\varphi_j$) is convex. Then the basis function is represented by the min-max formula*

$$\varphi_j(x) = \max\left(0, \min_{r \in N(j)} g_r^j(\mathbf{x})\right). \tag{2.17}$$

*Proof.* As usual when convexity is involved, the proof-by-drawing is almost evident. This is left to the reader. □

The formula (2.17) can be rewritten with the ReLU function in may ways. We particularize

$$\varphi_j(x) = R\left(\min_{r\in N(j)} g_r^j(\mathbf{x})\right) \text{ and } \varphi_j(x) = \min\left(R\left(g_1^j(\mathbf{x})\right), \dots, R\left(g_{N(j)}^j(\mathbf{x})\right)\right).$$

Next we formulate a recent result by He-Li-Xu-Zheng [43]. We consider a continuous function with compact support $f^{\text{obj}} = v \in C_0^0(\mathbb{R}^2)$ and $f = v_h \in V_h$ defined by

$$v_h = \sum_j \alpha_j \varphi_j, \qquad \alpha_j = v(\mathbf{x}_j). \tag{2.18}$$

Since the support of $v$ is compact, only a finite number of values $v_h(\mathbf{x}_j) = v(\mathbf{x}_j)$ are non zero.

**Theorem 2.4.8** (He-Li-Xu-Zheng [43])**.** *Consider $v \in C_0^0(\mathbb{R}^2)$ and a triangulation with the convexity property of Lemma 2.4.7 for all basis functions. Then $v_h \in V_h$ defined in (2.18) can be realized with a neural network with the ReLU function as activation function, with at most $O(\log_2 m_h)$ hidden layers and at most $O\left(m_h \; \#\left(\mathcal{T}_h \bigcap supp(v)\right)\right)$ neurons.*

*Proof.* • The first step of the proof is to bound the number of layers needed to evaluate one single basis function. The idea is that the evaluation

$$\varphi_j(x) = \min\left(R\left(g_1^j(\mathbf{x})\right), \dots, R\left(g_{N(j)}^j(\mathbf{x})\right)\right) \tag{2.19}$$

can be performed recursively. Indeed one can use the formula for the evaluation of the minimum of multiple values

$$\min\left(\alpha_1, \dots, \alpha_{2^{n+1}}\right) = \min\left(\min\left(\alpha_1, \dots, \alpha_{2^n}\right), \min\left(\alpha_{2^n+1}, \dots, \alpha_{2^{n+1}}\right)\right), \quad n \in \mathbb{N}, \tag{2.20}$$

which can be implemented recursively with $n+1$ levels. Here $n$ is chosen such that $2^n \le N(j) < 2^{n+1}$ where $N(j) \le m_h$ is the number of triangles around the node $\mathbf{x}_j$. So one has the bound

$$n + 1 \le \log_2 m_h + 1 \text{ for all } j.$$

With the formula (1.55), the min function can be evaluated with one ReLU function and more linear functions. With one additional ReLU function to evaluate the internal terms in (2.19), the total number of layers needed to implement (2.17) can be bounded by $\log_2 m_h + 2 = O(\log_2 m_h)$.

• As visible in (2.20), the number of neurons needed to implemented one basis function is 1 for the external layer, 2 for the first internal layer, .... So number of neurons is

$$\#\,(\text{neurons}) \le 1 + 2 + 4 + ... + O(\log_2 m_h) = O\left(2^{\log_2 m_h}\right) = O(m_h).$$

• Multiplied by the number of basis functions $\#\left(\mathcal{T}_h \bigcap \mathrm{supp}(v)\right)$, it yields the number of neurons $O\left(m_h \# \left(\mathcal{T}_h \bigcap \mathrm{supp}(v)\right)\right)$.
• A last linear combination based on (2.16) adds an extra linear cost. □

Recent and more comprehensive analysis of advanced type of finite elements which can be obtained with Neural Networks is in [77, 59].

## 2.5 ReLU function, power of depth

This discussion is based on a recent remarkable result by Yarosky [105], then exploited in [19] and other works. It is based on a series with the ReLU function with fast convergence and on the Takagi function.

### 2.5.1 An interesting series

Let $g : [0,1] \longrightarrow [0,1]$ be the hat function in dimension one

$$g(x) = 1 - |1-2x| = 2R(x) - 4R\left(x - \frac{1}{2}\right), \quad 0 \le x \le 1.$$

For $0 \le x \le 1$, w notice that $g(x/2) = x$ and $g(1-x) = g(x)$.

**Definition 2.5.1.** *One defines $g_1 = g$ and $g_{n+1} = gog_n$ by iterations.*

**Lemma 2.5.2.** *The family $g_n$ has a saw-tooth structure. More precisely*
• *For $n \ge 1$, $g_n$ is such that*

$$g_n(x) = 2^n x \text{ for } 0 \le x \le \frac{1}{2^n} \tag{2.21}$$

*and symmetric with respect to $\frac{1}{2^n}$*

$$g_n(x) = g_n\left(\frac{1}{2^{n-1}} - x\right) \text{ for } \frac{1}{2^n} \le x \le \frac{1}{2^{n-1}}. \tag{2.22}$$

• *For $n \ge 2$, $g_n$ is $\frac{1}{2^{n-1}}$-periodic on $[0,1]$*

$$g_n\left(x + \frac{1}{2^{n-1}}\right) = g_n(x) \text{ for } 0 \le x \le 1 - \frac{1}{2^{n-1}}. \tag{2.23}$$

*Proof.* • By definition, one has $g_1(x) = 2x$ for $0 \le x \le \frac{1}{2}$. Then (2.21) holds by iteration on $n$. Take $\frac{1}{2^n} \le x \le \frac{1}{2^{n-1}}$, then

$$g_n(x) = g\left(g_{n-1}(x)\right) = g\left(2^{n-1}x\right) = g\left(1 - 2^{n-1}x\right)$$

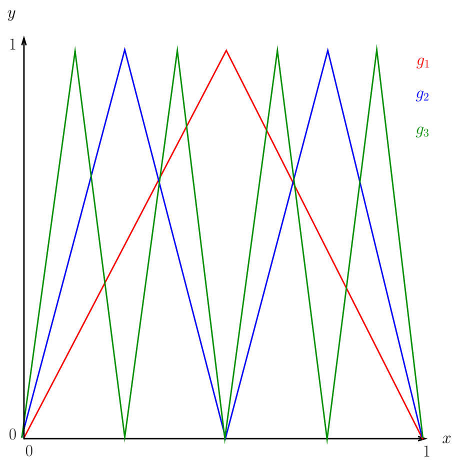

**Fig. 2.6:** Saw-tooth structure of the family $g_n$.

$$= g\left(g_{n-1}\left(\frac{1}{2^{n-1}} - x\right)\right) = g_n\left(\frac{1}{2^{n-1}} - x\right) \text{ which is (2.22).}$$

• Let us show (2.23) first for $n = 2$. For $0 \le x \le \frac{1}{2}$, one has

$$g_2\left(x + \frac{1}{2}\right) = g_2\left(1 - \left(x + \frac{1}{2}\right)\right) = g_2\left(\frac{1}{2} - x\right) = g_2(x) \qquad \text{(using (2.22)).}$$

For $n \ge 2$ and $0 \le x \le \frac{1}{2^{n-1}}$, one has

$$g_n\left(x + \frac{1}{2^{n-1}}\right) = g_2\left(g_{n-2}\left(x + \frac{1}{2^{n-1}}\right)\right) = g_2\left(2^{n-2}\left(x + \frac{1}{2^{n-1}}\right)\right)$$

$$= g_2\left(2^{n-2}x + \frac{1}{2}\right) = g_2\left(2^{n-2}x\right) = g_2\left(g_{n-2}(x)\right) = g_n(x) \text{ which is (2.23).}$$

□

**Lemma 2.5.3.** *Let* $0 \le x \le 1$. *One has the formula*

$$x^2 = x - \sum_{n\ge1} \frac{1}{4^n} g_n(x). \tag{2.24}$$

Since this formula plays a prominent role in the Yarotsky Theorem, we provide two different proofs.

*A first proof.* In this proof, one calculates directly the sum of the series. Set

$$H(x) = x - \sum_{n\ge1} \frac{1}{4^n} g_n(x) = \sum_{n\ge1} \frac{1}{4^n} h_n(x), \qquad \text{where } h_n(x) = 2^n x - g_n(x).$$

By construction $h_n(x/2) = h_{n-1}(x)$ and $h_1(x/2) = 0$, so

$$H\left(\frac{x}{2}\right) = \frac{1}{4}H(x). \tag{2.25}$$

One also has

$$H(1-x) = 1-x-\sum_{n\geq 1}\frac{1}{4^n}g_n(1-x) = 1-x-\sum_{n\geq 1}\frac{1}{4^n}g_n(x) = 1-2x+H(x). \tag{2.26}$$

It is shown in Lemma 2.5.4 that (2.25) and (2.26) imply (2.27). This property is rewritten as $H(x) = x^2$ for $x \in \left\{\frac{k}{2^n} : n \in \mathbb{N},\ k = 0, \ldots, 2^n\right\}$ which is a dense subset of the interval $[0,1]$. By direct calculation, one has that $\|g_n'\|_{L^\infty(0,1)} = 2^n$ so $H$ is Lipschitz

$$|H(x) - H(y)| = |x-y| + \sum_{n\geq 1}\frac{1}{4^n}2^n|x-y| = 2|x-y|,$$

and it is continuous. Therefore $H(x) = x^2$ for all $x$ in the interval $[0,1]$. □

**Lemma 2.5.4.** *One has*

$$H\left(\frac{k}{2^n}\right) = \left(\frac{k}{2^n}\right)^2, \qquad 0 \leq k \leq 2^n, \quad n \in \mathbb{N}. \tag{2.27}$$

*Proof.* By definition, $H(1) = 1$ and $H(0) = 0$ so (2.27) holds for $n = 0$. For $n = 1$, take (2.25) with $x = 1$. Assume now (2.27) for $n \geq 1$. Then (2.25) shows

$$H\left(\frac{k}{2^{n+1}}\right) = \frac{1}{4}\left(\frac{k}{2^n}\right)^2 = \left(\frac{k}{2^{n+1}}\right)^2, \qquad 0 \leq k \leq 2^n.$$

Next take $2^n < k \leq 2^{n+1}$ and note $k' = 2^{n+1} - k$ such that $0 \leq k' \leq 2^n$. Then, with (2.26), one has

$$\begin{aligned} H\left(\frac{k}{2^{n+1}}\right) &= H\left(1 - \frac{k'}{2^{n+1}}\right) = 1 - 2\frac{k'}{2^{n+1}} + H\left(\frac{k'}{2^{n+1}}\right) \\ &= 1 - 2\frac{k'}{2^{n+1}} + \left(\frac{k'}{2^{n+1}}\right)^2 = \left(\frac{k}{2^{n+1}}\right)^2 \end{aligned}$$

and which shows (2.27) for $n+1$. Iteration on $n$ ends the proof. □

*A second proof of the formula (2.24).* This second proof is based on a functional equation satisfied by the series. Let $p(x) = x - x^2$.
For $0 \leq x \leq \frac{1}{2}$, one checks that

$$p(x) = \frac{1}{4}(2x) + \frac{1}{4}(2x - (2x)^2) = \frac{1}{4}g(x) + \frac{1}{4}p(g(x)). \tag{2.28}$$

For $\frac{1}{2}x \le x \le 1$, one sets $y = 1 - x \in [0, \frac{1}{2}]$. By definition one has $p(y) = p(x)$ and $g(y) = g(x)$. So $p(x) = p(y) = \frac{1}{4}g(y) + \frac{1}{4}p(g(y)) = \frac{1}{4}g(x) + \frac{1}{4}p(g(x))$.
So the functional equation (2.28) holds for all $0 \le x \le 1$. The solution of the functional equation can be calculated as

$$\begin{array}{rcl} p(x) & = & \frac{1}{4}g(x) + \frac{1}{4}p(g(x)) \\ & = & \frac{1}{4}g(x) + \frac{1}{16}g \circ g(x)) + \frac{1}{16}p(g \circ g(x)) \\ & = & \frac{1}{4}g(x) + \frac{1}{16}g \circ g(x)) + \frac{1}{64}g \circ g \circ g(x) + \frac{1}{64}p(g \circ g \circ g(x)). \end{array}$$

One obtains at any order $p \ge 1$

$$p(x) = \sum_{n=1}^{p} \frac{1}{4^n} g_n(x) + \frac{1}{4^p} p(g_p(x)).$$

Letting $p \to \infty$, the last term tends to zero in norm $L^\infty(0,1)$. So $p(x) = \sum_{n=1}^{\infty} \frac{1}{4^n} g_n(x)$ for $0 \le x \le 1$, which ends this second proof. □

Next we consider the truncated series under the form

$$f_p(x) = x - \sum_{n \ge 1}^{p} \frac{1}{4^n} g_n(x). \tag{2.29}$$

The function $f_p(x)$ is an approximation of the square function $f^{\text{obj}}(x) = x^2$.

**Lemma 2.5.5.** *One has fast convergence of the truncated series*

$$\left\| f^{\text{obj}} - f_p \right\|_{L^\infty(0,1)} \le \frac{1}{3 \times 4^p}, \qquad 0 \le x \le 1. \tag{2.30}$$

*Proof.* Indeed $\left\| \sum_{n=p+1}^{\infty} \frac{1}{4^n} g_n \right\|_{L^\infty(0,1)} \le \sum_{n=p+1}^{\infty} \frac{1}{4^n} = \frac{1}{3 \times 4^p}$. □

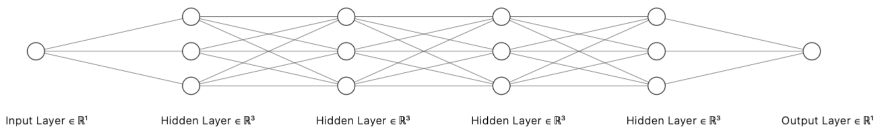

**Fig. 2.7:** Graph structure of the series $f_p$ for $p = 4$.

Let us now construct two networks which realize $f_p$. We see on Figure 2.7 such a network for $p = 4$. It is made of 4 hidden dense layers of width equal to 3, plus the input and output layer. The output layer uses a linear activation function. The implementation of $f$ is made with $p$ hidden layers with formula (1.25). The two top neurons of each layer on the figure are here to calculate the $g_n$ for increasing values of $n$. The bottom neurons make what is called a **collation channel** which calculates the sum of the series. This denomination comes from electrical engineering [19]).

**Definition 2.5.6.** *In a generic neural network with p hidden layers such as the one of Figure 1.2, a* **collation channel** *is the bottom line of hidden neurons with the additional condition*

$$(W_r)_{i,a_r} = 0 \text{ for } 1 \le r \le p-1 \text{ and } 1 \le i \le a_{r+1}-1.$$

The idea behind this definition is that no numerical information come from the bottom line until the last layer. This is evident to check on Figure 2.7.

The first network (2.31-2.33) that we consider uses the ReLU activation function and a collation channel

$$W_0 = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \quad b_0 = \begin{pmatrix} 0 \\ -\frac{1}{2} \\ 0 \end{pmatrix}, \quad g = R, \tag{2.31}$$

$$W_r = \begin{pmatrix} 2 & -4 & 0 \\ 2 & -4 & 0 \\ 2/4^r & -4/4^r & 1 \end{pmatrix}, \quad b_r = \begin{pmatrix} 0 \\ -\frac{1}{2} \\ 0 \end{pmatrix}, \quad g = R, \quad 1 \le r \le p-1, \tag{2.32}$$

$$W_p = \begin{pmatrix} 2 \times 4^{-p}, & -4 \times 4^{-p}, & 1 \end{pmatrix}, \quad b_p = \begin{pmatrix} 0 \end{pmatrix}, \quad g = \text{linear}. \tag{2.33}$$

A key property is that the ReLU is not activated on the collation channel, because all results are non positive.

**Lemma 2.5.7.** *The objective function $f^{\text{obj}}(x) = x^2$ can be approximated in the maximal norm on a neural network with p hidden dense layers with 3 neurons. One has fast convergence of the NN approximation with respect to the number of layers, in the sense that the error is $\varepsilon = \left\| f^{\text{obj}} - f_p \right\|_{L^\infty(0,1)} = O(4^{-p})$ and the scaling of the numerical cost with respect the error is* $\text{Cost} = O(\log 1/\varepsilon)$.

*Proof.* The scaling $\varepsilon = O(4^{-p})$ comes from (2.30-2.29). For large number $p$ of hidden layers, the cost is proportional is $\text{Cost} = O(3p)$. But $p = O(\log 1/\varepsilon)$. □

As a consequence of Lemma 1.2.12 which states that permutations of the weight and biases do not change the result of the function, the matrices of weights and biases in (2.31-2.33) can be changed by permutations. A more drastic change consists to changing the activation function, as achieved in the second neural network

```
Python-Keras-Tensorflow Initialization
fac=4; depth=2
def init_W0(shape, dtype=None): return K.constant(np.array([[1,1,0]]))
def init_b(shape, dtype=None): return K.constant(np.array([0,-0.5,0]))
def init_W1(shape, dtype=None):
 W= np.array([[2,2,2/fac],[-4,-4,-4/fac],[0,0,1]]);return K.constant(W)
def init_W2(shape, dtype=None):
 W= np.array([[2,2,2/fac**2],[-4,-4,-4/fac**2],[0,0,1]]);
 return K.constant(W)
def init_W3(shape, dtype=None):
 W=np.array([[2/fac**depth],[-4/fac**depth],[1]]); return K.constant(W)
model = Sequential()
model.add(Dense(3, input_dim=1,name="lay0",kernel_initializer=init_W0,
             use_bias=True,bias_initializer=init_b,activation='relu'))
model.add(Dense(3, input_dim=1,name="lay1",kernel_initializer=init_W1,
             use_bias=True,bias_initializer=init_b,activation='relu'))
model.add(Dense(3, input_dim=1,name="lay2",kernel_initializer=init_W2,
             use_bias=True,bias_initializer=init_b,activation='relu'))
model.add(Dense(3, input_dim=1,name="lay3",kernel_initializer=init_W3,
             use_bias=False,bias_initializer=init_b,activation='relu'))
x_p=np.linspace(0,1,100); y_p=model.predict(x_p);
plt.plot(x_p,y_predict_ini)
```

**Algorithm 6:** Implementation of $x \mapsto x(1-x)$ deduced from the NN of Figure 2.7.

defined by (2.34-2.36). This second network with the TReLU as activation function generates the same function $f$ as the first network (2.31-2.33) which uses the ReLU function

$$\overline{W}_0 = \begin{pmatrix} 2 \\ 2 \\ 0 \end{pmatrix}, \quad b_0 = \begin{pmatrix} 0 \\ 2 \\ 0 \end{pmatrix} \tag{2.34}$$

$$\overline{W}_r = \begin{pmatrix} 2 & 2 & 0 \\ -2 & -2 & 0 \\ 4^{-r} & 4^{-r} & 1 \end{pmatrix}, \quad b_r = \begin{pmatrix} -2 \\ 4 \\ -4^{-r} \end{pmatrix}, \quad 1 \le r \le p-1, \tag{2.35}$$

$$\overline{W}_p = \begin{pmatrix} -4^{-p}, & -4^{-p}, & 1 \end{pmatrix}, \quad b_p = \begin{pmatrix} 4^{-p} \end{pmatrix}, \quad g = \text{linear}. \tag{2.36}$$

There are some reasons to think this second implementation is slightly better than the first one for some practical calculations, because the sensibility to variations with respect of $x$ or to the weights and biases seems to be less.

**Exercise 2.5.8.** *Check (2.31-2.33) and (2.34-2.36) generates the same function.*

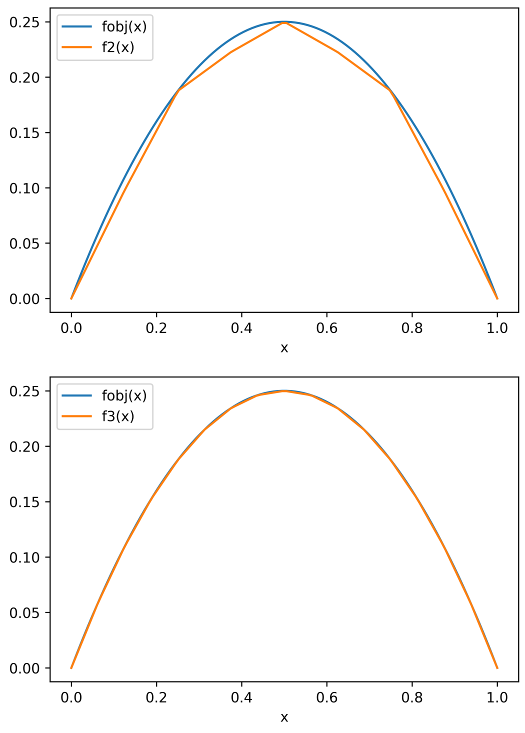

**Fig. 2.8:** Reconstruction of $f^{\mathrm{obj}}(x) = x(1-x)$ with Algorithm 6. On the left with 2 hidden layers. On the right with 3 hidden layers. Note that Algorithm 6 is not able to approximation more general objective functions such as in Figures 2.3 and 2.5.

The next question is to generates neural network approximation of all monomials $x \mapsto x^n$ for $n \in \mathbb{N}$. An easy answer follows which is based on the **polarization formula**

$$x^3 = \frac{1}{4}(x+x^2)^2 - \frac{1}{4}(x-x^2)^2. \tag{2.37}$$

The polarization formula expresses that a certain combination of linear functions and a quadratic function generates a cubic function. It can be generalized to generate monomials at any order.

**Lemma 2.5.9.** *Neural network approximations of $x \mapsto x^n$ are possible by various replications of a first one for $x \mapsto x^2$.*

*Proof.* Let us detail the proof for $n = 3$. Adding one more collocation channel to the structure of Figure 2.7, one constructs a neural network which realizes an approximation of the objective function $f_1^{\mathrm{obj}} : x \mapsto \begin{pmatrix} x^2 \\ x \end{pmatrix}$. One more objective function is

$$\begin{array}{rccl} f_2^{\mathrm{obj}} : & \mathbb{R}^2 & \to & \mathbb{R}^3 \\ & \begin{pmatrix} x \\ y \end{pmatrix} & \to & \begin{pmatrix} x \\ y \\ \frac{1}{2}(x+y) \end{pmatrix} . \end{array}$$

Since $f_2^{\mathrm{obj}}$ is linear, this construction is immediate. By construction one has $f_2^{\mathrm{obj}} \circ f_1^{\mathrm{obj}}(x) = \left(x^2, x, \frac{1}{2}(x^2+x)\right)^t$. A third objective function is $f_3^{\mathrm{obj}}(x,y,z) = (x^2, y^2, z^2)^t$. A fourth linear objective function is $f_4^{\mathrm{obj}}(x,y,z) = -\frac{1}{2}x - \frac{1}{2}y + 2z$. Then one checks that

$$f_4^{\mathrm{obj}} \circ f_3^{\mathrm{obj}} \circ f_2^{\mathrm{obj}} \circ f_1^{\mathrm{obj}}(x) = x^3 \tag{2.38}$$

which is a way to implement the polarization formula (2.37). With convenient neural network approximations to approximate $f_1^{\mathrm{obj}}$ and $f_3^{\mathrm{obj}}$, the proof is ended for $n = 3$. The construction is $x \mapsto x^4$ is done by squaring a square. Then $x^5 = \frac{1}{4}(x + x^4)^2 - \frac{1}{4}(x - x^4)^2$ allows to construct $x^5$. All monomials for $n \geq 6$ can be constructed in the same way. □

### 2.5.2 Yarotsky Theorem

The next result is a prelude to the Yarotsky Theorem [105].

**Definition 2.5.10.** *Given a small number $0 < \varepsilon < 1$, we will say that a function*

$$\mathrm{mul} : \mathbb{R}^2 \to \mathbb{R}$$

*is an approximate multiplier operator if it satisfies the properties*
- *for all $(x,y) \in \mathbb{R}^2$, one has $\mathrm{mul}(0,y) = \mathrm{mul}(x,0) = 0$,*
- *if $|x| \leq 1$ and $|y| \leq 1$, then $|\mathrm{mul}(x,y) - xy| \leq \varepsilon$,*

**Proposition 2.5.11.** *There exists a ReLU architecture which implements an approximate multiplier operator. The number of computational units of the network is bounded as $\#(N) \leq C\,(\log 1/\varepsilon + 1)$ where the constant $C > 0$ is independent of $\varepsilon$.*

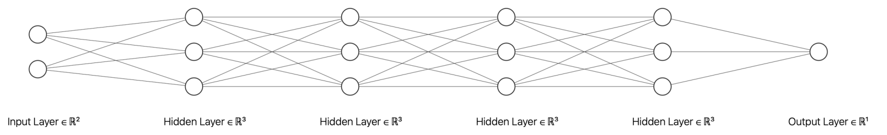

**Fig. 2.9:** Graph structure of an approximate multiplier $(x, y) \mapsto xy$ with $m = 2$, $p = 4$ and $n = 1$.

*Proof.* The proof is an elaborated version of the polarization formula (2.37). Consider the function $f_p$ defined in 2.29) and set

$$\text{mul}(x, y) = f_p\left(\frac{|x+y|}{2}\right) - f_p\left(\frac{|x-y|}{2}\right). \tag{2.39}$$

• One has by definition $\text{mul}(0, y) = \text{mul}(x, 0) = 0$.
• Next one notes that

$$\text{mul}(x,y) - xy = \left(f_p\left(\frac{|x+y|}{2}\right) - \left(\frac{|x+y|}{2}\right)^2\right) - \left(f_p\left(\frac{|x-y|}{2}\right) - \left(\frac{|x-y|}{2}\right)^2\right).$$

For $|x| \le 1$ and $|y| \le 1$, one has $0 \le \frac{|x \pm y|}{2} \le 1$. Therefore Lemma 2.5.7 can be used two times for the two terms between parentheses. It yields the bound $|\text{mul}(x,y) - xy| \le \frac{2}{3 \times 4^p}$. So one takes $\varepsilon = \frac{2}{3 \times 4^p}$ which shows that (2.39) is indeed an approximate multiplier operator.
• It remains to bound the number of computational units which is the product of the number of layers of the network times the width of the network. This is evident since

$$p = \frac{1}{4}\log\left(\frac{2}{3\varepsilon}\right) = \frac{1}{4}\log 1/\varepsilon + \frac{1}{4}\log\left(\frac{2}{3}\right).$$

The number of computational units is the number of layers times the number of neurons (3 for $f_p$), that is $O(p) \le C(\log 1/\varepsilon + 1)$. □

To go further, one defines a convenient partition of unity as follows. The partition of unity will be used to construct with local pieces an approximation of an objective function, as visible on the formula (2.49).

One starts with a continuous and piecewise affine cut-off function

$$\psi(x) = \begin{cases} 1 & |x| \le 1, \\ 2 - |x| & 1 \le |x| \le 2, \\ 0 & 2 \le |x|. \end{cases}$$

For $N \ge 1$ large enough, define

$$\phi_m(x) = \psi(3Nx - 3m) \text{ for } 0 \le m \le N. \tag{2.40}$$

The functions $(\phi_m)$ realize a partition of unity on the interval $[0,1]$

$$\sum_{m=0}^{N} \phi_m(x) = 1, \quad 0 \le x \le 1. \tag{2.41}$$

One defines the function $f_{mn}$

$$f_{mn}(x) = \mathrm{mul}\left( \phi_m(x), \underbrace{\mathrm{mul}\left(x - \frac{m}{N}, \mathrm{mul}\left(x - \frac{m}{N}, \dots\right)\dots\right)}_{n \text{ times}} \right). \tag{2.42}$$

The function $f_{mn}$ is a local approximation of $\phi_m(x)(x - m/N)^n$ so it will be possible to combine the partition of unity (2.41) with a local (around $m/N$) Taylor expansion based on shifted monomials $(x - m/N)^n$. One has first to characterize the properties of these functions.

**Lemma 2.5.12.** *The function $f_{mn}$ satisfies the properties*

- *For $x \notin supp(\phi_m)$, then $f_{mn}(x) = 0$.*
- *The error bound holds $\left|f_{mn}(x) - \phi_m(x)\left(x - \frac{m}{N}\right)^n\right| \le n\varepsilon$ for $0 \le x \le 1$.*
- *The function $f_{mn}$ can be implemented on a ReLU architecture with a number $\#(N_{mn})$ of computational unit bounded as $\#(N_{mn}) \le O(n)\,(\log 1/\varepsilon + 1)$.*

*Proof.* • The first property of a multiplier (see Definition 2.5.10) and the definition (2.42) yield $f_{mn}(x) = 0$.
• The second point comes from the telescopic decomposition with $n$ terms

$$\begin{aligned}
&f_{mn}(x) - \phi_m(x)(x - \tfrac{m}{N})^n \\
&= \left[ \mathrm{mul}\left( \phi_m(x), \underbrace{\mathrm{mul}\left(x - \frac{m}{N}, \dots\right)}_{n \text{ times}} \right) - \phi_m(x) \underbrace{\mathrm{mul}\left(x - \frac{m}{N}, \dots\right)}_{n \text{ times}} \right] \\
&+\phi_m(x) \left[ \underbrace{\mathrm{mul}\left(x - \frac{m}{N}, \dots\right)}_{n \text{ times}} - (x - \tfrac{m}{N}) \underbrace{\mathrm{mul}\left(x - \frac{m}{N}, \dots\right)}_{n-1 \text{ times}} \right] \\
&+ \dots \\
&+\phi_m(x)(x - \tfrac{m}{N})^{n-2} \left[ \underbrace{\mathrm{mul}\left(x - \frac{m}{N}, \dots\right)}_{2 \text{ times}} - (x - \tfrac{m}{N})^2 \right].
\end{aligned}$$

By construction $|\phi_m(x)| \le 1$ and $\left|x - \frac{m}{N}\right| \le 1$, so it is possible to bound the absolute value of each line by using the second estimate of Definition 2.5.10. All lines are

bound by $\varepsilon$. Adding the contributions, one gets $|f_{mn}(x) - \phi_m(x)(x-m/N)^n| \le n\varepsilon$ which is the second point.
• Finally let us evaluate the number of computational units $\#(N_{mn})$. A network that construct $f_{mn}$ can be obtained by placing $n$ successive multipliers one after the other and adding one more line of neurons (a collation channel) to propagate $x-m/N$ inside the neural network. Then one gets $\#(N_{mn}) \le Cn\,(\log 1/\varepsilon + 1)$. □

Next one constructs by tensorization a partition of unity in higher dimension for $\mathbf{x} = (x_1, \dots, x_d) \in [0,1]^d$

$$\phi_{\mathbf{m}}(\mathbf{x}) = \Pi_{k=1}^{d} \phi_{m_k}(x_k), \quad \mathbf{m} = (m_1, \dots, m_d) \in \{0, \dots, N\}^d \tag{2.43}$$

such that

$$\sum_{\mathbf{m}/N \in [0,1]^d} \phi_{\mathbf{m}}(\mathbf{x}) = 1, \quad \mathbf{x} \in [0,1]^d. \tag{2.44}$$

One also defines

$$f_{\mathbf{mn}}(\mathbf{x}) = \mathrm{mul}\left(f_{m_1 n_1}(x_1), \mathrm{mul}\left(f_{m_2 n_2}(x_2), \dots\right)\right). \tag{2.45}$$

**Corollary 2.5.13** (of Lemma 2.5.12). *The function $f_{\mathbf{mn}}$ satisfies the properties*

- *For $\mathbf{x} \notin supp(\phi_{\mathbf{m}})$, then $f_{\mathbf{mn}}(\mathbf{x}) = 0$.*
- *The error bound holds $\left| f_{\mathbf{mn}}(\mathbf{x}) - \phi_{\mathbf{m}}(\mathbf{x}) \Pi_{k=1}^{d} \left(x_k - \frac{m_k}{N}\right)^{n_k} \right| \le (d + |\mathbf{n}|)\varepsilon$ for $\mathbf{x} \in [0,1]^d$.*
- *The function $f_{\mathbf{mn}}$ can be implemented on a ReLU architecture with a number $\#(N_{\mathbf{mn}})$ of computational unit bounded as $\#(N_{\mathbf{mn}}) \le O(d + |\mathbf{n}|)(\log 1/\varepsilon + 1)$.*

*Proof.* • Necessarily, one has that $x_k \notin \mathrm{supp}(\phi_{m_k})$, at least for one index $k \in \{1, \dots, d\}$. The first property of an approximate multiplier yields the claim.
• All terms $f_{m_k n_k}(x_k)$ are less than one in absolute value for $|X_k| \le 1$. Therefore a telescopic decomposition like in the proof of Lemma 2.5.12 yields a first estimate

$$\left| f_{\mathbf{mn}}(\mathbf{x}) - \Pi_{k=1}^{d} f_{m_k n_k}(x_k) \right| \le d\varepsilon.$$

By using Lemma 2.5.12, the term $f_{m_k n_k}(x_k)$ approximates $\phi_{m_k}\left(x_k - \frac{m_k}{N}\right)^{n_k}$ with an error less than $n_k \varepsilon$. Since all these terms are less than one, a telescopic decomposition yields a second estimate

$$\left| \Pi_{k=1}^{d} f_{m_k n_k}(x_k) - \Pi_{k=1}^{d} \phi_{m_k}\left(x_k - \frac{m_k}{N}\right)^{n_k} \right| \le \left(\sum_{k=1}^{d} n_k\right) \varepsilon.$$

The triangular inequality and the notation $|\mathbf{n}| = \sum_{k=1}^{d} n_k$ shows the result.
• The cost is obtained by using Lemma 2.5.12 to estimate the cost of calculating all

individuals terms $f_{m_k n_k}(x_k)$ and the cost of multiplying these $d$ terms (Proposition 2.5.11). □

**Theorem 2.5.14** (Yarotsky ). *Let $\delta > 0$ and consider functions in the space $W^{r,\infty}[0,1]^d$. There exists a ReLU architecture with less than $O(\log 1/\delta + 1)$ layers and less than $C_r \delta^{-d/r}(\log 1/\delta + 1)$ computational units which is capable to reproduce any smooth function such that $\left\| f^{\text{obj}} \right\|_{W^{r,\infty}[0,1]^d} \leq 1$ with accuracy $\delta$.*

**Remark 2.5.15.** *This remarkable result shows that the* **best theoretical accuracy** *increases with respect to the regularity. Or more precisely the cost for obtaining a certain accuracy is asymptotically less for functions with increased regularity. However, once again, the increase of the best theoretical accuracy is not guarantee for practical calculations.*

*The inequality shows also a devastating possibility, which is that high dimensionality $d \gg 1$ dramatically increases the cost of obtaining this accuracy: this phenomenon is related to the* **curse of dimension**. *It will be discussed by other means in the sequel.*

*Proof.* The proof [105] is based on explicit manipulations and is reminiscent of high order Bernstein interpolation.

• The number of bounded derivatives of the objective function $f^{\text{obj}}$ is equal to $r$. Consider the polynomial $p_{\mathbf{m}}$ which is the truncation at order $r$ of the Taylor-expansion at $\frac{\mathbf{m}}{N}$

$$p_{\mathbf{m}}(\mathbf{x}) = \sum_{|\mathbf{n}| \leq r-1} \frac{1}{\mathbf{n}!} \frac{\partial^{|\mathbf{n}|} f^{\text{obj}}}{\partial x_1^{n_1} \dots \partial x_d^{n_d}} \left(\frac{\mathbf{m}}{N}\right) \left(\mathbf{x} - \frac{\mathbf{m}}{N}\right)^{|\mathbf{n}|}, \qquad \frac{\mathbf{m}}{N} \in [0,1]^d, \quad (2.46)$$

with the usual conventions $\mathbf{n}! = \Pi_{k=1}^d \frac{1}{n_k!}$ and $\left(\mathbf{x} - \frac{\mathbf{m}}{N}\right)^{|\mathbf{n}|} = \Pi_{k=1}^d \left(x_k - \frac{m_k}{N}\right)^{n_1 + \dots + n_k}$. Since $|\mathbf{n}| \leq r - 1$, then the total degree of these polynomials is bounded by $r - 1$. One has the bound

$$\left\| f^{\text{obj}} \right\|_{W^{r,\infty}[0,1]} = \max_{|\mathbf{n}| \leq r} \left\| \frac{\partial^{|\mathbf{n}|} f^{\text{obj}}}{\partial x_1^{n_1} \dots \partial x_d^{n_d}} \right\|_{L^\infty[0,1]} \leq 1,$$

so all coefficients are bounded

$$\left| \frac{1}{\mathbf{n}!} \frac{\partial^{|\mathbf{n}|} f^{\text{obj}}}{\partial x_1^{n_1} \dots \partial x_d^{n_d}} \left(\frac{\mathbf{m}}{N}\right) \right| \leq 1, \qquad 0 \leq |\mathbf{n}| \leq r - 1, \quad \frac{\mathbf{m}}{N} \in [0,1]^d. \quad (2.47)$$

• Then one constructs the function $f(x) = \sum_{\mathbf{m}/N \in [0,1]^d} \phi_{\mathbf{m}}(x) p_{\mathbf{m}}(x)$ which can be expressed as a double sum

$$f(\mathbf{x}) = \sum_{\mathbf{m}/N \in [0,1]^d} \sum_{|\mathbf{n}| \leq r-1} \frac{1}{\mathbf{n}!} \frac{\partial^{|\mathbf{n}|} f^{\text{obj}}}{\partial x_1^{n_1} \dots \partial x_d^{n_d}} \left(\frac{\mathbf{m}}{N}\right) \phi_{\mathbf{m}}(\mathbf{x}) \left(\mathbf{x} - \frac{\mathbf{m}}{N}\right)^{|\mathbf{n}|}.$$

• Using the partition of unity formula (2.43), the error between $f^{\mathrm{obj}}$ and $f$ can be bounded as

$$\begin{aligned}
\left|f^{\mathrm{obj}}(\mathbf{x}) - f(\mathbf{x})\right| &= \left|\sum_{\mathbf{m}/N\in[0,1]^d} \phi_{\mathbf{m}}(\mathbf{x})\left(f^{\mathrm{obj}}(\mathbf{x}) - p_{\mathbf{m}}(\mathbf{x})\right)\right| \\
&\le \sum_{\mathbf{m}\in\bigcap\{|x_k-m_k/N|<1/N\}} \left|f^{\mathrm{obj}}(\mathbf{x}) - p_{\mathbf{m}}(\mathbf{x})\right| \\
&\le 2^d \max_{\mathbf{m}\in\bigcap\{|x_k-m_k/N|<1/N\}} \left|f^{\mathrm{obj}}(\mathbf{x}) - p_{\mathbf{m}}(\mathbf{x})\right|.
\end{aligned}$$

A classical estimate of the remainder of the multivariate Taylor expansion between $f^{\mathrm{obj}}$ and $p_{\mathbf{m}}$ yields the bound

$$\left|f^{\mathrm{obj}}(\mathbf{x}) - f(\mathbf{x})\right| \le \frac{2^d d^r}{r!N^r}$$

Let us take $N$ equal to the integer part of $\left(\frac{r!\delta}{2^{d+1}d^r}\right)^{-1/r}$ plus one (the ceiling function) so that $N^r \ge \frac{r!\delta}{2^{d+1}d^r}$. Under this condition, one has $\left|f^{\mathrm{obj}}(x) - f(x)\right| \le \frac{\delta}{2}$. It is a standard exercise to show that there exists a constant $C > 0$ such that

$$N \le C\frac{d}{\delta^{1/r}}. \tag{2.48}$$

• In view of the technical Proposition 2.5.11, it is natural to define

$$g(\mathbf{x}) = \sum_{\mathbf{m}/N\in[0,1]^d}\sum_{|\mathbf{n}|\le r-1} \frac{1}{\mathbf{n}!}\frac{\partial^{|\mathbf{n}|}f^{\mathrm{obj}}}{\partial x_1^{n_1}\dots\partial x_d^{n_d}}\left(\frac{\mathbf{m}}{N}\right) f_{\mathbf{mn}}(\mathbf{x}). \tag{2.49}$$

The function $g$ is implementable in a neural network and is a priori close to $f$. The difference is

$$\begin{aligned}
& f(\mathbf{x}) - g(\mathbf{x}) \\
= & \sum_{\mathbf{m}/N\in[0,1]^d}\sum_{|\mathbf{n}|\le r-1} \frac{1}{\mathbf{n}!}\frac{\partial^{|\mathbf{n}|}f^{\mathrm{obj}}}{\partial x_1^{n_1}\dots}\left(\frac{\mathbf{m}}{N}\right)\left[\phi_{\mathbf{m}}(\mathbf{x})\left(\mathbf{x} - \frac{\mathbf{m}}{N}\right)^{|\mathbf{n}|} - f_{\mathbf{mn}}(\mathbf{x})\right] \\
= & \sum_{\mathbf{m}|\mathbf{x}\in\mathrm{supp}(\phi_{\mathbf{m}})}\sum_{|\mathbf{n}|\le r-1} \frac{1}{\mathbf{n}!}\frac{\partial^{|\mathbf{n}|}f^{\mathrm{obj}}}{\partial x_1^{n_1}\dots}\left(\frac{\mathbf{m}}{N}\right)\left[\phi_{\mathbf{m}}(\mathbf{x})\left(\mathbf{x} - \frac{\mathbf{m}}{N}\right)^{|\mathbf{n}|} - f_{\mathbf{mn}}(\mathbf{x})\right] \\
= & 2^d \max_{\mathbf{m}|\mathbf{x}\in\mathrm{supp}(\phi_{\mathbf{m}})} \sum_{|\mathbf{n}|\le r-1}\frac{1}{\mathbf{n}!}\left|\phi_{\mathbf{m}}(\mathbf{x})\left(\mathbf{x} - \frac{\mathbf{m}}{N}\right)^{|\mathbf{n}|} - f_{\mathbf{mn}}(\mathbf{x})\right| \\
\le & 2^d e^d (d+r)\varepsilon,
\end{aligned}$$

where the last inequality comes Corollary 2.5.13 and $\sum_{|\mathbf{n}|\le r-1}\frac{1}{\mathbf{n}!} \le \sum_{\mathbf{n}\in\mathbb{N}^d}\frac{1}{\mathbf{n}!} = \left(\sum_{n\in\mathbb{N}}\frac{1}{n!}\right) = e^d$.

• Take the multiplier accuracy $\varepsilon$ such that $2^d e^d (d+r)\varepsilon = \frac{\delta}{2}$. Then the triangular inequality yields

$$\left|f^{\text{obj}}(\mathbf{x}) - g(\mathbf{x})\right| \le \left|f^{\text{obj}}(\mathbf{x}) - f(\mathbf{x})\right| + |f(\mathbf{x}) - g(\mathbf{x})| \le \frac{\delta}{2} + \frac{\delta}{2} = \delta$$

which the accuracy claimed in the Theorem.
• Finally let us evaluate the complexity of the network needed to evaluate the function $g$. It is possible to consider a network which evaluates **in parallel** all the functions $f_{\mathbf{mn}}$, and then which makes a linear combination with a collation channel. So the number of layers scales as for the approximate multiplier.
The number of computational units is

$$\#(N_{\text{total}}) = O\left(\sum_{|\mathbf{n}|\le r-1}\ \sum_{|\mathbf{m}|/N\in[0,1]^d} \#\left(N_{\mathbf{mn}}\right)\right)$$

$$\le r(N+1)^d \left[C(d+r)\left(\log 1/\varepsilon + 1\right)\right] \le C(r,d)\delta^{-d/r}\left(\log 1/\delta + 1\right)$$

for some constant $C(r,d)$. □

### 2.5.3 Takagi function

The Takagi function [94] illustrates a fascinating property of neural networks, which is their ability to approximate very irregular functions. This feature is totally counterintuitive with respect to the standard approximation theory of smooth functions. The Takagi function is a particular infinite linear combination of the Haar functions, it is continuous but nowhere pointwise differentiable [2]. It was described in 1901 by Takagi in a 2-page note as a simple example of a continuous function without derivative. The Takagi function is connected to the theory of Brownian motion, one can consult [28][pages 46 to 53]. There is probably more to understand about the use of the Takagi function and similar functions for the approximation of noisy time series, even if it is completely out of the scope of these notes.

**Definition 2.5.16.** *The Takagi function* $\mathrm{Tak} \in C^0[0,1]$ *is*

$$\mathrm{Tak}(x) = \sum_{n\ge 1} \frac{1}{2^n} g_n(x).$$

*Its truncation* $\mathrm{Tak}_p \in C^0[0,1]$ *at stage* $p \ge 1$ *is:* $\mathrm{Tak}_p(x) = \sum_{n=1}^{p} \frac{1}{2^n} g_n(x)$.

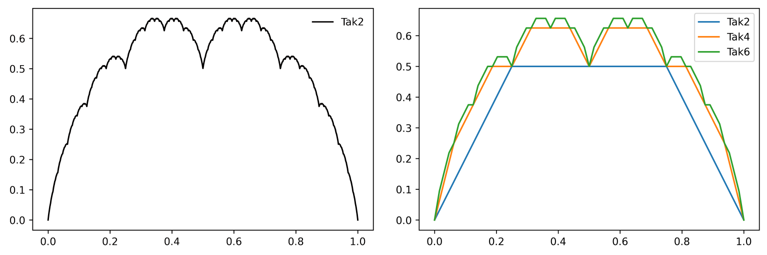

**Fig. 2.10:** On the left, plot of the Takagi function. On the right, plot of the truncated Takagi functions $\mathrm{Tak}_2$, $\mathrm{Tak}_4$ and $\mathrm{Tak}_6$ (algorithm 6 can be used to draw these curves).

**Lemma 2.5.17.** *One has the symmetry property* $\mathrm{Tak}(1-x) = \mathrm{Tak}(x)$ *and the self similar property*

$$\mathrm{Tak}\left(\frac{x}{2}\right) = \frac{1}{2}x + \frac{1}{2}\mathrm{Tak}(x). \tag{2.50}$$

*Proof.* The symmetry comes from the symmetry of $g$. The self similarity is deduced from

$$\mathrm{Tak}\left(\frac{x}{2}\right) = \frac{1}{2}g\left(\frac{x}{2}\right) + \sum_{n\geq 2}\frac{1}{2^n}g_n(x/2) = \frac{1}{2}x + \sum_{n\geq 2}\frac{1}{2^n}g_{n-1}(x) = \frac{1}{2}x + \frac{1}{2}\mathrm{Tak}(x).$$

□

**Lemma 2.5.18.** *The Takagi function can be approximated in the maximal norm on a neural network with p hidden dense layers with 3 neurons with the ReLU function as activation function, with error* $O(2^{-p})$.

*Proof.* Adapt the proof of Lemma 2.5.7. □

The following results propose a self contained characterization of the fact that the Takagi function is not differentiable in Lebesgue spaces $L^p(0,1)$, $1 \leq p \leq \infty$. This property can of course be deduced from the material in [2].

**Definition 2.5.19.** *Let* $g \in C^0[0,1]$ *be a continuous function. Let* $\sigma = (0 \leq a = x_0 < x_1 < \cdots < x_r = b \leq 1)$ *be a subdivision of the interval* $[a,b] \subset [0,1]$.
*The variation of* $g$ *relatively to the subdivision* $\sigma$ *is*

$$V(g,\sigma) = \sum_{s=0}^{r-1} |g(x_{s+1}) - g(x_r)|\,.$$

*The total variation of the function* $g$ *is* $TV(g) = \sup_\sigma V(g,\sigma)$.

For a $g \in C^0[0,1]$, there are two cases. Either the total variation is finite, or it is infinite. If it is infinite, then the $g$ cannot have a derivative in any $L^p(0,1)$, $1 \le p \le \infty$. Indeed one has the classical inequalities

$$V(g,\sigma) \le \|g'\|_{L^1(0,1)} \le \|g'\|_{L^p(0,1)} \quad 1 \le p \le \infty.$$

The first inequality comes from the evident bound $|g(x_{s+1}) - g(x_r)| \le \int_{x_r}^{x_{r+1}} |g'(x)|dx$. The second inequality comes from the Hölder inequality [88, 28]. The consequence is that a function with an infinite total variation is not differential in $L^p(0,1)$ for all $1 \le p \le \infty$.

**Proposition 2.5.20.** *The total variation of the Takagi function is infinite.*

*Proof.* • Take $p \in \mathbb{N}$ and the equi-distributed subdivision $\sigma_p$ defined by

$$x_i = \frac{i}{2^p}, \quad 0 \le i \le 2^p.$$

By (2.21), one has $g_n(0) = 0$. The periodicity of $g_n$ induced by (2.23) yields that

$$g_n(x_i) = 0 \text{ for all } x_i = \frac{i}{2^p} \text{ and all } n \ge p+1. \tag{2.51}$$

So

$$\text{Tak}(x_i) = \sum_{n=1}^{p} \frac{1}{2^n} g_n(x_i) + \sum_{n=p+1}^{\infty} \frac{1}{2^n} g_n(x_i) = \sum_{n=1}^{p} \frac{1}{2^n} g_n(x_i) = \text{Tak}_p(x_i).$$

So

$$V(\text{Tak}, \sigma_p) = V(\text{Tak}_p, \sigma_p), \tag{2.52}$$

where the function $\text{Tak}_p$ is easier to evaluate since it is generated with a finite sum.
• One has

$$\begin{aligned} V(\text{Tak}_p, \sigma_p) &= \sum_{i=0}^{2^p-1} \left| \text{Tak}_p\left(\frac{i+1}{2^p}\right) - \text{Tak}_p\left(\frac{i}{2^p}\right) \right| \\ &= \sum_{i=0}^{2^{p-1}-1} \left[ \left| \text{Tak}_p\left(\frac{i+1}{2^{p-1}}\right) - \text{Tak}_p\left(\frac{i}{2^{p-1}} + \frac{1}{2^p}\right) \right| \right. \\ &\quad \left. + \left| \text{Tak}_p\left(\frac{i}{2^{p-1}} + \frac{1}{2^p}\right) - \text{Tak}_p\left(\frac{i}{2^{p-1}}\right) \right| \right] \end{aligned}$$

and

$$V(\text{Tak}_p, \sigma_{p-1}) = \sum_{i=0}^{2^{p-1}-1} \left| \text{Tak}_p\left(\frac{i+1}{2^{p-1}}\right) - \text{Tak}_p\left(\frac{i}{2^{p-1}}\right) \right|$$

The difference is

$$V(\mathrm{Tak}_p,\sigma_p)-V(\mathrm{Tak}_p,\sigma_{p-1})=\sum_{i=0}^{2^{p-1}-1}\left[\left|\mathrm{Tak}_p\left(\frac{i+1}{2^{p-1}}\right)-\mathrm{Tak}_p\left(\frac{i}{2^{p-1}}+\frac{1}{2^p}\right)\right|\right.$$

$$\left.+\left|\mathrm{Tak}_p\left(\frac{i}{2^{p-1}}+\frac{1}{2^p}\right)-\mathrm{Tak}_p\left(\frac{i}{2^{p-1}}\right)\right|-\left|\mathrm{Tak}_p\left(\frac{i+1}{2^{p-1}}\right)-\mathrm{Tak}_p\left(\frac{i}{2^{p-1}}\right)\right|\right]. \tag{2.53}$$

The terms between $[\dots]$ in (2.53) are of the form $|\alpha-\beta|+|\beta-\gamma|-|\alpha-\gamma|$. The triangular inequality yields $|\alpha-\beta|+|\beta-\gamma|-|\alpha-\gamma|\geq 0$ so one obtains the inequality $V(\mathrm{Tak}_p,\sigma_p)-V(\mathrm{Tak}_p,\sigma_{p-1})\geq 0$. Using (2.51) one has also $V(\mathrm{Tak}_p,\sigma_{p-1})=V(\mathrm{Tak}_{p-1},\sigma_{p-1})$, so it yields the inequalities

$$V(\mathrm{Tak}_p,\sigma_p)-V(\mathrm{Tak}_{p-1},\sigma_{p-1})\geq 0\quad p\geq 1$$

which already shows that $p\mapsto V(\mathrm{Tak}_p,\sigma_p)$ is an increasing sequence.

• Next we assume that $p$ is odd and we pose $2q=p-1$. We use Lemma 2.5.21 to characterize all terms on the intervals $\left[\frac{i}{2^{p-1}},\frac{i+1}{2^{p-1}}\right]$ where $\mathrm{Tak}_{2q}=\mathrm{Tak}_{p-1}$ is constant. Let us set

$$\alpha=\mathrm{Tak}_{p-1}\left(\frac{i+1}{2^{p-1}}\right)\text{ and }\gamma=\mathrm{Tak}_{p-1}\left(\frac{i}{2^{p-1}}\right)$$

which are such that $\alpha=\gamma$. Using (2.51) once again, one has

$$\mathrm{Tak}_p\left(\frac{i+1}{2^{p-1}}\right)=\alpha=\gamma=\mathrm{Tak}_p\left(\frac{i}{2^{p-1}}\right).$$

One also has

$$\mathrm{Tak}_p\left(\frac{i}{2^{p-1}}+\frac{1}{2^p}\right)=\mathrm{Tak}_p\left(\frac{i}{2^{p-1}}+\frac{1}{2^p}\right)+\frac{1}{2^p}=\alpha+\frac{1}{2^p}.$$

It has been already noticed that the terms between $[\dots]$ in (2.53) form a signed triangular inequality. Here the value is

$$|\alpha-\beta|+|\beta-\gamma|-|\alpha-\gamma|=2\frac{1}{2^p}=\frac{1}{2^{2q}}.$$

• Therefore one has that $V(\mathrm{Tak}_{2q+1},\sigma_{2q+1})\geq V(\mathrm{Tak}_{2q},\sigma_{2q})+\frac{(2q)!}{(q!)^2 2^{2q}}$. Actually one has that the inequality is in fact an equality and that $V(\mathrm{Tak}_{2q+1},\sigma_{2q+2})=V(\mathrm{Tak}_{2q+1},\sigma_{2q+1})$ (left to the reader). A consequence of (2.52) is that one can write for all $Q\geq 1$

$$TV(\mathrm{Tak})\geq\sum_{q=1}^{Q}\frac{1}{\sqrt{q}}\frac{(2q)!}{(q!)^2 2^{2q}}. \tag{2.54}$$

• It is a standard application of the Stirling formula to show that

$$\frac{(2q)!}{(q!)^2 2^{2q}} \approx \frac{\sqrt{4\pi q}\,(2q/e)^{2q}}{\left(\sqrt{2\pi q}\,(q/e)^q\right)^2 2^{2q}} \approx \sqrt{\pi/q}.$$

Therefore the series on the right hand side of (2.54) diverges, so the claim. □

**Lemma 2.5.21.** *The function* $\mathrm{Tak}_{2q}$ *is constant on exactly* $\frac{(2q)!}{(q!)^2}$ *intervals of the form* $\left[\frac{i}{2^{2q}}, \frac{i+1}{2^{2q}}\right]$ *for* $0 \le i \le 2^{2q} - 1$.
*The function* $\mathrm{Tak}_{2q+1}$ *is constant on no intervals of the form* $\left[\frac{i}{2^{2q+1}}, \frac{i+1}{2^{2q+1}}\right]$ *for* $0 \le i \le 2^{2q+1} - 1$.

*Proof.* The claim is already visible on Figure 2.10.
Let $p \le n = 2^q$. Actually the function $\frac{1}{2^p} g_p(x)$ has a slope $\pm 1$ on all $2^{p-1}$ on successive intervals of length $\frac{1}{2^p}$. So one can as well consider that the function $\frac{1}{2^p} g_p(x)$ has a slope $\pm 1$ on all $2^{n-1}$ on the successive intervals of length $\frac{1}{2^n}$. The case $n = 4$ is illustrated in Table 2.1. So if one focuses on the interval $\left[\frac{i}{2^{2q}}, \frac{i+1}{2^{2q}}\right]$,

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $g_1$ | + | + | + | + | + | + | + | + | - | - | - | - | - | - | - | - |
| $g_2$ | + | + | + | + | - | - | - | - | + | + | + | + | - | - | - | - |
| $g_3$ | + | + | - | - | + | + | - | - | + | + | - | - | + | + | - | - |
| $g_4$ | + | - | + | - | + | - | + | - | + | - | + | - | + | - | + | - |

**Tab. 2.1:** The sign of the slope of $g_p$ $(1 \le p \le n)$ on $2^n$ successive intervals of length $\frac{1}{2^n}$. Here $n = 4$.

then the slope of $\mathrm{Tak}_{2q}$ is $\underbrace{\pm 1 \pm 1 \cdots \pm 1 \pm 1}_{\pm 2q \text{ times}}$. This number vanishes if and only if the number of $+1$ is exactly equal to $q$. Then the number of $-1$ is also exactly equal to $q$. The number of possibilities is exactly $\frac{(2q)!}{(q!)^2}$, so the claim for $\mathrm{Tak}_{2q}$.
For $\mathrm{Tak}_{2q+1}$ the number of lines in Table 2.1 is an odd integer, so the slope cannot be zero. □

Next we give a separate proof that the Takagi function is not in $H^1(0,1)$. The proof has its own interest because it uses the orthogonality of the functions $g_n'$.

**Proposition 2.5.22.** *Whatever* $0 \le a < b \le 1$, *then* $\mathrm{Tak} \notin H^1(a,b)$.

*Proof.* • Take a small enough interval $J = \left[\frac{k}{2^{m-1}}, \frac{k+1}{2^{m-1}}\right] \subset [a,b]$ for $0 \leq k \leq 2^{m-1}-1$ where $m$ is large enough. The self-similar properties of the $g_n$'s yield that

$$\mathrm{Tak}\left(\frac{k}{2^{m-1}}+x\right) = \sum_{n=1}^{m-1} \frac{1}{2^n} g_n(x) + \frac{1}{2^{m-1}}\mathrm{Tak}(2^{m-1}x).$$

The functions $g_n$'s are piecewise affine so they belong to the space $H^1(0,1)$. When $x$ spans the small interval $J \subset [0,1]$, then $2^{m-1}x - k$ spans the entire interval $[0,1]$. Therefore it is equivalent to study the function on the full interval $[0,1]$, (note that the same argument can be used for the study of the total variation of the Takagi function),

$$\mathrm{Tak} \notin H^1\left(\frac{k}{2^{m-1}}, \frac{k+1}{2^{m-1}}\right) \Longleftrightarrow \mathrm{Tak} \notin H^1(0,1).$$

• One notices the orthogonal properties of the derivatives of the $g_r$

$$\int_0^1 g_r'(x) g_s'(x) dx = 2^{2r}\delta_{rs}. \tag{2.55}$$

This property is natural because the derivatives of the $g_r$ are linear combinations of the Haar functions 2.57 and it is evident at the inspection of the saw-tooth structure in Figure 2.6 (the proof is left to the reader).
• Consider the truncated series

$$\mathrm{Tak}_N(x) = \sum_{n=1}^{N} \frac{1}{2^n} g_n(x). \tag{2.56}$$

A direct calculation shows that $\left\|(\mathrm{Tak}_N)'\right\|^2_{L^2(0,1)} = \sum_{n=1}^N \frac{1}{4^n}2^{2n} = N \underset{N\to\infty}{\longrightarrow} \infty$. Therefore $\mathrm{Tak} \notin H^1(0,1)$. □

**Definition 2.5.23.** *The Haar wavelet is*

$$h(x) = \begin{cases} 1 & 0 < x < \frac{1}{2}, \\ -1 & \frac{1}{2} < x < 1, \\ 0 & \textit{otherwise.} \end{cases}$$

*The Haar functions are rescaled and shifted from the Haar wavelet*

$$H_k^j(x) = 2^{\frac{j}{2}} h(2^j x - k), \quad j,k \in \mathbb{Z}. \tag{2.57}$$

The Haar functions are an orthonormal basis of $L^2(\mathbb{R})$, that is

$$\int_{\mathbb{R}} H_k^j(x) H_{k'}^{j'}(x) dx = \delta_{kk'}\delta_{jj'}$$

and
$$\forall f \in L^2(\mathbb{R}), \text{ one has } f = \sum_{jk} f_k^j H_k^j \text{ where } f_k^j = \int_{\mathbb{R}} f(x) H_k^j(x) dx.$$
The convergence of the series is in the quadratic norm. Schauder proved that the Haar functions are also a basis [6][page 110] for the spaces $L^p(\mathbb{R})$, $1 \le p$.

**Remark 2.5.24.** *For the functions $h_r = g_r - \frac{1}{2}$, one has the orthogonality property $\int_0^1 h_r(x) h_s(x) dx = 3\delta_{rs}$.*

## 2.6 Summary of the chapter

It is shown in this chapter that Approximation Theory provides many ways to explain that Neural-Network-generated functions can approximate general objective functions. The Cybenko Theorem focuses on functions with sigmoid activation functions. Among other ideas, the Finite Element Method (FEM) can be used to explain the accuracy of functions generated with ReLU activation function. The Yarotsky Theorem is the most general result adapted to the ReLU activation function. It is based on a series with fast convergence which is an adaptation of the generating series of the Takagi function.

A drawback of these theoretical results is that they do not provide practical prescriptions to construct Neural-Network-generated functions. After a detour next Chapter on a functional equation with strong contraction properties which provides an alternative to the polarization formula for the approximation of polynomial objective functions, practical methods will be evaluated in the other Chapters.