From Artificial Intelligence to a first neuron model

Frederic Precioso

Disclaimer

If any content in this presentation is yours but is not correctly referenced or if it should be removed, please contact me and I will correct it.

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Overview

- **Context & Vocabulary**
- Math Basics
- Simple Models

Overview

- **Context & Vocabulary**
  - ***What is Artificial Intelligence?***
  - ***Machine Learning without Maths***
  - ***Machine Learning & Statistics?***
- Math Basics
- Simple Models

CONTEXT & VOCABULARY

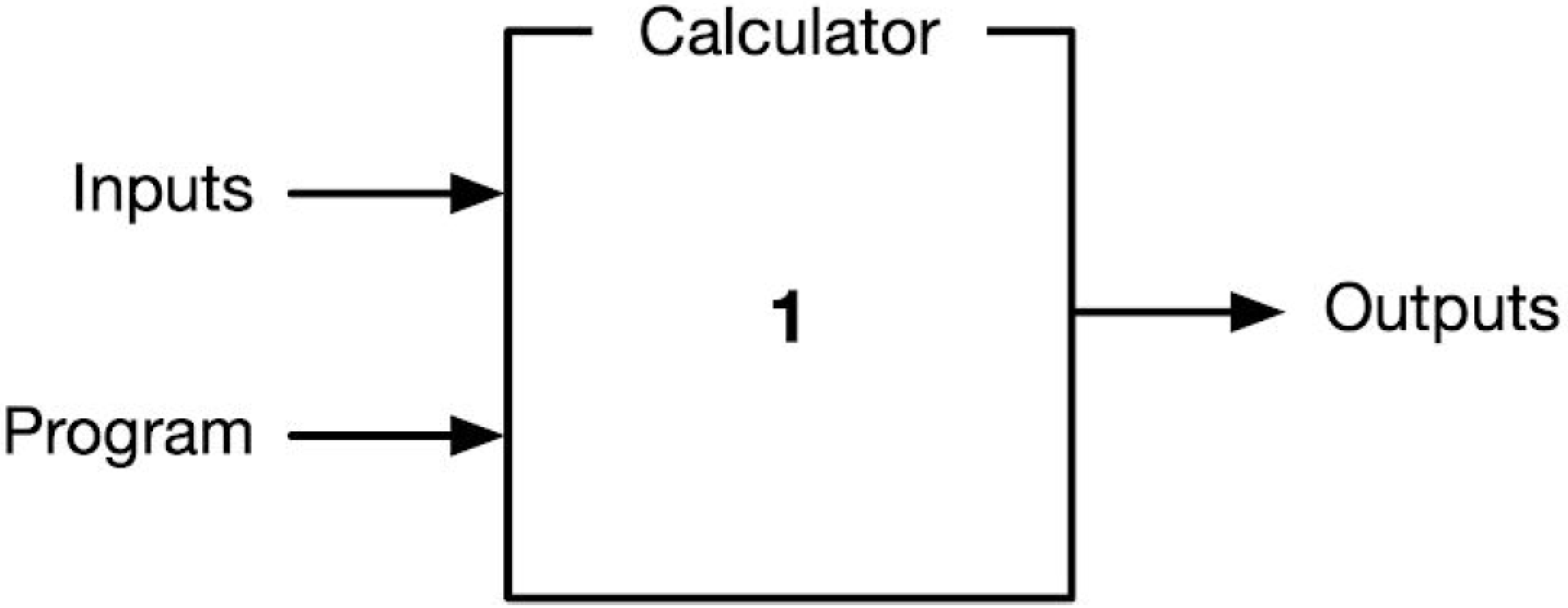
What is Artificial intelligence?

How is Artificial intelligence defined?

- The term ***Artificial Intelligence***, as a research field, was coined at the conference on the campus of Dartmouth College in the summer of **1956**, even though the idea was around since Antiquity: Hephaestus built automatons of metal to work for him or protect others, the Golem in Jewish folklore, etc.

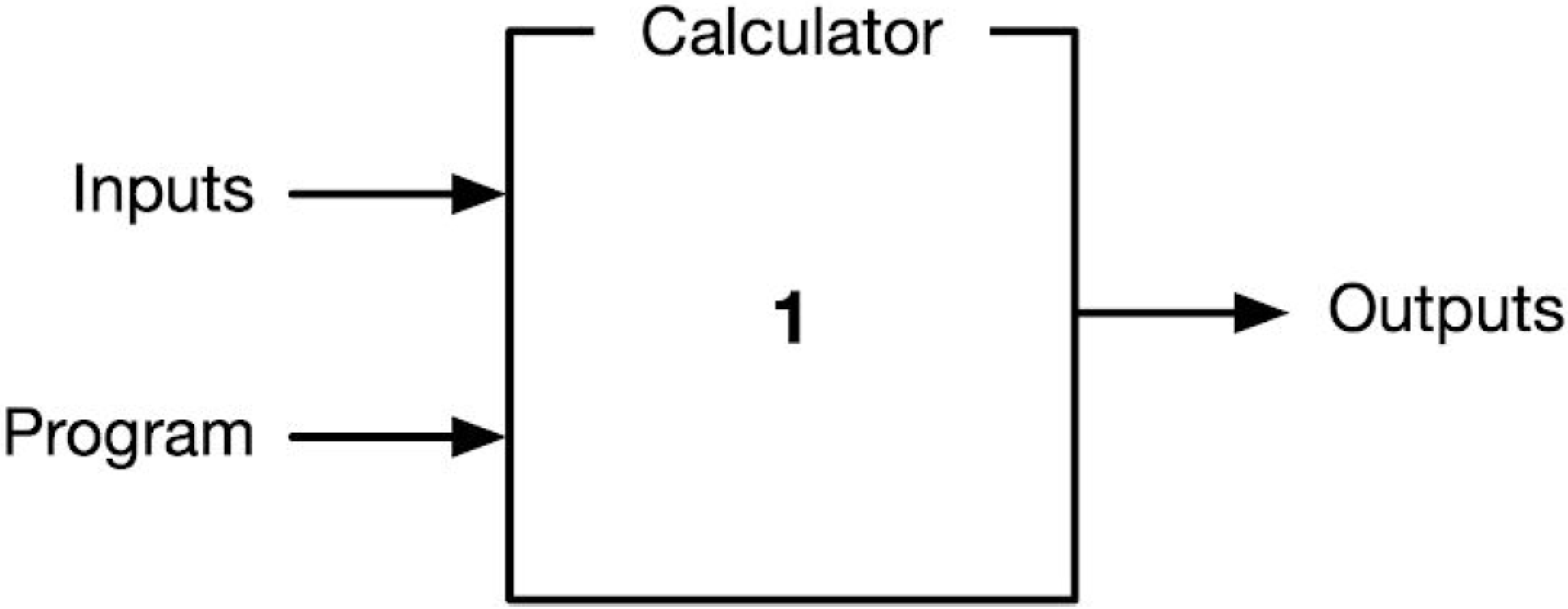
- Closer to the Dartmouth conference but still before, the first manifesto on Artificial Intelligence, an unpublished report ***"Intelligent Machinery"***, written by Alan Turing in **1948**. He already distinguished two different approaches to AI, which may be termed ***"top-down"*** and ***"bottom-up"*** *(now more commonly called knowledge-driven AI and data-driven AI respectively).*

(sources: Wikipedia, https://www.greeklegendsandmyths.com/automatons.html ,
http://www.alanturing.net/turing_archive/pages/Reference%20Articles/what_is_AI/What%20is%20AI02.html
Stanford Encyclopedia of Philosophy: https://plato.stanford.edu/entries/artificial-intelligence/*)*



(1) Hypothetical-deductive machines

(Figure from: Neurons spike back The invention of inductive machines and the artificial intelligence controversy", D. Cardon, J.-P. Cointet, A. Mazières, Translated by Elizabeth Libbrecht In Réseaux Volume 211, Issue 5, 2018, pages 173 to 220)



(2) inductive machines

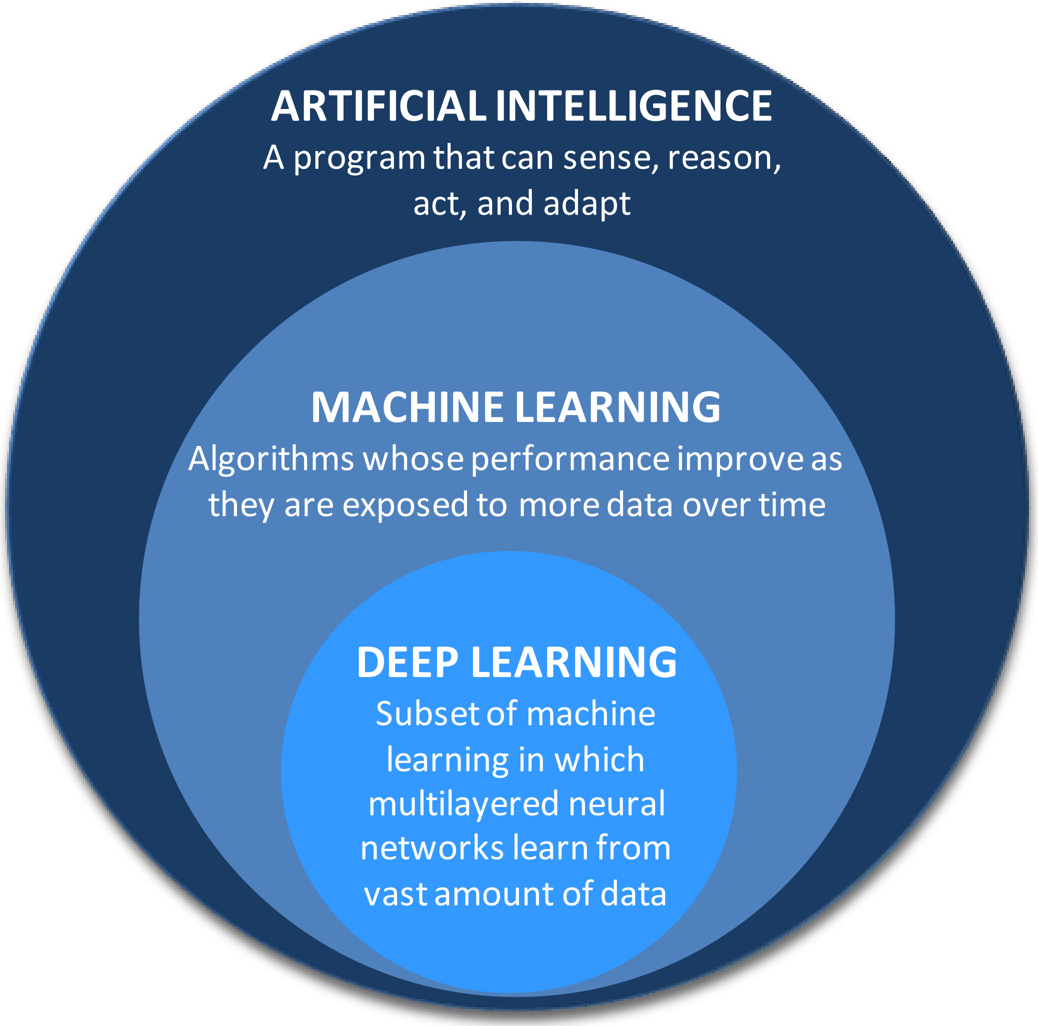
(Figure from: Neurons spike back The invention of inductive machines and the artificial intelligence controversy", D. Cardon, J.-P. Cointet, A. Mazières, Translated by Elizabeth Libbrecht In Réseaux Volume 211, Issue 5, 2018, pages 173 to 220)

Why Artificial Intelligence is so difficult to grasp?

- Frequently, when a technique reaches **mainstream use**, it is **no longer considered as artificial intelligence**; this phenomenon is described as the ***AI effect***: "AI is whatever hasn't been done yet.“ (***Larry Tesler's Theorem***)
  -> e.g. Path Finding (GPS), Chess electronic game, Alpha Go…

- Consequently, AI domain is continuously evolving and so very difficult to grasp.

AI vs Machine Learning vs Deep Learning



But what is Machine Learning?

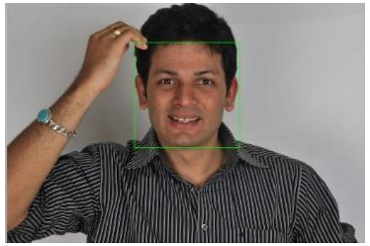
Machine Learning

$$\left(\mathbf{X}\right) \xrightarrow{f(\mathbf{X},\boldsymbol{\alpha})\ ?} y$$

$\left(\mathbf{X}\right)$

y

Face detection



Scores prediction

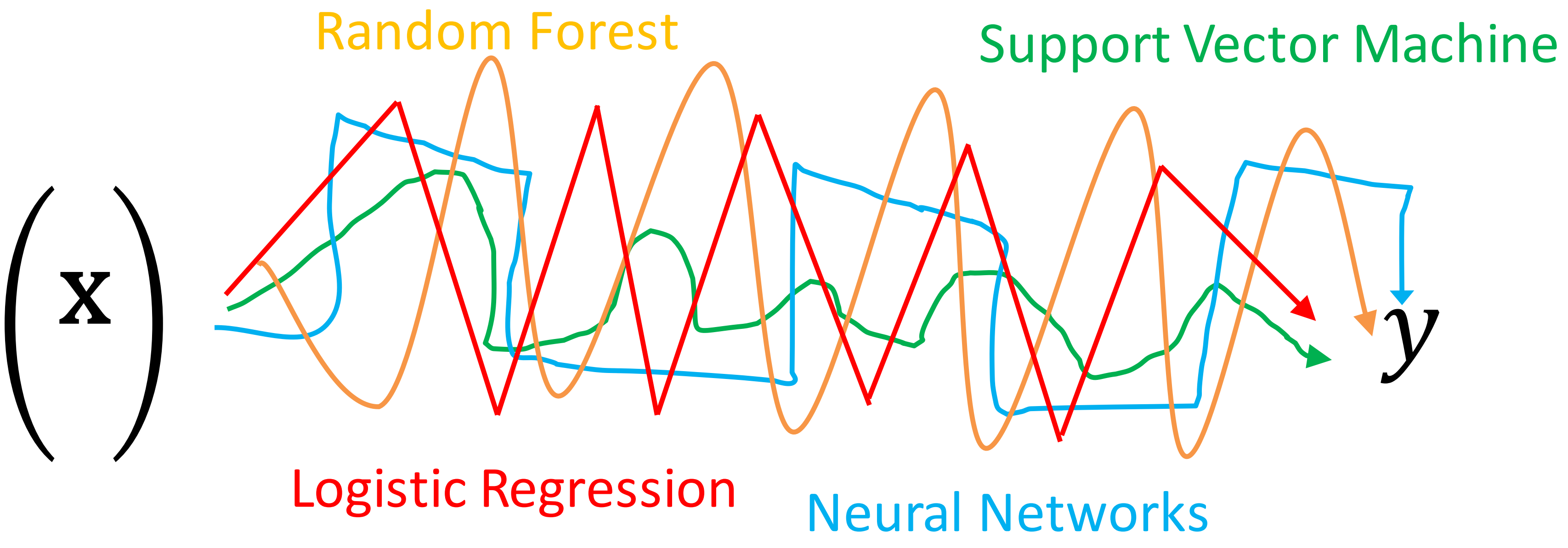
Sport bets

Voice recognition

univ-cotedazur.fr

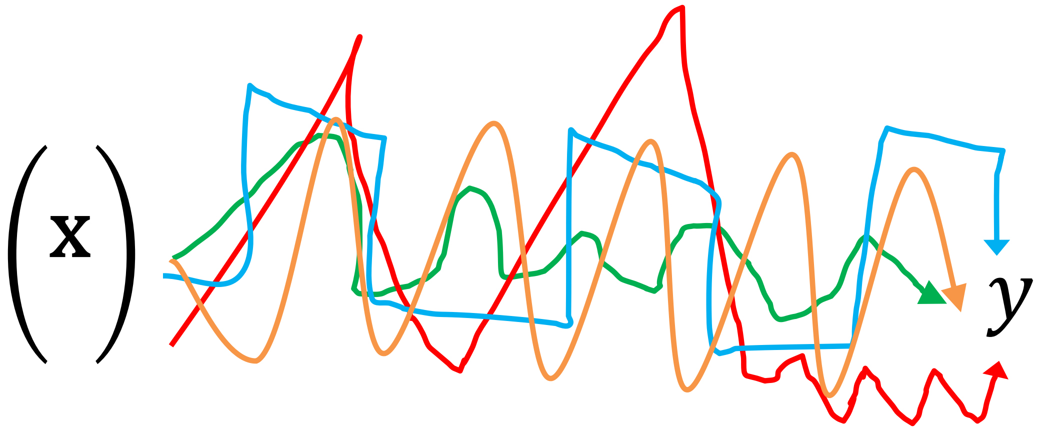
Machine Learning

$$\left(\mathbf{X}\right) \xrightarrow{f(\mathbf{X},\alpha)\ ?} y$$



Machine Learning

$$\left(\mathbf{x}\right) \xrightarrow{f(\mathbf{X},\alpha)\ ?} y$$



ML

$\Leftrightarrow$ "Weather Forcasting"

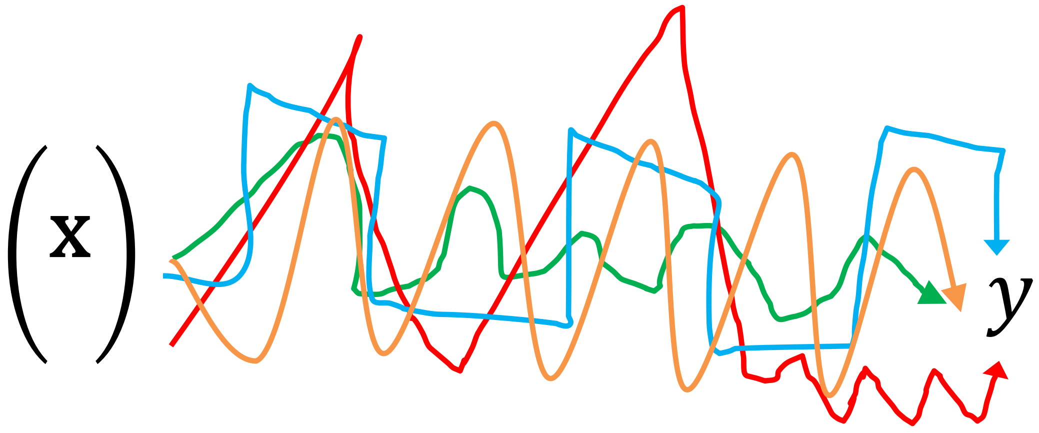
Cf. Question trolley dilemma

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Machine Learning

$$\left(\mathbf{X}\right) \xrightarrow{f(\mathbf{X},\alpha)\,?} y$$



ML

$\neq$ ***AI***

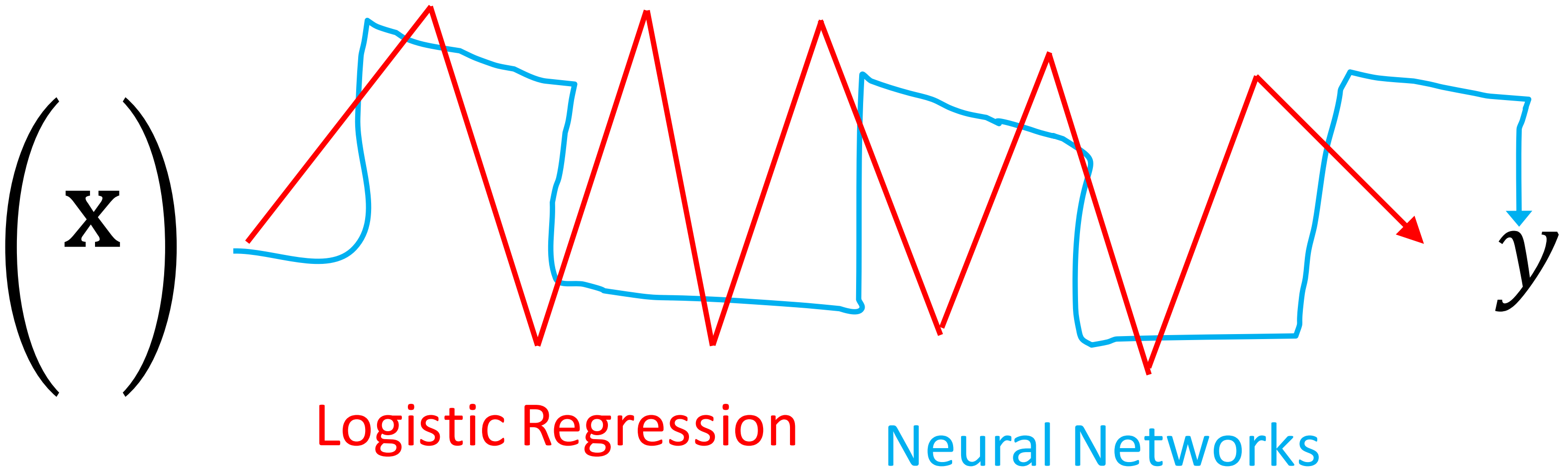
Francis Bach at *Frontier Research and Artificial Intelligence Conference*: **"Machine Learning is not AI"**
(https://erc.europa.eu/sites/default/files/events/docs/Francis_Bach-SEQUOIA-Robust-algorithms-for-learning-from-modern-data.pdf
https://webcast.ec.europa.eu/erc-conference-frontier-research-and-artificial-intelligence-25#)

Beware of the diversion!

Trolley dilemma

Machine Learning

$$\begin{pmatrix} \mathbf{x} \end{pmatrix} \xrightarrow{f(\mathbf{X},\alpha)\ ?} y$$

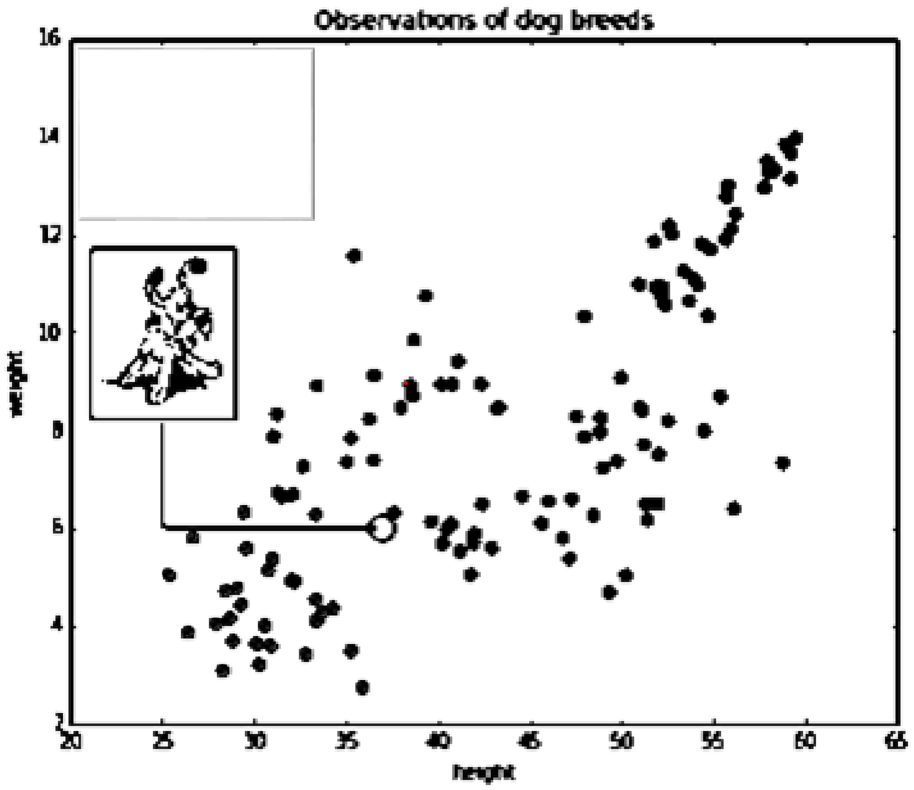


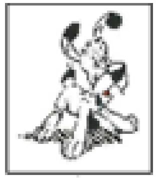
UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Machine Learning is Statistics?

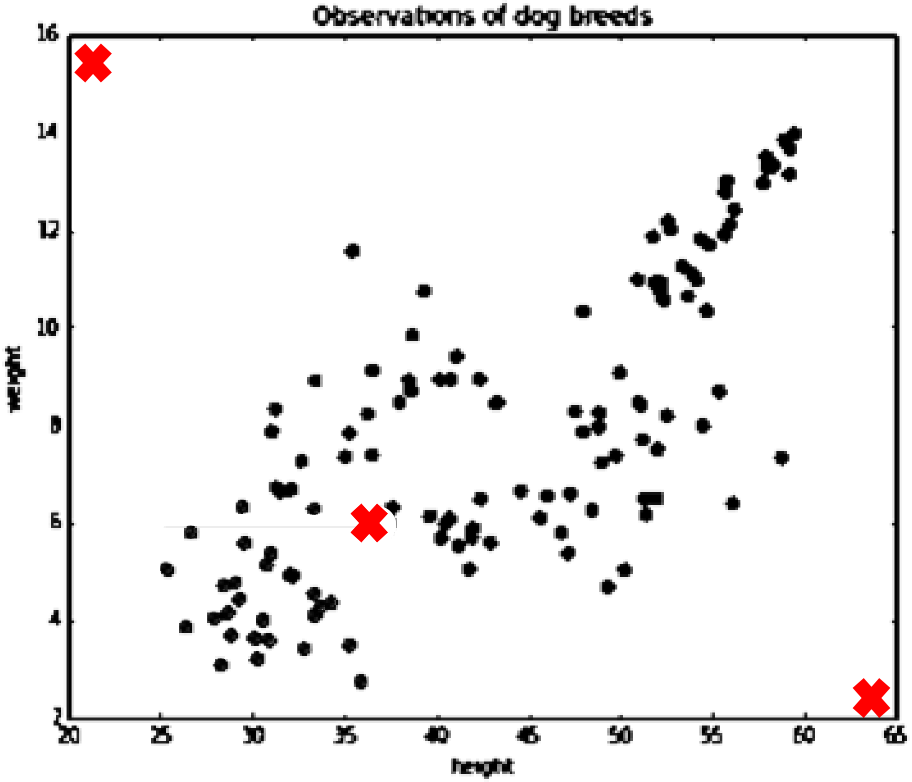
What breed is that Dogmatix (Idéfix)?





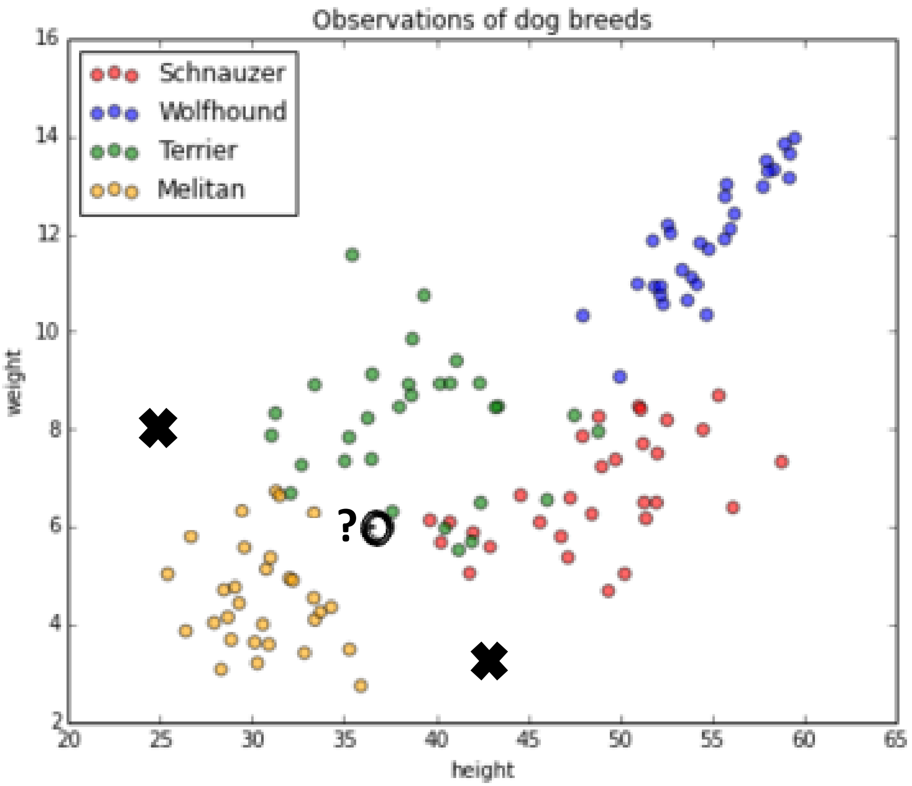
The illustrations of the slides in this section come from the blog "Bayesian Vitalstatistix: What Breed of Dog was Dogmatix?"

Does any real dog get this height and weight?



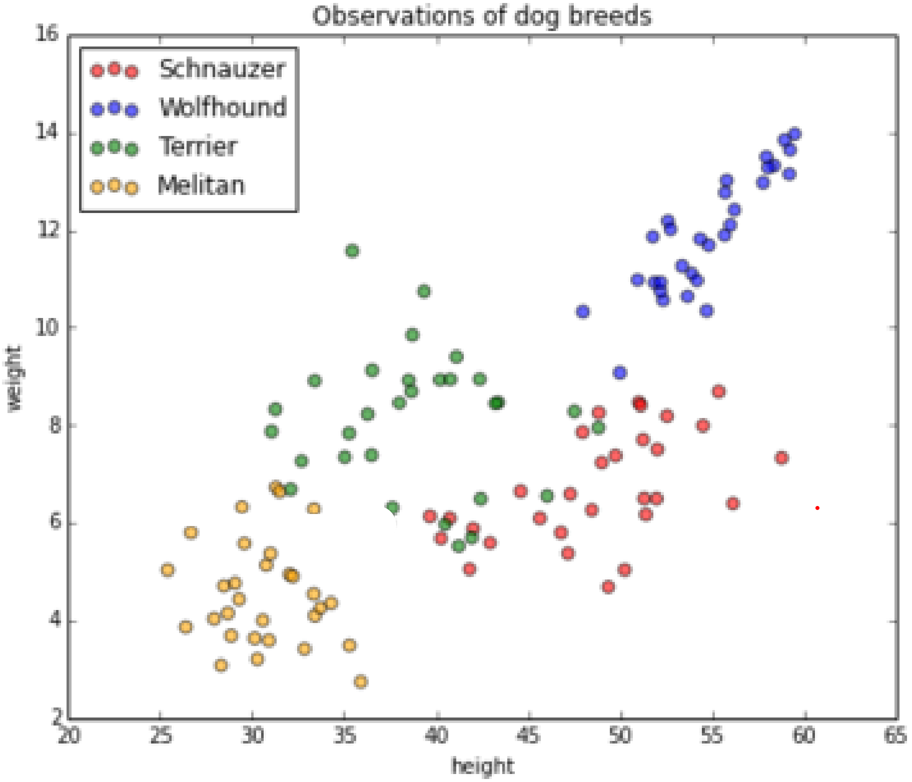
- Let us consider ***x*****,** vectors independently generated in $\boldsymbol{R}^d$ (here $\boldsymbol{R}^2$), following a probability distribution fixed but ***unknown*** $P(\mathbf{x})$.

What should be the breed of these dogs?



- An Oracle assigns a value *y* to each vector ***x*** following a probability distribution *P(y|**x**)* also fixed but ***unknown***.

An oracle provides me with examples?

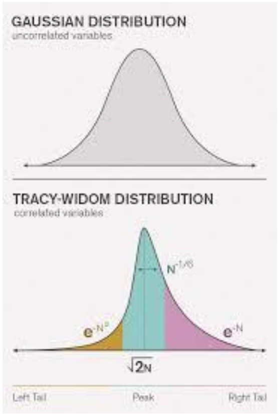


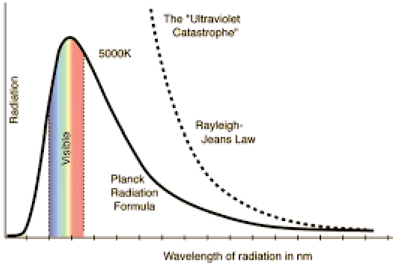
- Let S be a training set
  $S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_m, \mathbf{y}_m)\}$,
  with *m* **training samples i.i.d.** which follow the **joint probability**
  $P(\mathbf{x}, y) = P(\mathbf{x})P(y|\mathbf{x})$.

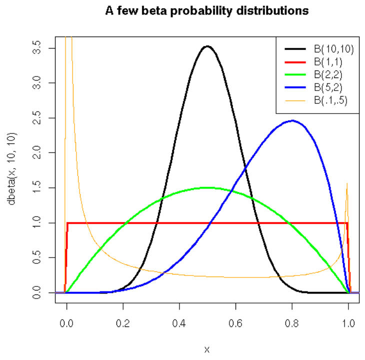
UNIVERSITÉ CÔTE D'AZUR

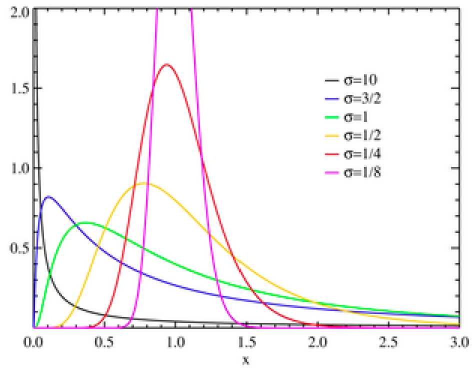
univ-cotedazur.fr

Statistical solution: Models, Hypotheses on data distribution…

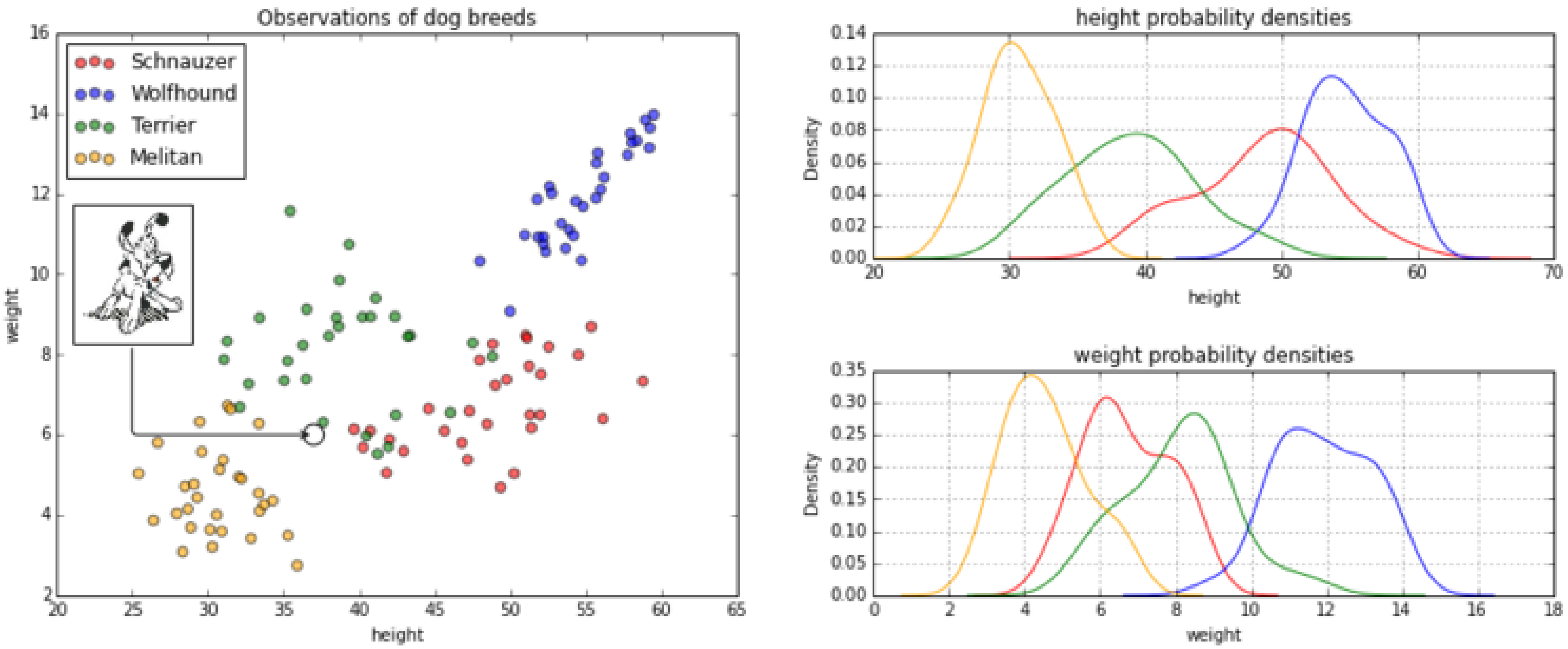




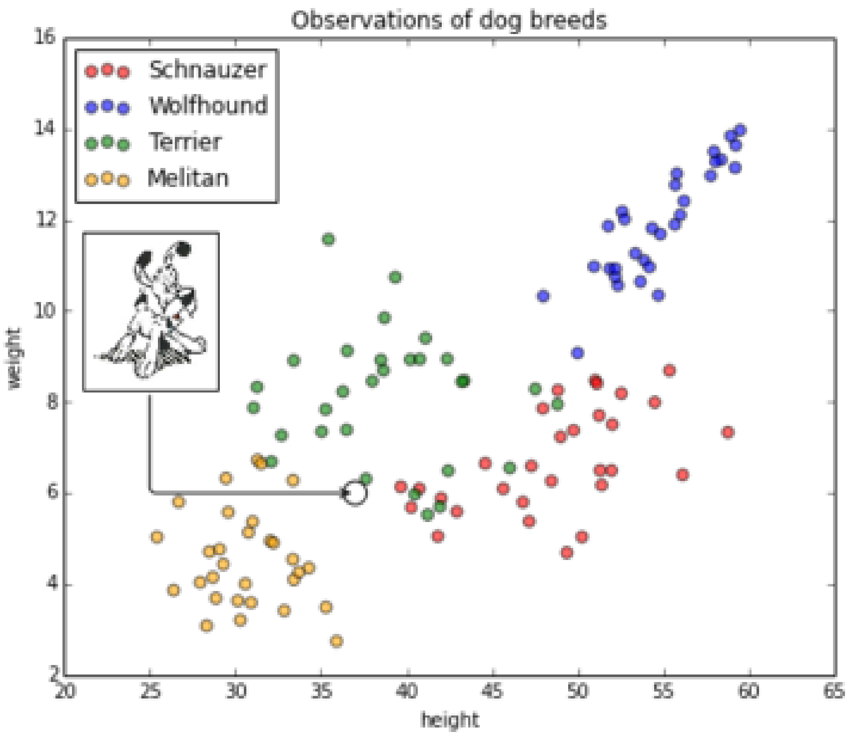


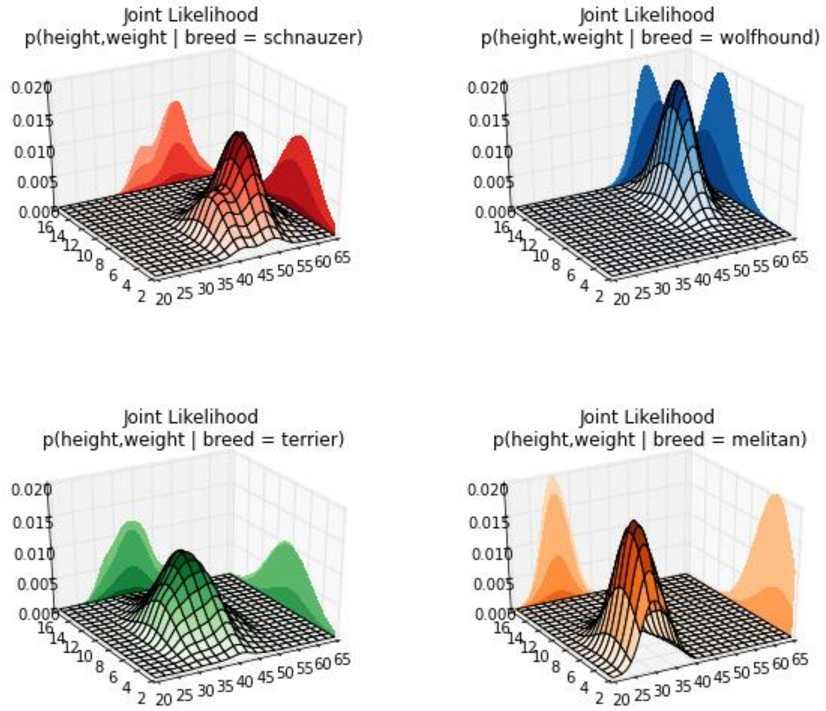


Statistical solution P(height, weight|breed)…

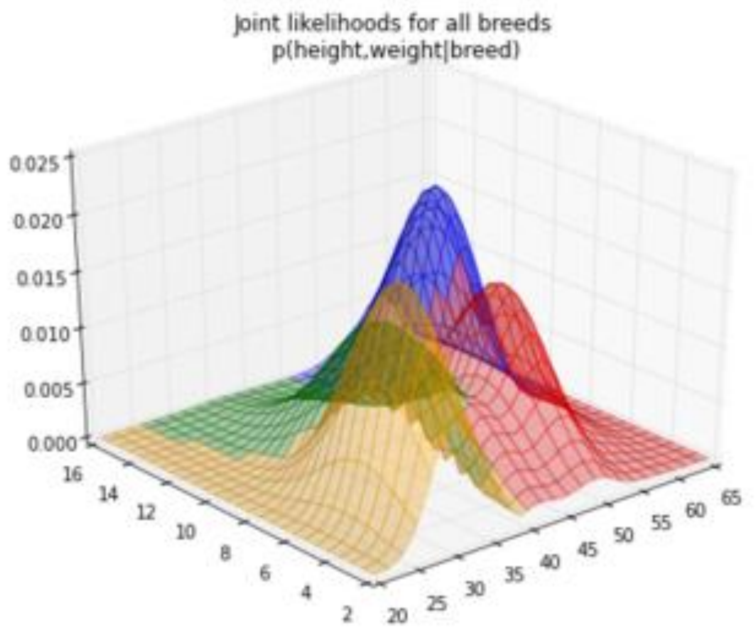


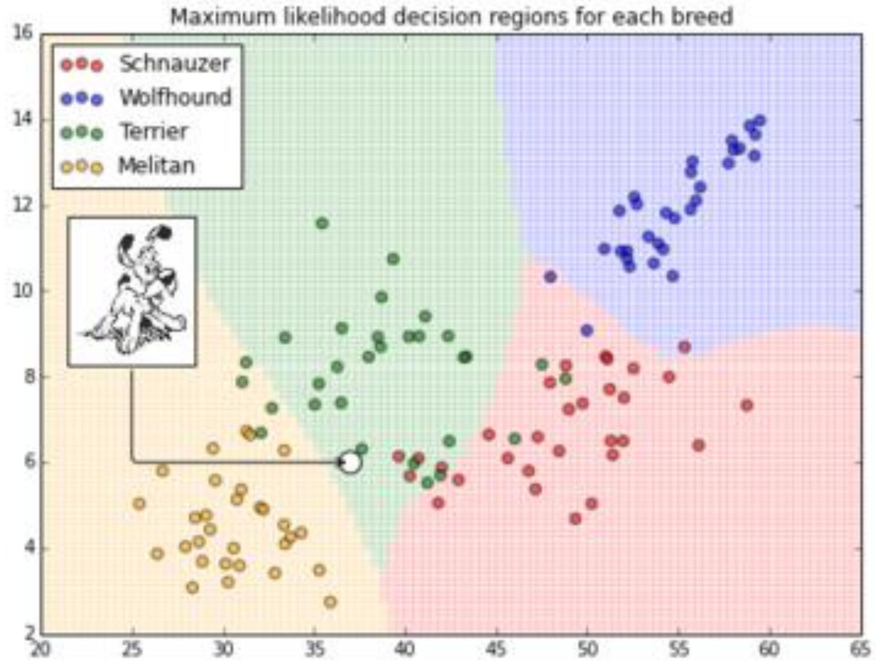
Statistical solution P(height, weight|breed)…



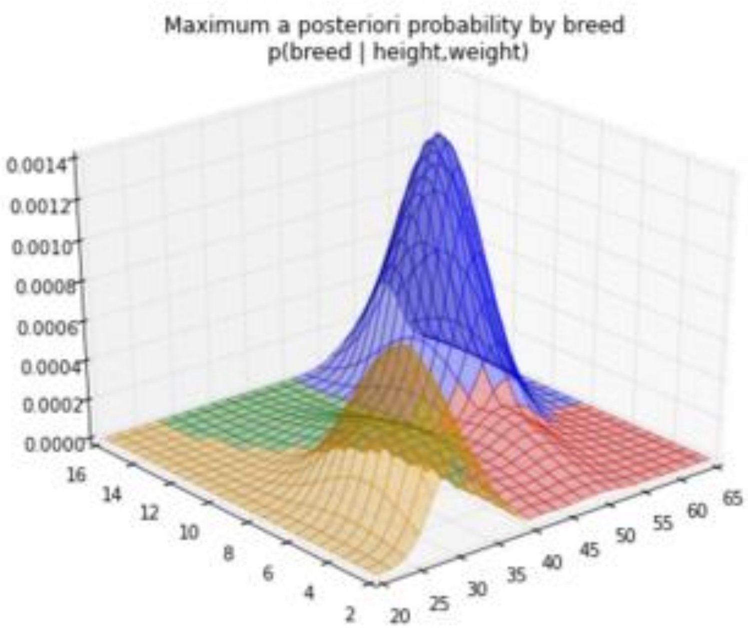


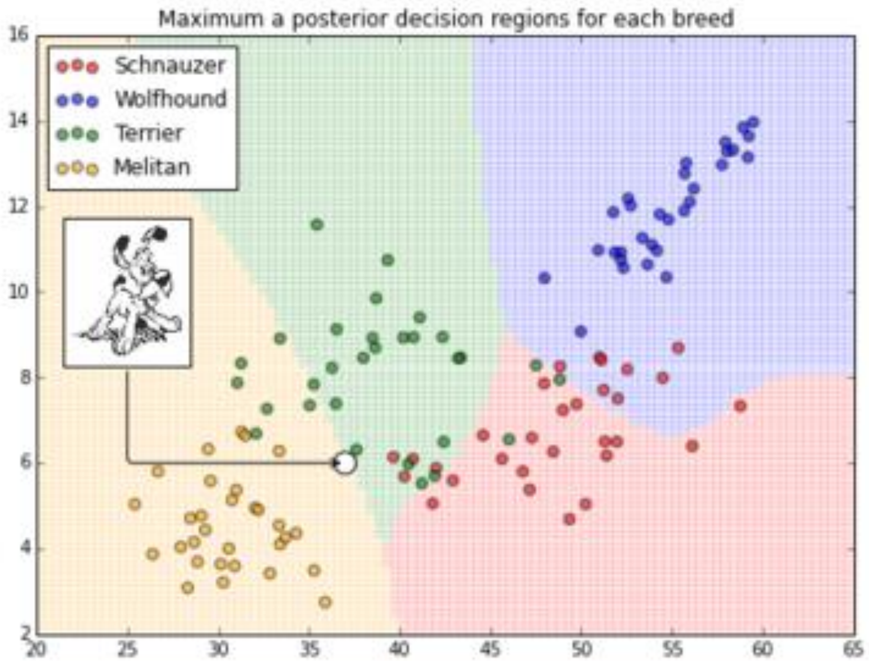
Statistical solution P(height,weight|breed)…





Statistical solution: Bayes, P(breed|height, weight)…

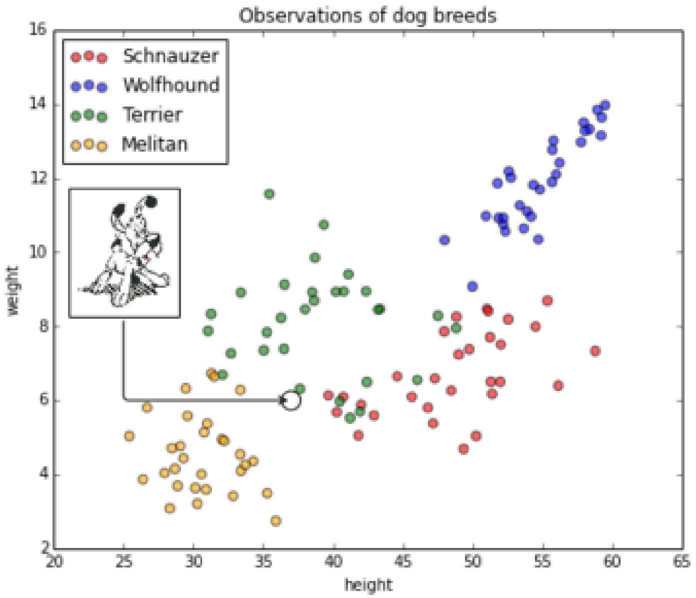




UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Machine Learning



- we have a learning machine (i.e. an algorithm) which can provide a family of functions *{f(**x**;α)}*, where *α* is a set of parameters.

$$\begin{pmatrix} \mathbf{X} \end{pmatrix} \xrightarrow{f(\mathbf{X},\boldsymbol{\alpha})\,?} y$$

The problem of (Machine) Learning

$$\left(\mathbf{X}\right) \xrightarrow{f\,(\mathbf{X},\boldsymbol{\alpha})\,?} y$$

- The problem of learning consists in finding the function (among the $\{f\ (\mathbf{x};\alpha)\}$) which provides the best approximation $\hat{y}$ of the true label y given by the Oracle.

- "***best***" is defined in terms of minimizing a specific *error measure/cost/loss* ***related to your problem/objectives***

  $\boldsymbol{L}((\mathbf{x}, y), \alpha) \in [a; b]$.

The problem of (Machine) Learning

- Thus, the objective is to minimize the *(real)* ***Risk***, i.e. the expectation of the error cost:

$$R(\alpha) = \int \boldsymbol{L}((\mathbf{x}, y), \alpha) dP(\mathbf{x}, y)$$

where P(**x**, y) is unknown.

- The training set **S** = $\{(\mathbf{x}_i, y_i)\}_{i=1,\dots,m}$ is built through an i.i.d. sampling according to P(**x** ,y). Since we cannot compute $R(\alpha)$, we look for minimizing the ***Empirical Risk*** instead:

$$R_{emp}(\alpha) = \frac{1}{m} \sum_{k=1}^{m} \boldsymbol{L}((\mathbf{x}_i, y_i), \alpha)$$

Machine Learning fundamental Hypothesis

$S = \{(x_i, y_i)\}_{i=1,\dots,m}$ is built through an *i.i.d.* sampling according to $P(x, y)$.

Machine Learning $\Leftrightarrow$ ***Statistics***

Train through Cross-Validation

Machine Learning $\Leftrightarrow$ ***Statistics***

Training set & Test set have to be distributed according to the same law

Vapnik learning theory (1995)

Vapnik had proven the following equation $\forall m$ with a probability at least equal to $1 - \eta$:

$$R(\alpha_m) \leq R_{emp}(\alpha_m) + (b-a)\sqrt{\frac{d_{VC}(ln(2m/d_{VC})+1) - ln(\eta/4)}{m}}$$

Training Error

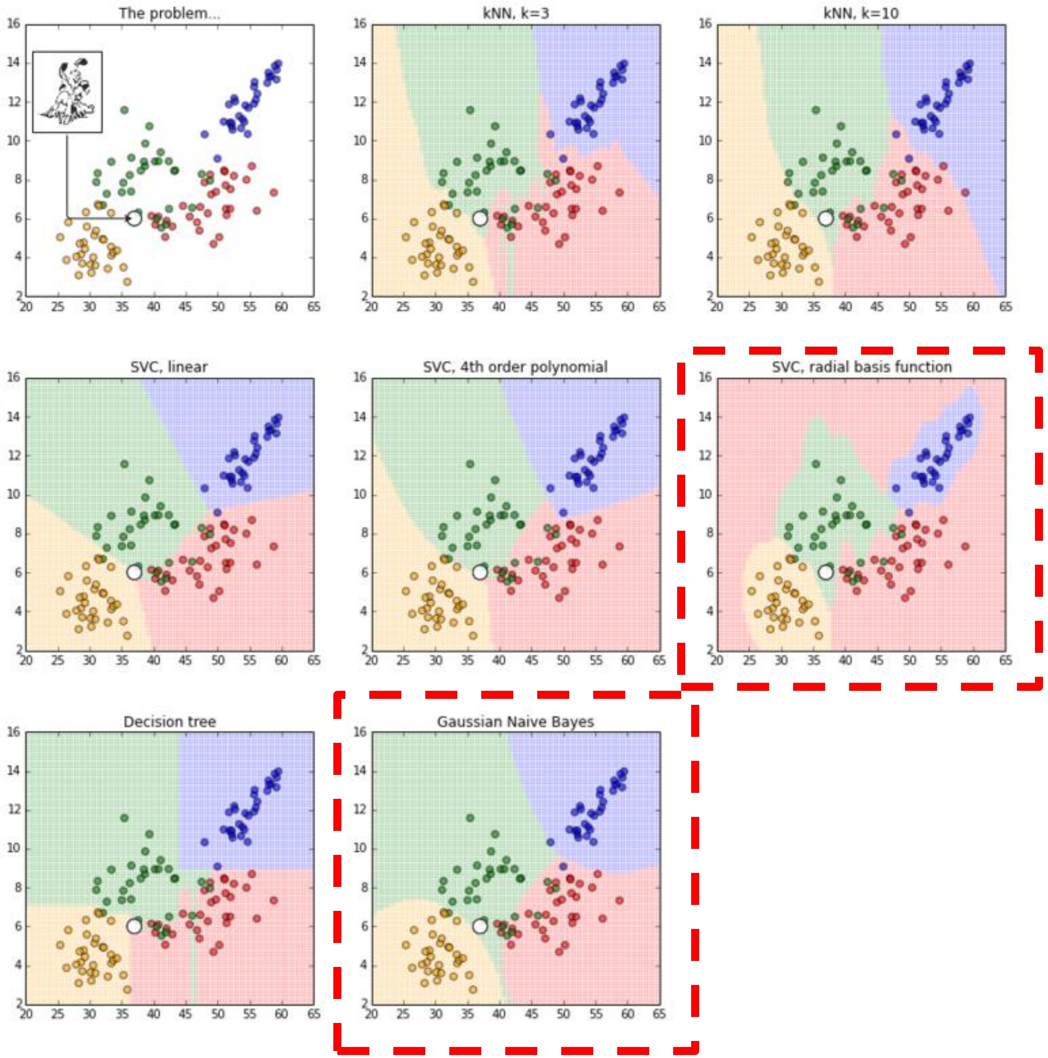
Generalization Error

Thus minimizing the ***Risk*** depends on minimzing the ***Empirical Risk*** and the ***Generalization Error*** of the model which depends on *m* (the number of training sample), and d_{VC} (the complexity of the model family chosen, also called *Vapnik-Chervonenkis Dimension).*

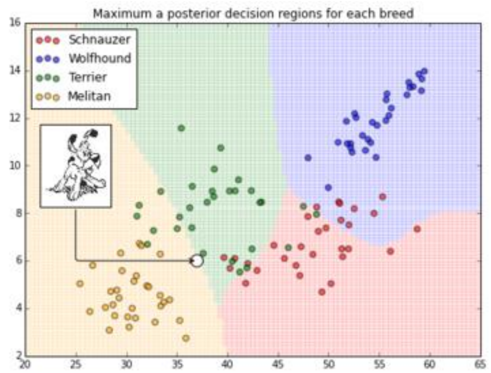
UNIVERSITÉ CÔTE D'AZUR

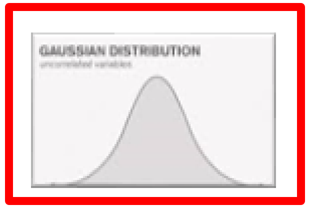
univ-cotedazur.fr

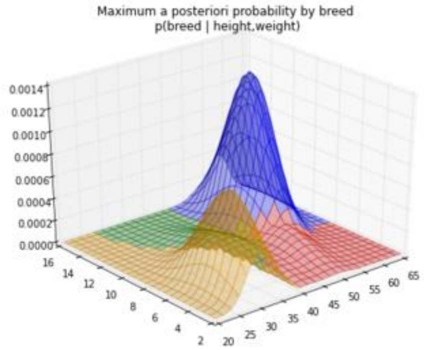
Machine Learning vs Statistics

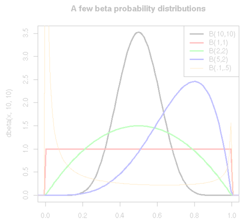


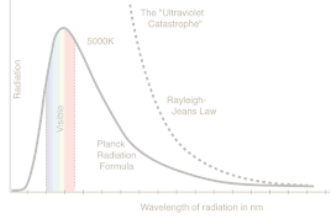
VS









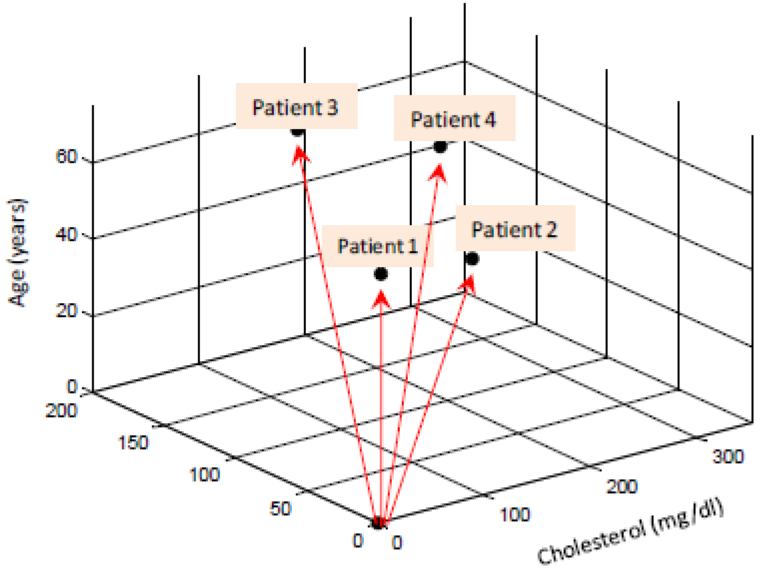


MATHEMATICAL BASICS

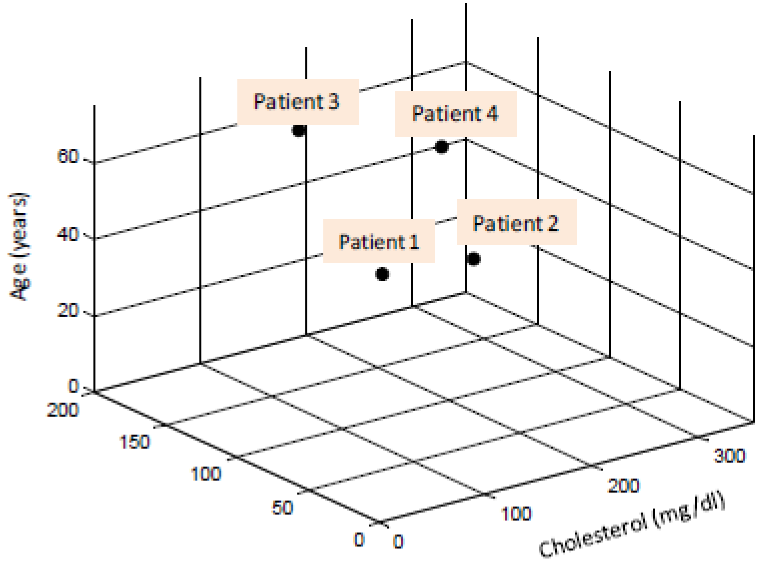
How to represent samples geometrically? Vectors & points in n-dimensional space (R^n)

Patient id	*Cholesterol* (mg/dl)	*Systolic BP* (mmHg)	*Age* (years)	**Tail of the vector**	**Arrow-head of the vector**
1	150	110	35	(0,0,0)	(150, 110, 35)
2	250	120	30	(0,0,0)	(250, 120, 30)
3	140	160	65	(0,0,0)	(140, 160, 65)
4	300	180	45	(0,0,0)	(300, 180, 45)

<u>Vector representation</u>



<u>Point representation</u>



From "A Gentle Introduction to Support Vector Machines in Biomedicine", A. Statnikov, et al, AMIA 2010.

Basic operation on vectors in R^n

1. Multiplication by a scalar

Consider a vector $\mathbf{a} = (a_1, a_2, \dots, a_n)$ and a scalar c

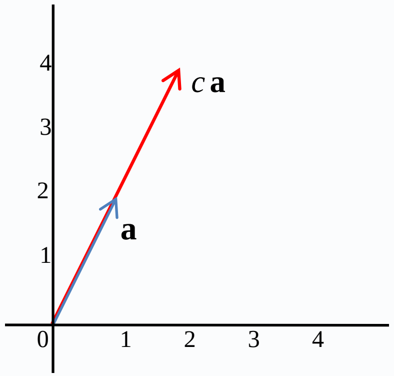
Define: $c\,\mathbf{a} = (ca_1, ca_2, \dots, ca_n)$

When you multiply a vector by a scalar, you "stretch" it in the same or opposite direction depending on whether the scalar is positive or negative.

$\mathbf{a} = (1, 2)$

$c = 2$

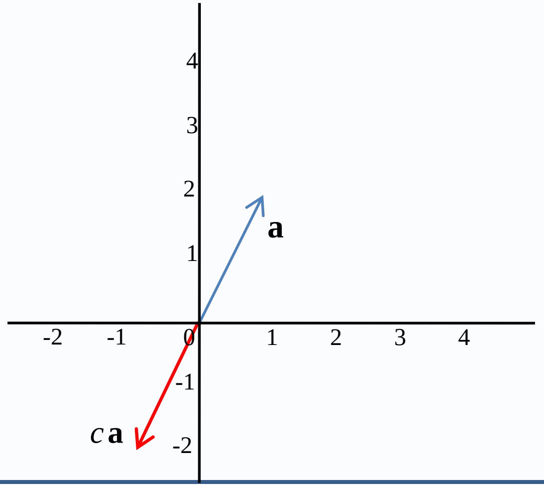
$c\,\mathbf{a} = (2, 4)$



$\mathbf{a} = (1, 2)$

$c = -1$

$c\,\mathbf{a} = (-1, -2)$



Basic operation on vectors in $\mathbf{R}^n$

2. Addition

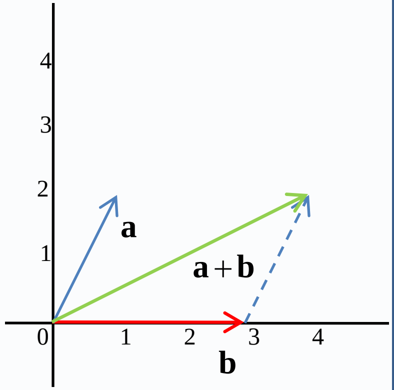
Consider vectors $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ and $\mathbf{b} = (b_1, b_2, \ldots, b_n)$

Define: $\mathbf{a} + \mathbf{b} = (a_1 + b_1, a_2 + b_2, \ldots, a_n + b_n)$

$\mathbf{a} = (1, 2)$

$\mathbf{b} = (3, 0)$

$\mathbf{a} + \mathbf{b} = (4, 2)$



Recall addition of forces in classical mechanics.

From "A Gentle Introduction to Support Vector Machines in Biomedicine", A. Statnikov, et al, AMIA 2010.

Basic operation on vectors in R^n

3. Subtraction

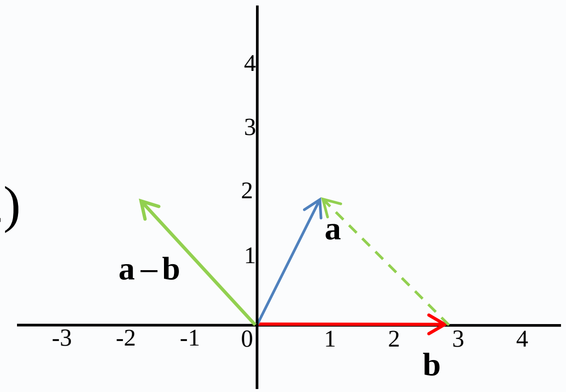
Consider vectors $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ and $\mathbf{b} = (b_1, b_2, \ldots, b_n)$

Define: $\mathbf{a} - \mathbf{b} = (a_1 - b_1, a_2 - b_2, \ldots, a_n - b_n)$

$\mathbf{a} = (1, 2)$

$\mathbf{b} = (3, 0)$

$\mathbf{a} - \mathbf{b} = (-2, 2)$



What vector do we need to add to $\vec{b}$ to get $\vec{a}$? I.e., similar to subtraction of real numbers.

From "A Gentle Introduction to Support Vector Machines in Biomedicine", A. Statnikov, et al, AMIA 2010.

Basic operation on vectors in R^n

4. Euclidian length or L2-norm

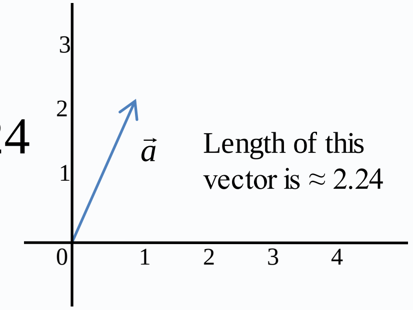
Consider a vector $\mathbf{a} = (a_1, a_2, ..., a_n)$

Define the L2-norm: $\|\mathbf{a}\|_2 = \sqrt{a_1^2 + a_2^2 + ... + a_n^2}$

We often denote the L2-norm without subscript, i.e. $\|\mathbf{a}\|$

$\mathbf{a} = (1, 2)$

$\|\mathbf{a}\|_2 = \sqrt{5} \approx 2.24$



L2-norm is a typical way to measure length of a vector; other methods to measure length also exist.

From "A Gentle Introduction to Support Vector Machines in Biomedicine", A. Statnikov, et al, AMIA 2010.

Basic operation on vectors in R^n

5. Dot product

Consider vectors $\mathbf{a} = (a_1, a_2, ..., a_n)$ and $\mathbf{b} = (b_1, b_2, ..., b_n)$

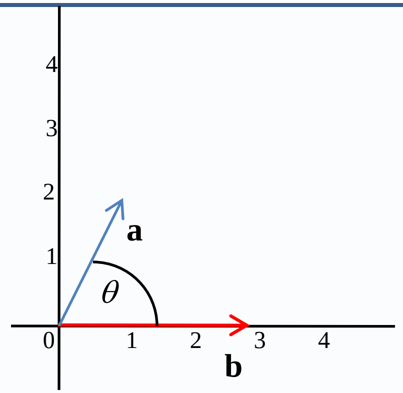
Define dot product: $\mathbf{a} \cdot \mathbf{b} = a_1 b_1 + a_2 b_2 + ... + a_n b_n = \sum_{i=1}^{n} a_i b_i$

The law of cosines says that $\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\|_2 \|\mathbf{b}\|_2 \cos\theta$ where θ is the angle between $\mathbf{a}$ and $\mathbf{b}$. Therefore, when the vectors are perpendicular $\mathbf{a} \cdot \mathbf{b} = 0$.

$\mathbf{a} = (1, 2)$

$\mathbf{b} = (3, 0)$

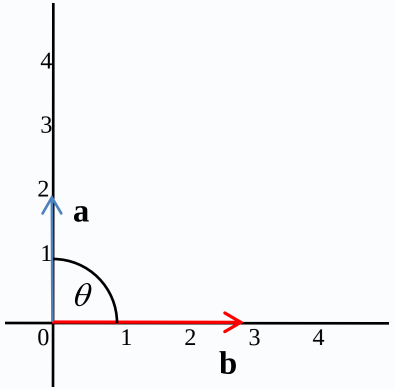
$\mathbf{a} \cdot \mathbf{b} = 3$



$\mathbf{a} = (0, 2)$

$\mathbf{b} = (3, 0)$

$\mathbf{a} \cdot \mathbf{b} = 0$



Basic operation on vectors in R^n

5. Dot product (continued)

$$\mathbf{a} \cdot \mathbf{b} = a_1 b_1 + a_2 b_2 + \ldots + a_n b_n = \sum_{i=1}^{n} a_i b_i$$

- Property: $\mathbf{a} \cdot \mathbf{a} = a_1 a_1 + a_2 a_2 + \ldots + a_n a_n = \|\mathbf{a}\|_2^2$
- In the classical regression equation $y = \mathbf{w} \cdot \mathbf{x} + b$

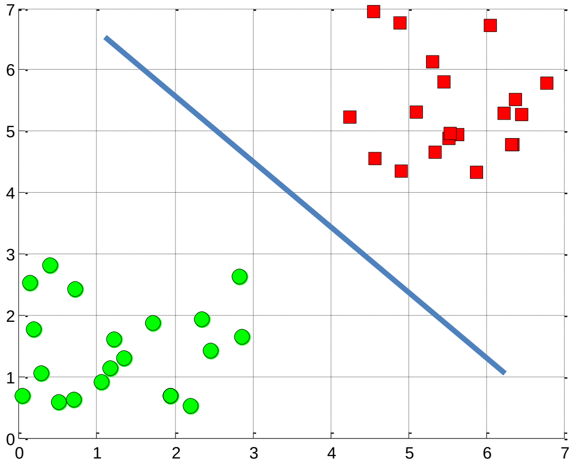
  the response variable *y* is just a dot product of the vector representing patient characteristics ($\mathbf{x}$) and the regression weights vector ($\mathbf{w}$) which is common across all patients plus an offset b.

From "A Gentle Introduction to Support Vector Machines in Biomedicine", A. Statnikov, et al, AMIA 2010.

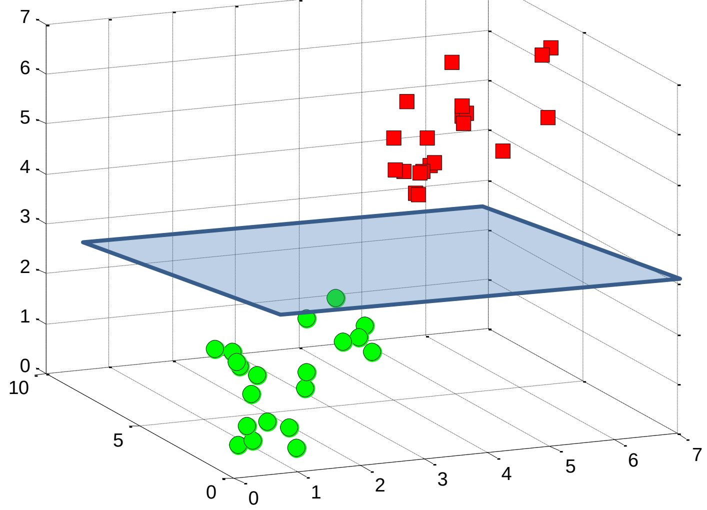
Hyperplanes as decision surfaces

- A hyperplane is a linear decision surface that splits the space into two parts;
- It is obvious that a hyperplane is a binary classifier.

A hyperplane in $\mathbb{R}^2$ is a line

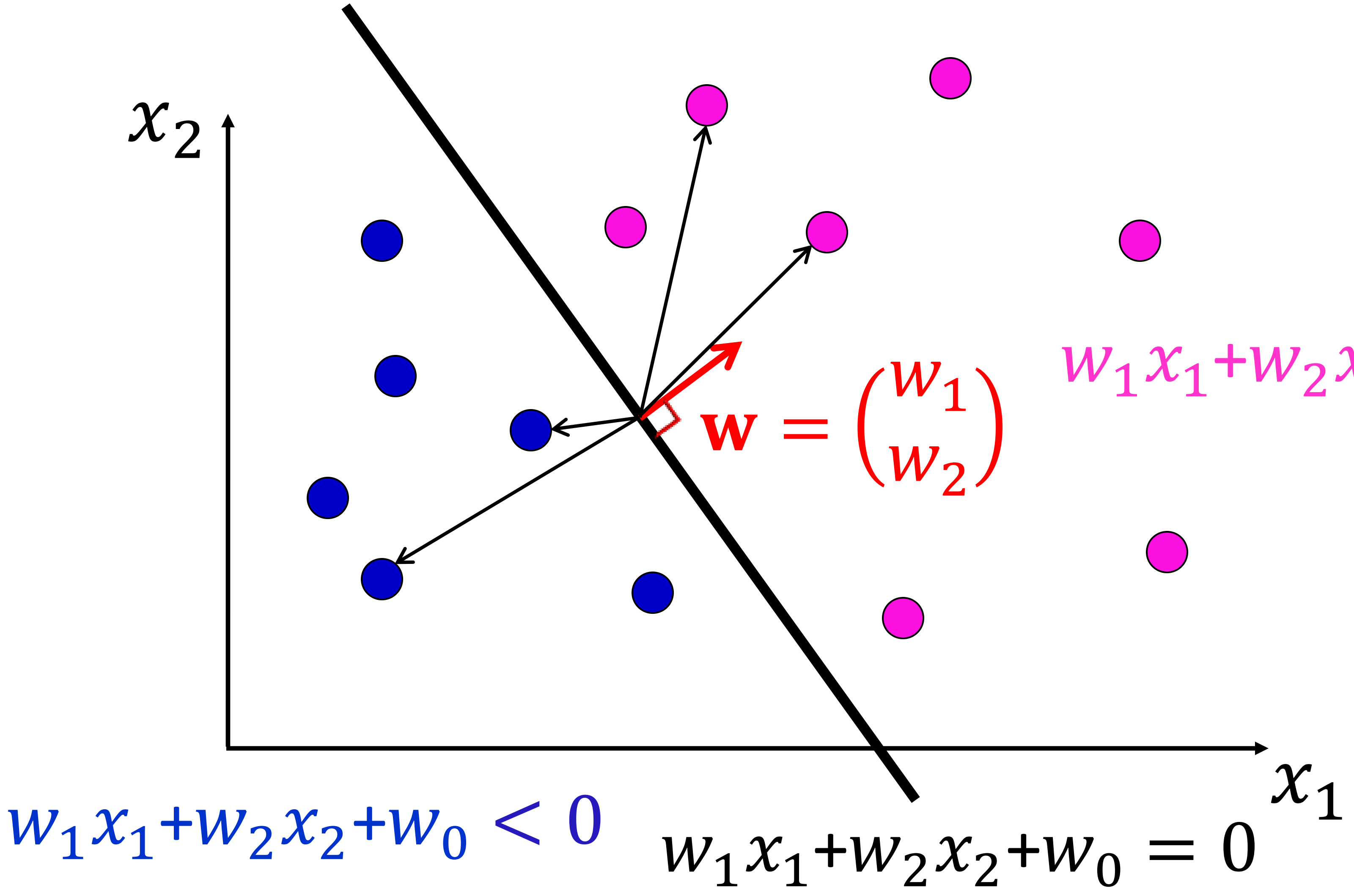


A hyperplane in $\mathbb{R}^3$ is a plane



A hyperplane in $\mathbb{R}^n$ is an n-1 dimensional subspace

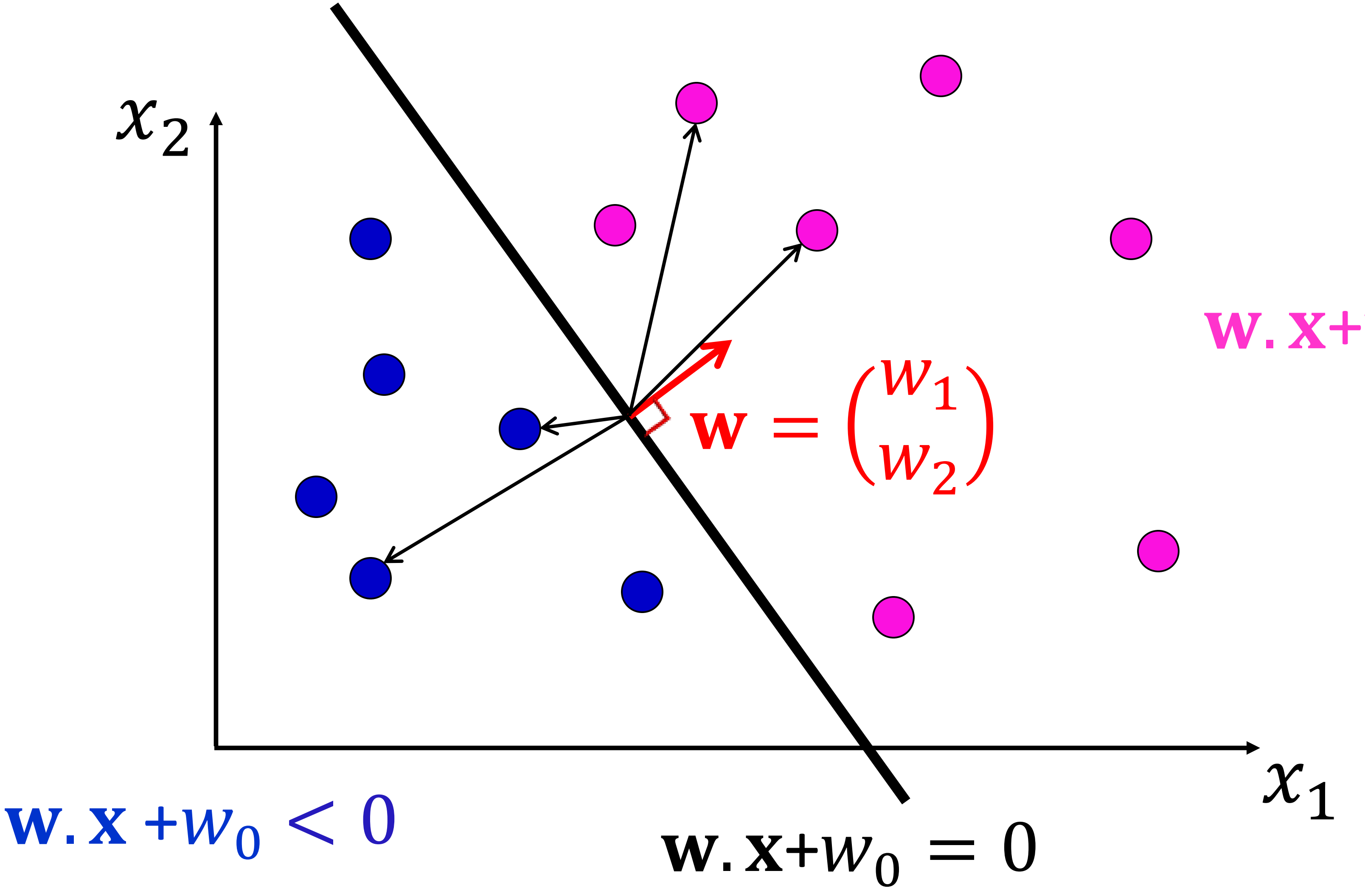
Geometry and Algebra



UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Geometry and Algebra



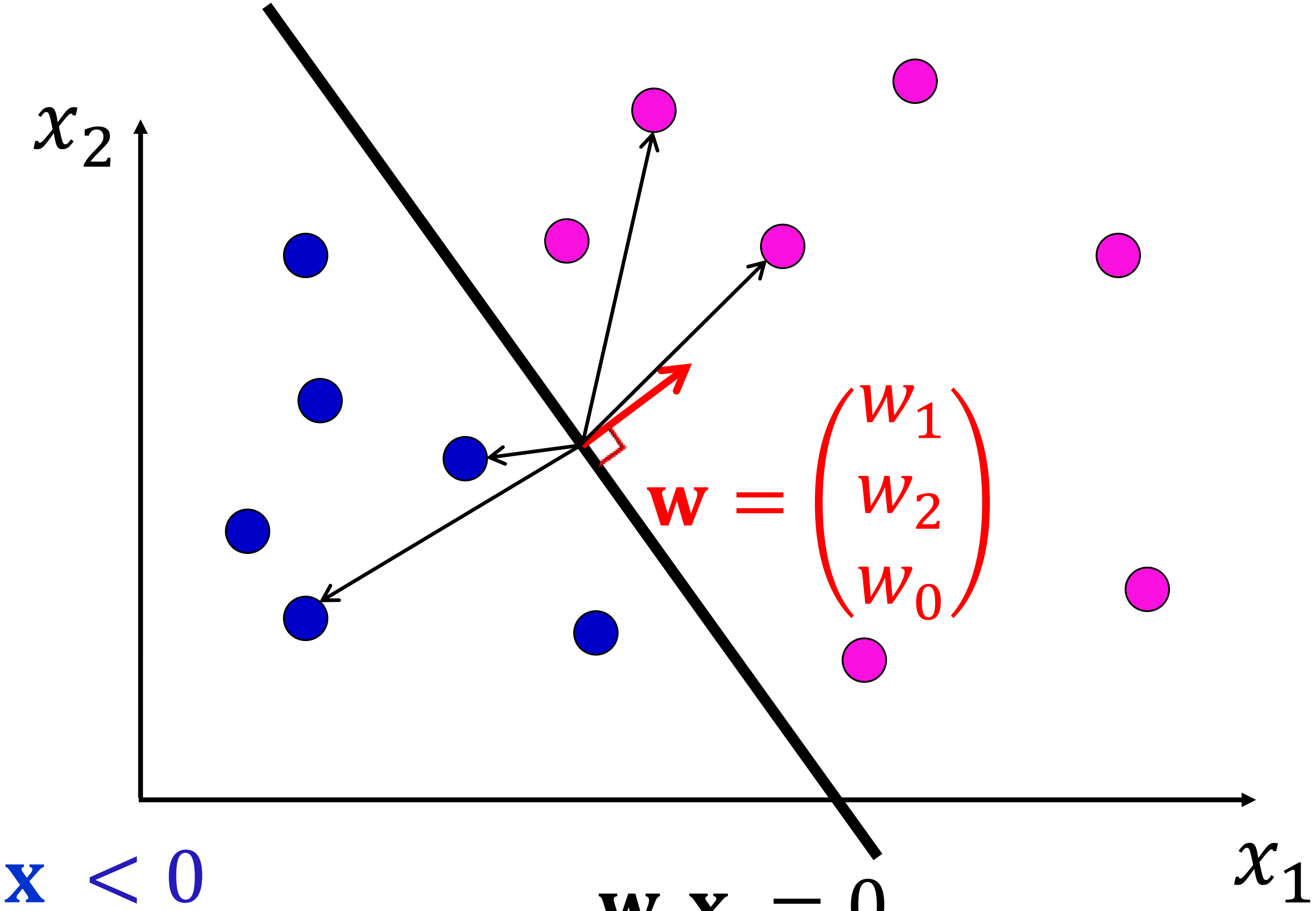
UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Simplification

$$\begin{pmatrix} w_1 \\ w_2 \\ w_0 \end{pmatrix} . \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix} > 0$$

$$\mathbf{w.x} > 0$$

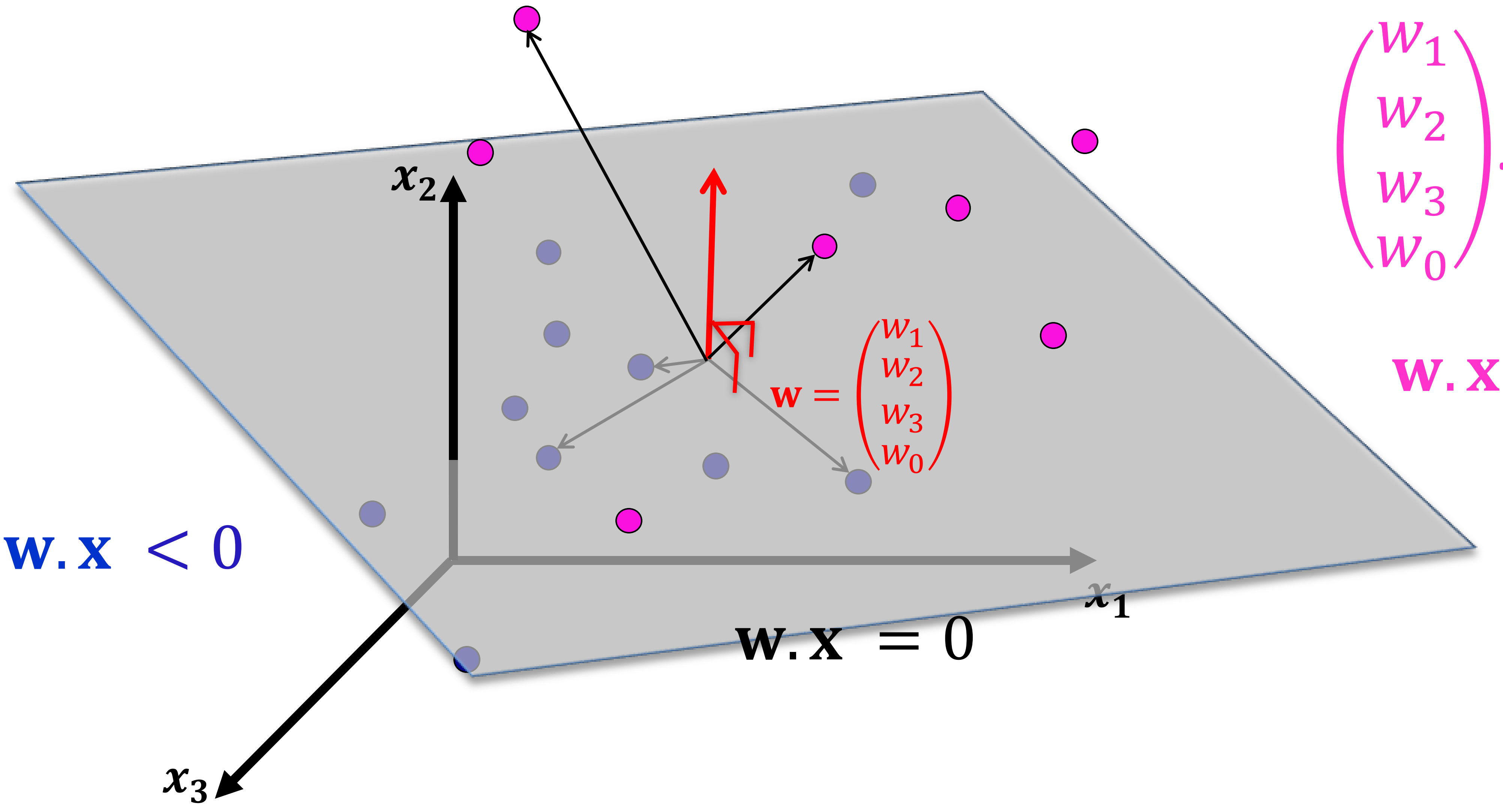


$\mathbf{w.x} < 0$

$\mathbf{w.x} = 0$

univ-cotedazur.fr

Geometry and Algebra



$$\begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_0 \end{pmatrix} . \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{pmatrix} > 0$$

$$\mathbf{w.x} > 0$$

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Derivative

- Derivative of a scalar function (of one variable)
  - $(ax)' = a$
  - $(ax + b)' = a$
  - $(g(f(x)))' = g'(f(x))f'(x)$ **Chain rule**

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Gradient operator

- Gradient of a multivariate function ($\mathbf{x}$ is a vector)
  - $\nabla_{\mathbf{x}}(\boldsymbol{a}\mathbf{x}) = \boldsymbol{a}$
  - $\nabla_{\mathbf{x}}(\boldsymbol{a}\mathbf{x} + b) = \boldsymbol{a}$
  - $\nabla_{\mathbf{x}}(\mathrm{g}(\mathrm{f}(\mathbf{x}))) = \nabla_{\mathbf{x}}\mathrm{g}\,(\mathrm{f}(\mathbf{x}))\,\nabla_{\mathbf{x}}\mathrm{f}(\mathbf{x})$ **(Chain rule)**

  - $\nabla_{\mathbf{x}_\mathbf{i}}(\mathbf{v}.\mathbf{x}) = \nabla_{\mathbf{x}_\mathbf{i}}(v_1.x_1 + v_2.x_2 + \cdots + v_n.xn) = v_i$

Overview

- Machine Learning vs Statistics
- Math Basics
- **Simple Model**
- From Simple to Complex

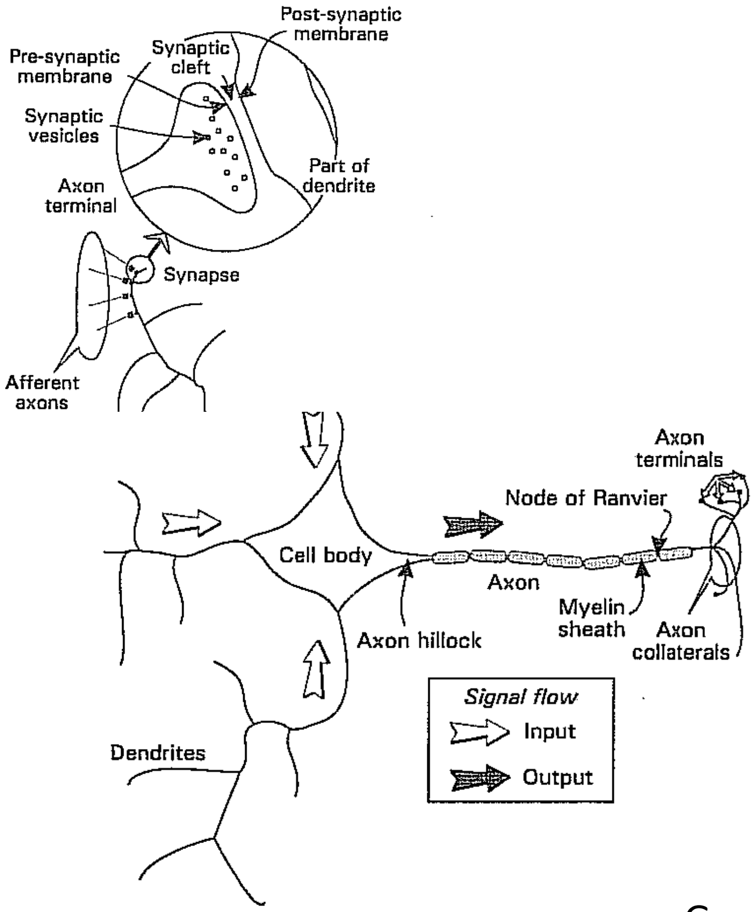
SIMPLE MODEL

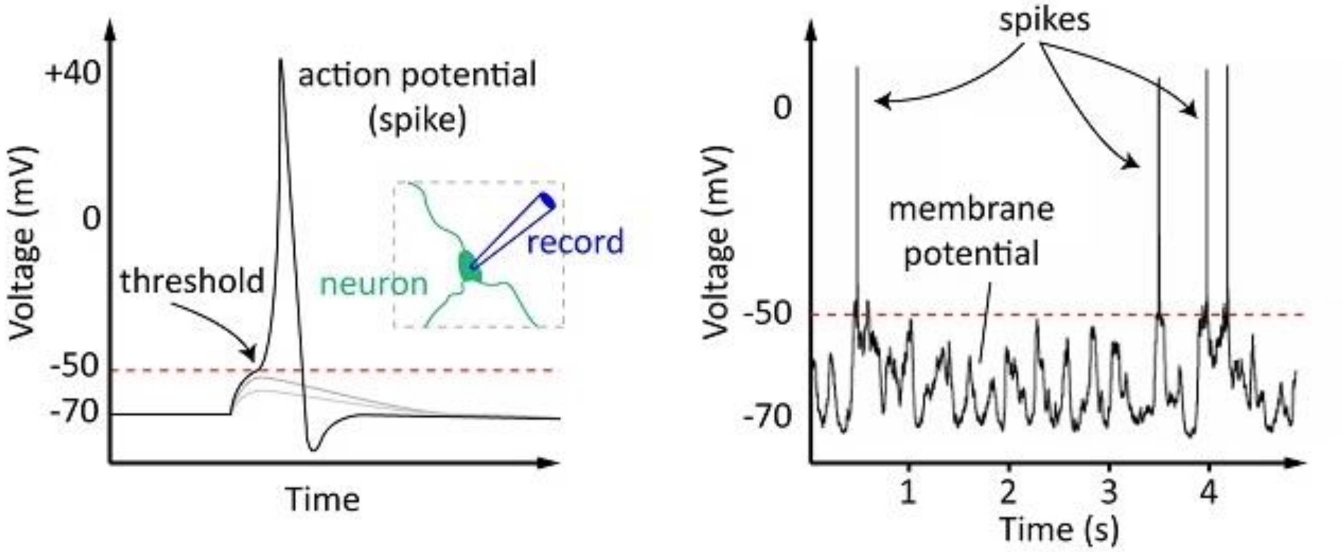
UNIVERSITÉ CÔTE D'AZUR
univ-cotedazur.fr

Initial Model: Perceptron

Biological neuron

- Before we study artificial neurons, let's look at a biological neuron





Gurney, K. (1997). *An introduction to neural networks*. CRC press.

First, biological neurons

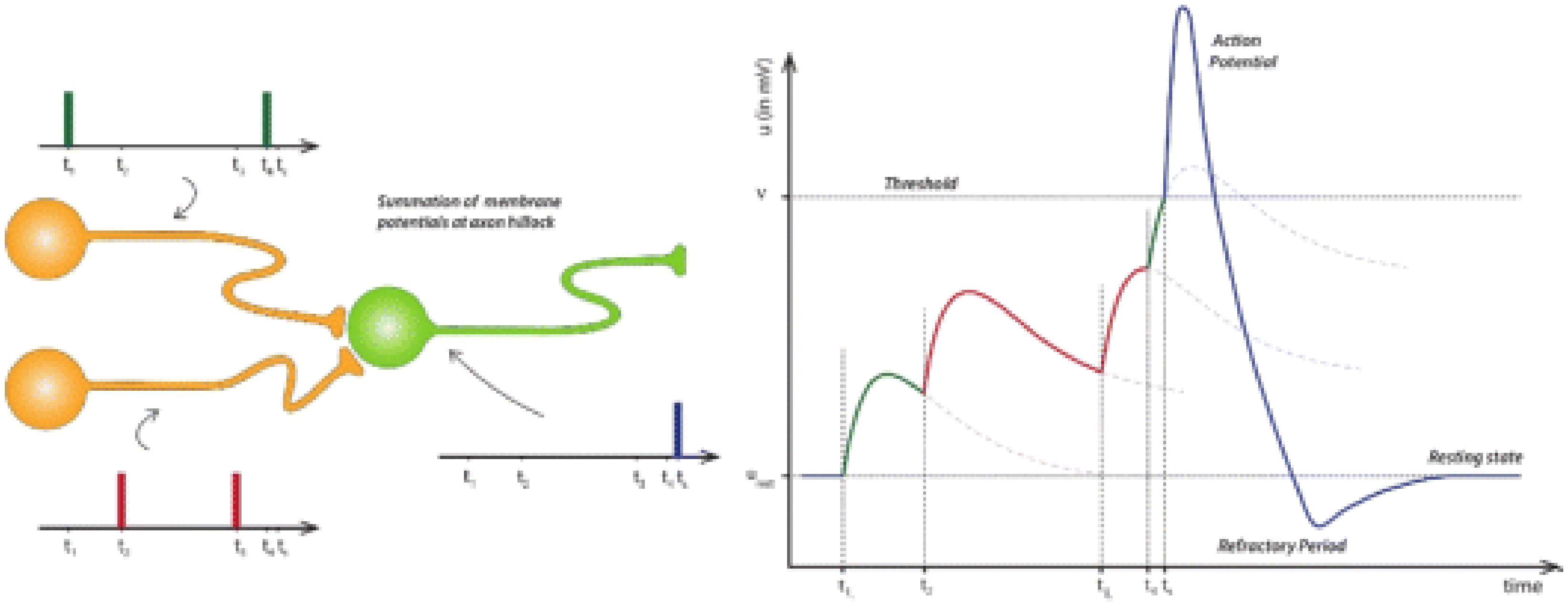
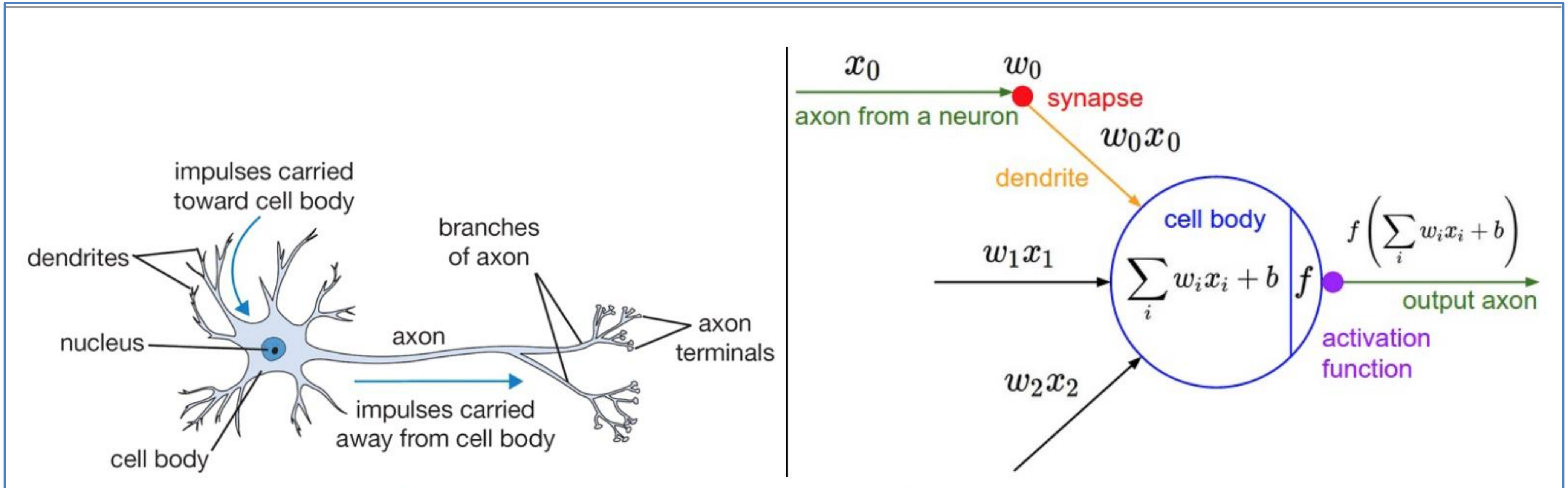


Figure from: Iakymchuk, T., et al. "Simplified spiking neural network architecture and STDP learning algorithm applied to image classification". In Journal of Image Video Proc. 2015, 4 (2015).

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Then, artificial neurons



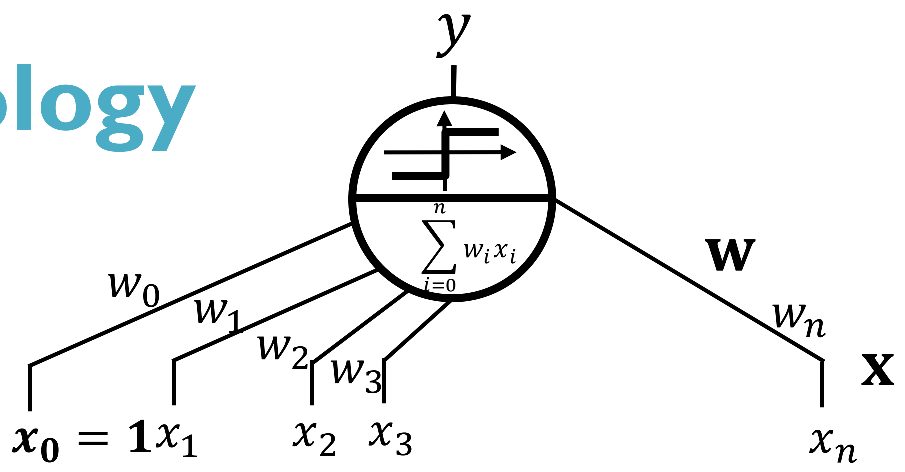
A cartoon drawing of a biological neuron (left) and its mathematical model (right).

Pitts & McCulloch (1943), binary inputs & activation function *f* thresholding

Rosenblatt (1956), real inputs & activation function *f* thresholding

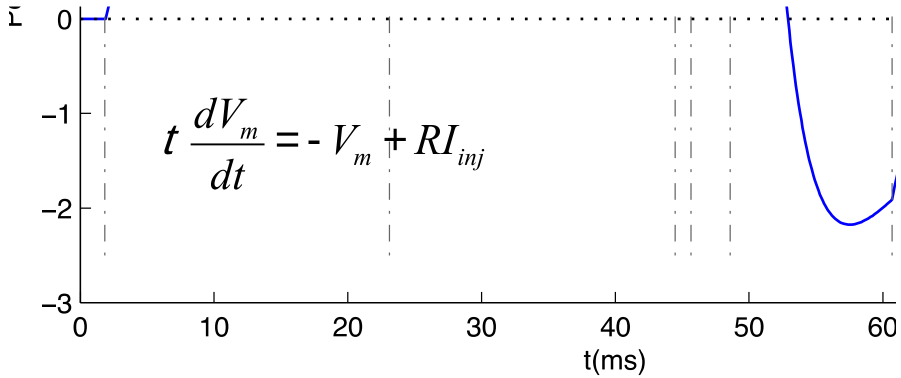
(Schéma : Isaac Changhau)

Artificial vs biology



mes

Spike-based description

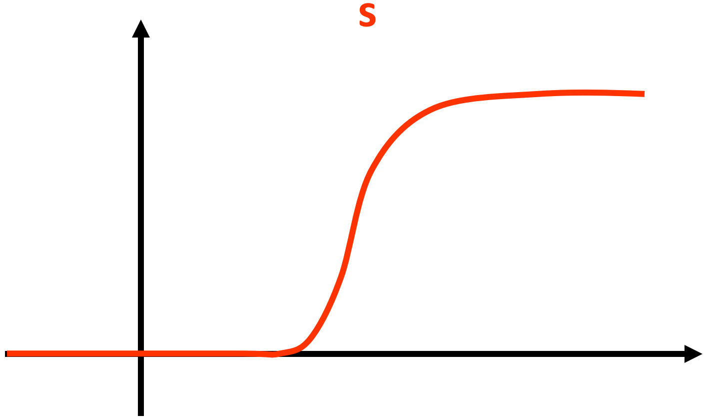


$$t\frac{dV_m}{dt} = -V_m + RI_{inj}$$

Gradient descent: KO

Rate-based description

Steady regime

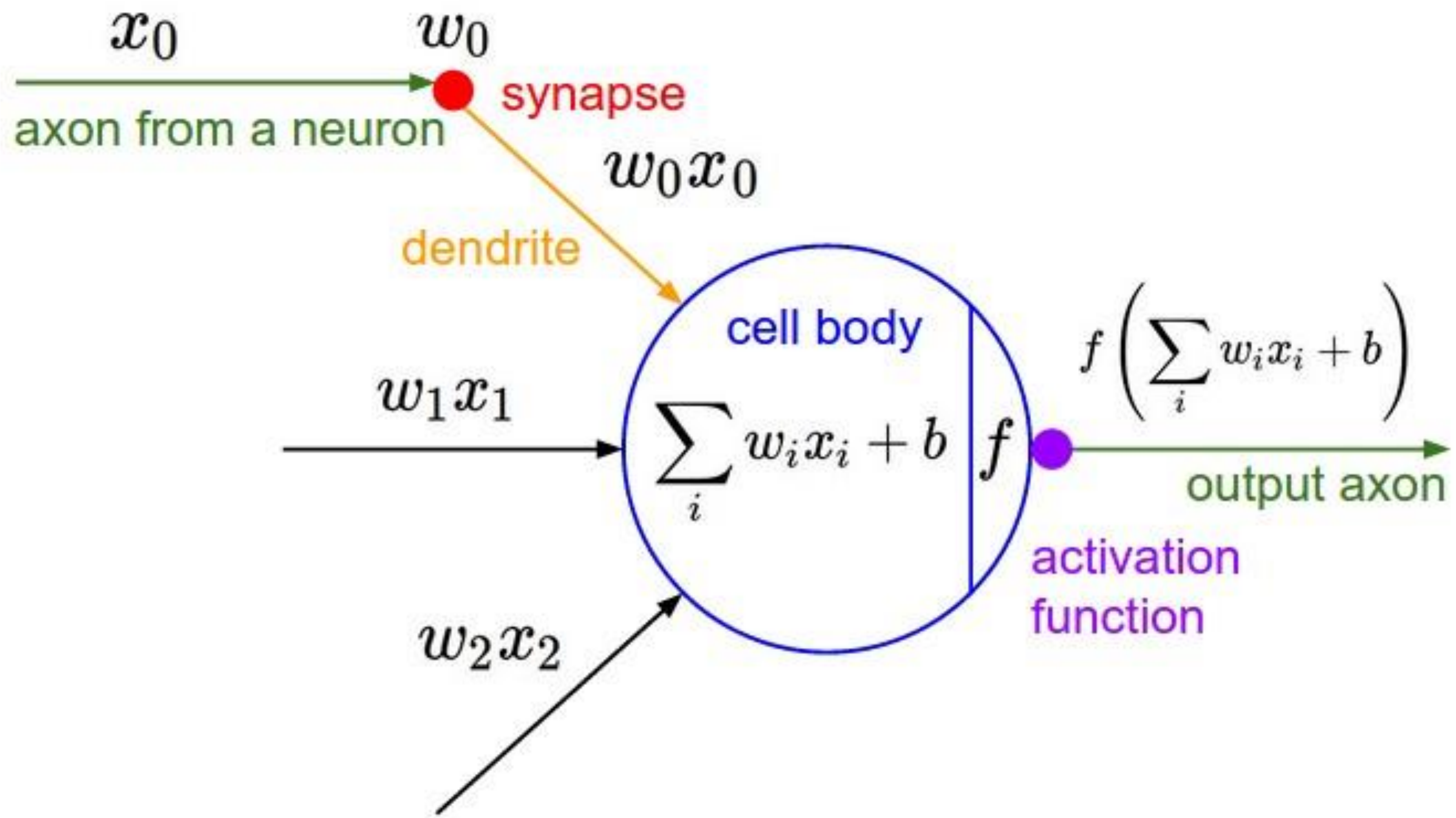


$$y = s(\Sigma\, w_i\, x_i)$$

Gradient descent: OK

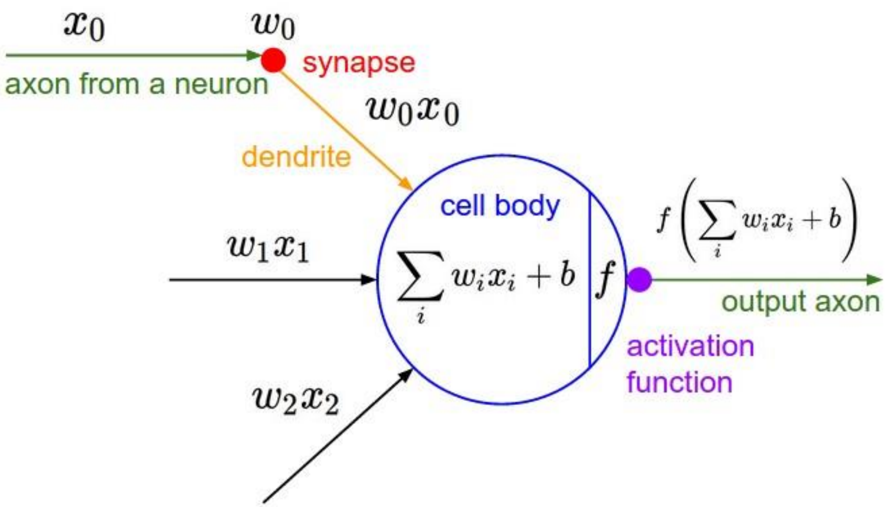
A single artificial neuron

- It is a **very simple abstract** of a biological neuron (McCulloch & Pitts, 1943)



A single artificial neuron

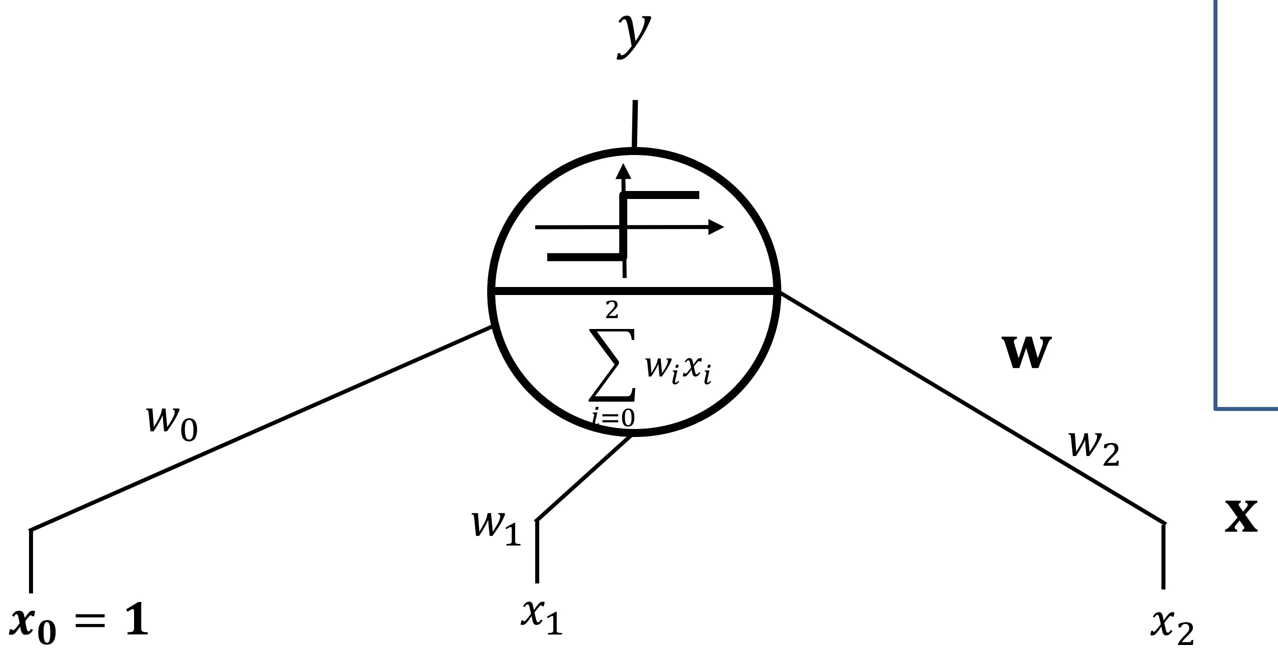
- Each input x has an associated **weight w** which can be modified
- Inputs x corresponds to signals from other neuron axons
  - x_0 - Bias are 'special' inputs, with weight w_0

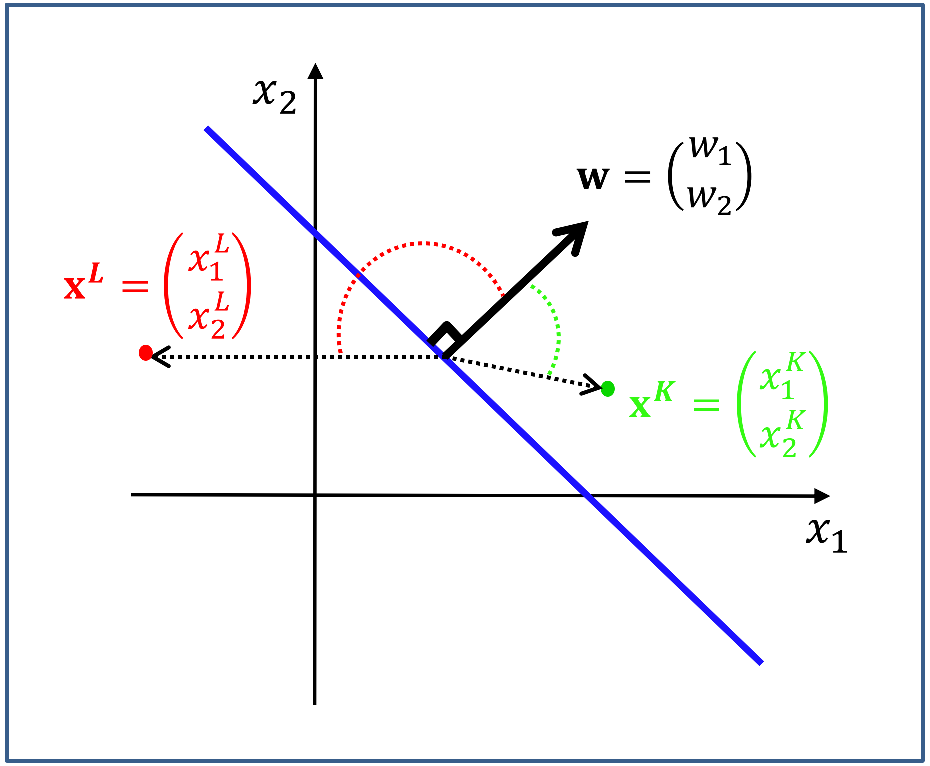


- Weights W corresponds to synaptic modulation (i.e. something like strength/amount of neurotransmitters)
- The summation corresponds to 'cell body'
- The activation function corresponds to axon hillock - computes some function f of the weighted sum of its inputs
- So, output $y=f(z)$, corresponds to axon signal

UNIVERSITÉ CÔTE D'AZUR
univ-cotedazur.fr

Artificial neuron = a linear classifier

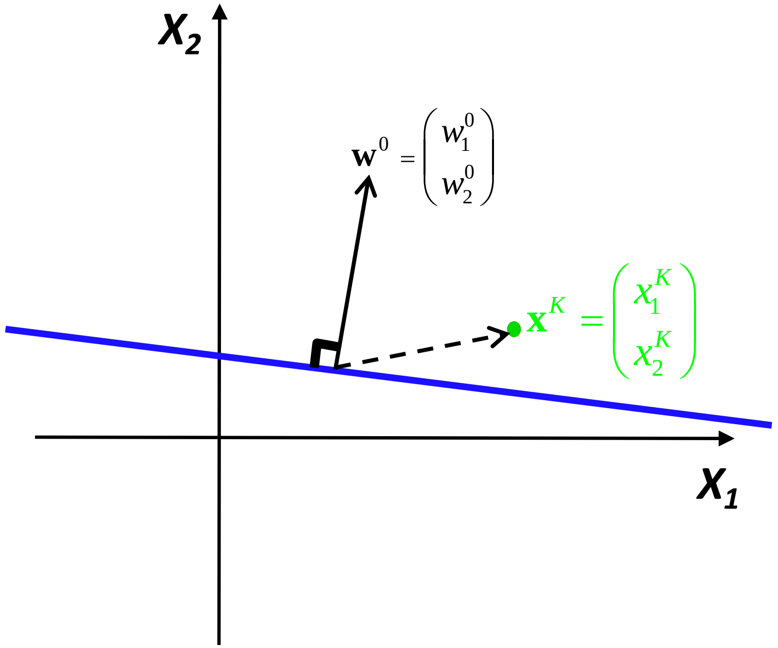


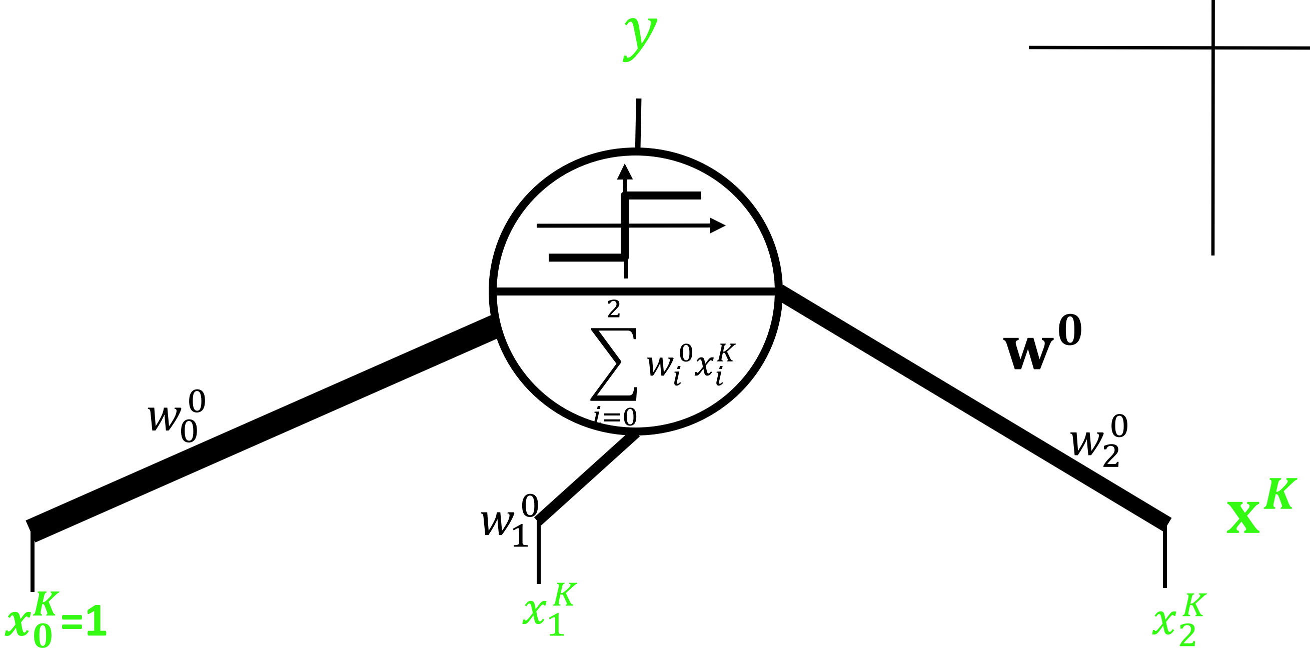


univ-cotedazur.fr

One artificial neuron (perceptron)

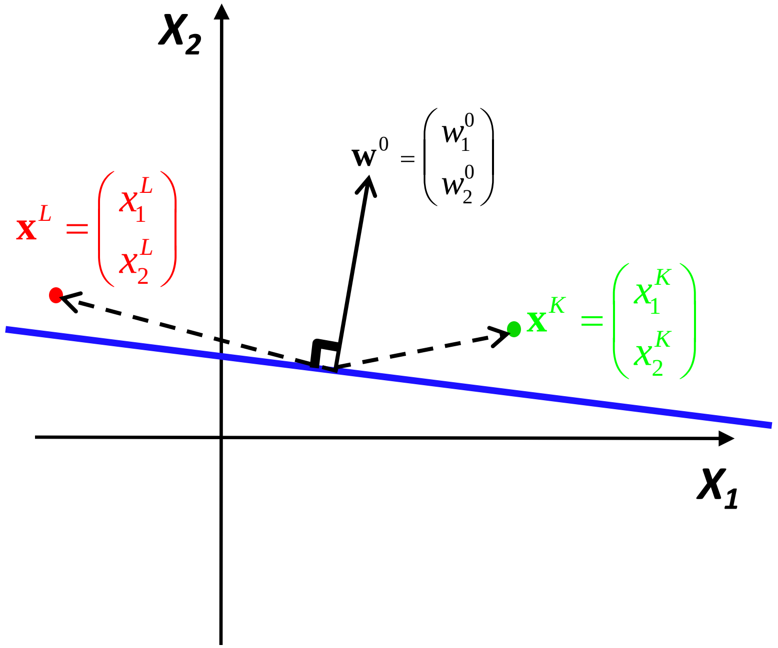
At training you want to set the weights,
so that your training samples are correctly classified:

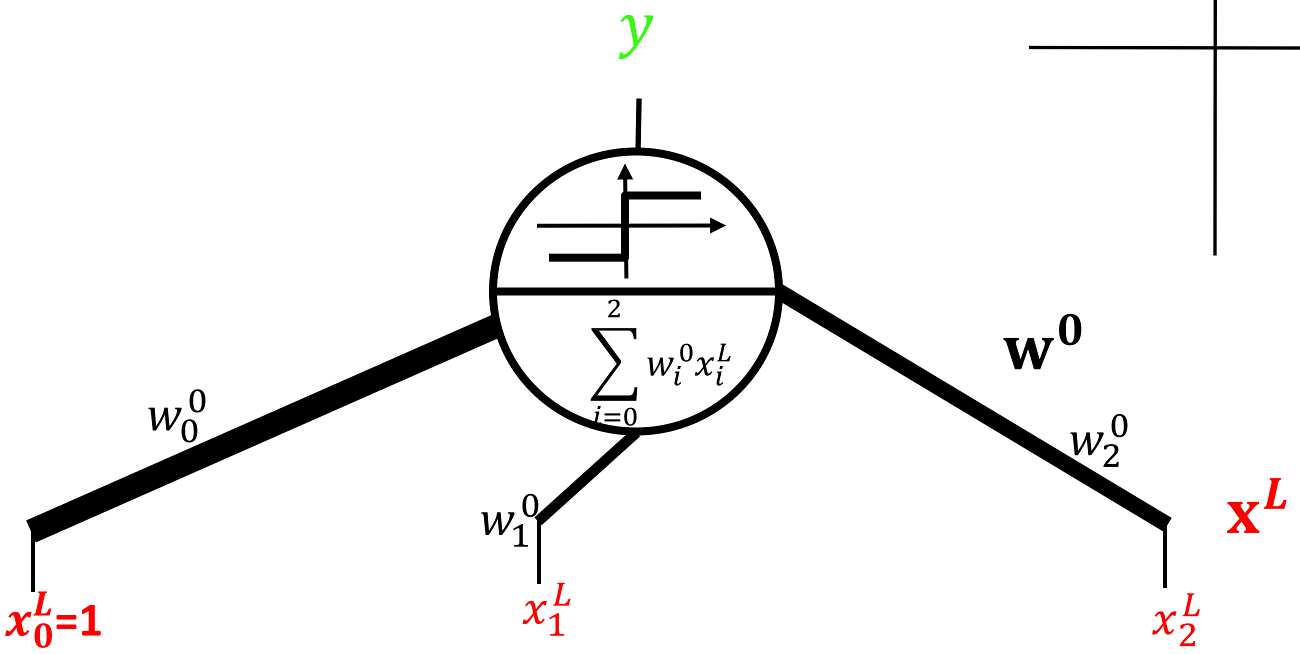




One artificial neuron (perceptron)

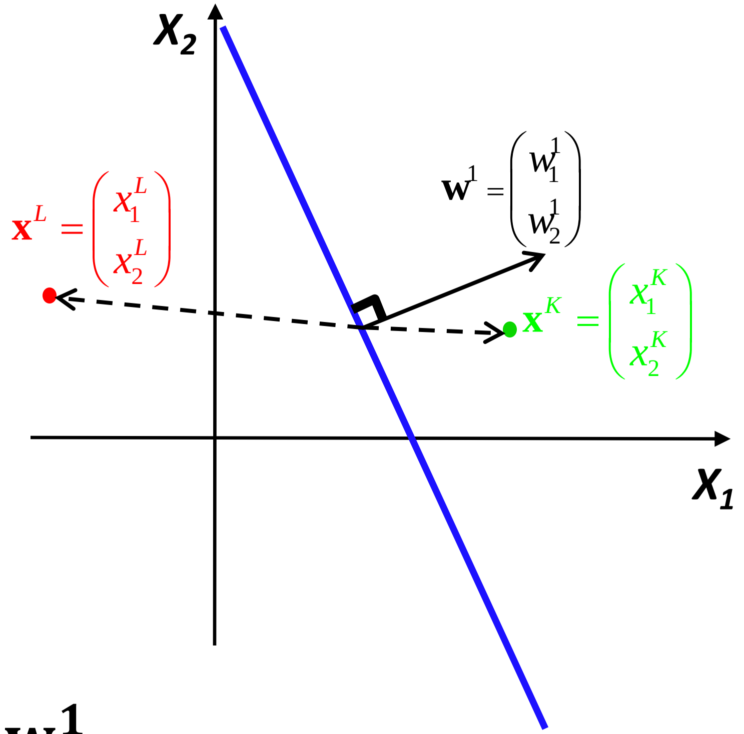
At training you want to set the weights,
so that your training samples are correctly classified:

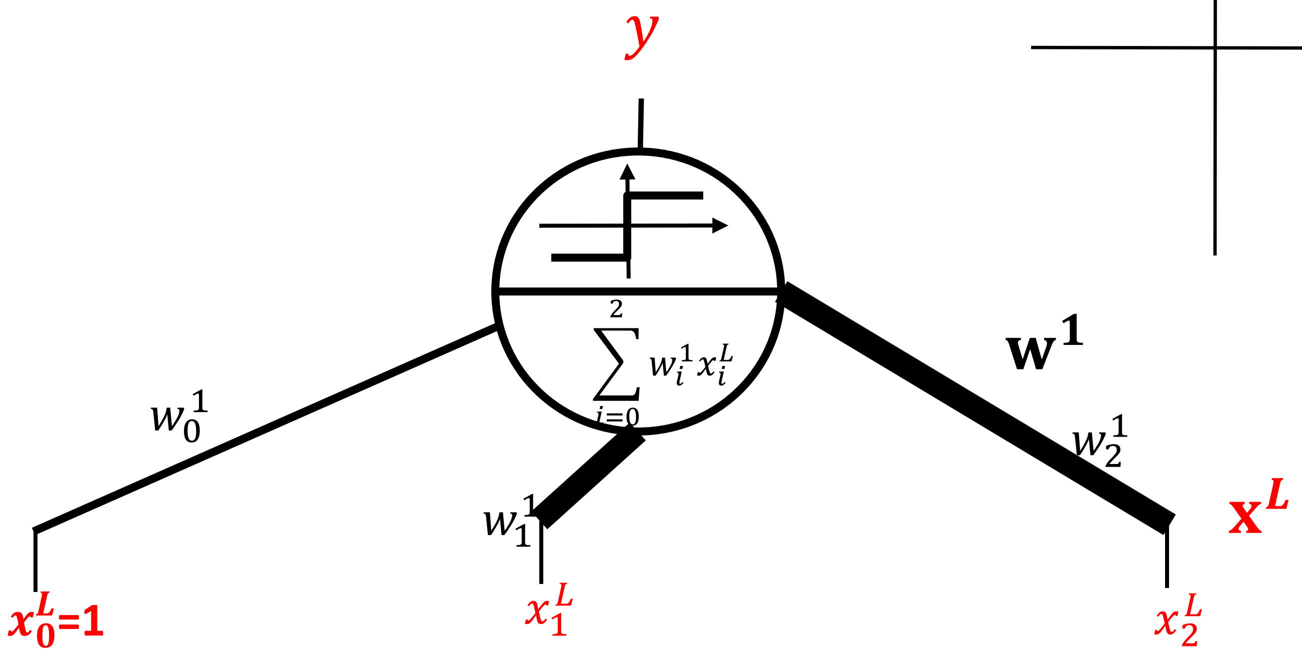




One artificial neuron (perceptron)

At training you want to set the weights,
so that your training samples are correctly classified:



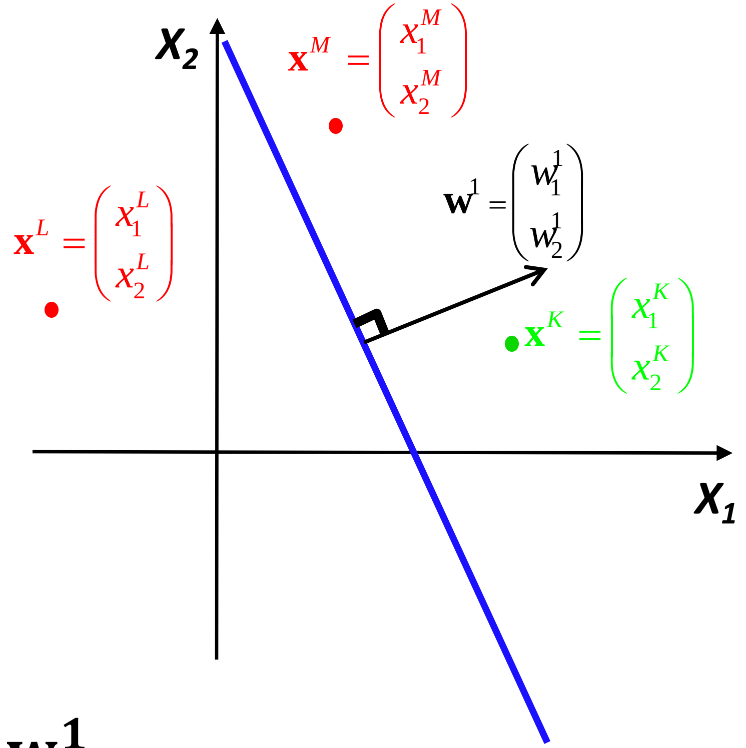


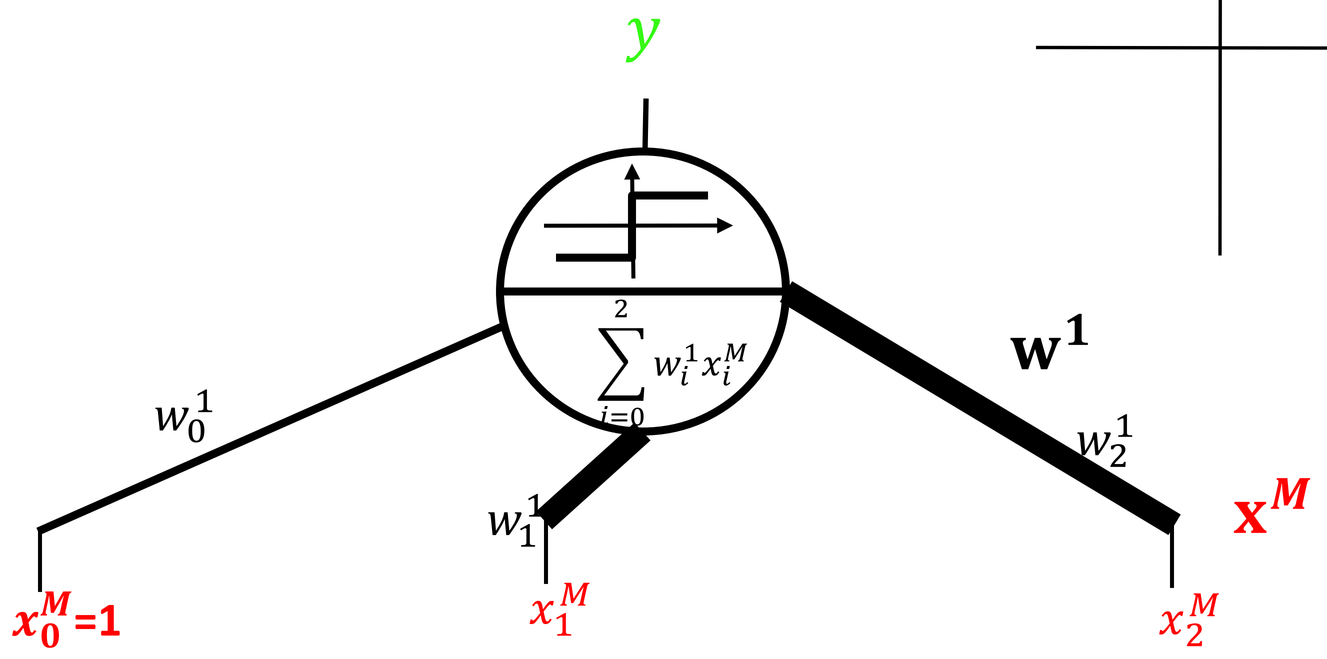
UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

One artificial neuron (perceptron)

At training you want to set the weights,
so that your training samples are correctly classified:



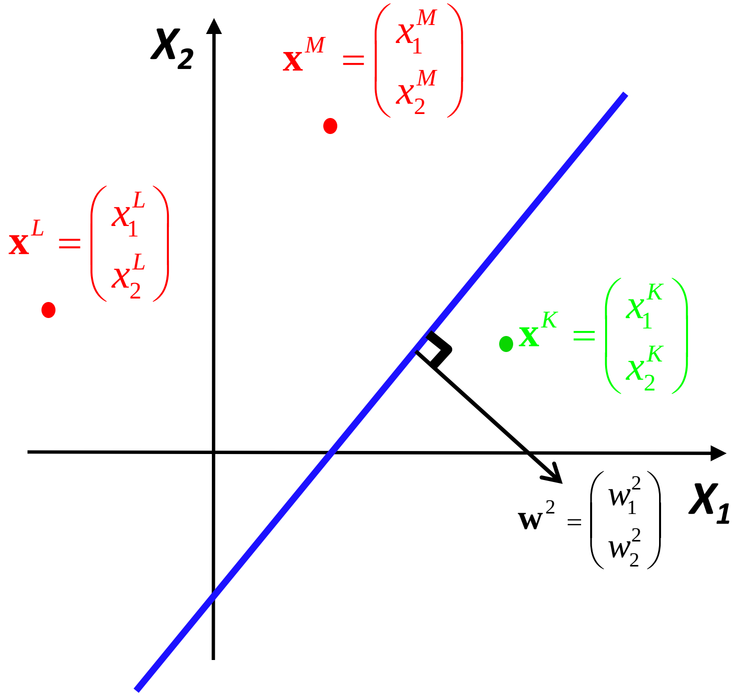


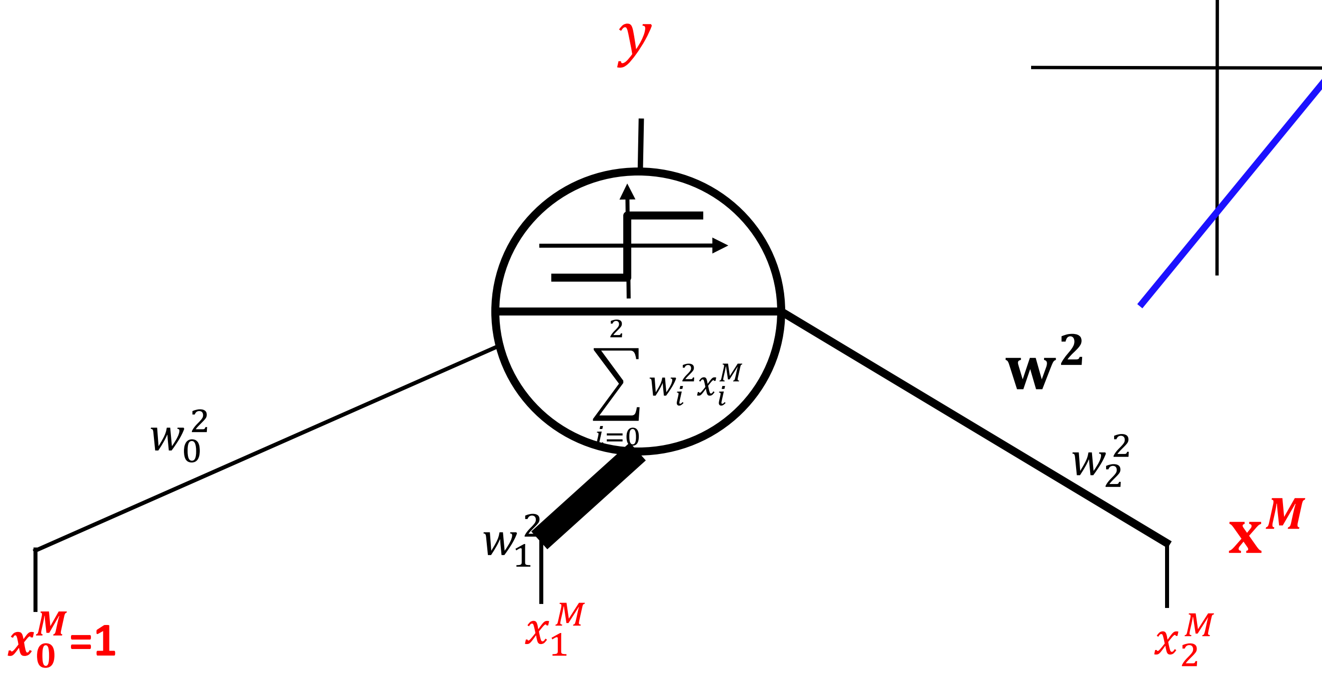
UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

One artificial neuron (perceptron)

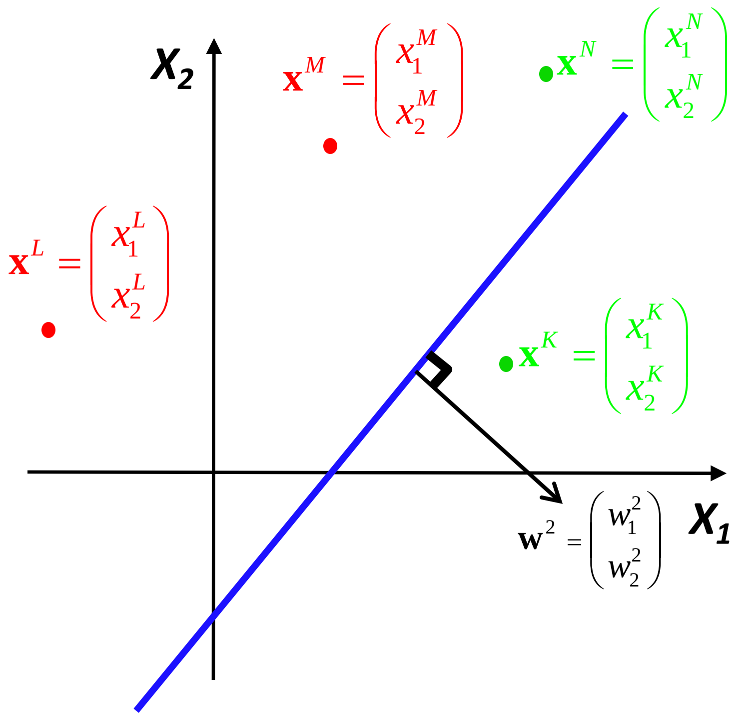
At training you want to set the weights,
so that your training samples are correctly classified:

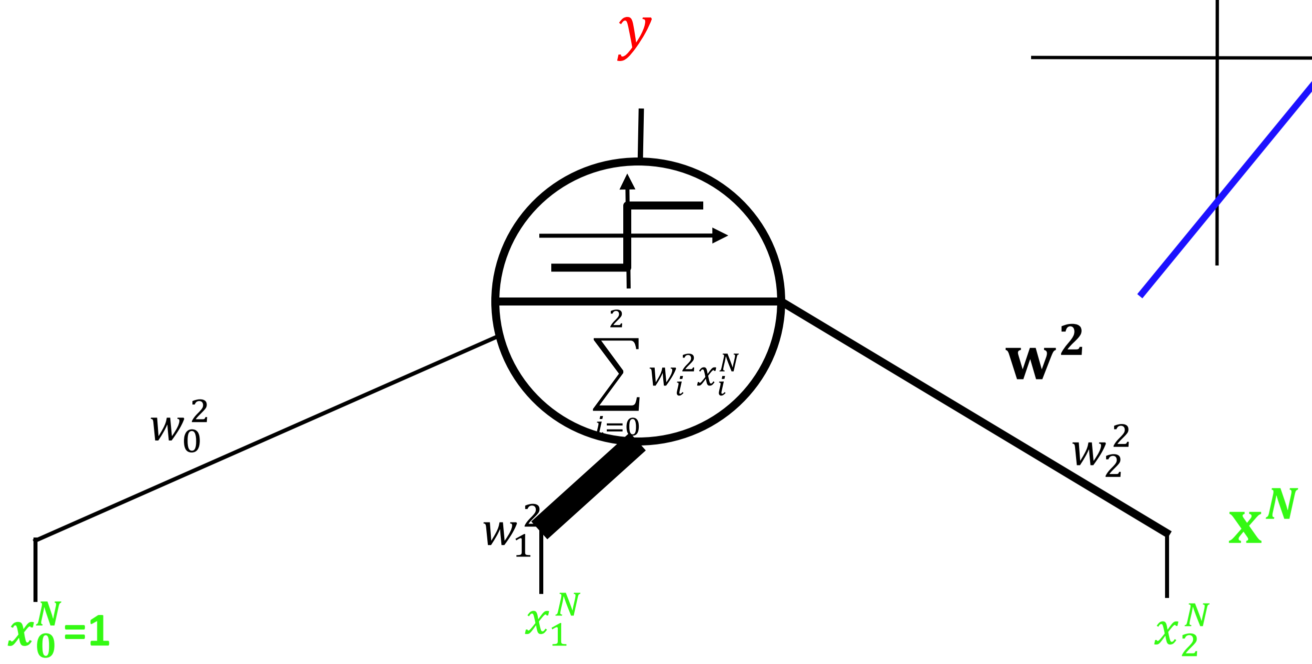




One artificial neuron (perceptron)

At training you want to set the weights,
so that your training samples are correctly classified:



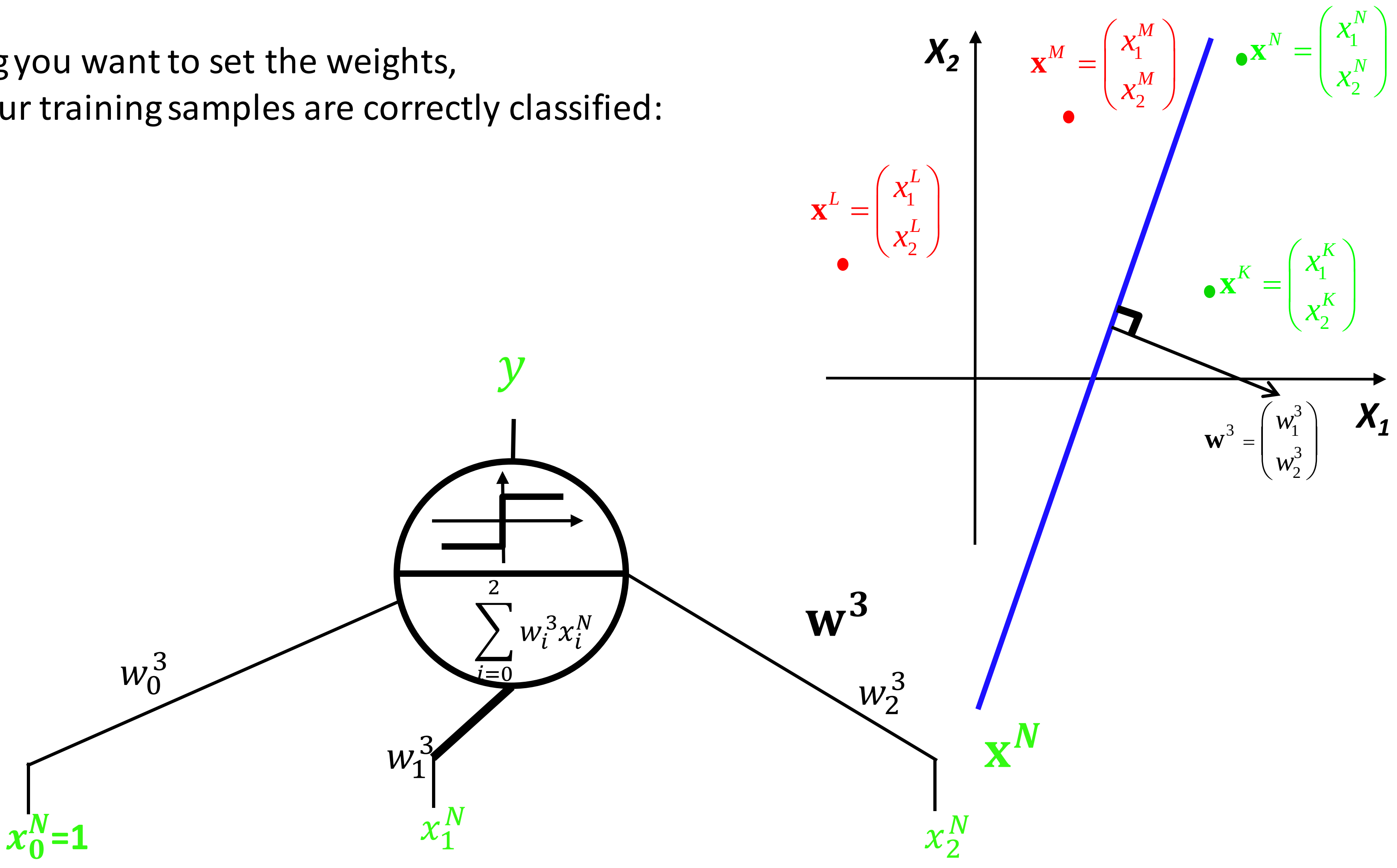


UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

One artificial neuron (perceptron)

At training you want to set the weights,
so that your training samples are correctly classified:

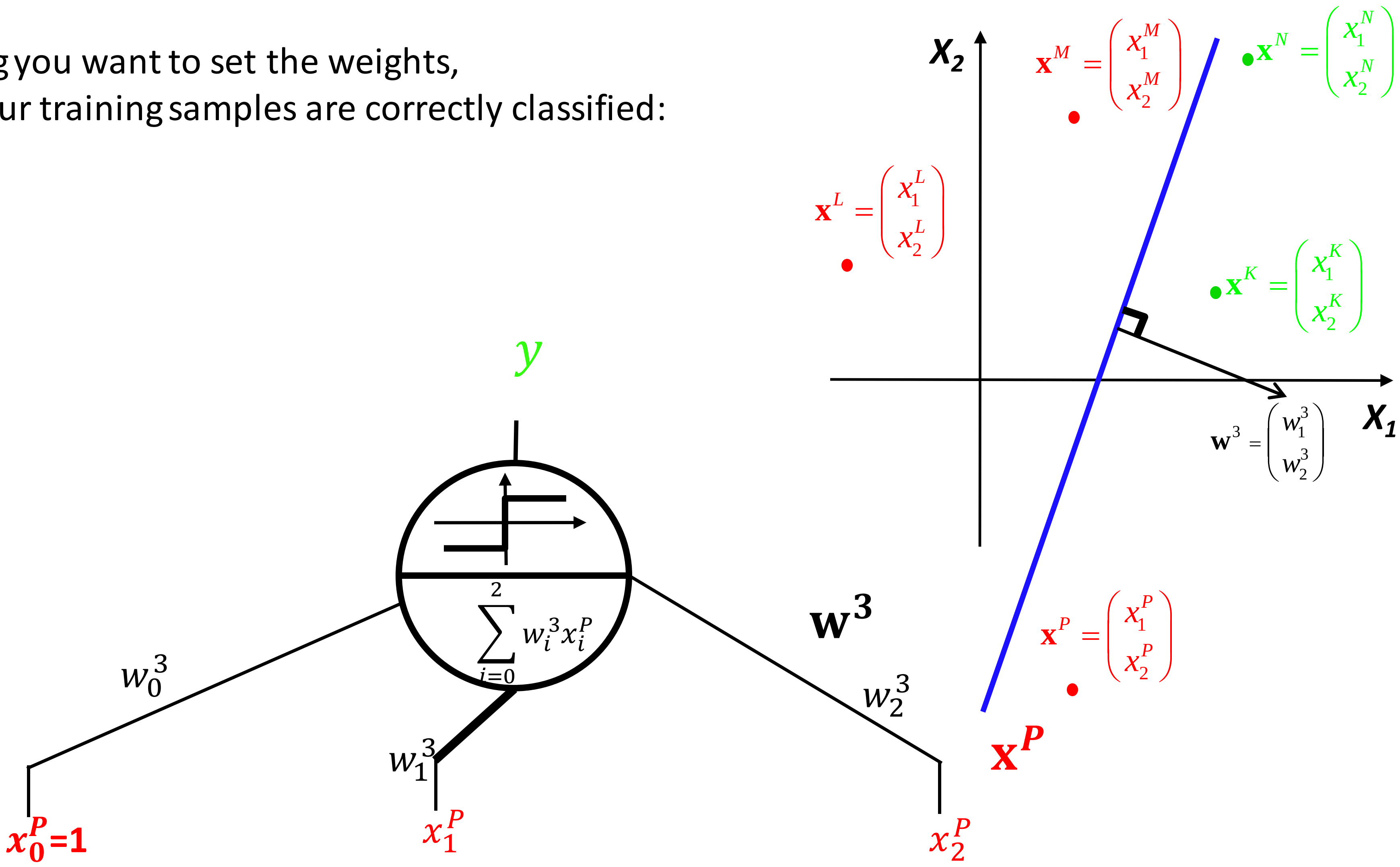


UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

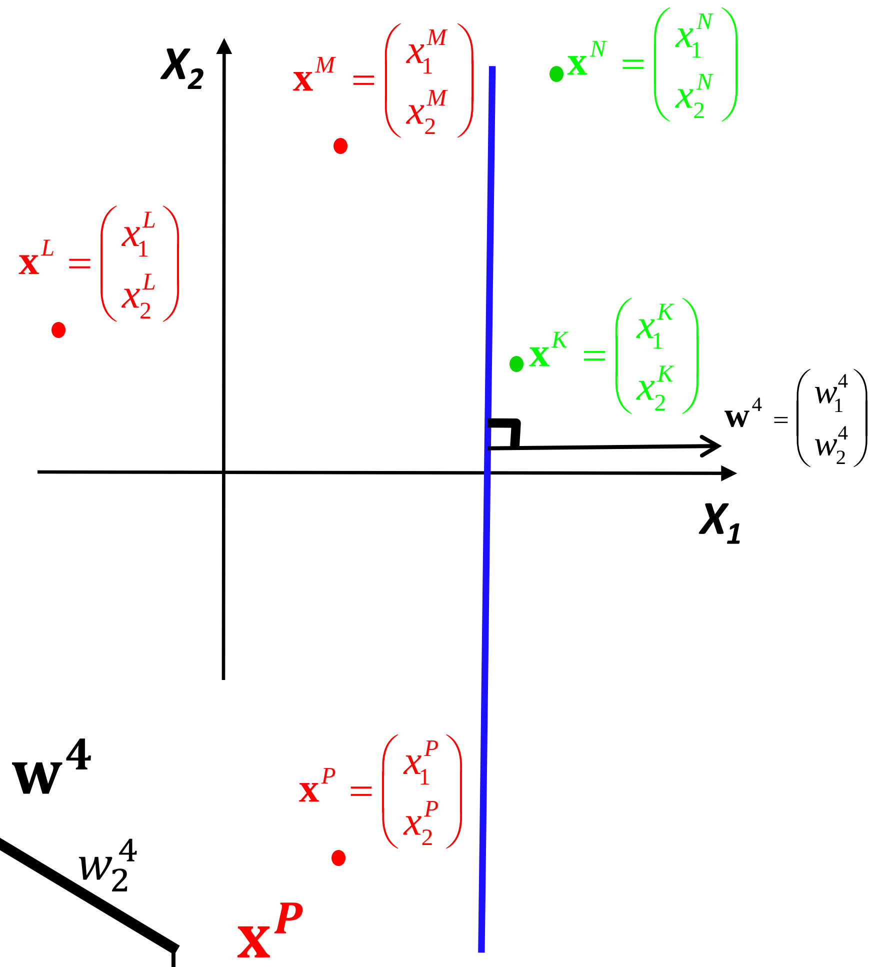
One artificial neuron (perceptron)

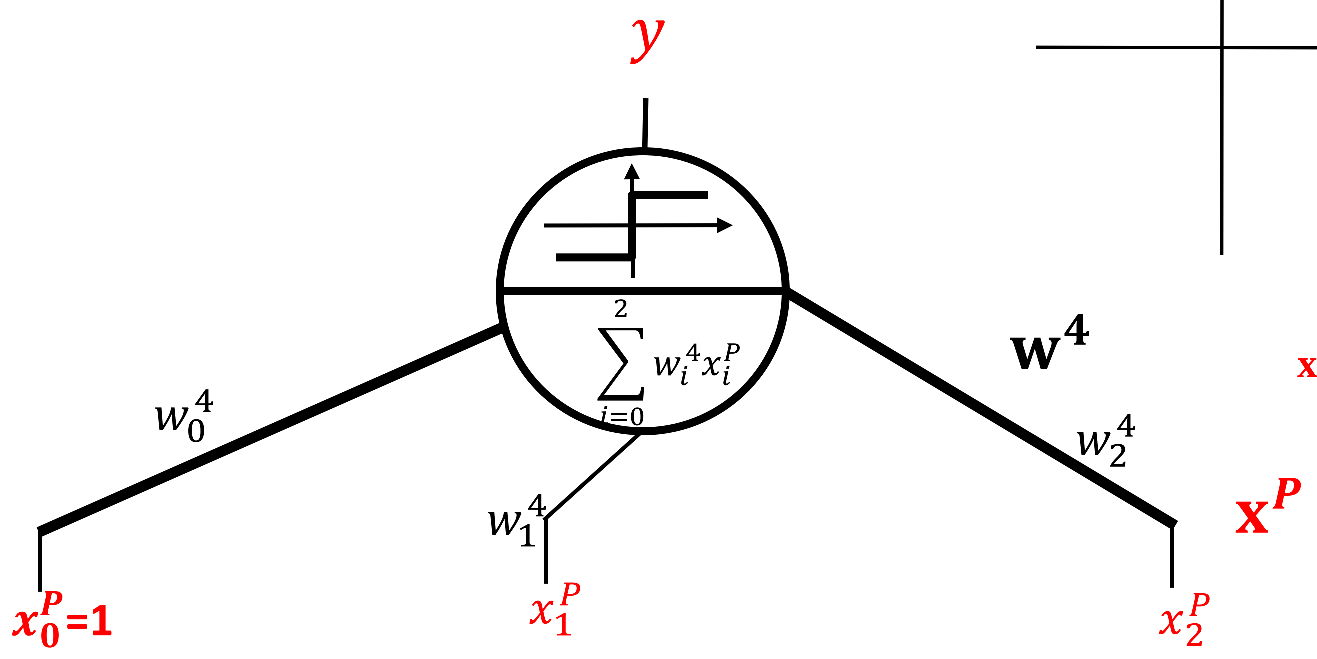
At training you want to set the weights,
so that your training samples are correctly classified:



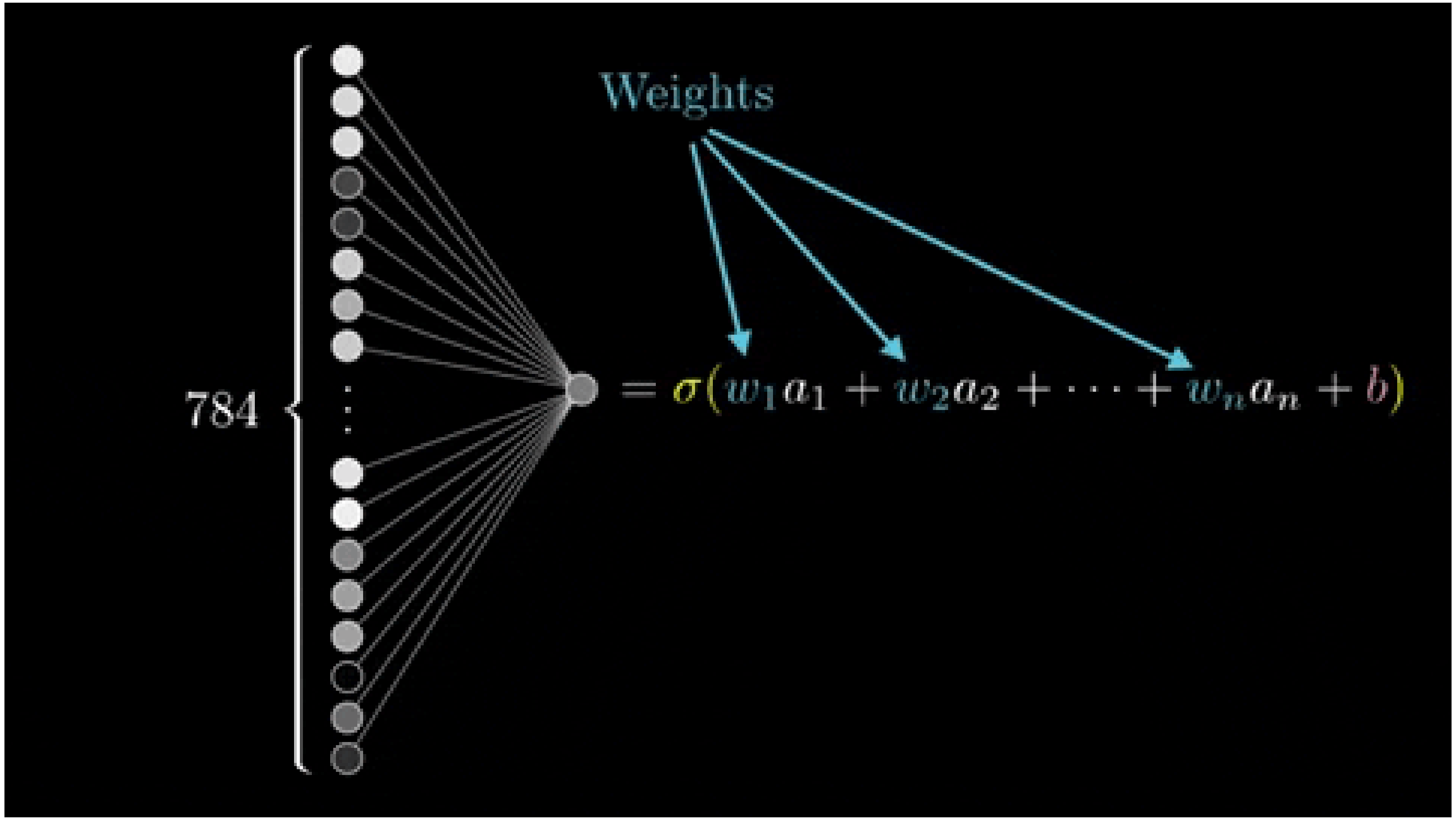
One artificial neuron (perceptron)

At training you want to set the weights,
so that your training samples are correctly classified:





One neuron = simple linear decision



Source: https://www.3blue1brown.com/neural-networks

UNIVERSITÉ CÔTE D'AZUR
univ-cotedazur.fr

Perceptron: Rosenblatt's Algorithm (1956-1958)

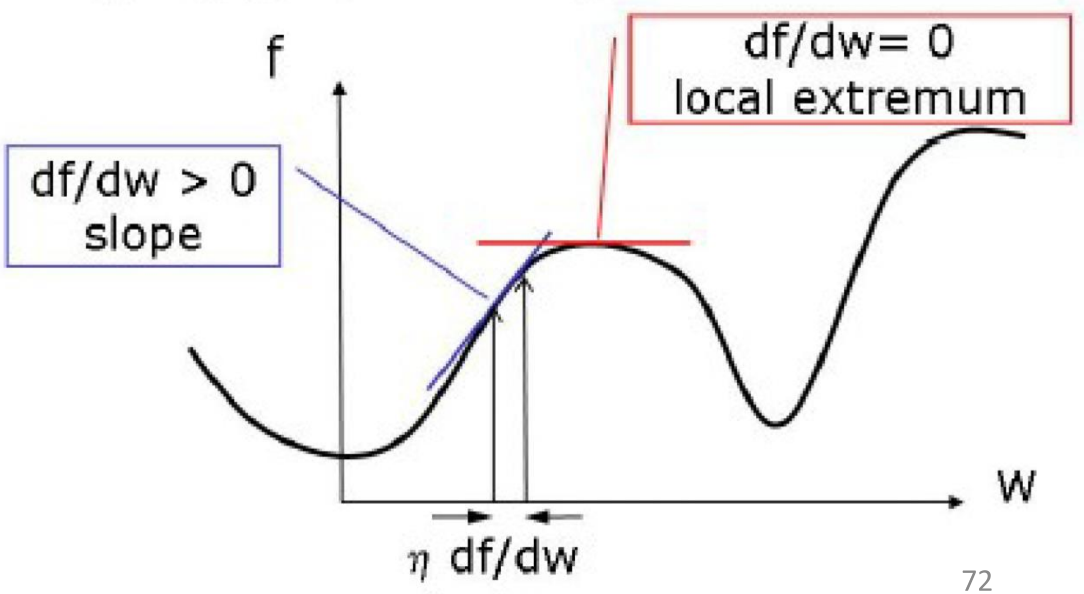
Perceptron Algorithm

- Pick initial weight vector (including w_0), e.g. (0, 0,…,0)
- Repeat until all points are correctly classified
  - *Repeat for each point*
    - Calculate $y^i\mathbf{w}\mathbf{x}^\mathrm{i}$ for point i
    - If $y^i\mathbf{w}\mathbf{x}^\mathrm{i} > 0$, the point is correctly classified
    - Else change the weights to increase the value of $y^i\mathbf{w}\mathbf{x}^\mathrm{i}$; change in weight proportional to $y^i\mathbf{x}^\mathrm{i}$

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review, 65(6):386.

Gradient Ascent

- Why pick $y^i\mathbf{x}^i$ as increment to weights?
- To maximize scalar function of one variable $f(\mathbf{w})$
  - Pick initial $\mathbf{w}$
  - Change $\mathbf{w}$ to $\mathbf{w} + \eta\, \mathrm{d}f/\mathrm{d}\mathbf{w}$ ($\eta > 0$, small)
  - Until f stops changing ($\mathrm{d}f/\mathrm{d}\mathbf{w} \approx 0$)



Gradient Ascent

- To maximize a multivariate function $f(\mathbf{w})$
  - Pick initial $\mathbf{w}$
  - Change $\mathbf{w}$ to $\mathbf{w} + \eta \nabla f_{\mathbf{w}}$ ($\eta > 0$, small)
  - Until f stops changing ($\nabla f_{\mathbf{w}} \approx 0$)
- Find local maximum, unless function is globally convex

$$\nabla f_{\mathbf{w}} = \left[\frac{\partial f}{\partial \mathbf{w}_1}, \quad \dots \quad , \frac{\partial f}{\partial \mathbf{w}_n}\right]$$

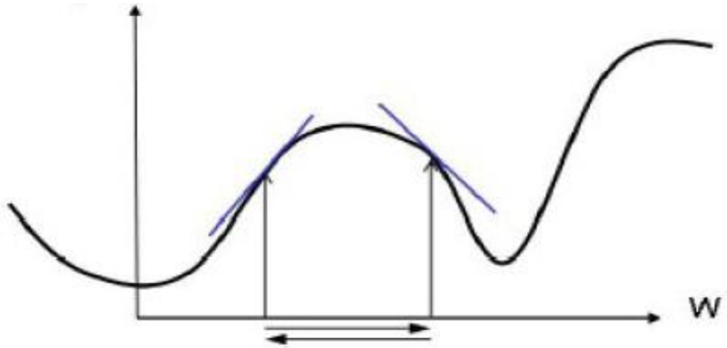
univ-cotedazur.fr

Gradient Ascent

- To maximize a multivariate function $f(\mathbf{w})$
  - Pick initial $\mathbf{w}$
  - Change $\mathbf{w}$ to $\mathbf{w} + \eta \nabla f_{\mathbf{w}}$ ($\eta > 0$, small)
  - Until f stops changing ($\nabla f_{\mathbf{w}} \approx 0$)

$$\nabla f_{\mathbf{w}} = \left[\frac{\partial f}{\partial \mathbf{w}_1}, \quad \dots \quad, \frac{\partial f}{\partial \mathbf{w}_n}\right]$$

- Find local maximum, unless function is globally convex
- If f is non-linear, the learning rate η has to be chosen very carefully
  - Too small $\Rightarrow$ slow convergence
  - Too big $\Rightarrow$ osccilations



Gradient Ascent

- Maximize margin of misclassified points

$$f(\mathbf{w}) = \sum_{on\ \mathbf{i}\ misclassified\ points} y^i \mathbf{w} \mathbf{x}^{\mathrm{i}}$$

$$\nabla_{\mathbf{w}} f(\mathbf{w}) = \sum_{on\ \mathbf{i}\ misclassified\ points} y^i \mathbf{x}^{\mathrm{i}}$$

- Off-line training: Compute, at each iteration, the gradient as sum over all training points
- On-line training: Approximate gradient by one of the terms in the sum: $y^i \mathbf{x}^{\mathrm{i}}$
  (principle of the *Stochastic Gradient Descent*, SGD)

Perceptron Algorithm

- Each change of **w** decreases the error on a specific point. However, changes for several points are correlated, that is different points could change the weights in opposite directions. Thus, this iterative algorithm requires several loops to converge.

- Guarantee to find a separating hyperplane if one exists – if data is linearly separable.

- If data are not linearly separable, then this algorithm loops indefinitely.

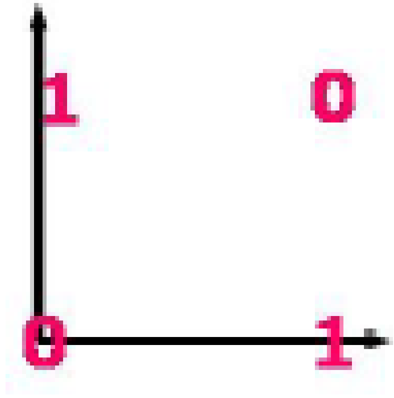
Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review, 65(6):386.

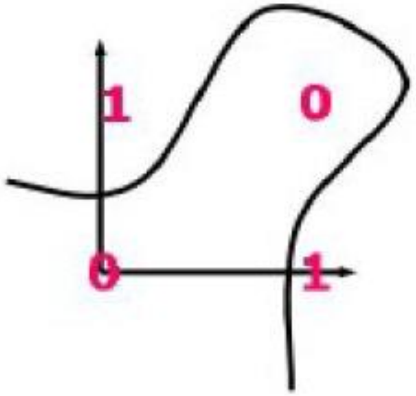
UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Beyond Linear Separability

- Values of the XOR boolean function cannot be separated by a single perceptron unit [Minsky and Papert, 1969].





Minsky, M. and Papert, S. (1969). Perceptrons: An Introduction to Computational Geometry. MIT Press.

Overview

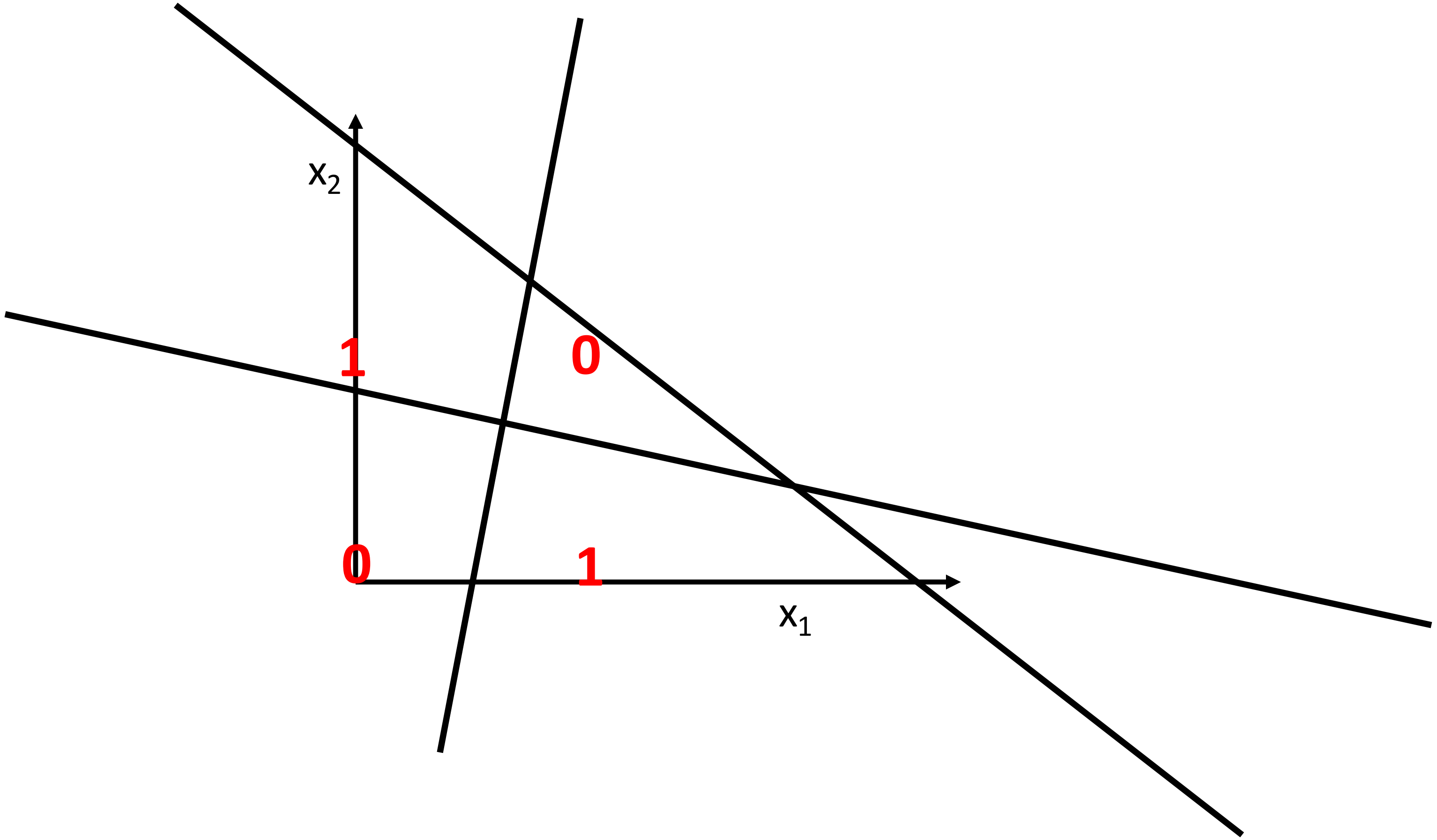
- Machine Learning vs Statistics
- Math Basics
- Simple Model
- **From Simple to Complex**

FROM SIMPLE TO COMPLEX

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

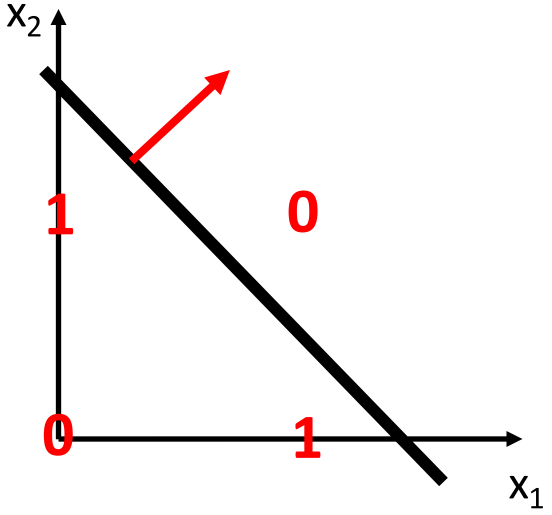
Problem which cannot be solved with a unique straight line

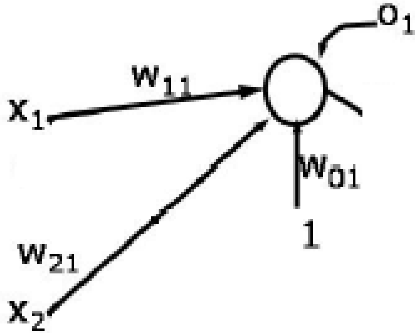


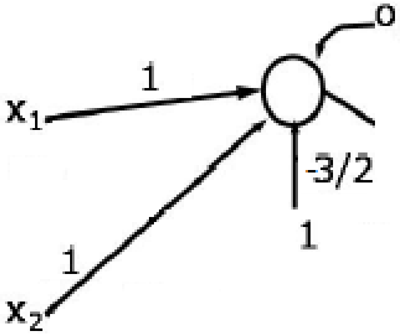
w_{01} =-3/2 w_{11} = w_{21} = 1

x_1	x_2	o_1
0	0	0
0	1	0
1	0	0
1	1	1

Problem which cannot be solved with a unique straight line



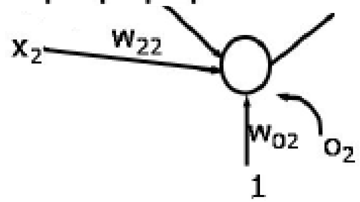


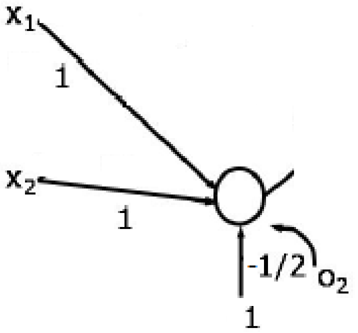


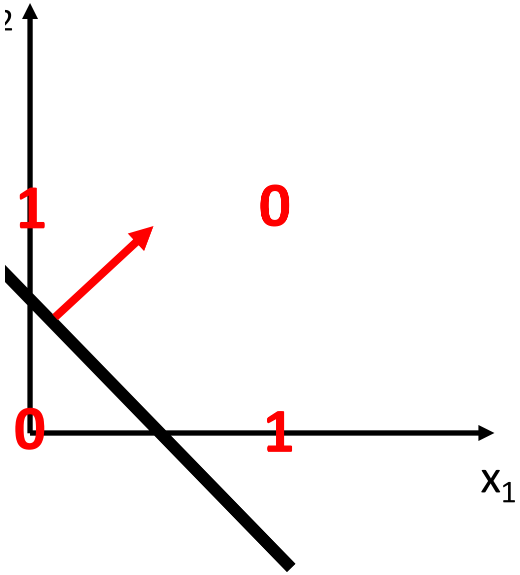
0
0
0
1

Problem which cannot be solved with a unique straight line

x_1	x_2	o_1	o_2
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1



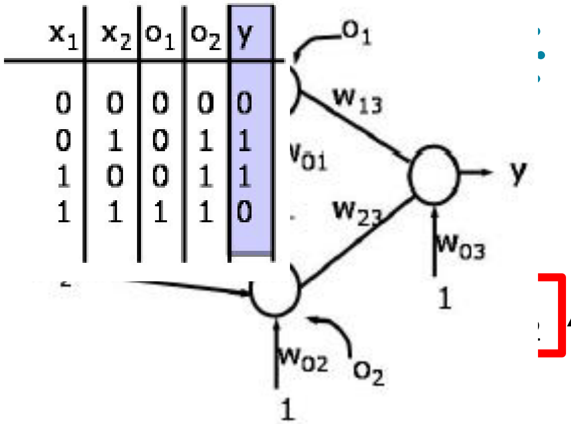


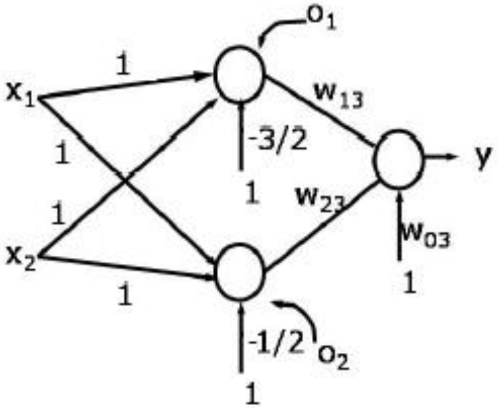


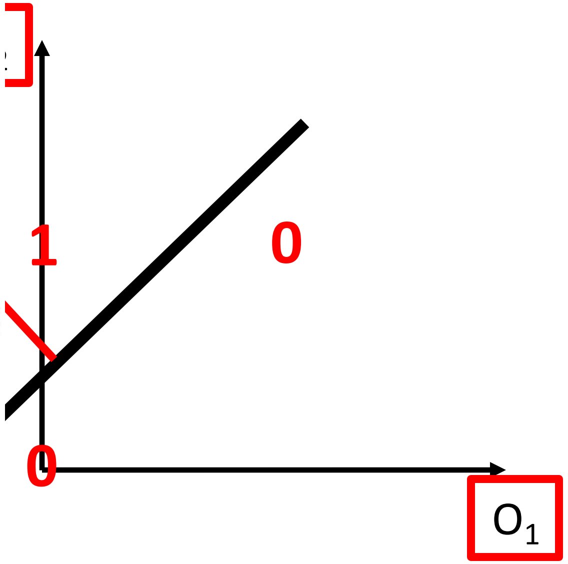
0	0
0	1
0	1
1	1

$w_{02} = -1/2 \quad w_{12} = w_{22} = 1$

Problem which cannot be solved with a unique straight line





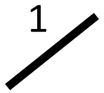


O_1

O_2

O_1

1



-1/2

O_2-O_1-1/2=0

0 0
0 1
0 1
1 1

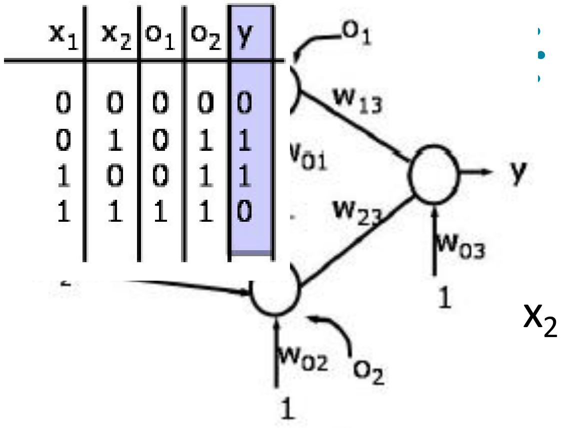
O_2

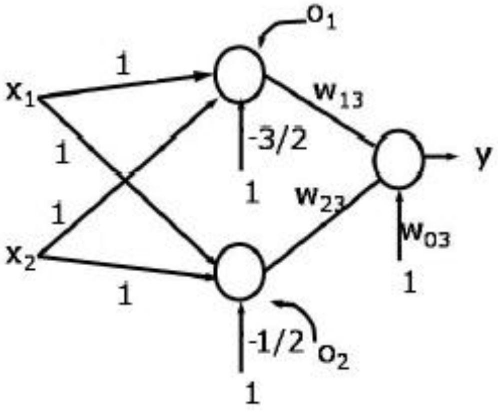
$w_{23}o_2+w_{13}o_1+w_{03}=0$

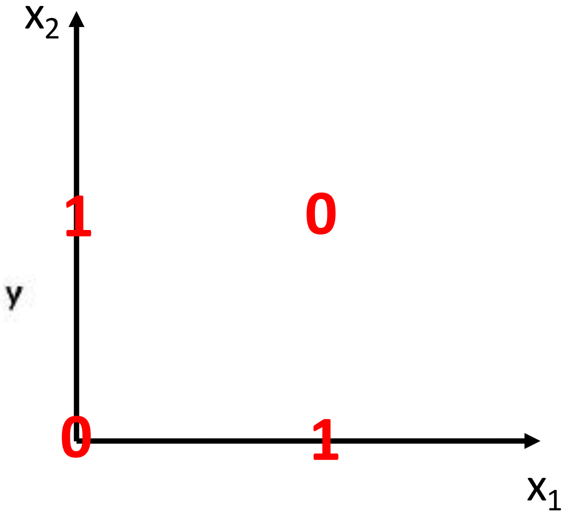
$w_{03}=-1/2$, $w_{13}=-1$, $w_{23}=1$

Problem which cannot be solved with a unique straight line

x_1	x_2	o_1	o_2	y
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0







$\boxed{O_1}$

$\boxed{O_2}$

$\boxed{O_1}$

1

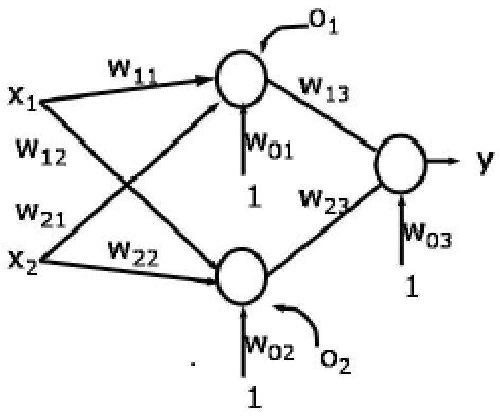
1

-1/2

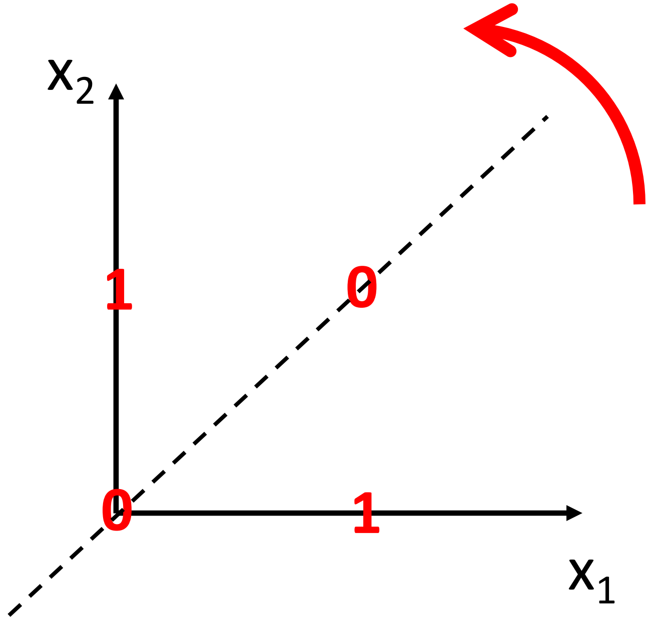
0 0
0 1
0 1
1 1

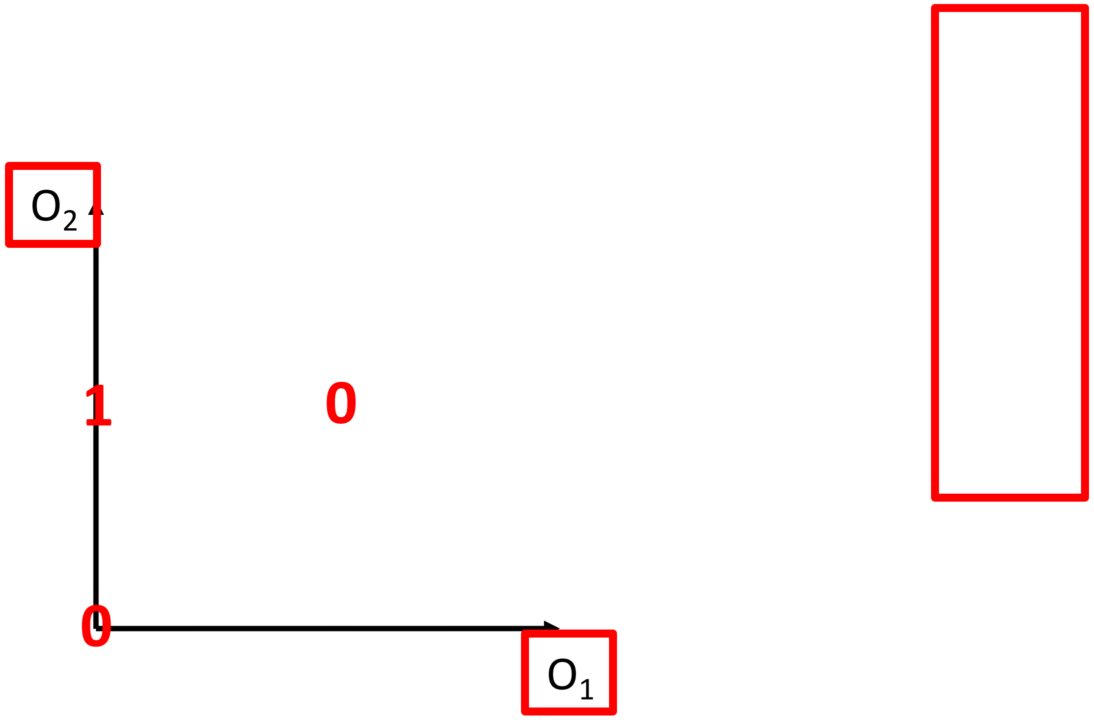
$\boxed{O_2}$

$w_{23}o_2+w_{13}o_1+w_{03}=0$
$w_{03}=-1/2,\ w_{13}=-1,\ w_{23}=1$



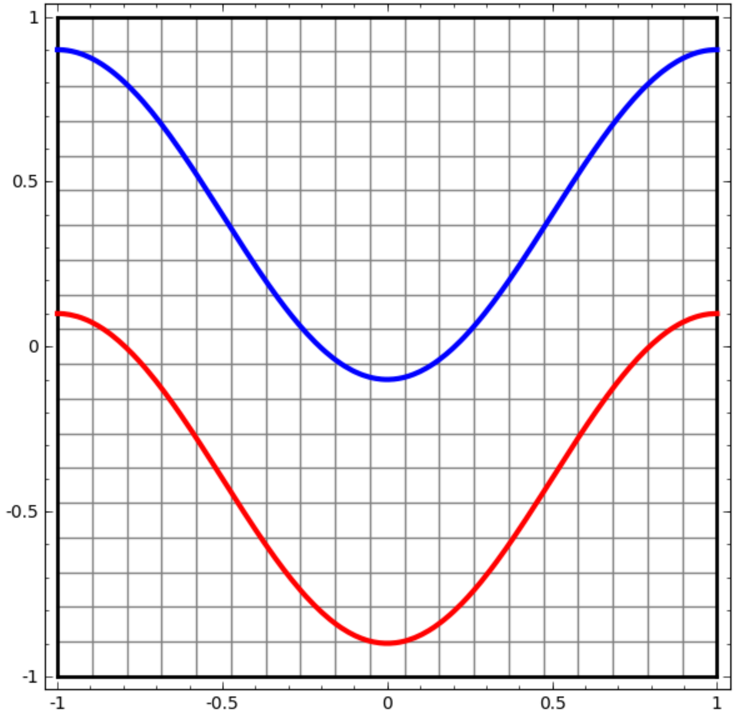
Problem which cannot be solved with a unique straight line

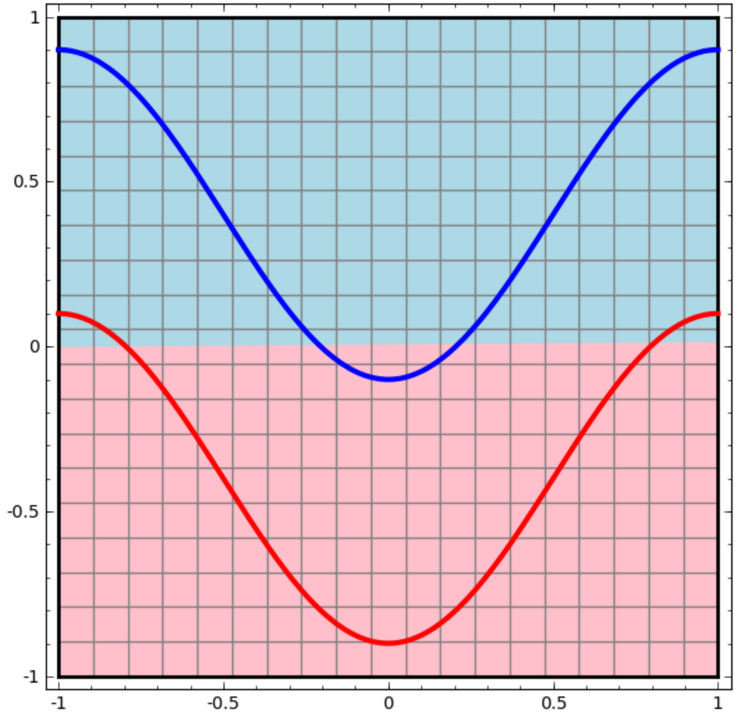




Adding a hidden layer of neurons has fold the input space

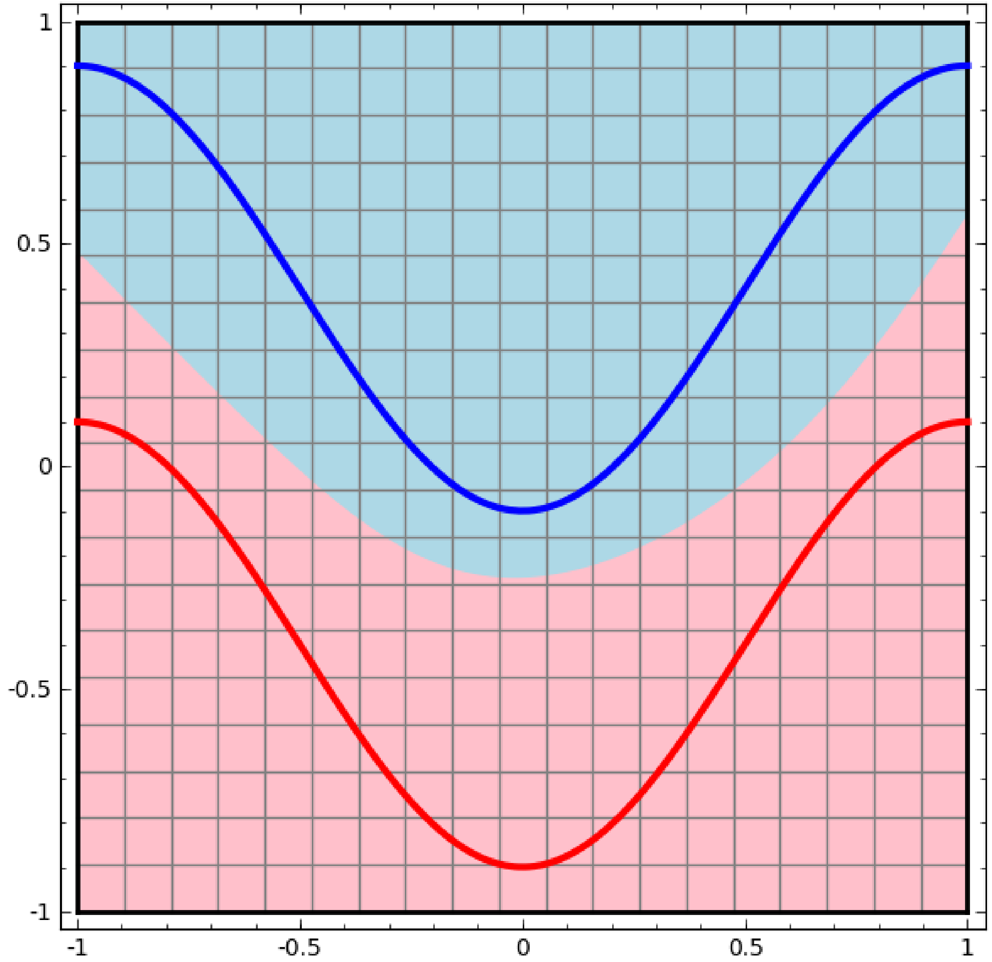
One perceptron





Illustrations from: http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

Multi-Layer Perceptron, manifold disentanglement

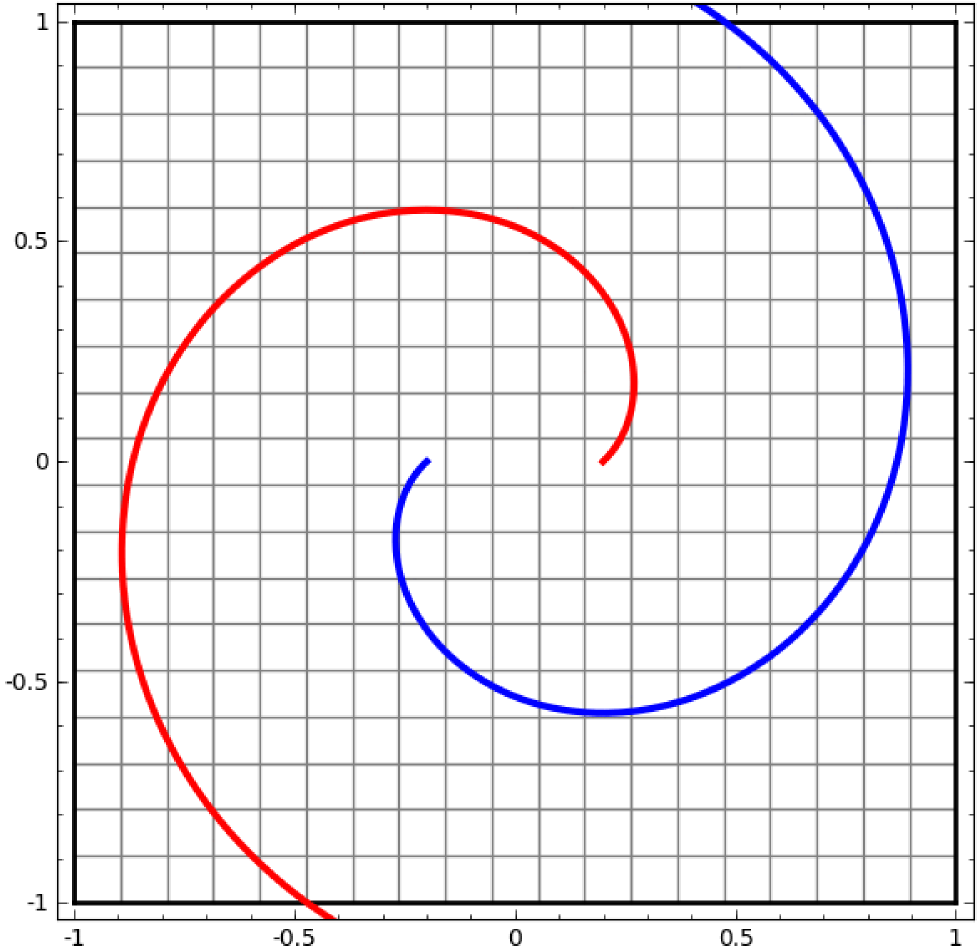


Illustrations from: http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

UNIVERSITÉ CÔTE D'AZUR

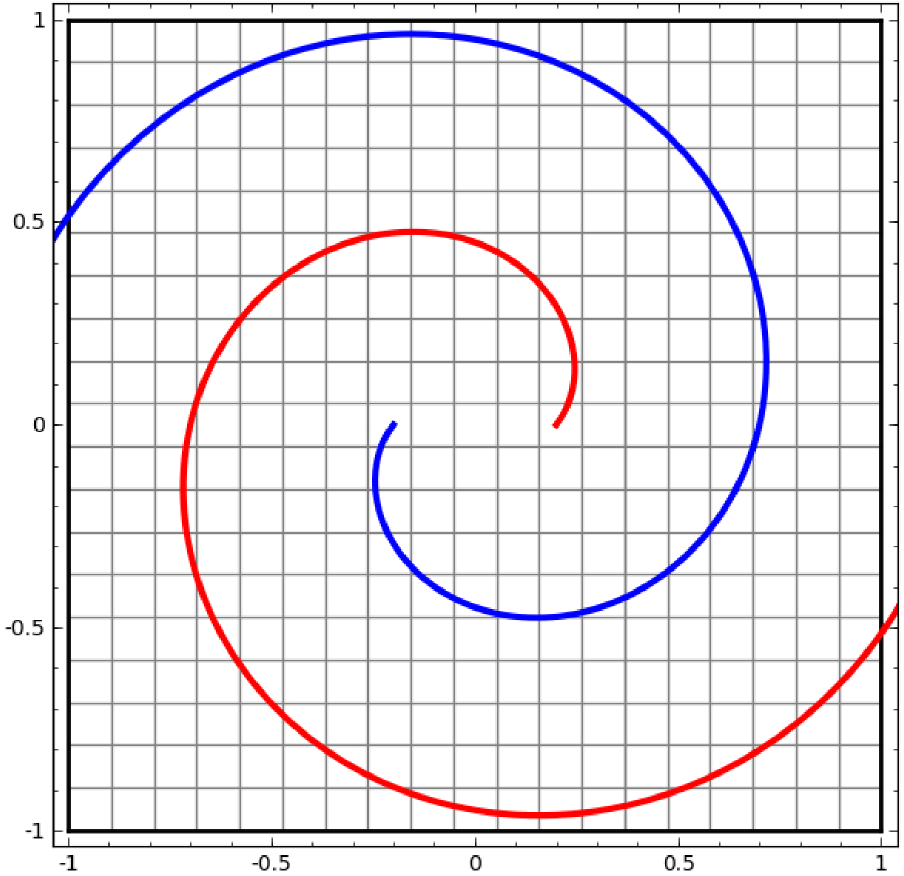
univ-cotedazur.fr

Multi-Layer Perceptron, manifold disentanglement



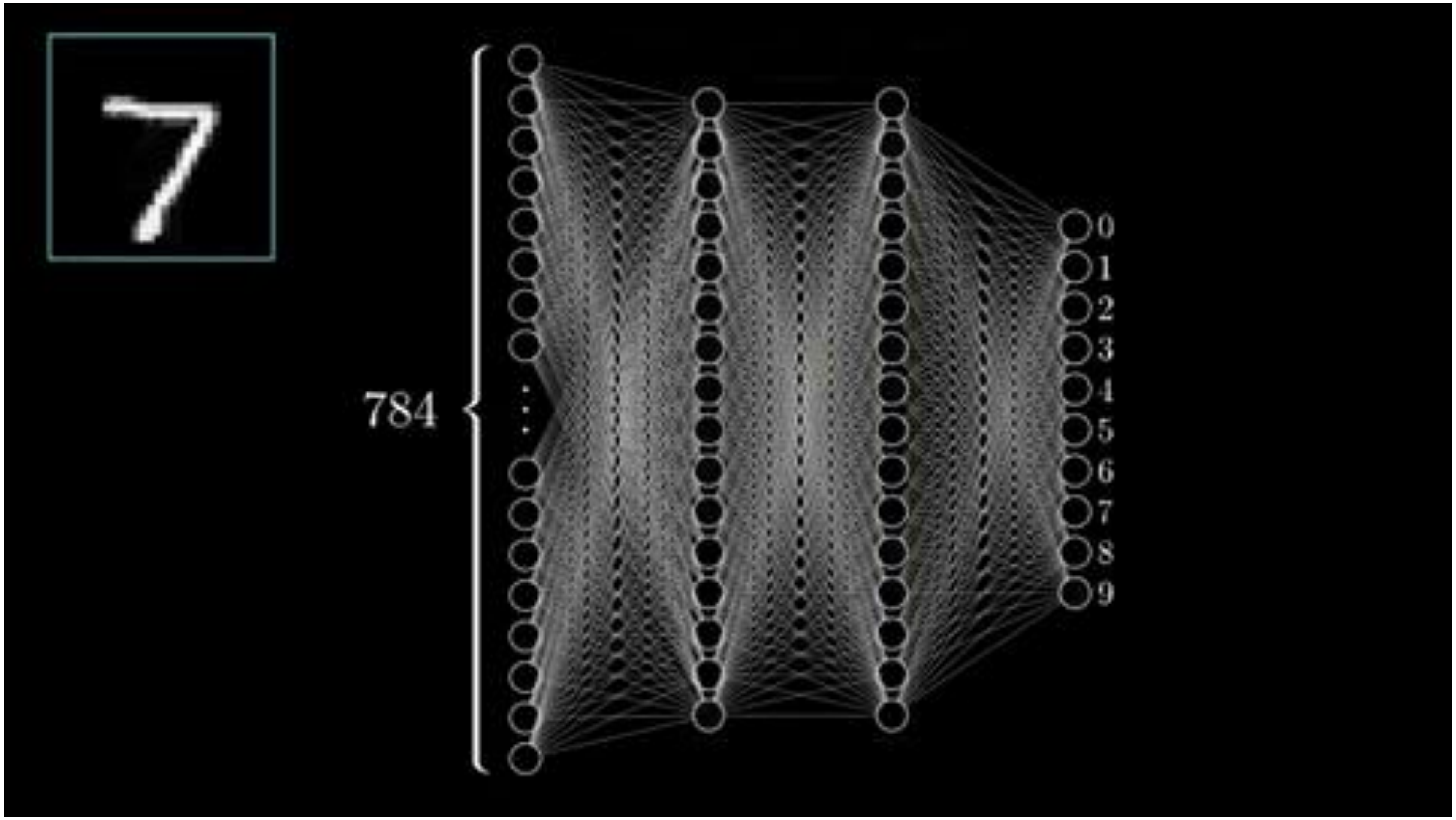
Illustrations from: http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

Multi-Layer Perceptron, manifold disentanglement



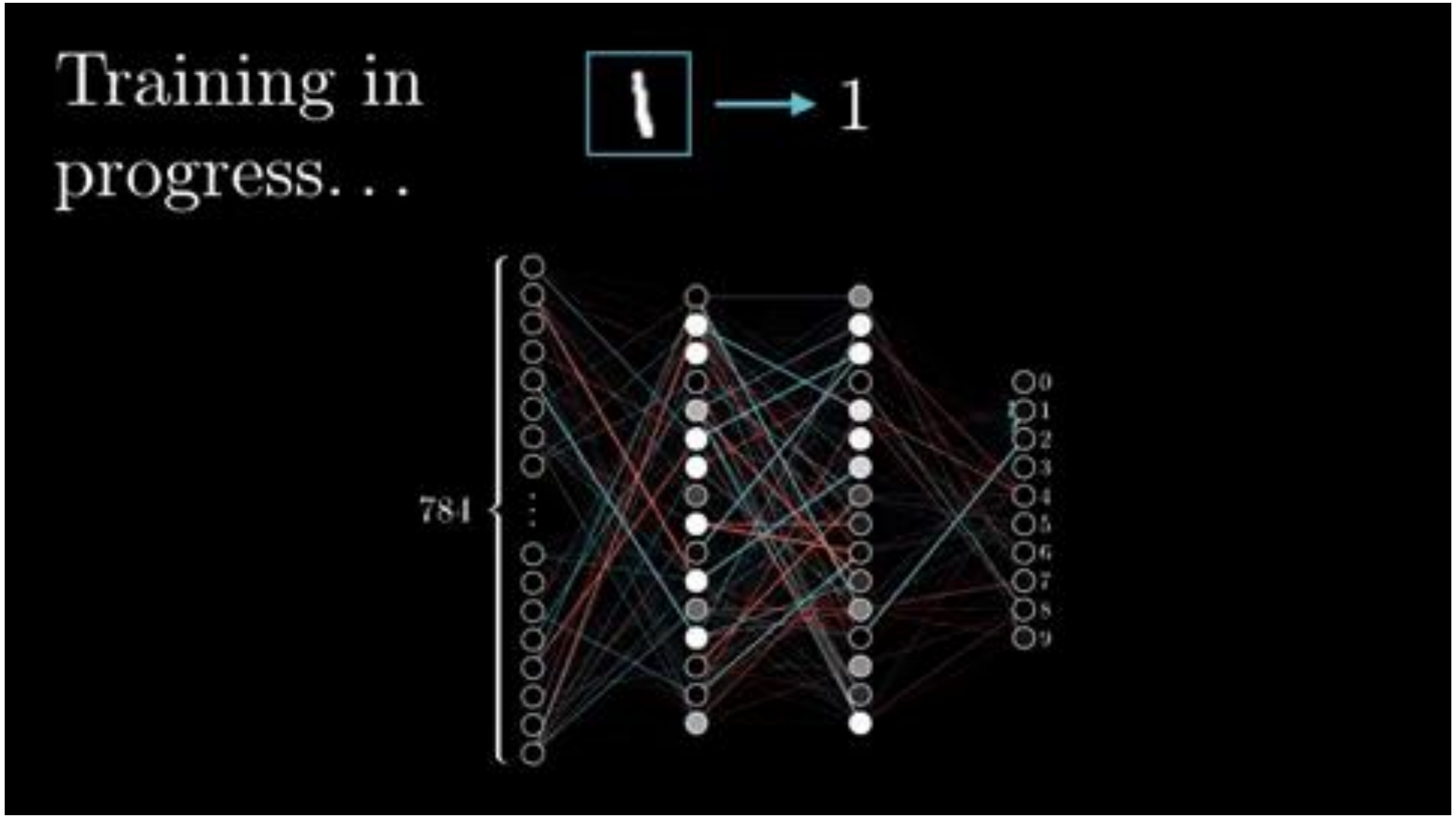
Illustrations from: http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

Solution: Neural Network



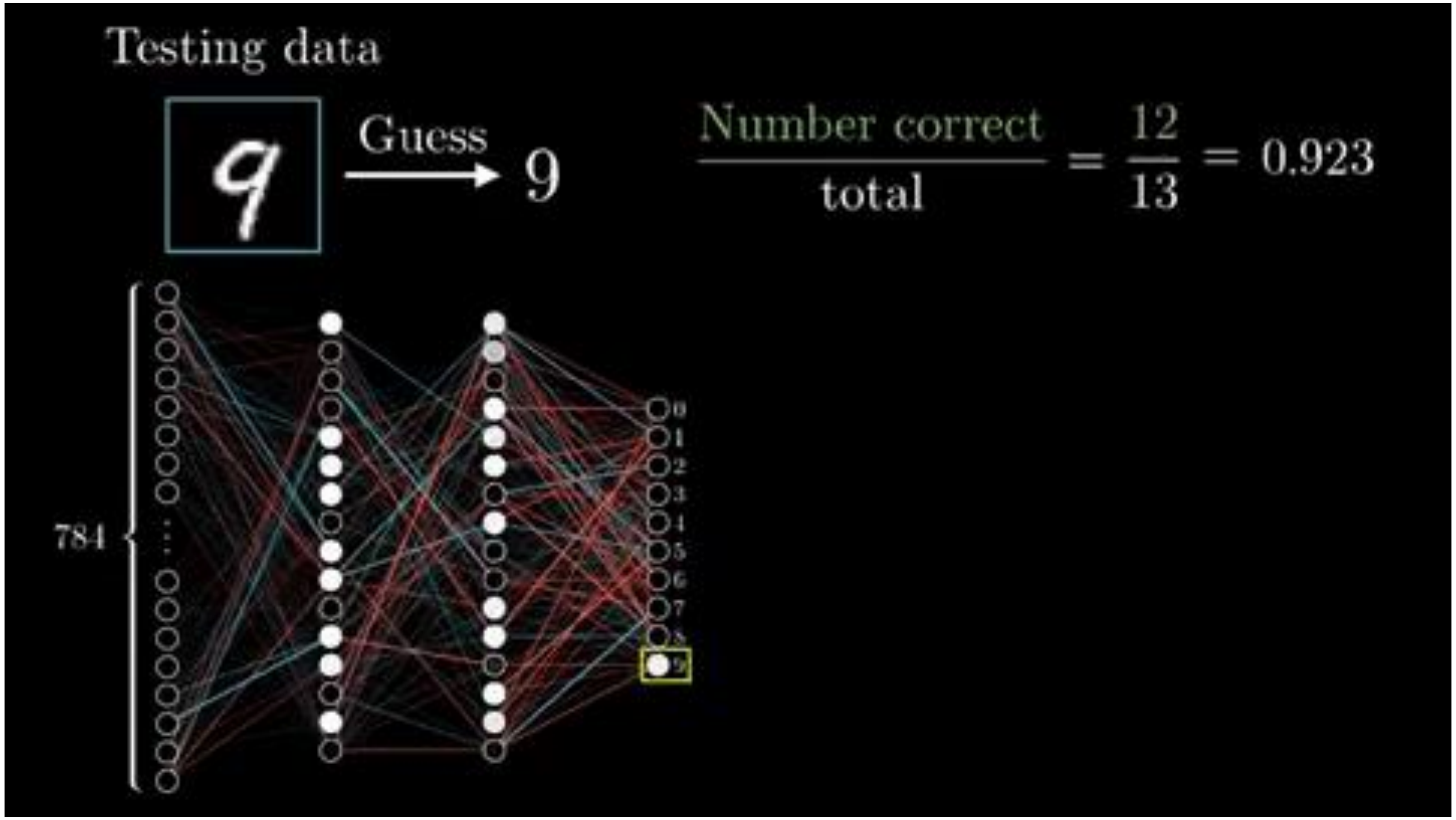
Source: https://www.3blue1brown.com/neural-networks

Training



Source: https://www.3blue1brown.com/neural-networks

Inference/Test

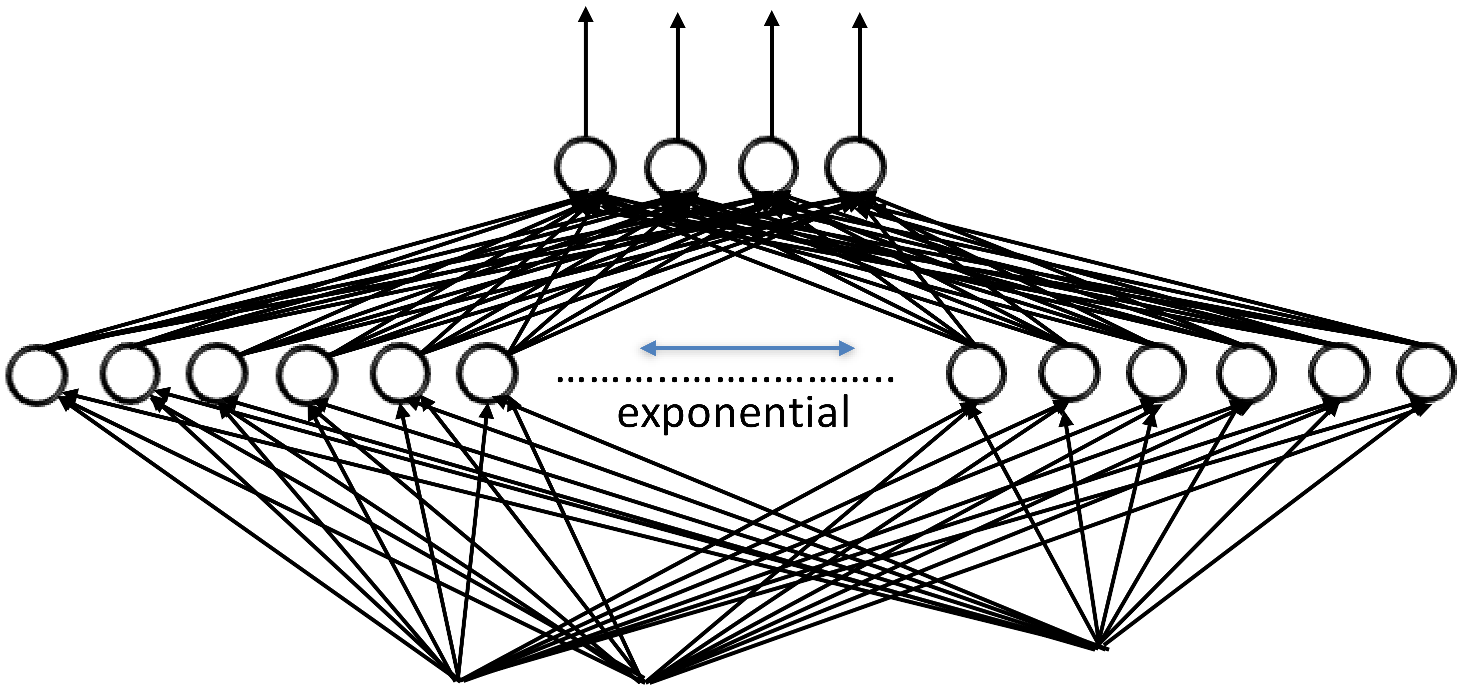


Source: https://www.3blue1brown.com/neural-networks

DEEP LEARNING PRINCIPLES

Deep representation origins

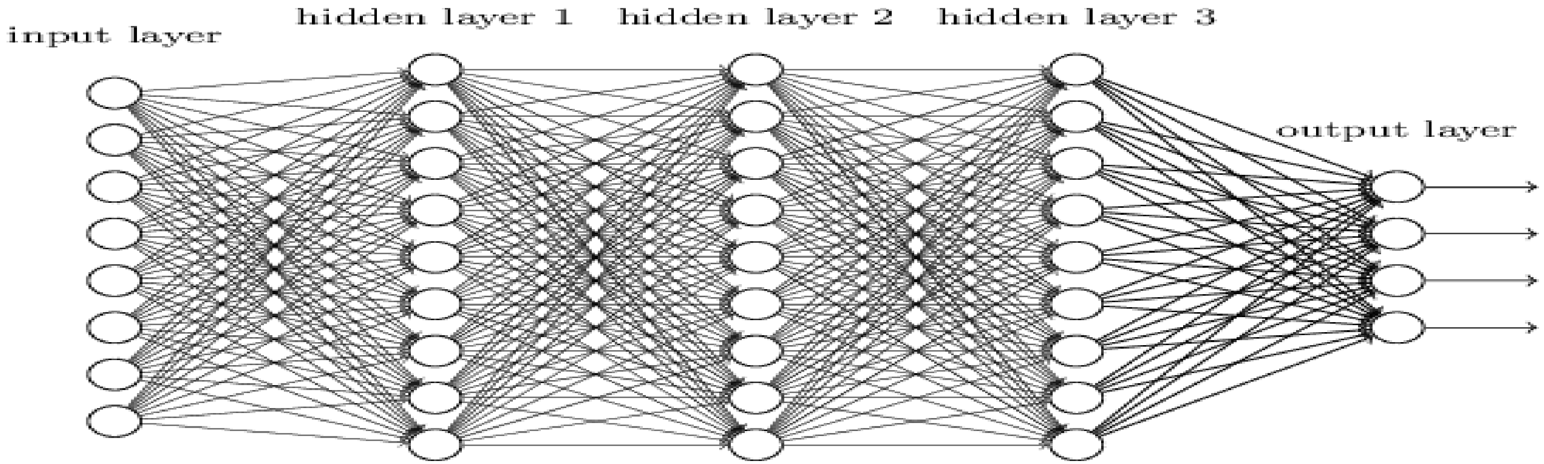
- **Theorems** (Cybenko (1989), Hornik & Stinchcombe & White (1989))
  A neural network with one single hidden layer is a universal "approximator", it can represent any continuous function on compact subsets of $\mathbf{R}^n \Rightarrow$ *2 layers are enough…***but** *hidden layer size may be exponential for error* ε *(or even infinite for error 0), and there is no efficient learning rule known.*



Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. Mathematics of control, signals and systems, 2(4), 303-314.
Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. Neural networks, 2(5), 359-366.

Deep representation origins

- **Theorem Hastad** (1986), **Bengio** et al. (2007) Functions representable compactly with *k* layers may require exponentially size with *k-1* layers

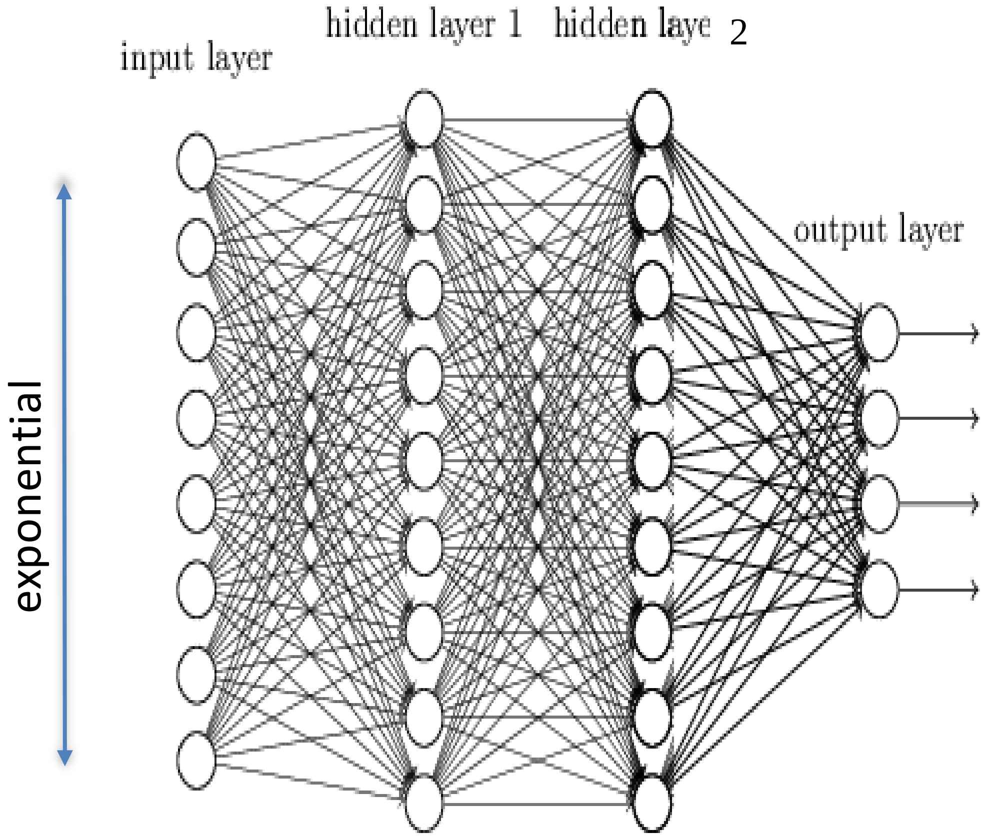


Johan T. Håstad. Computational Limitations for Small Depth Circuits. MIT Press, Cambridge, MA, 1987.
Bengio, Y., & LeCun, Y. (2007). Scaling learning algorithms towards AI. Large-scale kernel machines, 34(5), 1-41.

univ-cotedazur.fr

Deep representation origins

- **Theorem Hastad** (1986), **Bengio** et al. (2007) Functions representable compactly with *k* layers may require exponentially size with *k-1* layers



The Blessing of dimensionality: Thomas Cover's Theorem (1965)

Cover's theorem states: A complex pattern-classification problem cast in a high-dimensional space nonlinearly is more likely to be linearly separable than in a low-dimensional space.
(repeated sequence of Bernoulli trials)

The number of groupings that can be formed by (l-1)-dimensional hyperplanes to separate N points in two classes is:

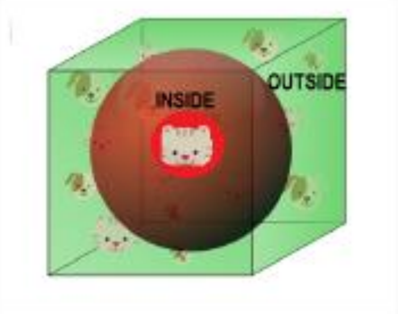
$$O(N,l) = 2\sum_{i=0}^{l} \frac{(N-1)!}{(N-1-i)!i!}$$

Notice: The total number of possible groupings is 2^N

The curse of dimensionality [Bellman, 1956]

- Euclidian distance is not relevant in high dimension: d ≥10
  1. look at the examples at distance at most r
  2. the hypersphere volume is too small: practically empty of examples

$$\frac{\textit{volume of the sphere of radial } r}{\textit{hypersphere of } 2r \textit{ width}} \rightarrow_{d\rightarrow\infty} 0$$



  3. need a number of examples exponential in d

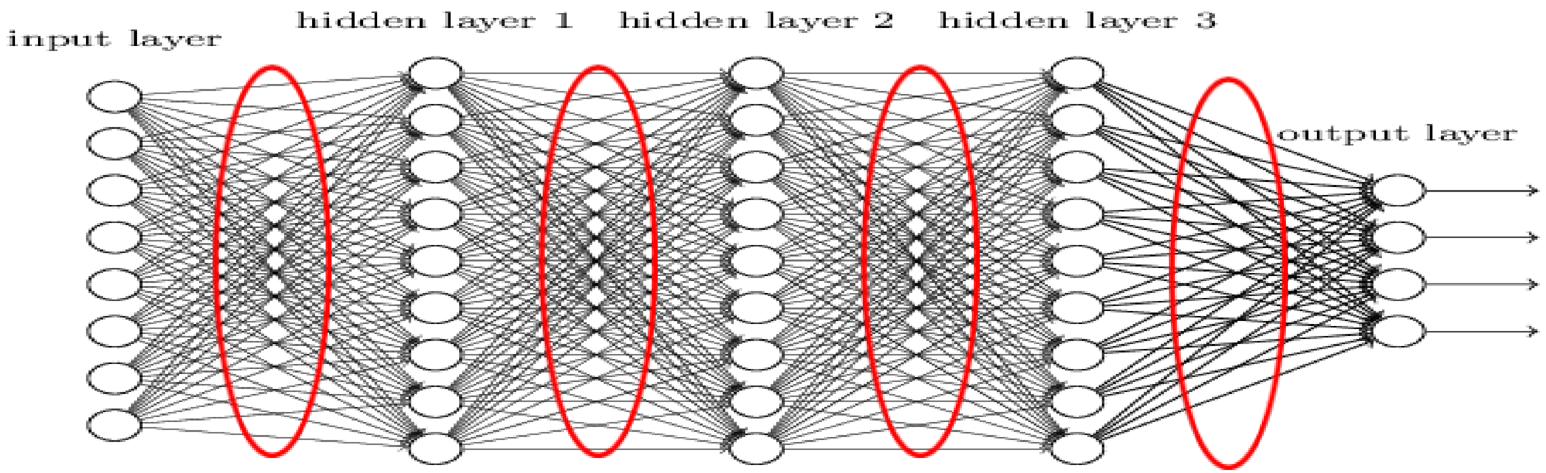
Remark

Specific care for data representation

Structure the network?

- Can we put any structure reducing the space of exploration and providing useful properties (invariance, robustness, sequentiality…)?

$$y = s(\underbrace{w_{13}s(\overbrace{w_{11}x_1 + w_{21}x_2 - w_{01}}^{z_1}) + w_{23}s(\overbrace{w_{12}x_1 + w_{22}x_2 - w_{02}}^{z_2})}_{z_3} - w_{03})$$



Enabling factors

- Why do it now ? Before 2006, training deep networks was unsuccessful because of practical aspects
  - faster CPU's
  - parallel CPU architectures
  - advent of GPU computing
  - Advances in ML/Optim (1995 -> 2005)

- Hinton, Osindero & Teh « A Fast Learning Algorithm for Deep Belief Nets », *Neural Computation*, 2006
- Bengio, Lamblin, Popovici, Larochelle « Greedy Layer-Wise Training of Deep Networks », *NIPS'2006*
- Ranzato, Poultney, Chopra, LeCun « Efficient Learning of Sparse Representations with an Energy-Based Model », *NIPS'2006*

- Results…
  - 2009, sound, interspeech +~24%
  - 2011, text, +~15% without linguistic at all
  - 2012, images, ImageNet +~20%
  - 2020, molecules/graphs, AlphaFold, +~24%

    (sorry in French, https://www.youtube.com/watch?v=OGewxRMME8o)

Slide from Yoshua Bengio

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

Deep learning = Learning representations/features

- The traditional model of pattern recognition (since the late 50's)
  - Fixed/engineered features (or fixed kernel) + trainable classifier

Hand-crafted Feature Extractor → "Simple" Trainable Classifier →

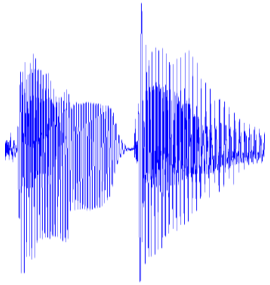
- End-to-end learning / Feature learning / Deep learning
  - Trainable features (or kernel) + trainable classifier

Trainable Feature Extractor → Trainable Classifier →

Slide from Yann LeCun

Architecture of "mainstream" pattern recognition systems

- Modern architecture for pattern recognition
  - Speech recognition: early 90's – 2011



MFCC → Mix of Gaussians → Classifier →

fixed | unsupervised | supervised

  - Object Recognition: 2006 - 2012

SIFT HoG → K-means Sparse Coding → Pooling → Classifier →

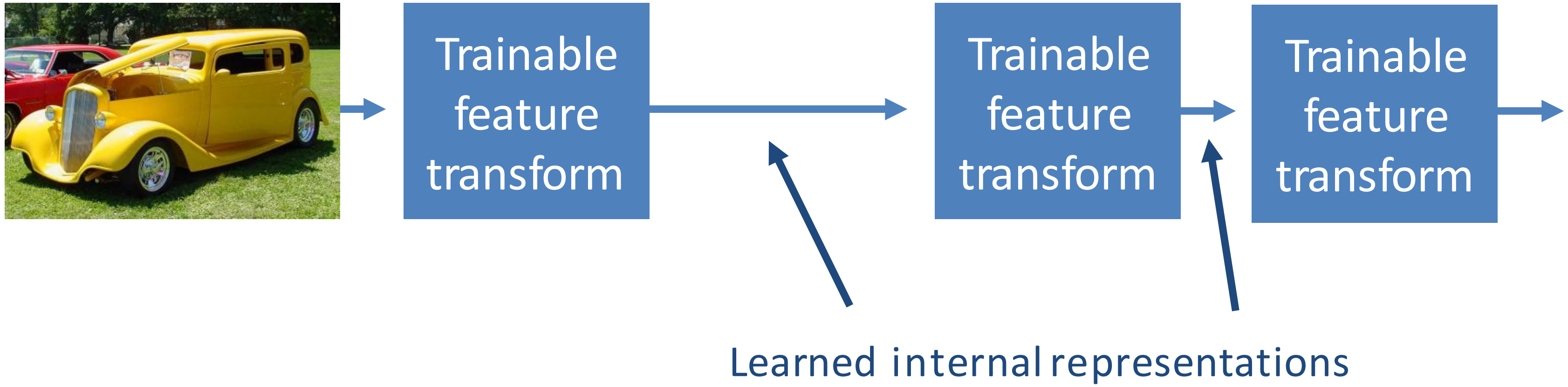
fixed | unsupervised | supervised

Low-level Features | Mid-level Features

Slide from Yann LeCun

Trainable feature hierarchies: end-to-end learning

- A hierarchy of trainable feature transforms
  - Each module transforms its input representation into a higher-level one.
  - High-level features are more global and more invariant
  - Low-level features are shared among categories



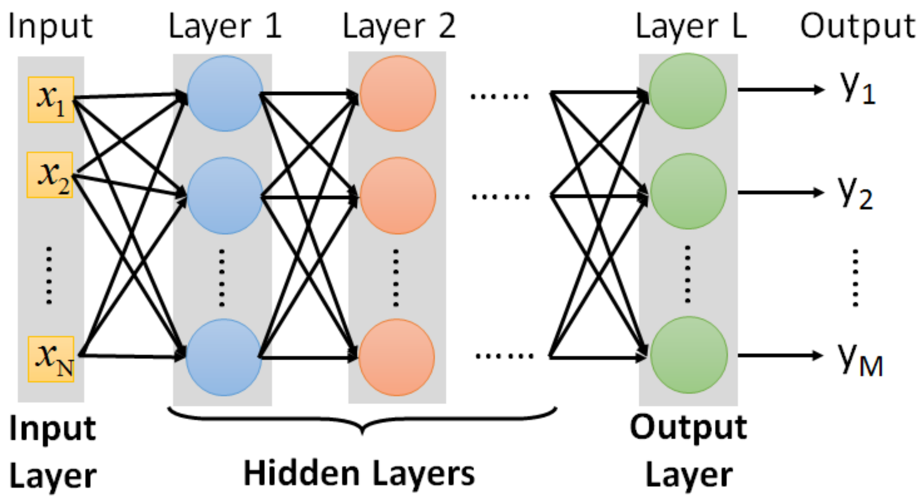
- How can we make all the modules trainable and get them to learn appropriate representations?

Slide from Yann LeCun

UNIVERSITÉ CÔTE D'AZUR

univ-cotedazur.fr

FAQ



- Q: How many layers? How many neurons for each layer?

Trial and Error + Intuition

- Q: Can we design a specific network structure?

Lecture 2

- Q: Can the structure be automatically determined?
  - Yes, intense research in the last 2 years (e.g. **AutoML, AdaNet**).