Deep learning

3.4. Multi-Layer Perceptrons

François Fleuret

https://fleuret.org/dlc/

Jan 1, 2021

A linear classifier of the form

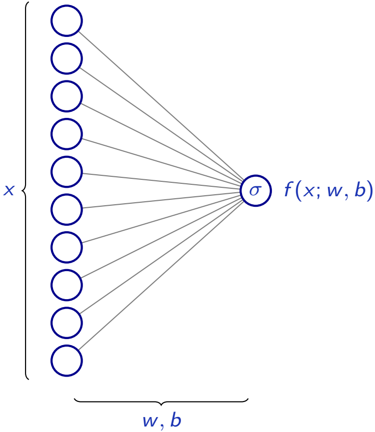
$$\begin{aligned} \mathbb{R}^D &\to \mathbb{R} \\ x &\mapsto \sigma(w \cdot x + b), \end{aligned}$$

with $w \in \mathbb{R}^D$, $b \in \mathbb{R}$, and $\sigma : \mathbb{R} \to \mathbb{R}$, can naturally be extended to a multi-dimension output by applying a similar transformation to every output

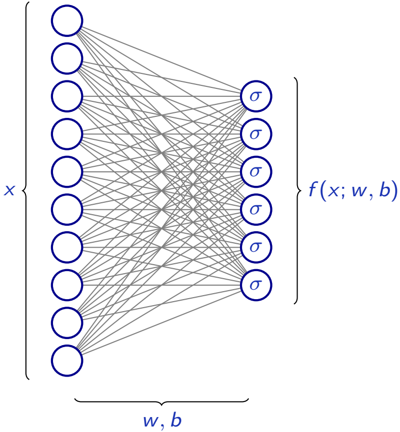
$$\begin{aligned} \mathbb{R}^D &\to \mathbb{R}^C \\ x &\mapsto \sigma(wx + b), \end{aligned}$$

with $w \in \mathbb{R}^{C \times D}$, $b \in \mathbb{R}^C$, and σ is applied component-wise.

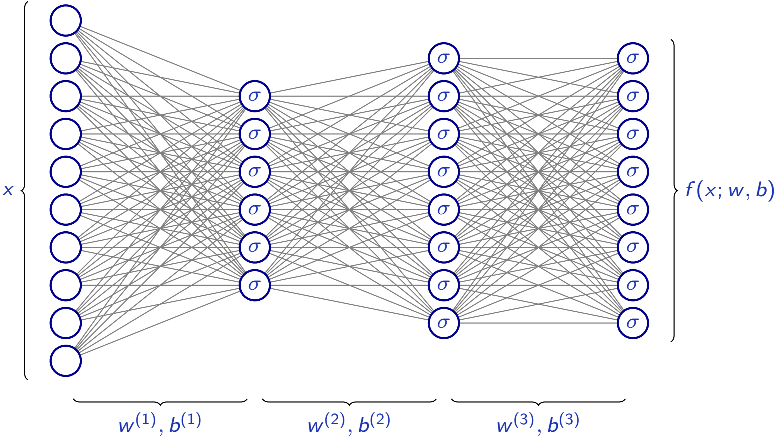
Even though it has no practical value implementation-wise, we can represent such a model as a combination of units. More importantly, we can extend it.



Even though it has no practical value implementation-wise, we can represent such a model as a combination of units. More importantly, we can extend it.



Even though it has no practical value implementation-wise, we can represent such a model as a combination of units. More importantly, we can extend it.



This latter structure can be formally defined, with $x^{(0)} = x$,

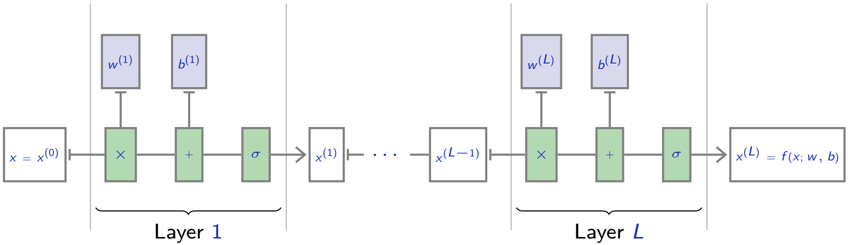
$$\forall l = 1, \dots, L, \; x^{(l)} = \sigma\left(w^{(l)} x^{(l-1)} + b^{(l)}\right)$$

and $f(x; w, b) = x^{(L)}$.

This latter structure can be formally defined, with $x^{(0)} = x$,

$$\forall l = 1, \dots, L,\ x^{(l)} = \sigma\left(w^{(l)} x^{(l-1)} + b^{(l)}\right)$$

and $f(x; w, b) = x^{(L)}$.



Such a model is a **Multi-Layer Perceptron (MLP).**

Note that if σ is an affine transformation, the full MLP is a composition of affine mappings, and itself an affine mapping.

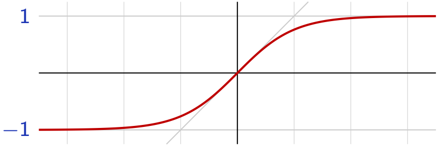
Note that if σ is an affine transformation, the full MLP is a composition of affine mappings, and itself an affine mapping.

Consequently:

⚠ **The activation function σ should be non-linear**, or the resulting MLP is an affine mapping with a peculiar parametrization.

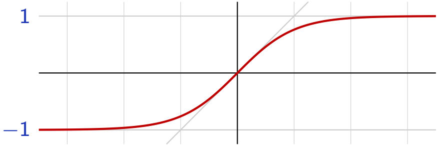
The two classical activation functions are the hyperbolic tangent

$$x \mapsto \frac{2}{1+e^{-2x}} - 1$$



The two classical activation functions are the hyperbolic tangent

$$x \mapsto \frac{2}{1+e^{-2x}} - 1$$

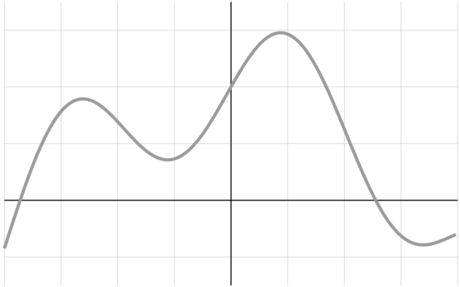


and the rectified linear unit (ReLU)

$$x \mapsto \max(0, x)$$

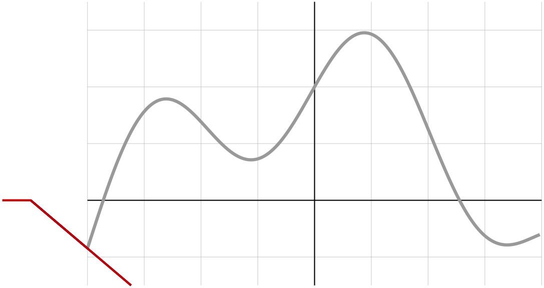
Universal approximation

We can approximate any $\psi \in \mathscr{C}([a, b], \mathbb{R})$ with a linear combination of translated/scaled ReLU functions.



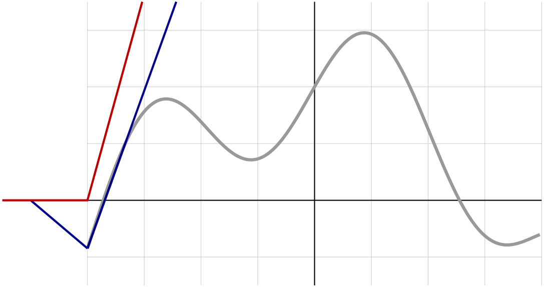
We can approximate any $\psi \in \mathscr{C}([a, b], \mathbb{R})$ with a linear combination of translated/scaled ReLU functions.

$$f(x) = \sigma(w_1 x + b_1)$$



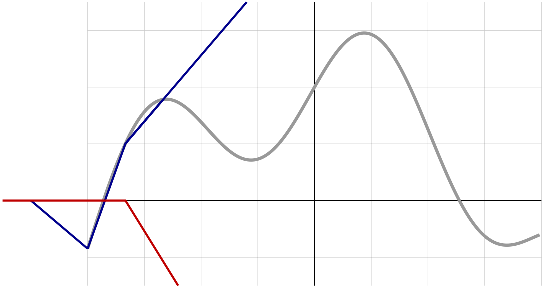
We can approximate any $\psi \in \mathscr{C}([a, b], \mathbb{R})$ with a linear combination of translated/scaled ReLU functions.

$$f(x) = \sigma(w_1 x + b_1) + \sigma(w_2 x + b_2)$$



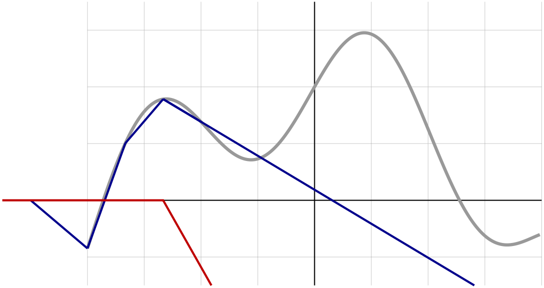
We can approximate any $\psi \in \mathscr{C}([a,b],\mathbb{R})$ with a linear combination of translated/scaled ReLU functions.

$$f(x) = \sigma(w_1 x + b_1) + \sigma(w_2 x + b_2) + \sigma(w_3 x + b_3)$$



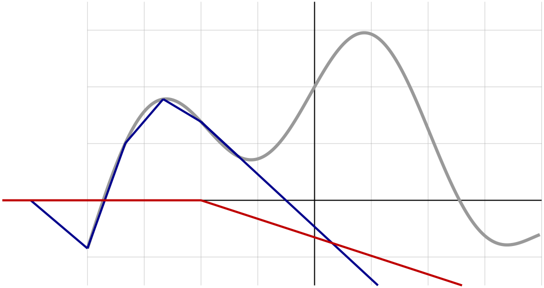
We can approximate any $\psi \in \mathscr{C}([a, b], \mathbb{R})$ with a linear combination of translated/scaled ReLU functions.

$$f(x) = \sigma(w_1 x + b_1) + \sigma(w_2 x + b_2) + \sigma(w_3 x + b_3) + \dots$$



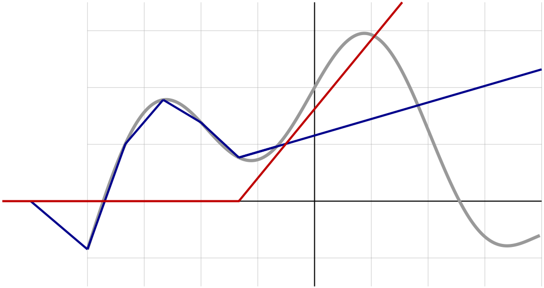
We can approximate any $\psi \in \mathscr{C}([a, b], \mathbb{R})$ with a linear combination of translated/scaled ReLU functions.

$$f(x) = \sigma(w_1 x + b_1) + \sigma(w_2 x + b_2) + \sigma(w_3 x + b_3) + \dots$$



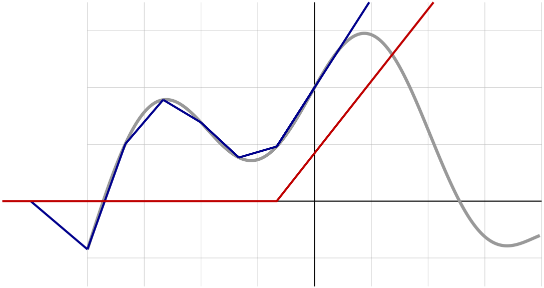
We can approximate any $\psi \in \mathscr{C}([a, b], \mathbb{R})$ with a linear combination of translated/scaled ReLU functions.

$$f(x) = \sigma(w_1 x + b_1) + \sigma(w_2 x + b_2) + \sigma(w_3 x + b_3) + \dots$$



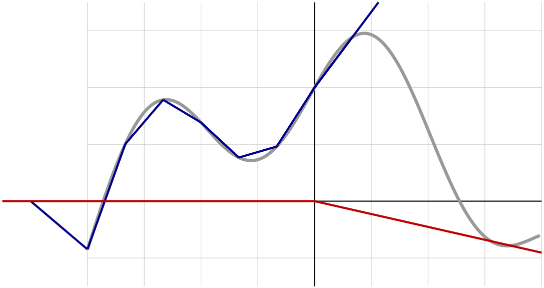
We can approximate any $\psi \in \mathscr{C}([a, b], \mathbb{R})$ with a linear combination of translated/scaled ReLU functions.

$$f(x) = \sigma(w_1 x + b_1) + \sigma(w_2 x + b_2) + \sigma(w_3 x + b_3) + \dots$$



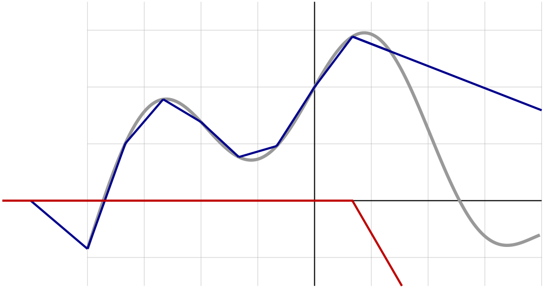
We can approximate any $\psi \in \mathscr{C}([a, b], \mathbb{R})$ with a linear combination of translated/scaled ReLU functions.

$$f(x) = \sigma(w_1 x + b_1) + \sigma(w_2 x + b_2) + \sigma(w_3 x + b_3) + \dots$$



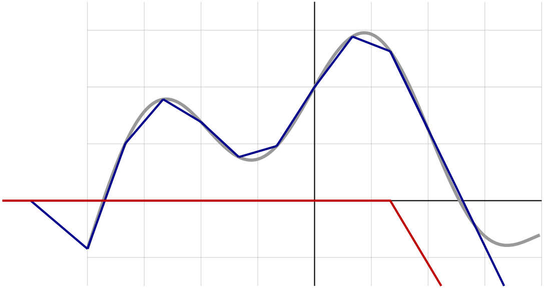
We can approximate any $\psi \in \mathscr{C}([a, b], \mathbb{R})$ with a linear combination of translated/scaled ReLU functions.

$$f(x) = \sigma(w_1 x + b_1) + \sigma(w_2 x + b_2) + \sigma(w_3 x + b_3) + \dots$$



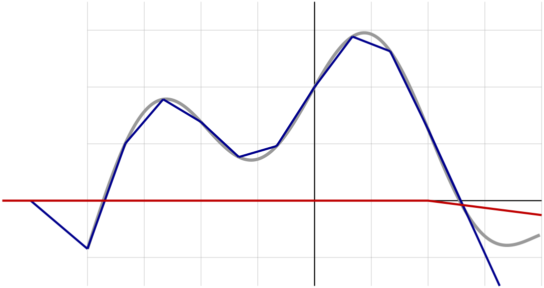
We can approximate any $\psi \in \mathscr{C}([a, b], \mathbb{R})$ with a linear combination of translated/scaled ReLU functions.

$$f(x) = \sigma(w_1 x + b_1) + \sigma(w_2 x + b_2) + \sigma(w_3 x + b_3) + \dots$$



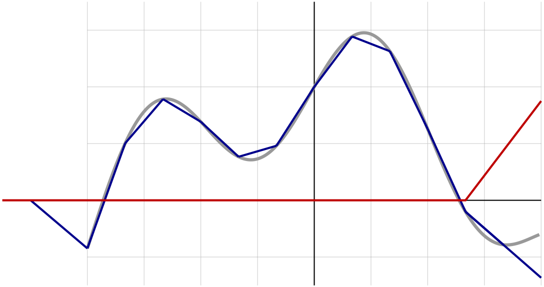
We can approximate any $\psi \in \mathscr{C}([a, b], \mathbb{R})$ with a linear combination of translated/scaled ReLU functions.

$$f(x) = \sigma(w_1 x + b_1) + \sigma(w_2 x + b_2) + \sigma(w_3 x + b_3) + \dots$$



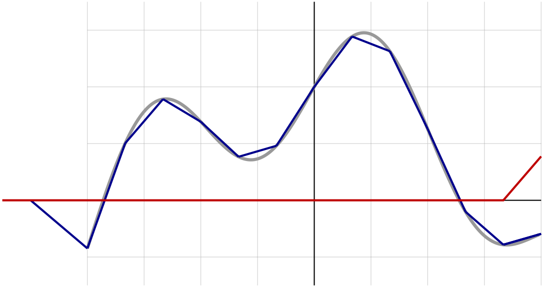
We can approximate any $\psi \in \mathscr{C}([a, b], \mathbb{R})$ with a linear combination of translated/scaled ReLU functions.

$$f(x) = \sigma(w_1 x + b_1) + \sigma(w_2 x + b_2) + \sigma(w_3 x + b_3) + \dots$$



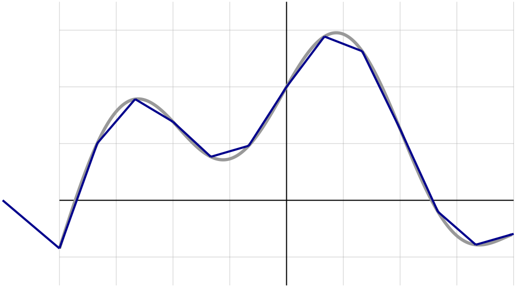
We can approximate any $\psi \in \mathscr{C}([a, b], \mathbb{R})$ with a linear combination of translated/scaled ReLU functions.

$$f(x) = \sigma(w_1 x + b_1) + \sigma(w_2 x + b_2) + \sigma(w_3 x + b_3) + \dots$$



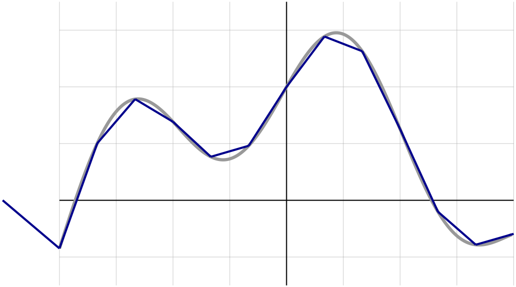
We can approximate any $\psi \in \mathscr{C}([a, b], \mathbb{R})$ with a linear combination of translated/scaled ReLU functions.

$$f(x) = \sigma(w_1 x + b_1) + \sigma(w_2 x + b_2) + \sigma(w_3 x + b_3) + \dots$$



We can approximate any $\psi \in \mathscr{C}([a, b], \mathbb{R})$ with a linear combination of translated/scaled ReLU functions.

$$f(x) = \sigma(w_1 x + b_1) + \sigma(w_2 x + b_2) + \sigma(w_3 x + b_3) + \dots$$



This is true for other activation functions under mild assumptions.

Extending this result to any $\psi \in \mathscr{C}([0,1]^D, \mathbb{R})$ requires a bit of work.

We can approximate the sin function with the previous scheme, and use the density of Fourier series to get the final result:

$$\forall \epsilon > 0, \exists K, w \in \mathbb{R}^{K \times D}, b \in \mathbb{R}^K, \omega \in \mathbb{R}^K, \text{ s.t.}$$

$$\max_{x \in [0,1]^D} |\psi(x) - \omega \cdot \sigma(w\, x + b)| \leq \epsilon.$$

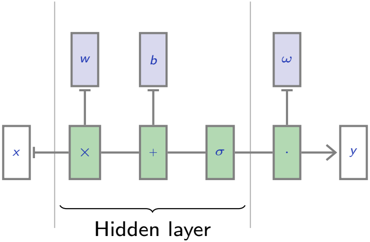
So we can approximate any continuous function

$$\psi : [0,1]^D \to \mathbb{R}$$

with a one hidden layer perceptron

$$x \mapsto \omega \cdot \sigma(w\,x + b),$$

where $b \in \mathbb{R}^K$, $w \in \mathbb{R}^{K \times D}$, and $\omega \in \mathbb{R}^K$.



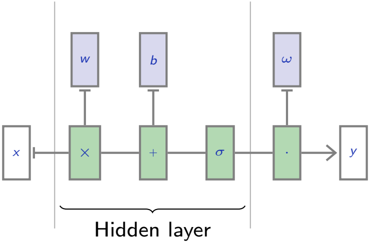
So we can approximate any continuous function

$$\psi : [0, 1]^D \to \mathbb{R}$$

with a one hidden layer perceptron

$$x \mapsto \omega \cdot \sigma(w\,x + b),$$

where $b \in \mathbb{R}^K$, $w \in \mathbb{R}^{K \times D}$, and $\omega \in \mathbb{R}^K$.



This is the **universal approximation theorem.**

⚠ A better approximation requires a larger hidden layer (larger K), and this theorem says nothing about the relation between the two.

So this results states that we can make the **training error** as low as we want by using a larger hidden layer. It states nothing about the **test error**

⚠ A better approximation requires a larger hidden layer (larger K), and this theorem says nothing about the relation between the two.

So this results states that we can make the **training error** as low as we want by using a larger hidden layer. It states nothing about the **test error**

Deploying MLP in practice is often a balancing act between under-fitting and over-fitting.

The end