Neural Networks (Part I)

Machine Learning, A.Y. 2022/23, Padova

Fabio Aiolli

October 24th, 2022

Artificial Neural Networks: Generalities

Two different reasons (lines of research) have historically pushed the study of Artificial Neural Networks (ANN):

- Reproduce/understand the human brain
  - devise "realistic" computational models of (part of) the human brain;
  - reproduce neuro-physiological phenomena;
  - assess empirically if the computational model reproduces biological data.
- Understand the general computational principles used by the human brain
  - it does not matter to reproduce the human brain, but only to infer what are the fundamental computational principles it uses;
  - simplifications and abstractions of brain functions/structures are the main working tools;
  - devise an artificial system that is possibly different from the human brain, but that reproduces some of its functions, perhaps in a faster and more efficient fashion (planes vs. birds methaphor).

The Da Vinci's flying machine (Ornithopter, 15th century) vs the Wright brothers' aircraft (Wright Flyer, 20th century)

We are interested in the second line of research!

There are a lot of different neural network models, which answer different computational/learning needs:

- Supevised learning (classification, regression, time series, ...);
- Unsupervised learning (clustering, representation learning, self-organized maps, associative memories, ...).

All these models differ for:

- Network topology;
- Function computed by a single neuron;
- Training algorithm;
- How training proceeds (how training data are used).

When to use a NN?

Characteristics of the problem:

- Input: discrete and/or real-valued vectors, high dimensionality;
- Output: discrete (classification) or real-valued (regression) vectors;
- Data (input and/or output) can be noisy and the form of the target function unknown (no apriori information);
- Having long learning times is acceptable and a quick evaluation of the learned function is required;
- The final solution DOES NOT need to be understood by a human expert ("black box problem")

Examples of applications:

- Speech recognition
- Image classification
- Time series prediction

Comparison with the human brain

NNs are inspired by the human brain:

- The brain is constituted by $\sim 10^{11}$ strongly interconnected neurons;
- Each neuron is connected with other $\sim 10^4$ neurons
- The response time of a neuron is about ~ 0.001 seconds

Considering that:

- Recognizing the content of a scene takes about ~ 0.1 seconds for a human;
- It cannot make more than ~ 100 serial calculations $[0.1/0.001 = 100]$;

It follows that the human brain heavily exploits parallel computing.

A bit of history

- 1943 McCulloch and Pitts: first mathematical models of a NN, binary inputs, linear combination with threshold, binary output.
- 1958 first neural network scheme, called Perceptron (perceptor), forerunner of current neural networks. Great enthusiasm.
- 1969 Minsky and Papert show limitations of Perceptron. Research in this field has a noticeable drop in funding.
- 1986 Rumelhart et al. propose the Backpropagation algorithm for learning the weights of a multilayer network. New life for research.

Single neuron - Perceptron

Consider the space of hyperplanes in $\mathbb{R}^n$ (n is the dimension of the input):

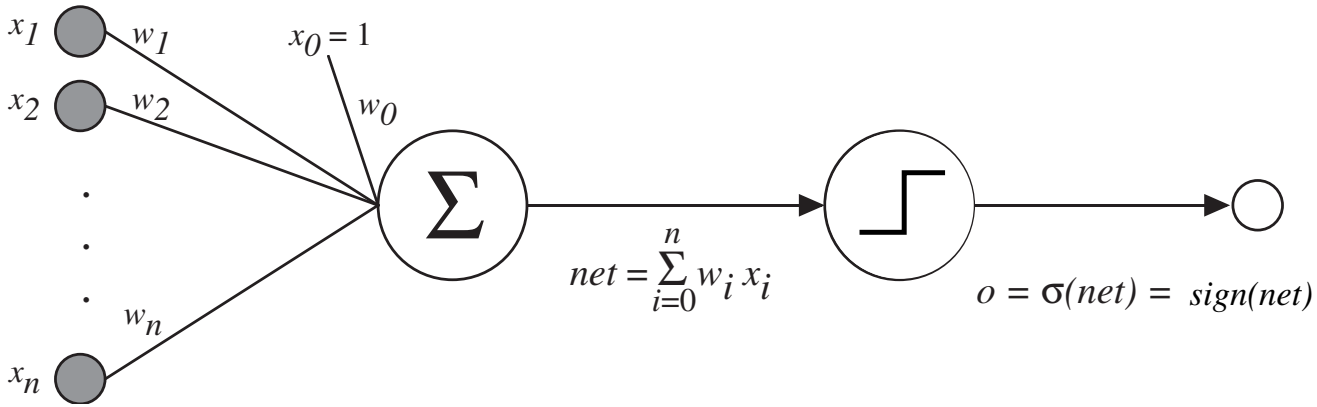
$$\mathcal{H} = \{f_{(\mathbf{w},b)}(\mathbf{x}) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b) : \mathbf{w}, \mathbf{x} \in \mathbb{R}^n, b \in \mathbb{R}\}$$

that we can rewrite as:

$$\mathcal{H} = \{f_{\mathbf{w}'}(\mathbf{x}') = \text{sign}(\mathbf{w}' \cdot \mathbf{x}') : \mathbf{w}', \mathbf{x}' \in \mathbb{R}^{n+1}\}$$

after the following change of variables:

$$\mathbf{w}' = [b, \mathbf{w}], \quad \mathbf{x}' = [1, \mathbf{x}]$$



We will refer to this neuron (and the associated learning algorithm) as Perceptron.

Boolean functions and Perceptron

Consider binary inputs and Boolean functions.

- Any Boolean function can be represented by combinations of operators or, and, not;
- Can a Perceptron implement the or? YES!
  Ex. $\mathbf{x} \in \{0,1\}^{n+1}, \mathbf{w}'_0 = -0.5, \{\mathbf{w}_i = 1\}_{i=1,\dots,n}$;
- Can a Perceptron implement the and? YES!
  Ex. $\mathbf{x} \in \{0,1\}^{n+1}, \mathbf{w}'_0 = -n + 0.5, \{\mathbf{w}_i = 1\}_{i=1,\dots,n}$;
- The not can be easily implemented by a Perceptron with a single connection (left as exercise).

Hence, any Boolean function can be implemented as a combination of Perceptrons!
Is there a "simple" Boolean function that, instead, a single Perceptron CANNOT IMPLEMENT?
YES, seen when we talked about VC-dim of the space of hyperplanes.

Perceptron: learning algorithm

Assume to have training examples in $\mathbb{R}^n$ that are linearly separable, that is, such that there is a hyperplane in $\mathbb{R}^n$ that can separate them.

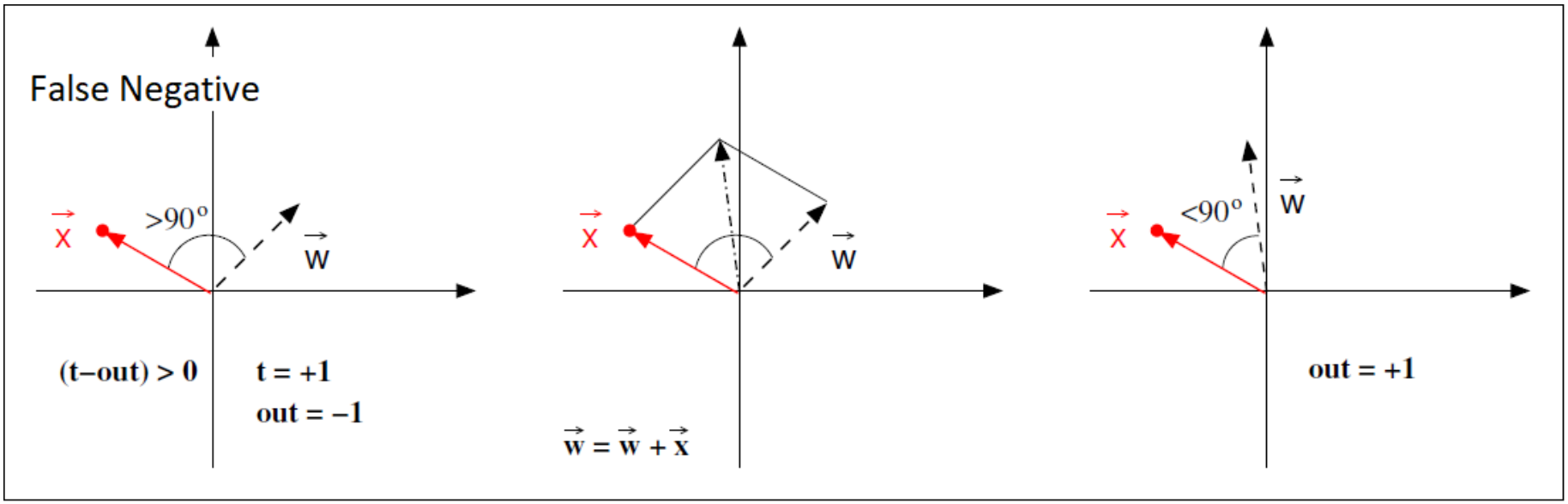
Perceptron

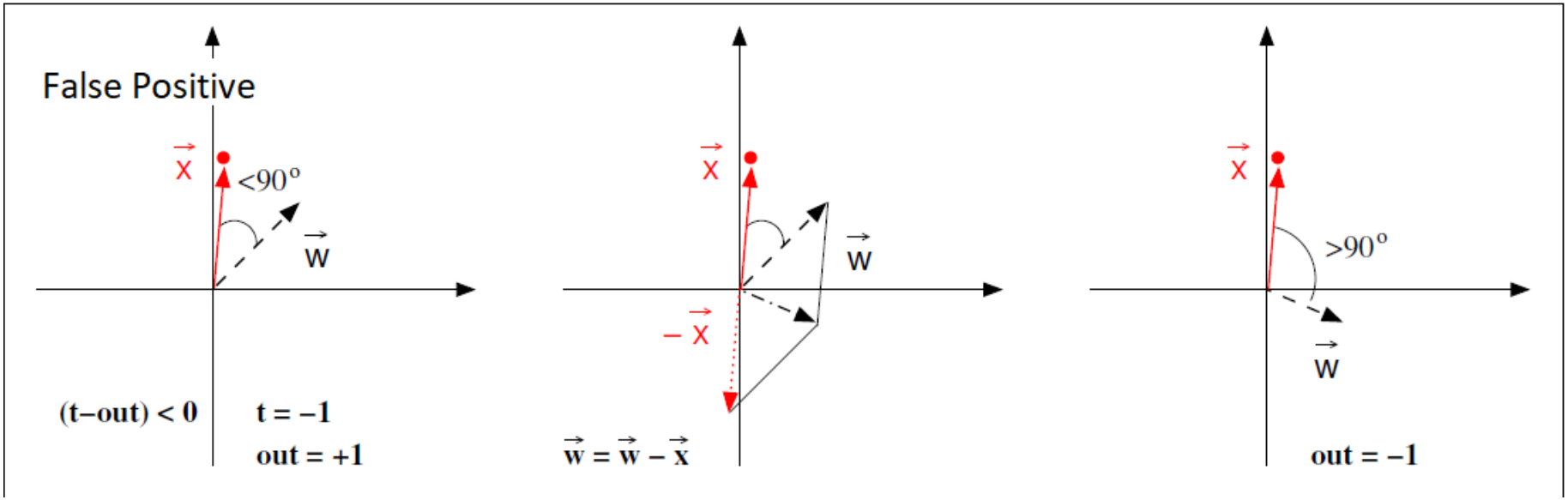
Input: Training set $\mathcal{S} = \{(\mathbf{x}, t)\}, \mathbf{x} \in \mathbb{R}^{n+1}, t \in \{-1, +1\}$, $\eta \geq 0$ (learning rate)

1. Initialize the value of the weights $\mathbf{w}$ randomly
2. Repeat
   a. Select (randomly) one of the training examples $(\mathbf{x}, t)$
   b. If $o = \text{sign}(\mathbf{w} \cdot \mathbf{x}) \neq t$, then

$$\mathbf{w} \leftarrow \mathbf{w} + \eta(t - o)\mathbf{x}$$

Geometric Interpretation



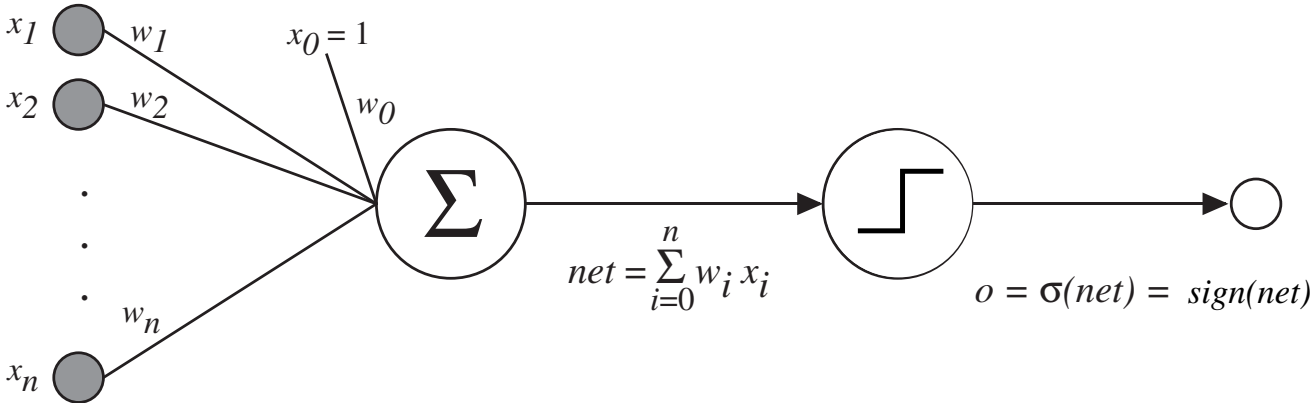


Remarks

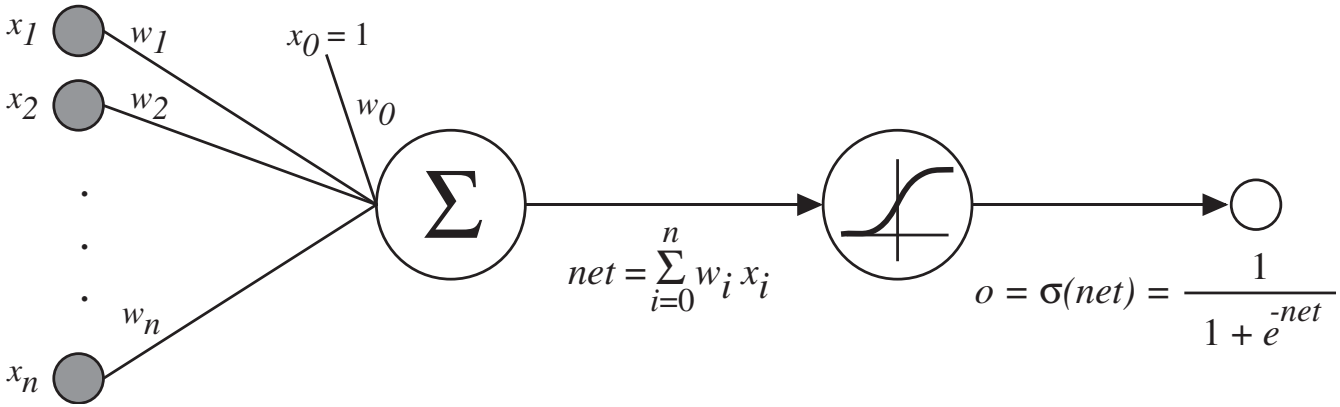
- Not necessarily a single learning step 2(b) is capable to change the sign of the output for an instance, many steps may be needed;
- A small value for the learning rate η makes the learning more stable, that is, it prevents the weight vector to undergo too "sharp" changes whenever step 2(b) is executed;
- If the training set is linearly separable, it can be shown that the Perceptron training algorithm terminates in a finite number of steps;
- Let R be the radius of the smallest hyper-sphere centered in the origin enclosing all the instances, i.e. $||\mathbf{x}_i|| <= R$. Let γ be the maximal value such that $t_i o_i = t_i(\mathbf{w} \cdot \mathbf{x}_i) \geq \gamma > 0$, then it can be shown that the number of steps of the Perceptron algorithm is bounded from above by the quantity R^2/γ^2.

Artificial neuron

Alternative 1: Hard-threshold (NOT derivable)



Alternative 2: Neuron with sigmoid (derivable)

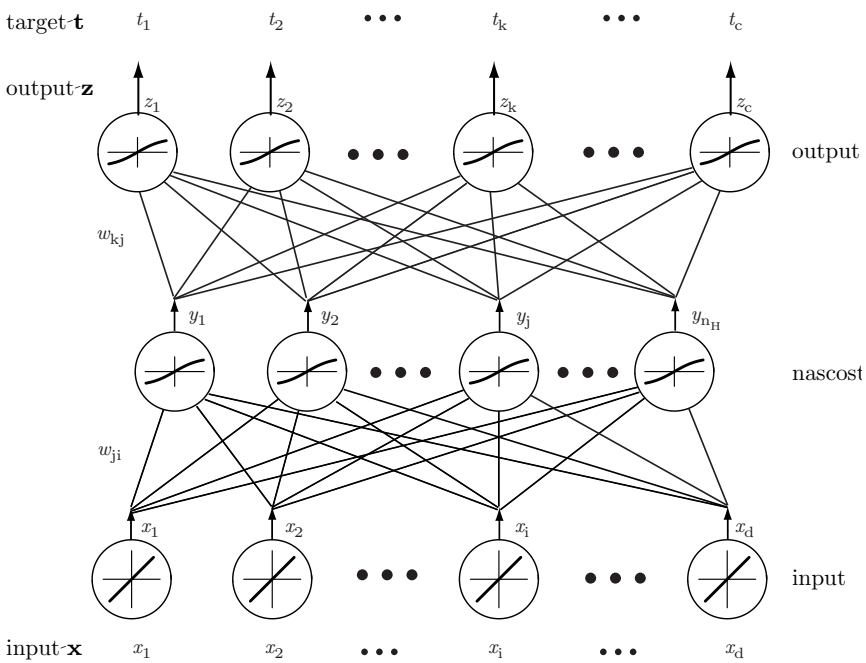


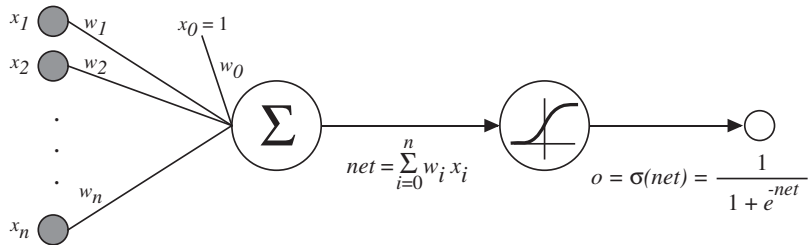
Multilayer Neural Networks

A Multilayer Neural Network is a system composed by inter-connected units that computes non-linear numeric functions:

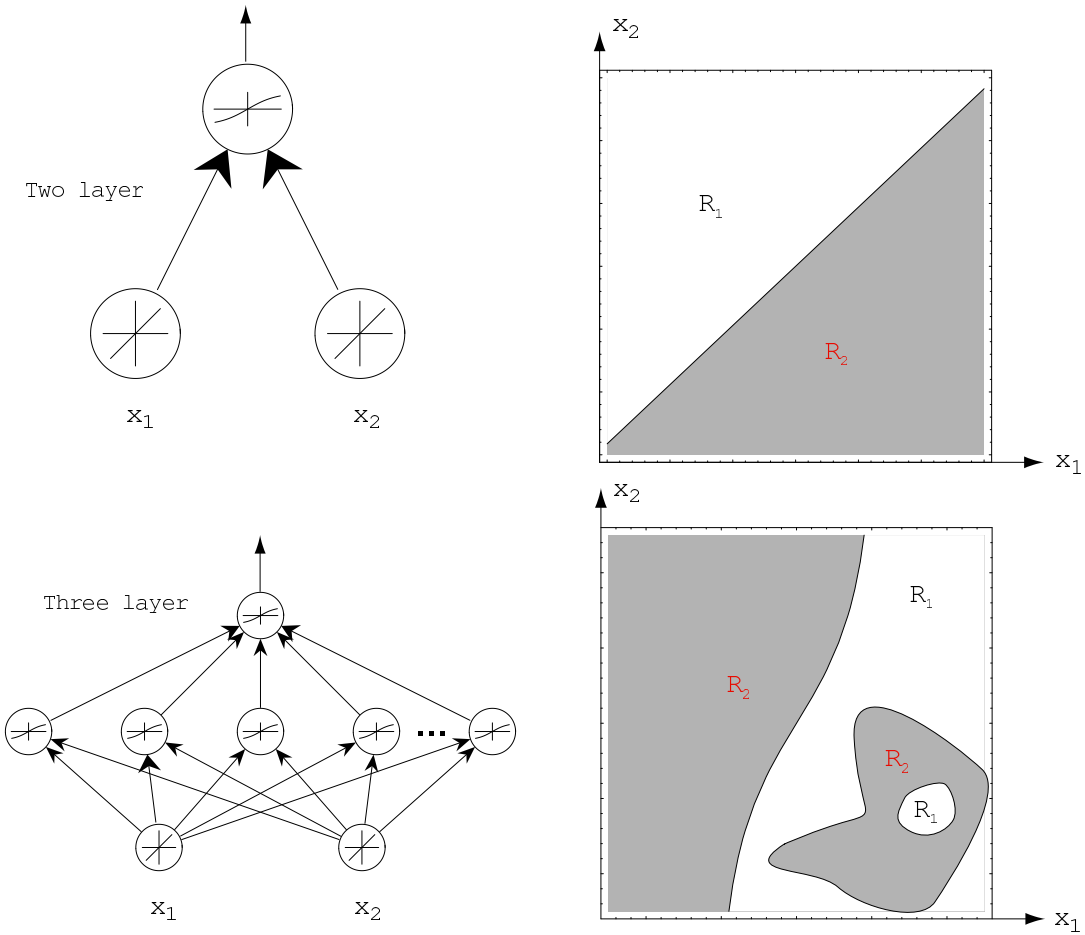
- The input units represent the input variables;
- The output units represent the output variables;
- The hidden units (if there are any) represent inner variables that (after the training) codify correlations among input variables, with respect to the output values we want to generate;
- Adaptable weights are defined on the connections between units.

Multilayer Neural Networks





Decision surfaces



Recap

Notions

- Perceptron
- Boolean functions and Perceptron
- Training algorithm for the Perceptron
- Convergence of the Perceptron
- Introduction to multi-layer neural networks

Exercises

- Give Perceptron-based multi-layer networks with hard thresholds and relative weights (without using learning) that implement simple Boolean functions such as: `A and (not B)`, `A xor B`, ...
- Prove that, if the Perceptron algorithm is initialized with the null vector, then the coefficient η does not affect learning.
- Implement the Perceptron algorithm.