Neural Networks

Prof. Alessandro Lucantonio

Aarhus University - Department of Mechanical and Production Engineering

?/?/2023

Motivations

- Neural Networks (NN) are one of the most flexible ML tools.
- Universal approximators!
- Can manipulate real and discrete data $\rightsquigarrow$ regression and classification problems.
- Not a single model: many type of NN (e.g. MLP, CNN, RNN).
- Inspired by biological systems.

Perceptron - Visualization

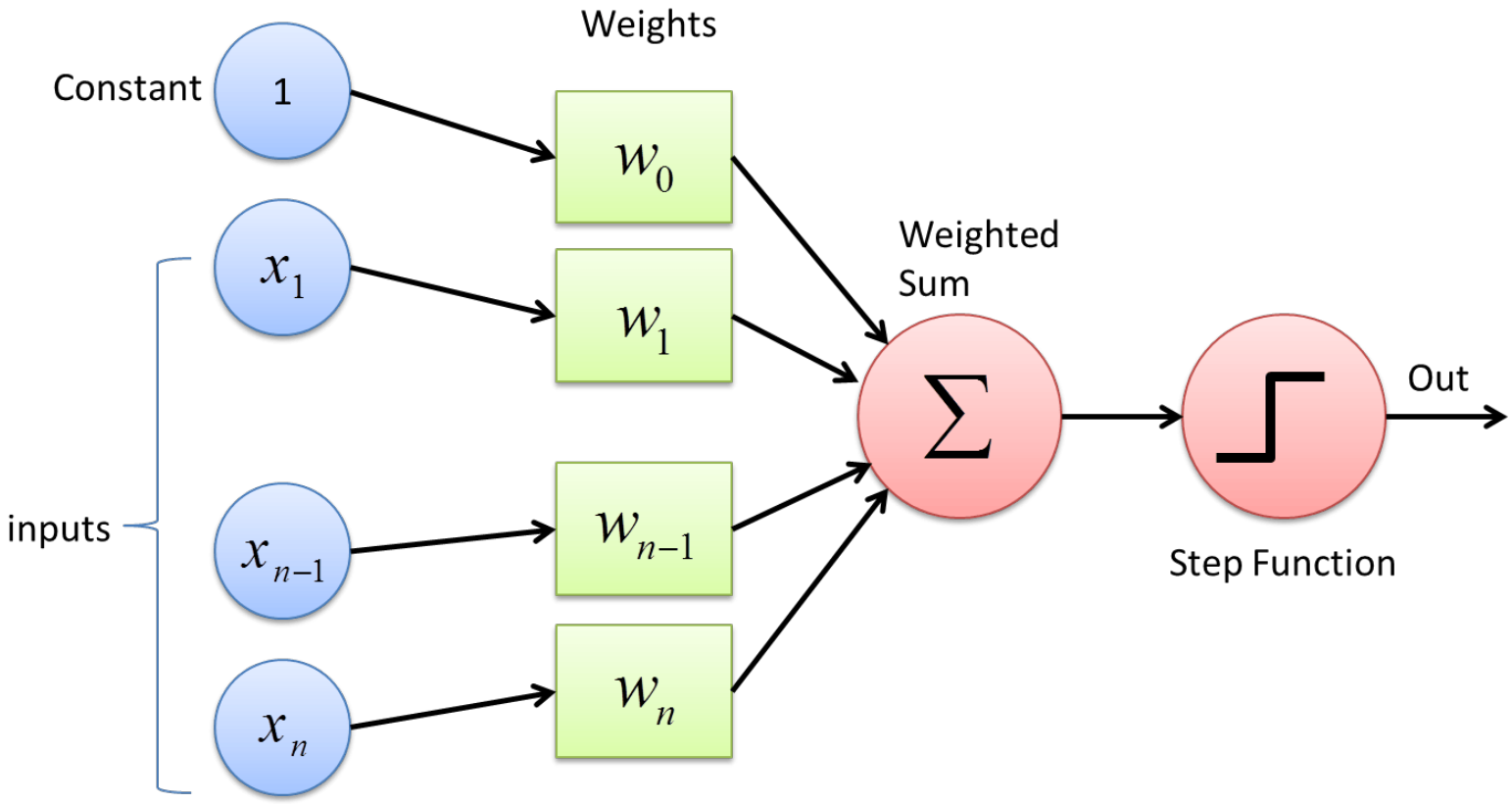


Figure: Representation of a perceptron.

Perceptron - Formal

Notation: n is as usual the number of features. x is an input sample of length $n+1$ with $x_0 = 1$.

Perceptron:

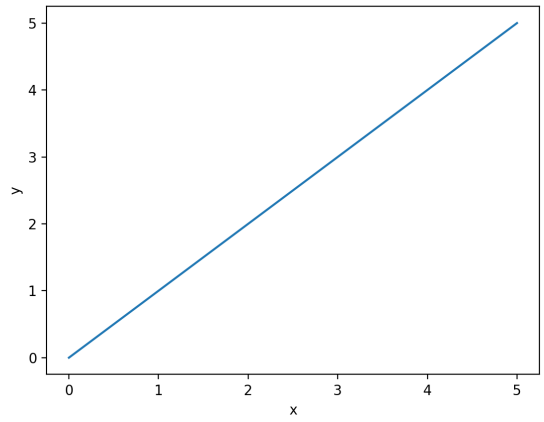
$$\begin{cases} n(\boldsymbol{x}) := \boldsymbol{w}^T \boldsymbol{x} = \sum_{j=0}^{n} w_j x_j, \\ h(\boldsymbol{x}) := f(n(\boldsymbol{x})). \end{cases}$$

n is the **net input** to the neuron.

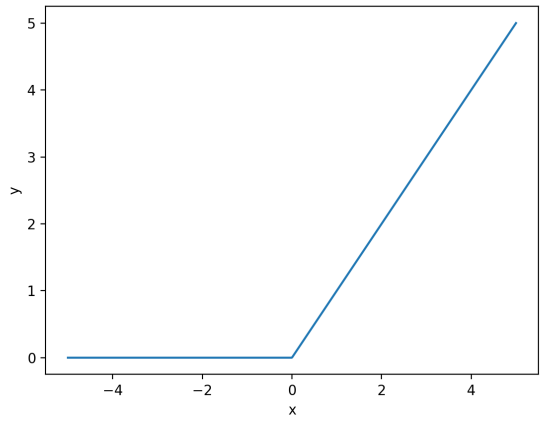
f is called **activation function**. Some common examples are:

- Linear: $f(t) = at + b$.
- ReLU (Rectified Linear Unit): $f(t) := \max\{0, t\}$.
- Sigmoid: $f(t) := \frac{1}{1+e^{-t}}$.
- Tanh (Hyperbolic tangent): $f(t) := \frac{e^{2t}-1}{e^{2t}+1}$

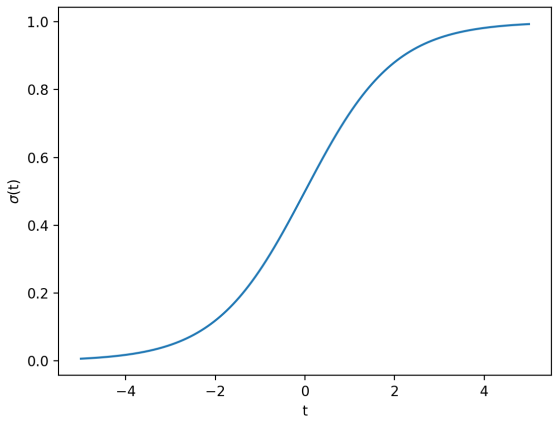
Activation Functions - Visualization



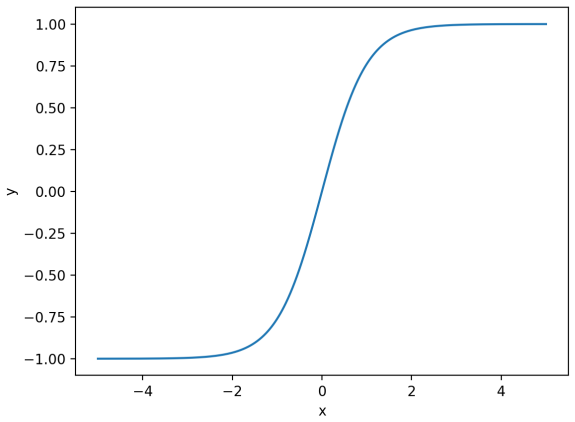
(a) Linear



(b) ReLU

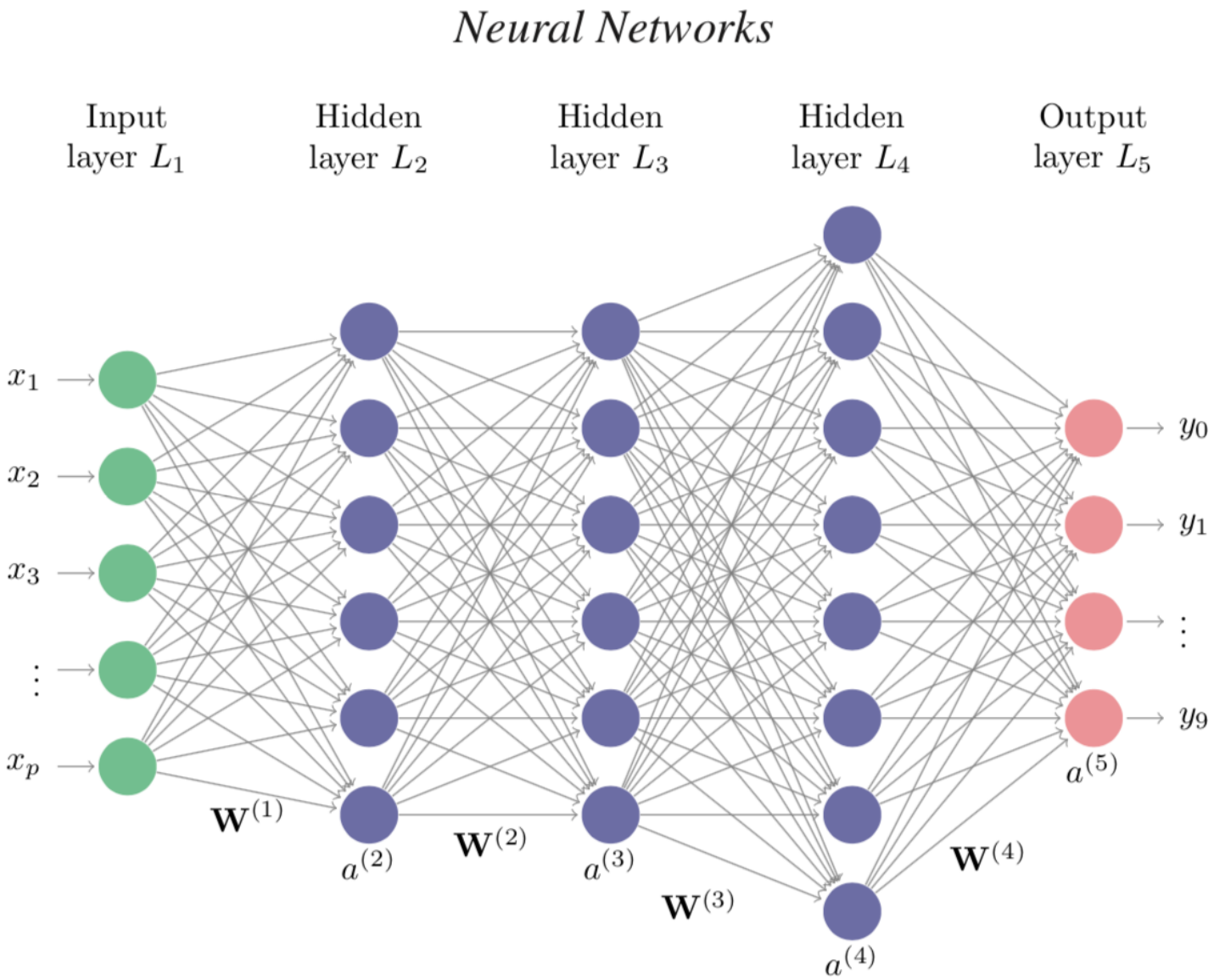


(c) Sigmoid



(d) Tanh

MultiLayer Perceptron (MLP) - Visualization



MLP - Formal

Notation: $\boldsymbol{a}^{(i)}$ is the input vector of the i-th layer and $\mathrm{W}^{(i)}$ is the weight matrix for each layer. m is the number of layers.

Parenthesis: Why do we have a weight matrix for each layer? Each layer i consists of m_i neurons, hence necessarily has m_i weight vectors, which we can arrange row-wise to form the matrix $\mathrm{W}^{(i)}$.

For each layer i compute

$$\begin{cases} n_i(\boldsymbol{a}^{(i)}) := \mathrm{W}^{(i)}\boldsymbol{a}^{(i)}, \\ h_i(\boldsymbol{a}^{(i)}) := f_i(n_i(\boldsymbol{a}^{(i)})). \end{cases}$$

Compact form:

$$h(\boldsymbol{x}) := f_{m-1}\big(\mathrm{W}^{(m-1)} f_{m-2}\big(\cdots f_1\big(\mathrm{W}^{(1)}\boldsymbol{x}\big)\cdots\big)\big)$$

MLP - Some comments

- This kind of NN are also called **feedforward NN**, since we transfer information from input to output.
- The hypothesis function is non-convex! This is due to the fact that composition of convex functions is not necessarily convex. Hence, in the learning problem we will have many local minima and saddle points.
- Theory tells us that MLP with just 1 hidden layer are universal approximators, in the sense that they approximate as well as you want "each" continuous function (not a formal statement, just to provide intuition).

Tips and Tricks - Is one layer really enough?

Theory suggests us that the answer is yes, but pay attention.
There exists cases for which an exponential number of units (w.r.t. the input dimension) are required to approximate well the data.

Introducing many layers helps to reduce the total number of units but complicate a bit the learning procedure (see later).

Then the question becomes: how to find the optimal number of layers? Validation is your best friend :)
Include in your model selection process as many different architectures/activation functions as you can.