IFT 6085 - Lecture 10
Expressivity and Universal Approximation Theorems Part 1

This version of the notes has not yet been thoroughly checked. Please report any bugs to the scribes or instructor.

Scribes
Winter 2020: Téo Orthlieb
Winter 2019: Moustafa Elarabi, Kun Ni

Instructor: Ioannis Mitliagkas

1 Summary

In this lecture we will discuss the expressivity of neural networks, that is what kind of functions a neural network could approximate. We will see the limitations of wide neural networks and the expressive power of going deeper.

2 Introduction

2.1 Capacity of the Perceptron

What kind of function can a neural network represent ?

Let's start with the simplest neural network, the Perceptron. It is effectively a NN with a single hidden layer having 1 hidden unit with an activation function σ. Both the input layer and the weights are a $1 \times n$ vector.

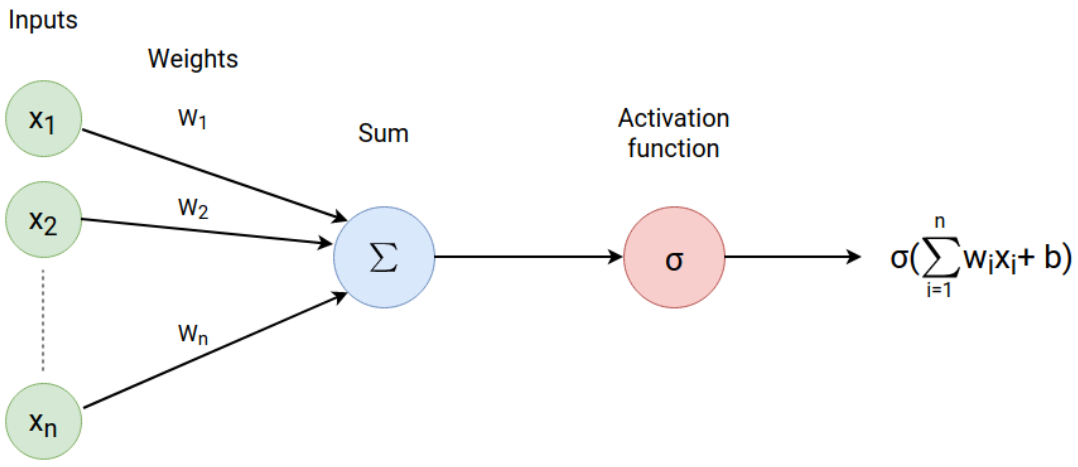


Figure 1: The Perceptron

The perceptron can model a function of this form: $\sigma(\sum_i^n \mathbf{w_i} x_i + b) = \sigma(\mathbf{w}^\top x + b)$. Typical σ can be:
sigmoid: $\frac{1}{1+e^{-x}}$ tanh: $\frac{e^{2x}-1}{e^{2x}+1}$ ReLU: $max(0, x)$ $\cdots$

Notably, the Perceptron is a linear classifier, and as such it famously can't model an XOR [7].

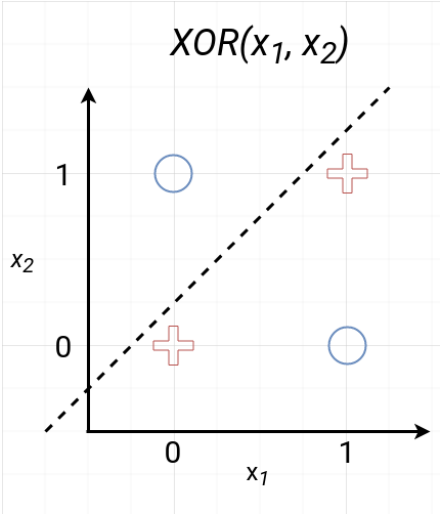


Figure 2: XOR is not linearly separable

2.2 Capacity of multiple neurons

By allowing ourselves more than 1 neuron in the hidden layer, we can model a XOR and in fact, we get the simplest **universal approximator**.

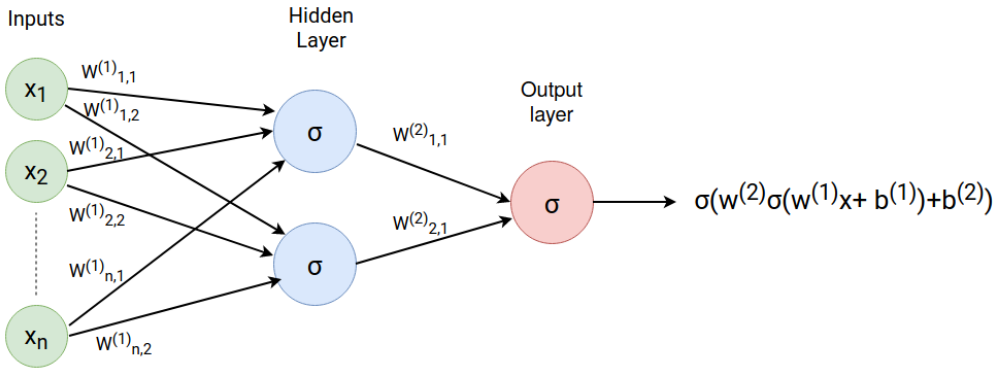


Figure 3: A NN with 2 hidden units

3 Universal Approximation Theorem

The universal approximation theorem states that any continuous function $f : [0, 1]^n \to [0, 1]$ can be approximated arbitrarily well by a neural network with at least 1 hidden layer with a finite number of weights, which is what we are going to illustrate in the next subsections.

3.1 Visual proof of Universal Approximation

In this section we will present a good intuition for the universal approximation theorem by making a summary of this page http://neuralnetworksanddeeplearning.com/chap4.html. (All credit is due to Michael A. Nielsen for all the

pictures in this subsection)

Say we want to approximate a function with 1 input and 1 output like so:

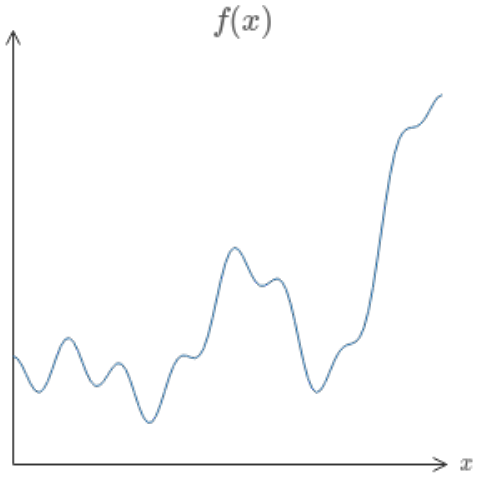


Figure 4: A continuous function

We will first consider a simple NN with 2 hidden neurons that have a sigmoid activation function, and for now the output neuron will just be linear.

Step 1 Make a step function with 1 of the neuron.

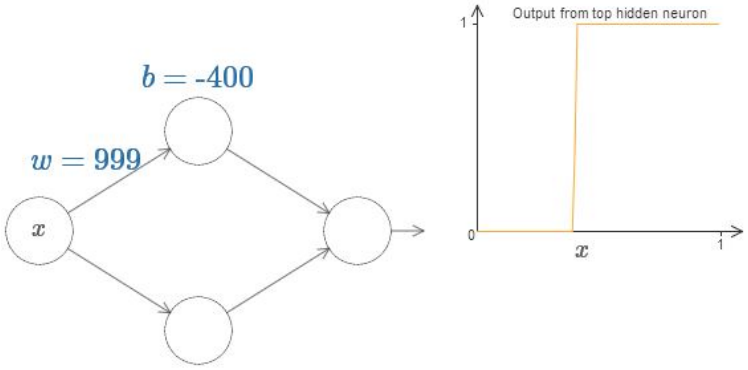


Figure 5: Making a step function with the top neuron

Let's focus on the top hidden neuron first, by using a big weight on the top neuron we can approximate the step function with a sigmoid arbitrarily well, and by adjusting the bias we can place it anywhere.
(Sidenote: the same argument can be made for the tanh activation, but not for ReLu)
In this toy example, we won't be interested in changing the weights of the first layer, they just have to be high enough, so we will just consider them to be constant.
Additionnaly, to make the plots clearer, we will display the position of the step instead of the bias, which is easily computed with $s = -\frac{b}{w}$.
With these changes, the plot above becomes:

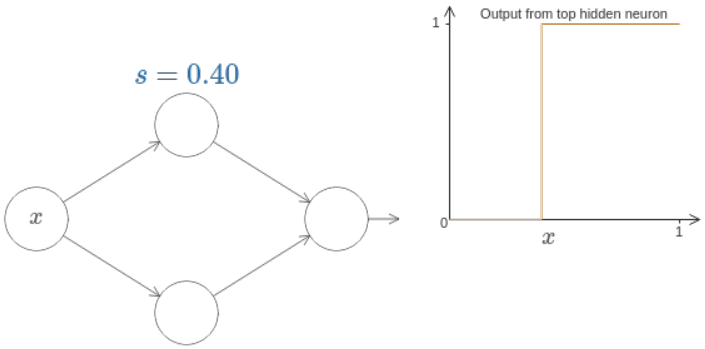


Figure 6: Making a step function with the top neuron

Step 2 Make a ”bin” with an opposite step function.

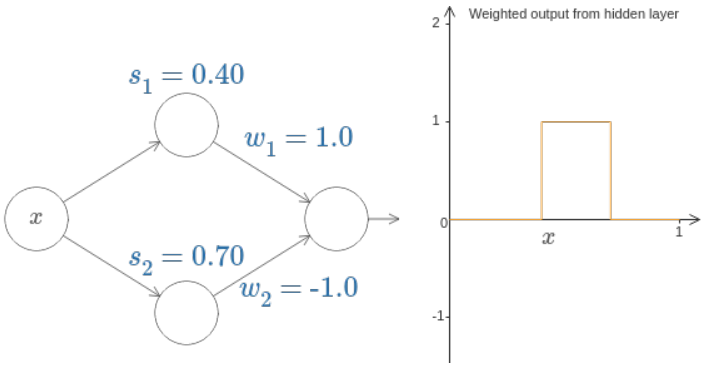


Figure 7: Making a bin with 2 opposite step functions

As illustrated above, by using the other neuron to make a step function, and setting opposing weights in the second layer, we can effectively approximate a bin and control its position, size and height.
Now you can probably see where this is going, to make things even clearer, we will just use 1 value for both w_1 and $-w_2$, called h, representing the height of the ”bin”.

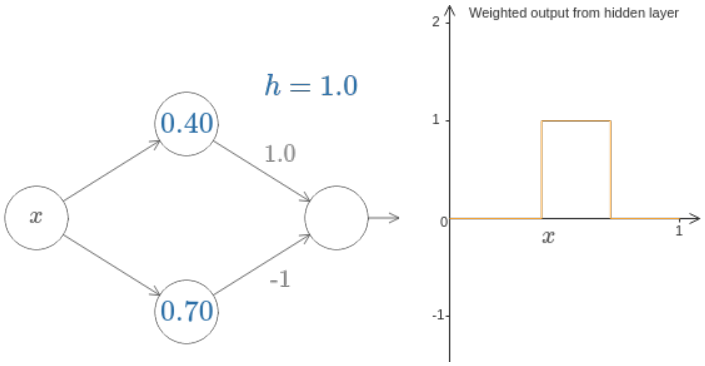


Figure 8: Making a bin with 2 opposite step functions

Step 3 Discretize the function.

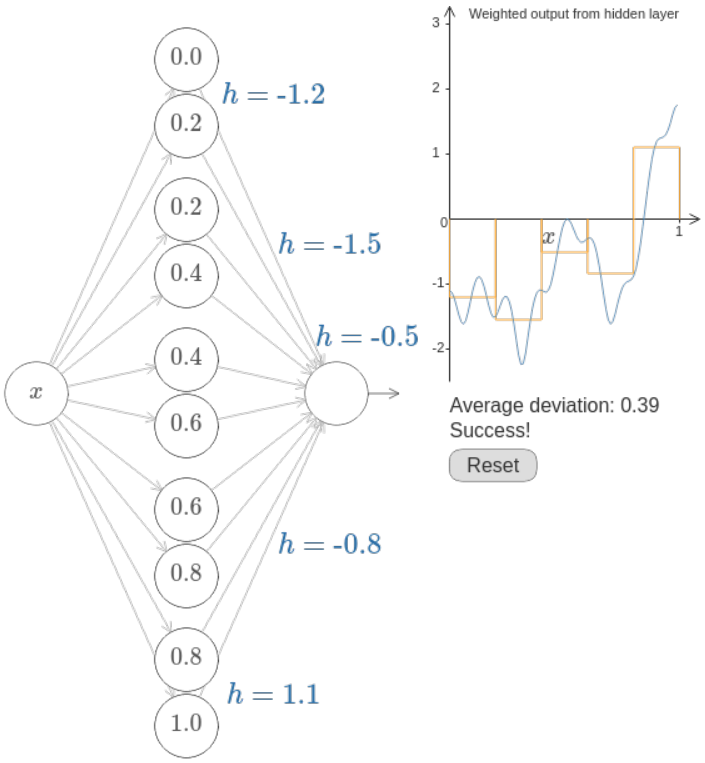


Figure 9: Approximating f with an histogram

In this final step, we combine several "bins" to make an histogram approximating the function. Illustrated above is a very rough approximation using only 5 bins (10 hidden units), but we can obviously make it sharper by simply adding more bins.

$\rightarrow$ Exercise for the reader: if we wanted to use this technique to approximate an *L-Lipschitz* function $f : \mathbb{R} \rightarrow \mathbb{R}$ on the interval $[0, 1]$ with an error at most ϵ at any point, how many bins would we need ?

what if i don't want linear neurons in the output layer ? The above network's output layer is linear, giving us the histogram approximating f, if we add a sigmoid activation function on the output we just have to approximate $\sigma^{-1} \circ f$ instead of f, which we can do with the same method.

3.2 Cybenko Approximation by Superposition of Sigmoidal Function

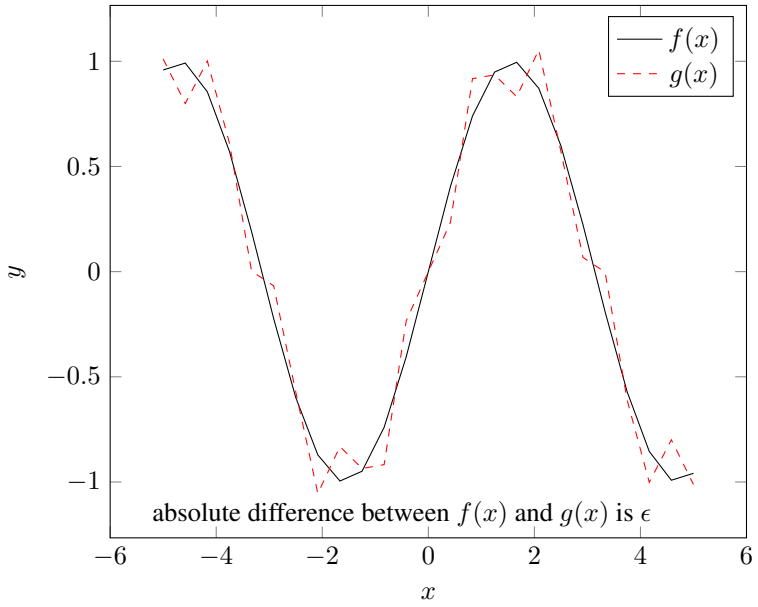
In this subsection we will present the first proven result for Universal Approximation limited to sigmoidal functions, by Cybenko [2].
We first define a *sigmoidal function* σ as:

$$\sigma(x) \rightarrow \begin{cases} 1 & \text{as } x \rightarrow +\infty \\ 0 & \text{as } x \rightarrow -\infty \end{cases}$$

Note: while sigmoidal functions are usually assumed to be monotonic increasing, this assumption is not necessary for this result.

Theorem 1. *Let $C([0,1]^n)$ denote the set of all continous function $[0,1]^n \rightarrow \mathbb{R}$, let σ be any sigmoidal activation function then the finite sum of the form $f(x) = \sum_{i=1}^{N} \alpha_i \sigma(\mathbf{w_i}^\top x + b_i)$ is dense in $C([0,1]^n)$*

Informaly, this theorem is saying that for any $g \subset C([0,1]^n)$ and any $\epsilon > 0$, there exists $f : x \to \sum_{i=1}^{N} \alpha_i \sigma(\mathbf{w_i}^\top x + b)$ such that $|f(x) - g(x)| < \epsilon$ for all $x \subset [0,1]^n$.



A similar result was independently obtained by Hornik[4] and also by Funahashi[3] using different tools. Hornik's proof relies on the Stone-Weierstrass Theorem which states that every continuous function defined on a closed interval $[a, b]$ can be uniformly approximated as closely as desired by a polynomial function.

3.3 Kolmogorov-Arnold Representation theorem

The Kolmogorov-Arnold representation theorem (or superposition theorem) [5] states that every multivariate continuous function can be represented as a superposition of continuous functions of one variable.
It solved a more general form of Hilbert's thirteenth problem [1] which was questioning whether a solution to 7^{th} degree equations could be expressed by a finite sum of two-variable functions.

Theorem 2. *Any continuous function $f : [0,1]^n \to \mathbb{R}$ can be written as*

$$f(x) = f(x_1, .., .., x_n) = \sum_{q=1}^{Z_m} \phi_q \left(\sum_{q=1}^{m} \Psi_q(x_q) \right)$$

This implies, among other things, that if we could chose the non-linearity of each unit we can represent any continuous function **exactly** with a NN with 1 hidden layer.

4 The expressive power of Deep neural networks

4.1 A view from the width (Lu et al. 2017)

In the previous sections, we focused on the setting of depth-bounded (e.g. depth-2) neural networks. With [6] we're going to see some interesting results for **width-bounded** neural networks instead !

Theorem 3. *(Universal Approximation Theorem for Width-Bounded ReLU Networks). For any Lebesgue-integrable function $f : \mathbb{R}^n \to \mathbb{R}$ and any $\epsilon > 0$, there exists a fully-connected ReLU network A with width $d_m \leq n + 4$, such that the function F_A represented by this network satisfies*

$$\int_{\mathbb{R}^n} |f(x) - F_A(x)| dx < \epsilon$$

This theorem states that any continuous function $f : \mathbb{R}^n \to \mathbb{R}$ can be approximated by a deep ReLU network with width $\leq n + 4$.

Theorem 4. *Let n be the input dimension. For any integer $k \geq n + 4$ there exists $F_\alpha : \mathbf{R^n} \to \mathbf{R}$ represented by a relu neural network α with width $d_m = 2k^2$ and depth $h = 3$ such that for any constant $b > 0$, there exists an $\epsilon > 0$ and for any function $F_\beta : \mathbf{R^n} \to \mathbf{R}$ represented by a ReLu neural network β whose parameters are bounded in $[-b, b]$ with width $d_m \leq k^{\frac{3}{2}}$ and depth $h \leq k + 2$ the following inequality holds*

$$\int_{\mathbf{R}} |F_\alpha - F_\beta| dx \geq \epsilon$$

This theorem states that there are networks such that reducing width requires increasing in the size to compensate, which is similar to that of depth qualitatively.

4.2 Representation benefits of deep NN (Telgarsky 2015)

In this section, we want to show interesting results from [8], that will allow us to compare the expressivity of wide networks against deep and recurrent networks, on a specific classification problem defined below:

Let n-ap (*n-alternating-points*) be the set of $n := 2^k$ points uniformly spaced within $[0, 1 - 2^{-k}]$ with alternating labels.

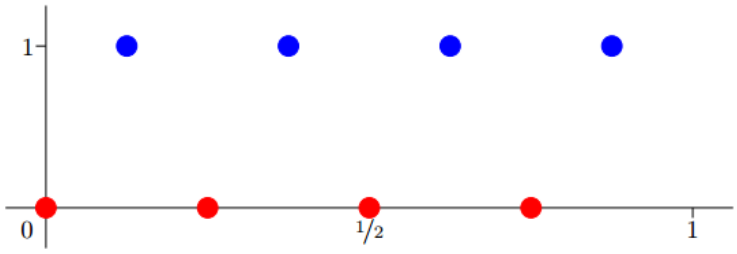


Figure 10: The 2^3-ap [8]

In the following theorems we will note $\mathscr{N}(\sigma, l, m)$ the set of function given by a feedforward neural network with activation σ and l layers with at most m nodes. The ReLU activation will be noted σ_R and the classification error $\mathcal{R}_z$.

Theorem 5. *With $m \leq 2^{(k-3)/(l-1)}$, for any positive integer k, $\exists$ a collection of $n := 2^k$ points $S = (x_i, y_i)_{i=1}^n$ where $x_i \in [0, 1], y \in \{0, 1\}$ such that*

$$\min_{f \in \mathscr{N}(\sigma_R, m, l)} \mathcal{R}_z(f) = \frac{1}{6} \qquad \textit{and} \qquad \min_{g \in \mathscr{N}(\sigma_R, 2, 2k)} \mathcal{R}_z(g) = 0$$

Notably, if we look at the case of 2 layered networks, this tells us that even with 2^{k-3} units in the hidden layer, the wide network is going to missclassify $\frac{1}{6}$ of the points, whereas a deep network with $2k$ hidden layers with 2 units each can achieve 0 classification error.

Now we will refine this result, let $\mathscr{R}(\sigma, l, m, k)$ denote k iterations of a recurrent neural network, every $f \in \mathscr{R}(\sigma, l, m, k)$ can be expressed as some fixed network $g \in \mathscr{N}(\sigma, l, m)$ applied k times:

$$f(x) = g^k(x) = \underbrace{(g \circ g \cdots \circ g)}_{\text{k times}}(x)$$

Consequently, $\mathscr{R}(\sigma, l, m, k) \subseteq \mathscr{N}(\sigma, l, mk)$ but the former has $\mathcal{O}(ml)$ parameters whereas the latter has $\mathcal{O}(mlk)$ parameters.
Lastly we define the following functions:

Definition 6 (Sawtooth)**.** *A function $f : \mathbb{R} \to \mathbb{R}$ is t-sawtooth if it is piece-wise linear with t-pieces*

We can say for example that σ_R (ReLU) is a 2-sawtooth function, decision stumps used in boosting are also 2-sawtooth, and decision trees with t-1 nodes are t-sawtooth.

Theorem 7. *Let positive integer k, number of layers l, and number of nodes per layer m be given. Given a t-sawtooth $\sigma : \mathbb{R} \to \mathbb{R}$ and $n := 2k$ points as specified by the n-ap, then*

$$\min_{f \in \mathcal{N}(\sigma,m,l)} \mathcal{R}_z(f) \geq \frac{n - 4(tm)^l}{3n} \qquad \textit{and} \qquad \min_{g \in \mathscr{R}(\sigma_R,2,2,k)} \mathcal{R}_z(g) = 0$$

In summary, this means that on the 2^k-ap, one needs exponentially (in k) many parameters with a wide network, linearly many parameters with a deep network and constantly many parameters with a recurrent network.

4.2.1 Upper bound proof [WIP]

Lemma 1. *If f is t-sawtooth, g is s-sawtooth, then we have*
$f + g$ is (s+t)-sawtooth *$f \circ g$ is st-sawtooth*

Lemma 2. *If σ is t-sawtooth, then every $f \in \mathcal{N}(\sigma;m,l)$ is $(tm)^l$-sawtooth*

Let us define the "mirror map" $f_m^k : \mathbb{R} \to \mathbb{R}$ as figure shows:

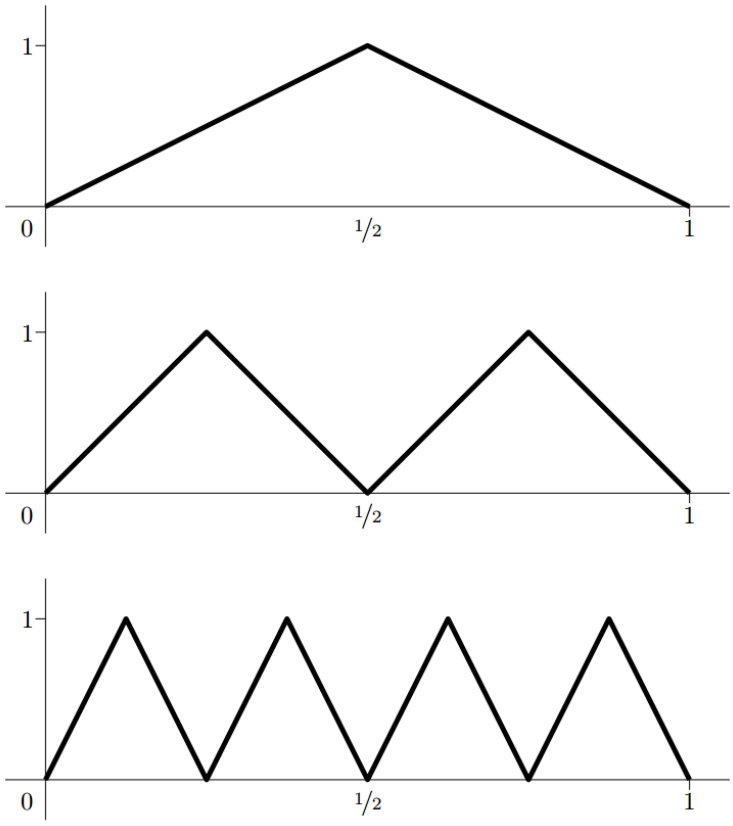


Figure 11: f_m, f_m^2 and f_m^3 [8]

Note that $f_m^k \in \mathscr{R}(\sigma_R; 2, 2, k) \subseteq \mathcal{N}(\sigma_R; 2, 2k)$.

References

[1] S. S. ABHYANKAR. Hilbert's thirteenth problem. In *Proc. of the Franco-Belgian Conference in Reims, Société Mathématique de France, Séminaires et Congres*, volume 2, page 1, 1997.

[2] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[3] K.-I. Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural networks*, 2 (3):183–192, 1989.

[4] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[5] A. N. Kolmogorov. On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. In *Doklady Akademii Nauk*, volume 114, pages 953–956. Russian Academy of Sciences, 1957.

[6] Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang. The expressive power of neural networks: A view from the width. In *Advances in Neural Information Processing Systems*, pages 6231–6239, 2017.

[7] M. Minsky and S. Papert. Perceptrons: an introduction to computational geometry. 1969.

[8] M. Telgarsky. Representation benefits of deep feedforward networks. *arXiv preprint arXiv:1509.08101*, 2015.