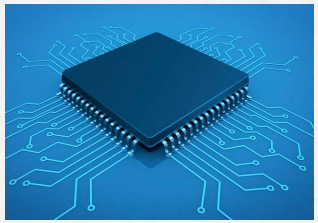
2. PERCEPTRONS

Stephan Robert-Nicoud
HEIG-VD/HES-SO

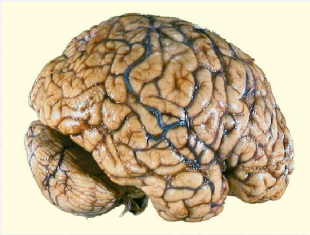
APPRENTISSAGE PAR RESEAUX DE NEURONES ARTIFICIELS

Objectives

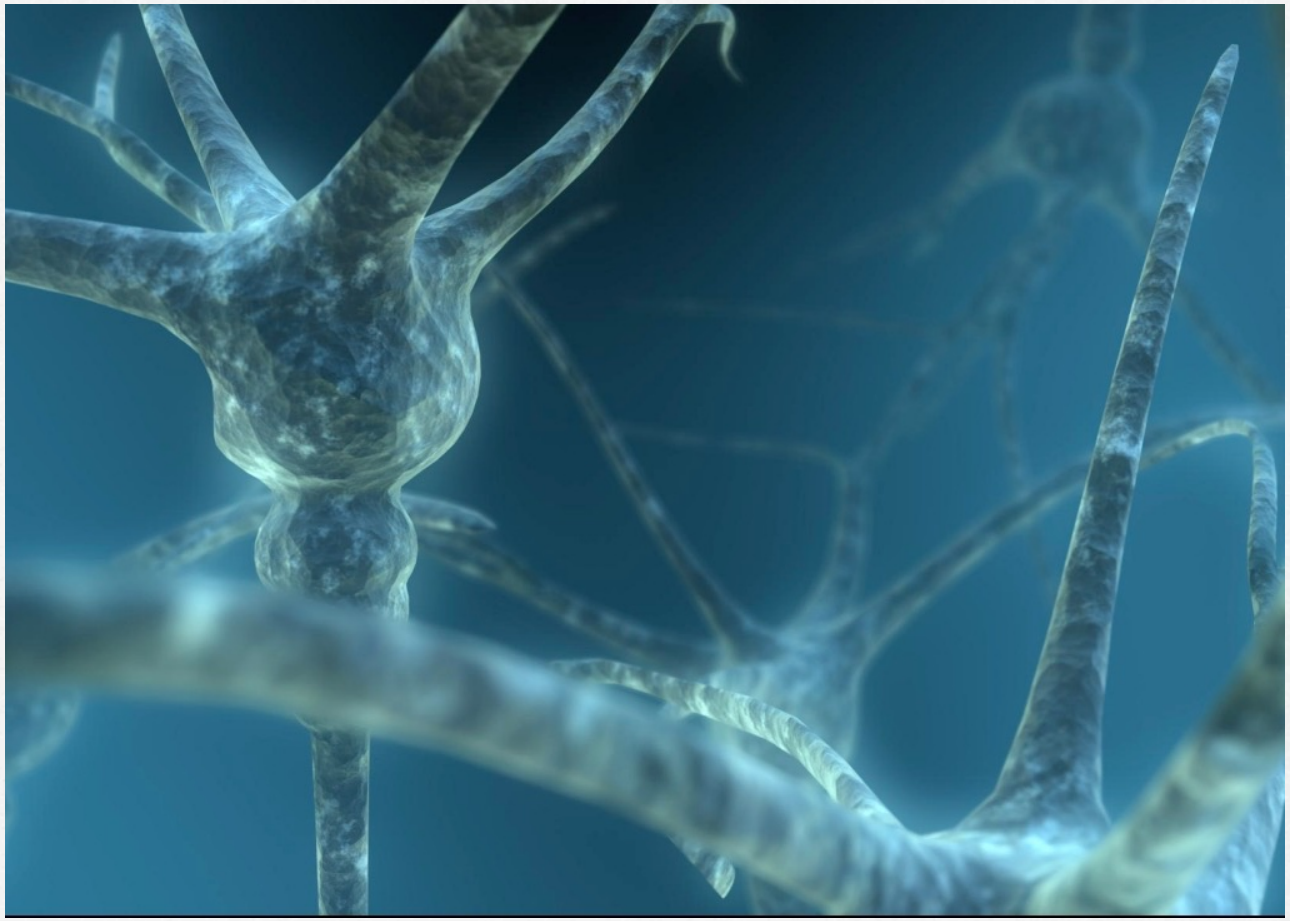
- Understand what motivates the inspiration from biological neural networks
- Study the basic models of artificial neurons and their functioning
- Understand what does an artificial neuron do from the mathematical point of view
- Understand how does an artificial neuron "learn" from training data



VS



	von Neumann computer	Brain
processor	Complex High frequency (GHz) one or several (cores)	Simple (neuron) low frequency (Hz) Many!
memory	Separated from the processor Address-based access	Integrated to computing (distributed) Content-addressable
computing	centralized sequential Programmable	Distributed Parallel Learning
reliability	Highly vulnerable	Very robust

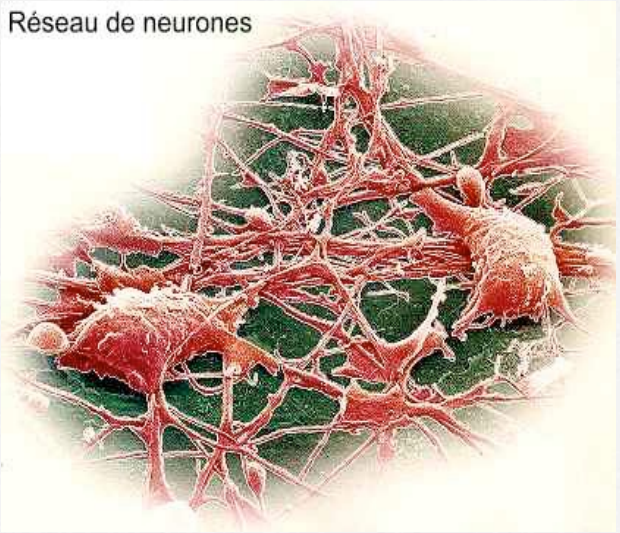


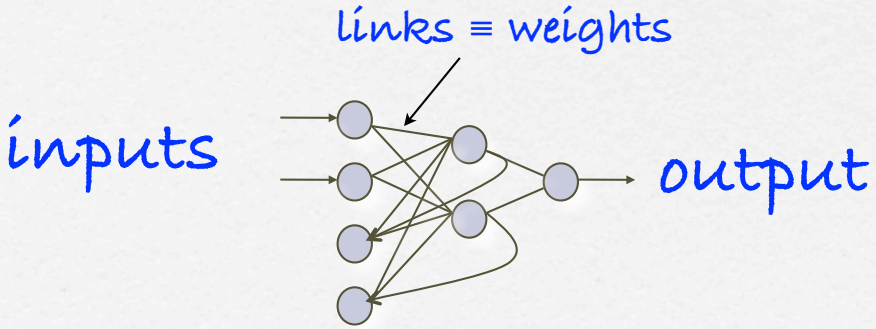
Connectionism: modeling of mental or behavioral phenomena as emergent processes of interconnected networks of simple units

Connectionist systems

We ideally use a massively parallel architecture where each computational element (neuron) performs a sort of a correlation between inputs and stored values called synaptic weights.

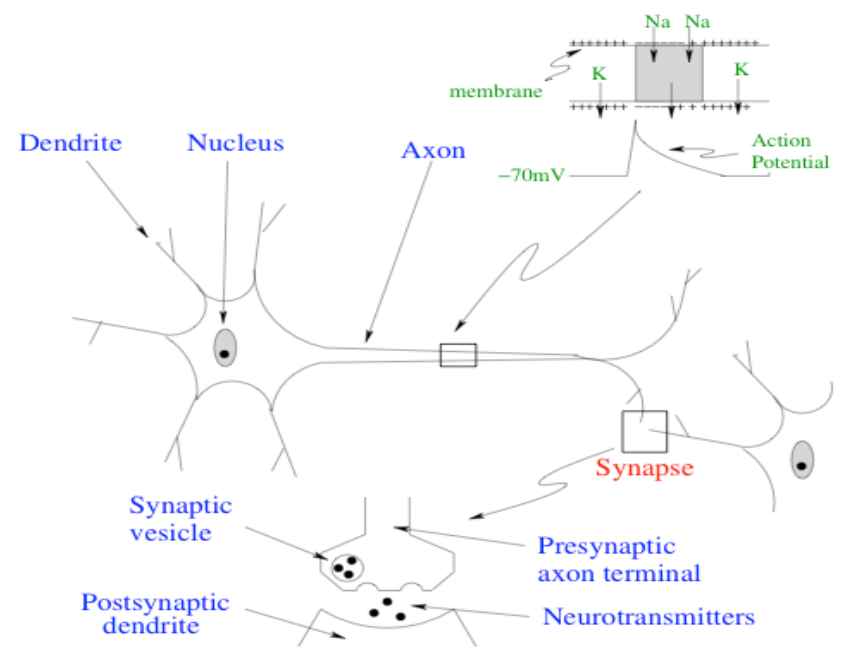
Paradigm shift: we replace programming by learning



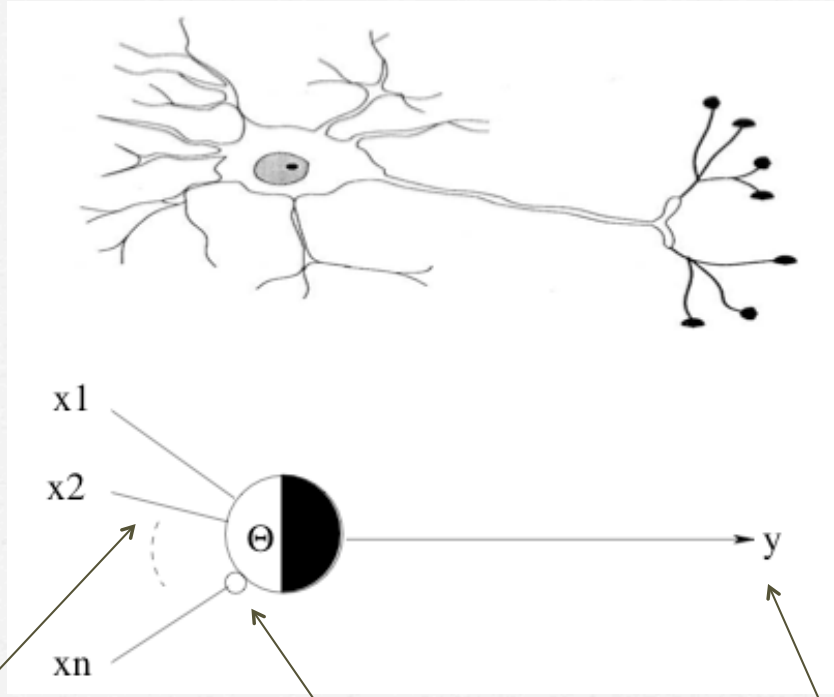


The artificial neuron

MCulloch-Pitts (1948)



biological neuron



excitatory input

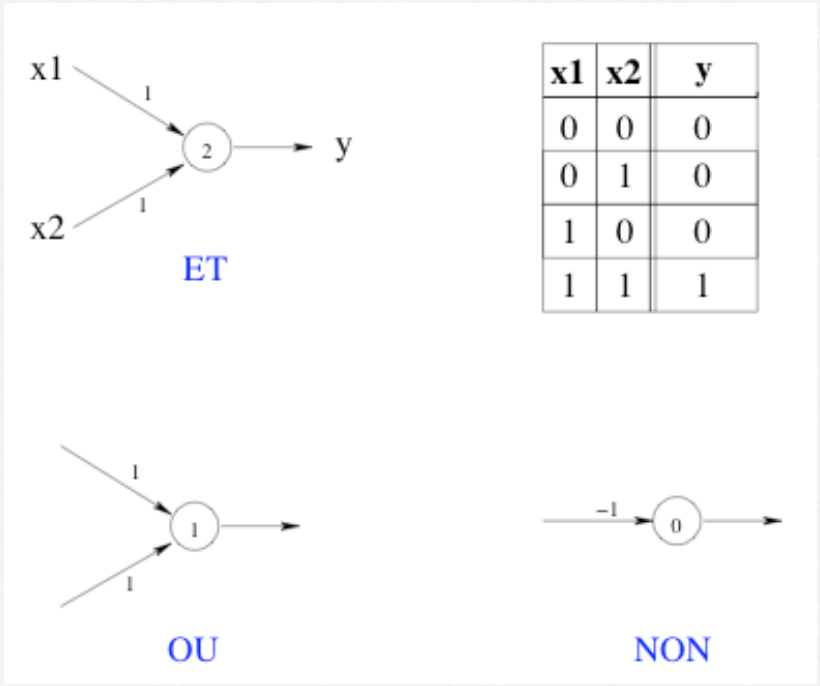
inhibitory input

binary output

IF $(x_1 + x_2 + \ldots - x_n) \geq \Theta$ then y = 1, else y = 0

Threshold logic

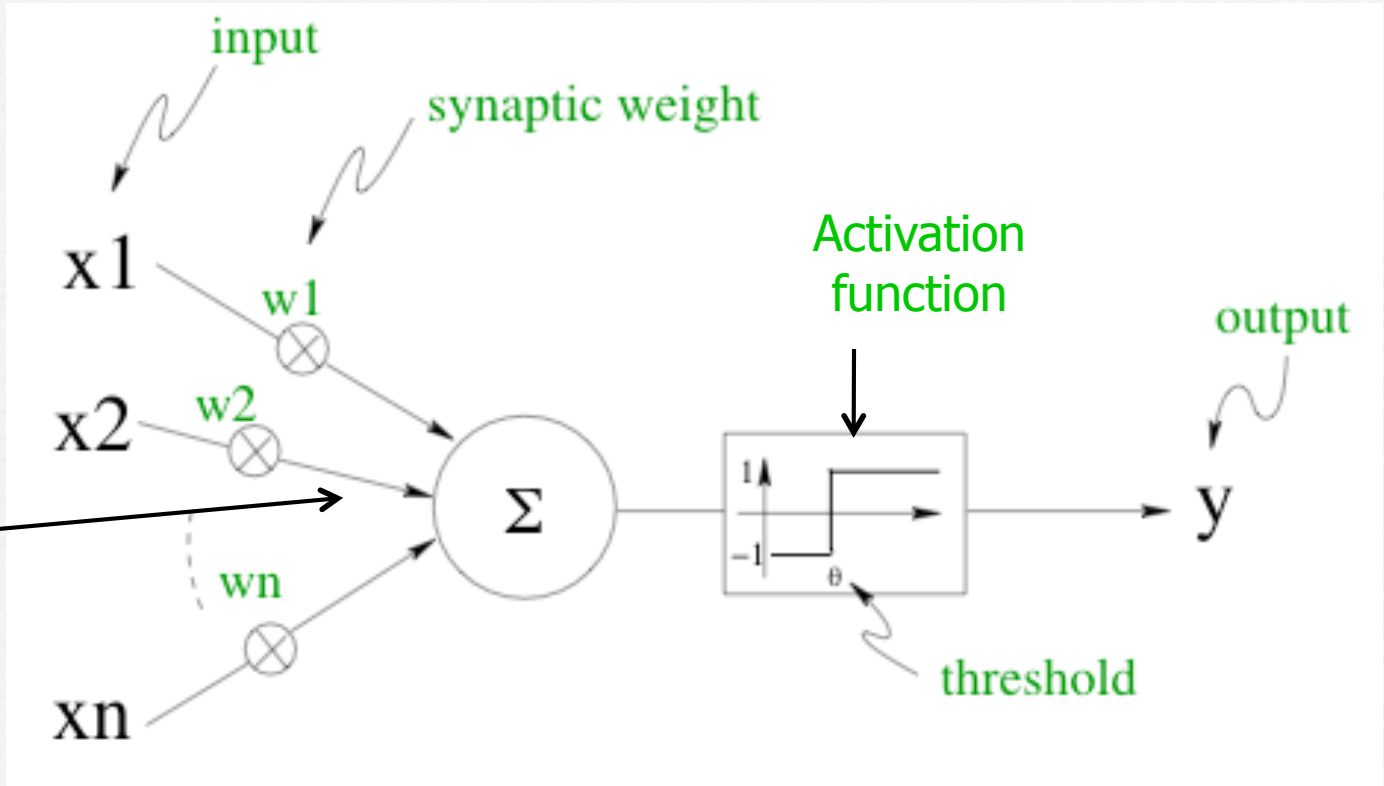
We can interpret that excitatory inputs are weighted by 1 and the inhibitory inputs are weighted by -1



x1	x2	y
0	0	0
0	1	0
1	0	0
1	1	1

if (x1 + x2) ≥ 2 then y = 1, otherwise y = 0

Rosenblatt's Perceptron (1958)

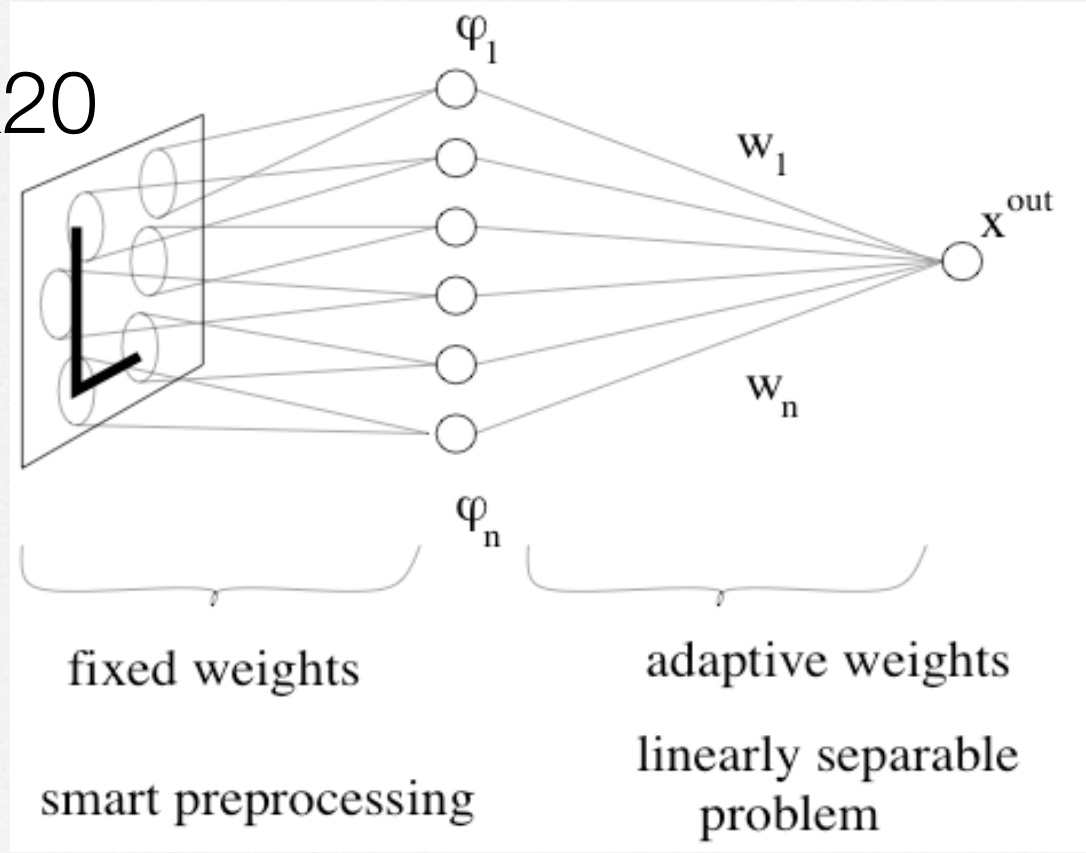


If $(w_1x_1 + w_2 x_2 + \ldots + w_n x_n) \geq \Theta$ then y = 1, otherwise y = -1

If $\Sigma\ w_i x_i \geq \Theta$ then y = 1, otherwise y = -1

Perceptron

20x20



NEW NAVY DEVICE LEARNS BY DOING

Psychologist Shows Embryo of Computer Designed to Read and Grow Wiser

WASHINGTON, July 7 (UPI) —The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.

The embryo—the Weather Bureau's $2,000,000 "704" computer—learned to differentiate between right and left after fifty attempts in the Navy's demonstration for newsmen.

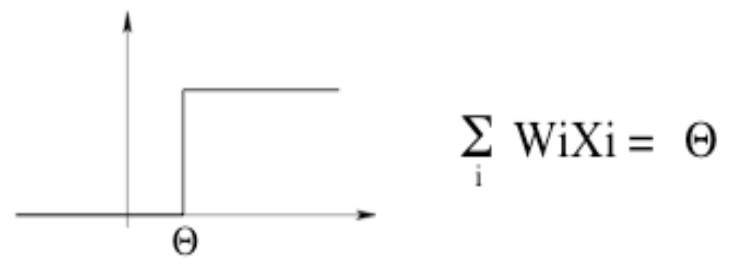
The service said it would use this principle to build the first of its Perceptron thinking machines that will be able to read and write. It is expected to be finished in about a year at a cost of $100,000.

The New York Times, July 8 1958

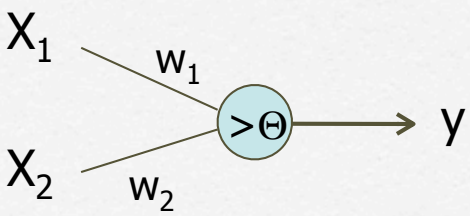
What a Perceptron can do ?

geometric interpretation

The activation function is discontinuous, what happens when the sum of weighted inputs equals the threshold ?

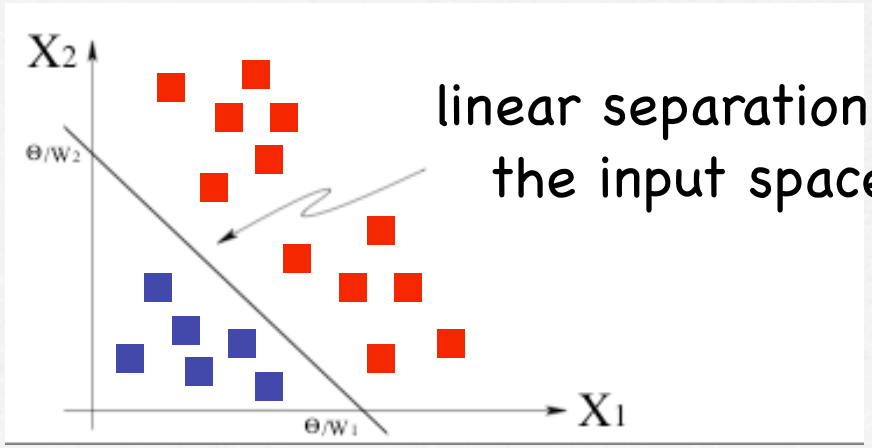


2D case: $w_1 x_1 + w_2 x_2 = \Theta$, therefore $x_2 = -(w_1/w_2)\, x_1 + \Theta/w_2$



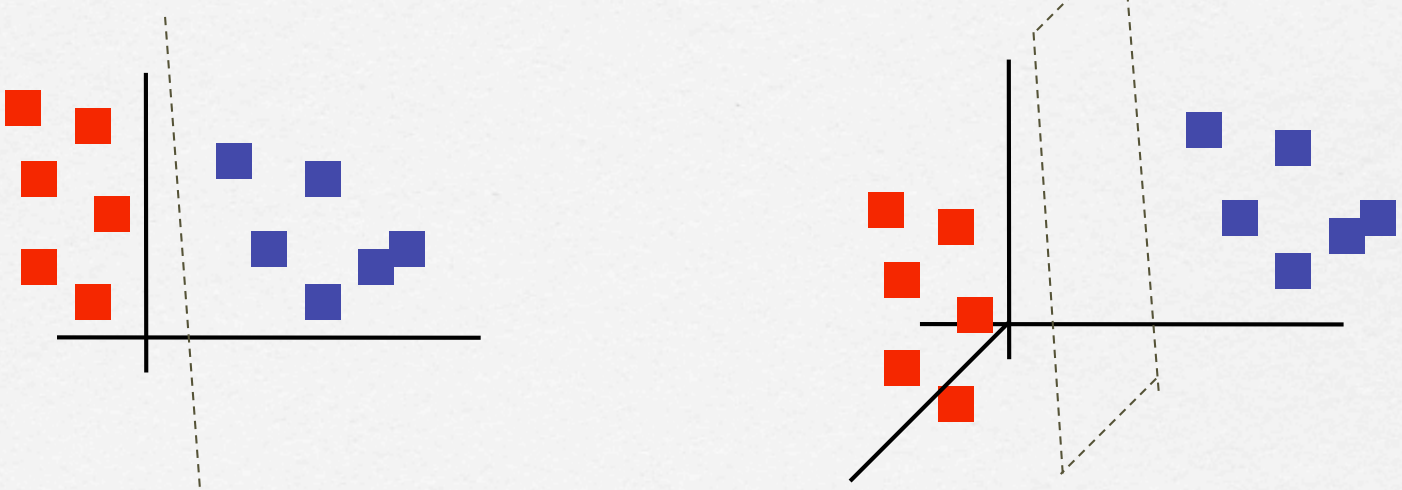
y=0 :

y=1 :



The Perceptron as a pattern classifier

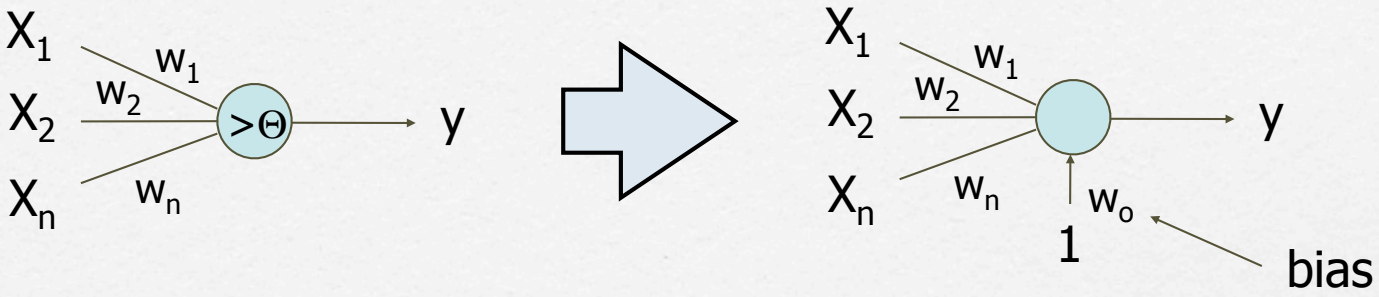
A Perceptron is capable of linearly separating the space into two distinct regions. In 2D the separation is a line, in 3D it is a plane and in n-dimensions it is a hyperplane



Learning problem: in a classification problem, it consists on finding the weights and threshold values to correctly separate blue from red data points

Bias: a learnable threshold of activation

We would like to get rid of the threshold



If $\Sigma\ w_i x_i \geq \Theta$ then y = 1,
otherwise y = 0

or

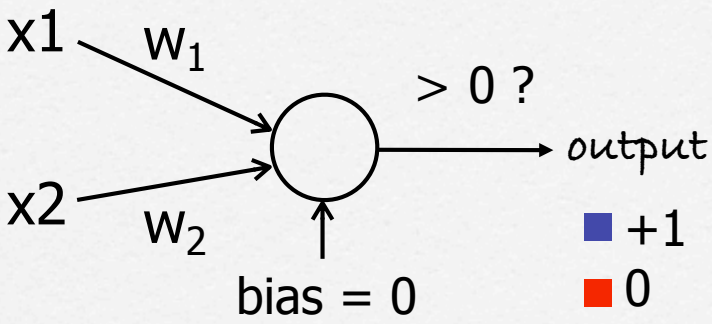
If $\Sigma\ w_i x_i - \Theta \geq 0$ then y = 1,
otherwise y = 0

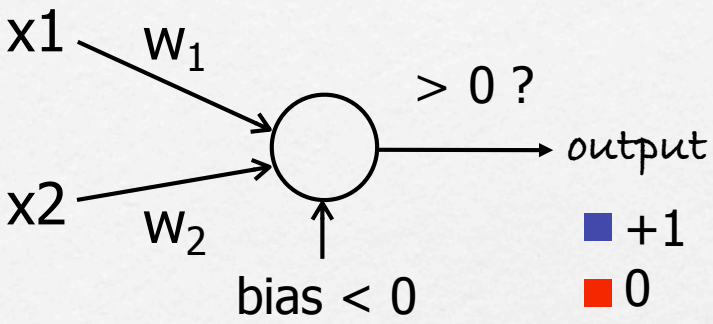
Let's introduce $W_0 = -\Theta$:

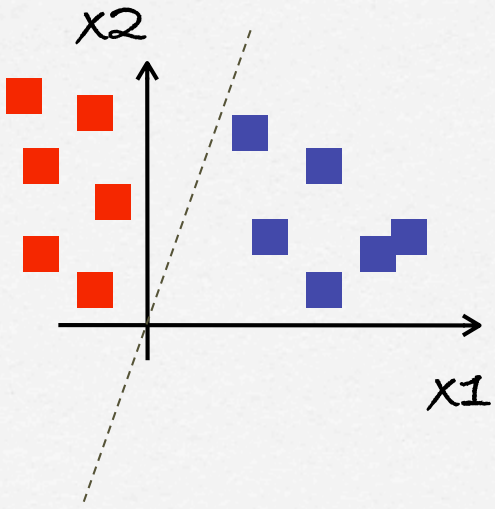
If $\Sigma\ w_i x_i + w_0 \geq 0$ then y = 1,
otherwise y = 0

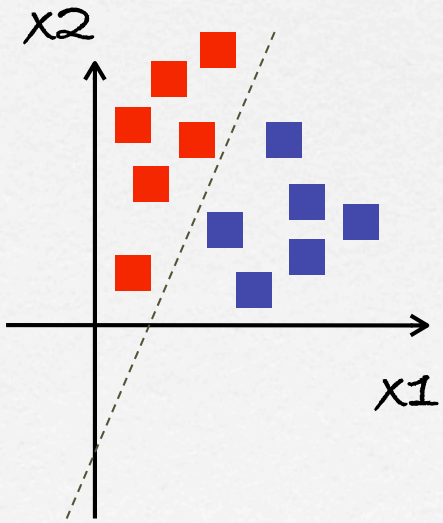
Wo is called a "bias" and can be seen as an extra weight that can be learned using a learning algorithm

Biasing the output for zero inputs









Perceptron learning algorithm

1. Randomly initialize weights
2. Compute the neuron's output **y** for a given input vector **x**

   $y = \Sigma_j w_j x_j$

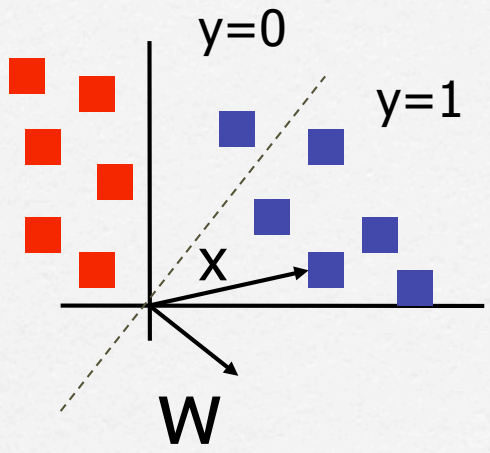
3. Update weights: $W_j(t+1) = W_j(t) + \eta(d-y)x$ //so-called delta rule

   d is the desired output and η is the learning rate, **0.0 < η < 1.0**

4. Repeat 2 and 3 for a given number of steps or until the error is smaller than a given threshold, i.e., **error < 1/2 $\Sigma_p (y_p-d_p)^2$**

Learning in action (weight updating)

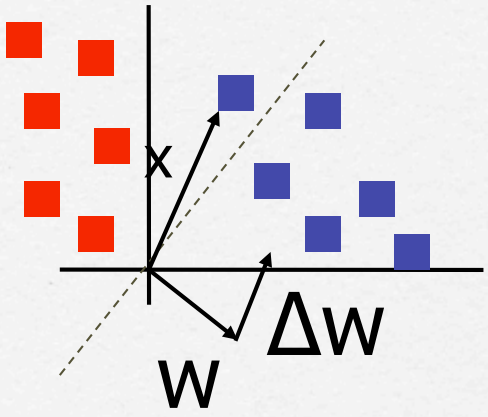
1) No classification error

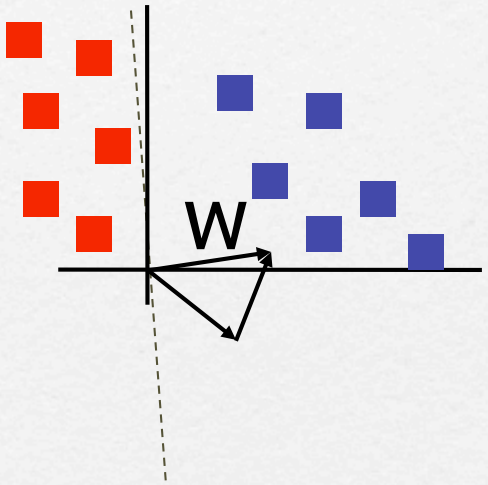


$W_j(t+1) = W_j(t) + \eta(d-y)\mathbf{x}$

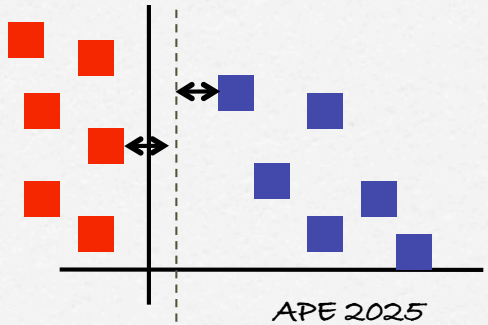
d=y, so $W_j(t+1) = W_j(t)$

2) Error correction after a weight update





$W_j(t+1) = W_j(t) + \eta(d-y)\mathbf{x}$

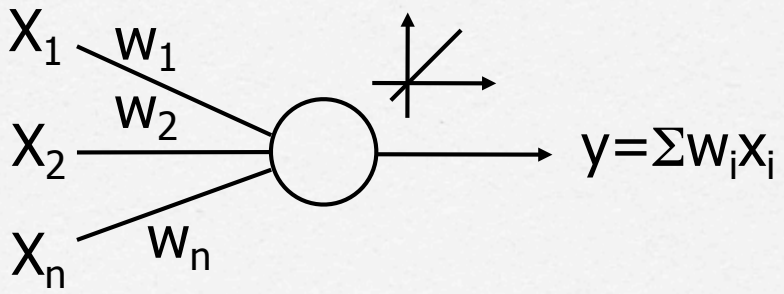


optimal separation

Gradient descent

Widrow-Hoff algorithm / Delta rule

The activation function of the Perceptron has a discontinuity, thus it is not derivable at that point. By using a linear activation function, Widrow and Hoff introduced a new model called ADALINE or ADAptive LINear Element



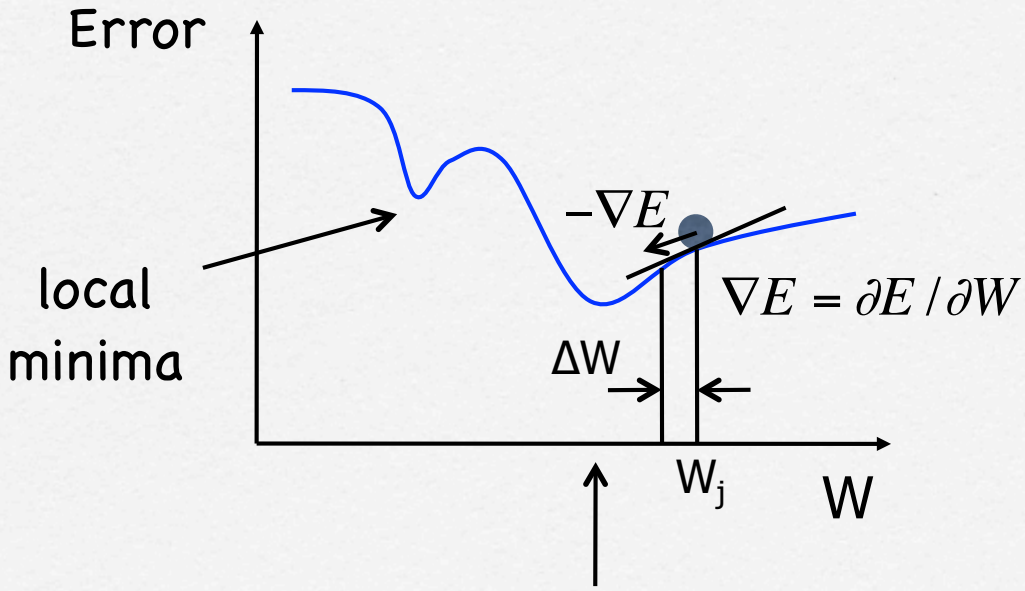
So, what is learning ? : the idea is to find the weights Wi that minimize an error function E, such as this one:

$$E = 1/2\sum_{i=1}^{p}\|y_i - d_i\|^2, y = \sum_j w_j x_j$$

A way to do this exploits the first derivative of E with respect to the weights. This method is called gradient descent, and dates back to Cauchy (1847).

Gradient descent (2)

Widrow-Hoff algorithm / Delta rule



Set of weights minimizing the error

$$E = 1/2 \sum_{i=1}^{p} \|y_i - d_i\|^2, y = \sum_j w_j x_j$$

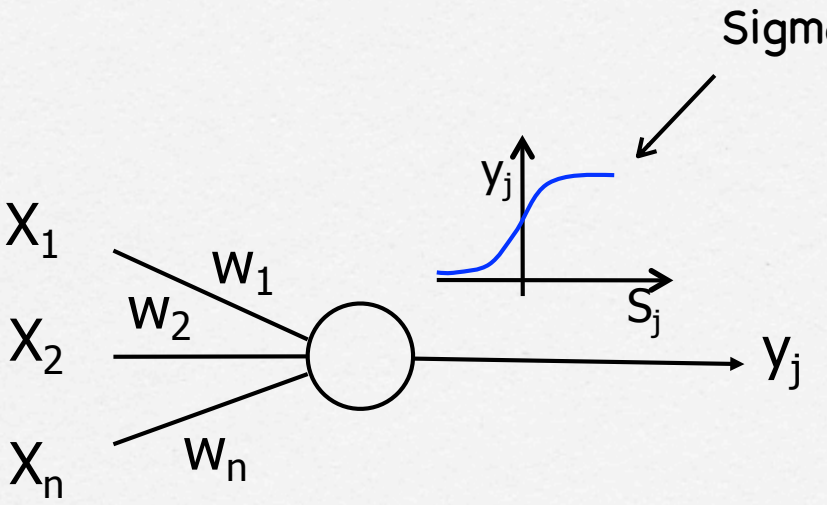
$$\nabla E = \partial E / \partial w = \partial E / \partial y \cdot \partial y / \partial w$$

$$\partial E / \partial y = y - d$$

$$\partial y / \partial w_j = \frac{\partial}{\partial w_j} \sum_j w_j x_j = x_j$$

$$\Delta w_j = -\nabla E = \eta (d - y) x_j$$

Generalized Delta rule



$$y_j = \frac{1}{1+e^{-S_j}}, \quad S_j = \sum w_i x_i$$

$$\Delta w_j = \eta(d-y) y_j' x_j$$

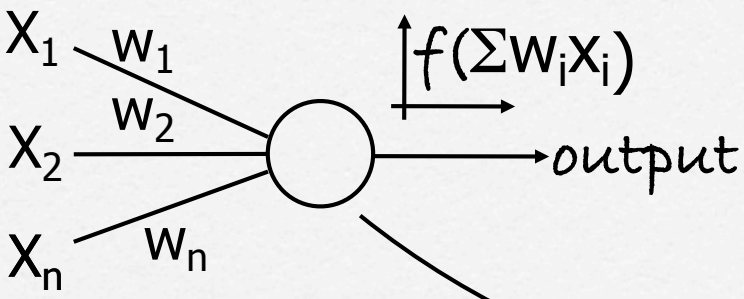
$$y_j' = y_j(1-y_j)$$

The tanh **and** ReLu are other widely used activation functions

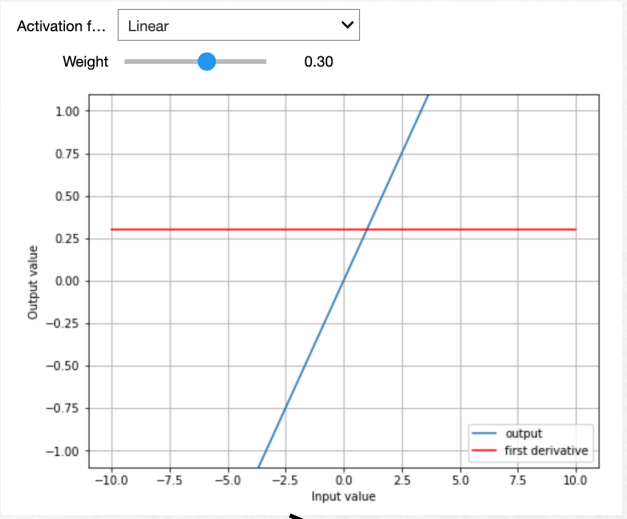
```
import numpy as np

# sigmoid function and derivative
def sigmoid(neta):
    output = 1 / (1 + np.exp(-neta))
    d_output = output * (1 - output)
    return (output, d_output)

# input dataset
X = np.array([  [0.2,0.2],
                [0.2,0.5],
                  .......
                [0.8,0.7],
                [0.8,0.9] ])

# output classes
y = np.array([[0,0,0,0, ..... 1,1,1,1]]).T

# seed random numbers to make calculation
# deterministic (just a good practice)
np.random.seed(1)

# initialize weights randomly with mean 0
weights = np.random.normal(size=2)
bias = np.random.normal(size=1)
```

```
inputs = X
target = y

for iter in xrange(100):

    # forward propagation
    neta = np.dot(inputs, weights) + bias
    output, d_output = sigmoid(neta)

    # how much did we miss?
    error = target - output

    # learning rate
    alpha = 0.1

    d_w_x = alpha * error * d_output * inputs[:,0]
    d_w_y = alpha * error * d_output * inputs[:,1]
    d_b = alpha * error * d_output

    # update weights
    weights += np.array([np.sum(d_w_x), np.sum(d_w_y)])
    bias += np.sum(d_b)

print "Network After Training:"
print weights, bias
```

Activation functions (1)



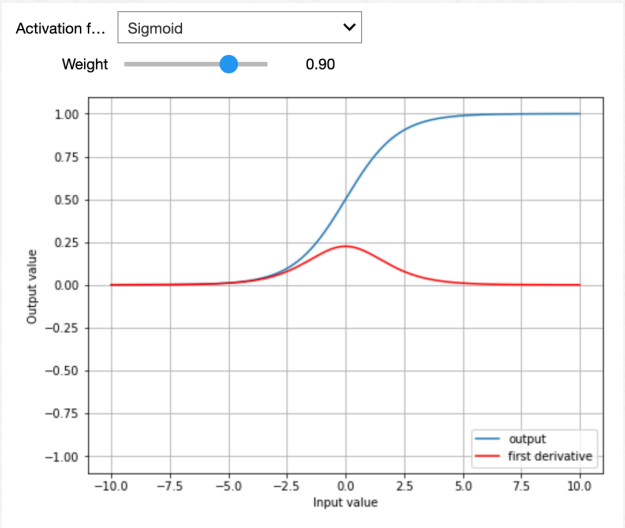
Linear



+∞

-∞

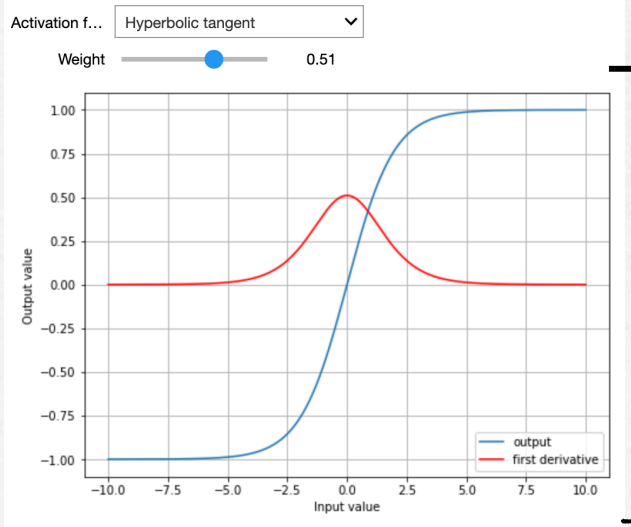
sigmoid



1

0

tanh



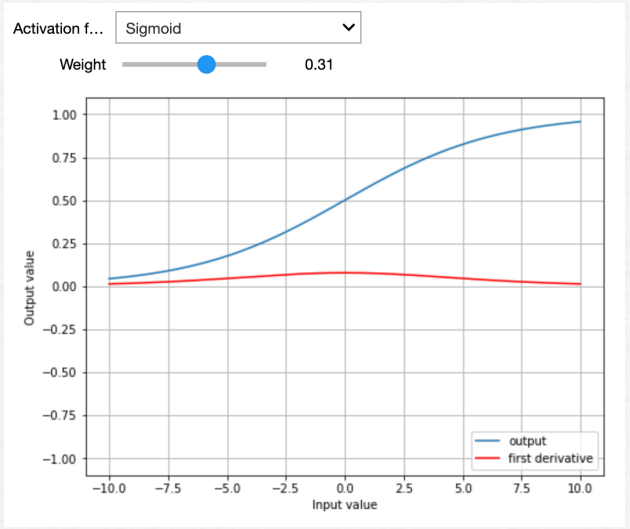
+1

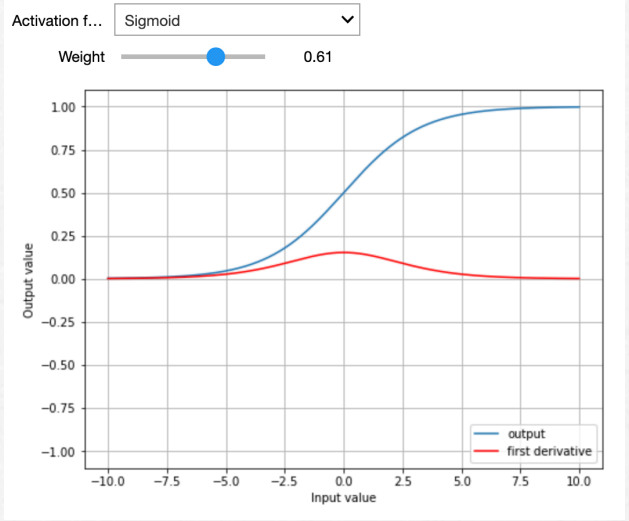
-1

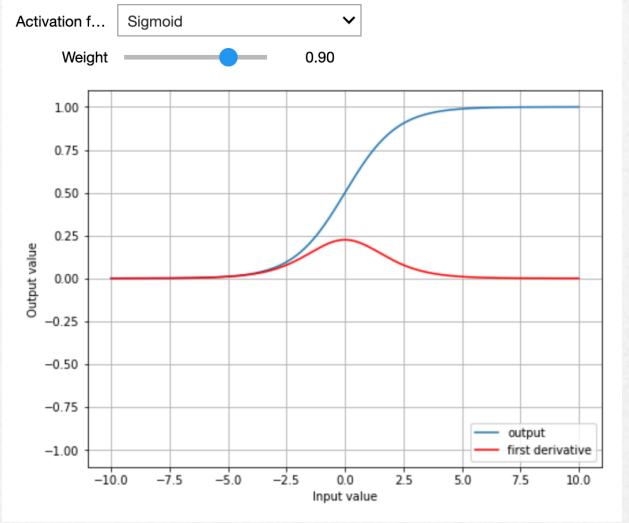
HE VD IG

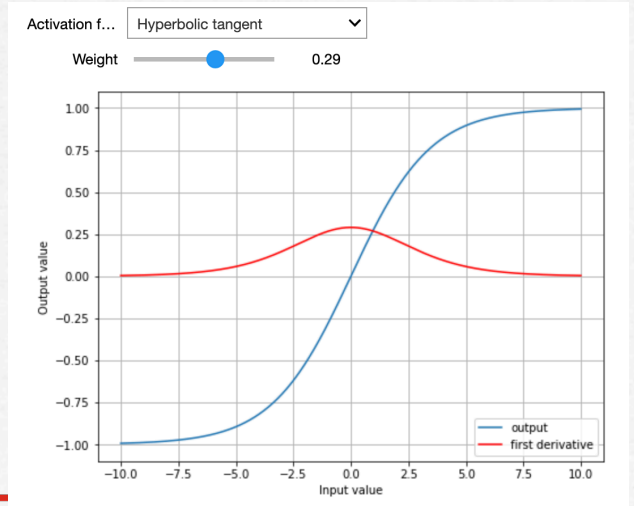
APE 2025

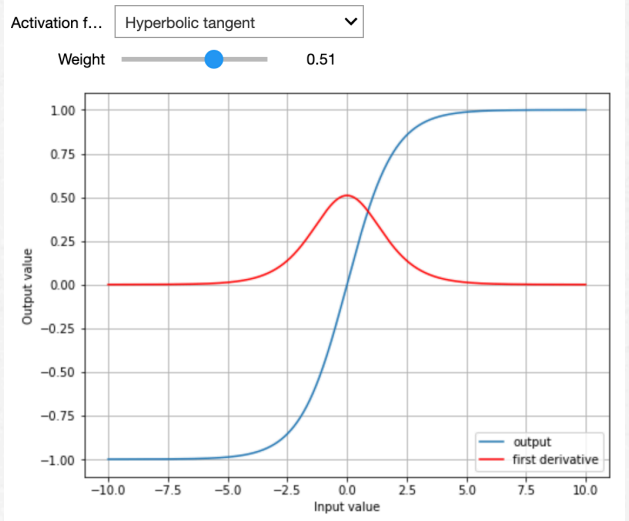
Activation functions (2)

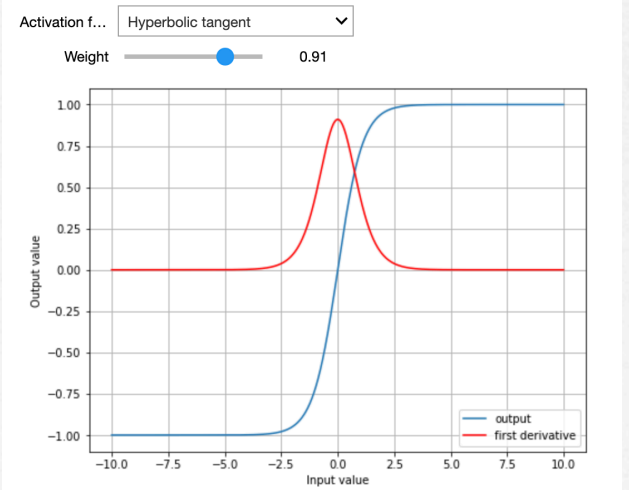








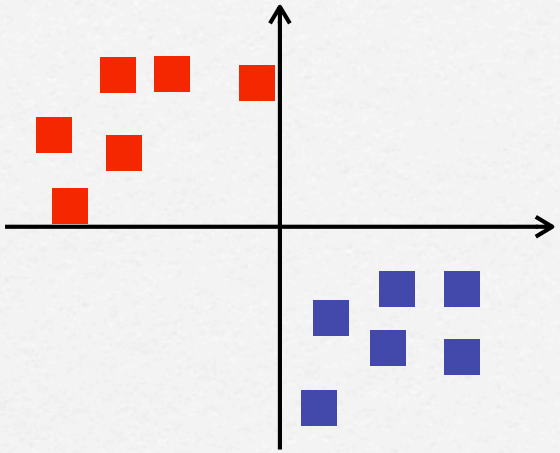


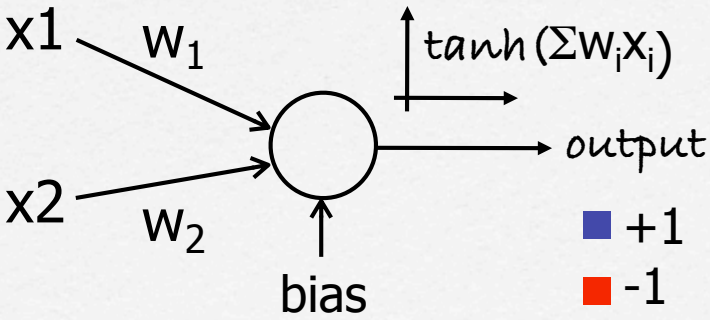


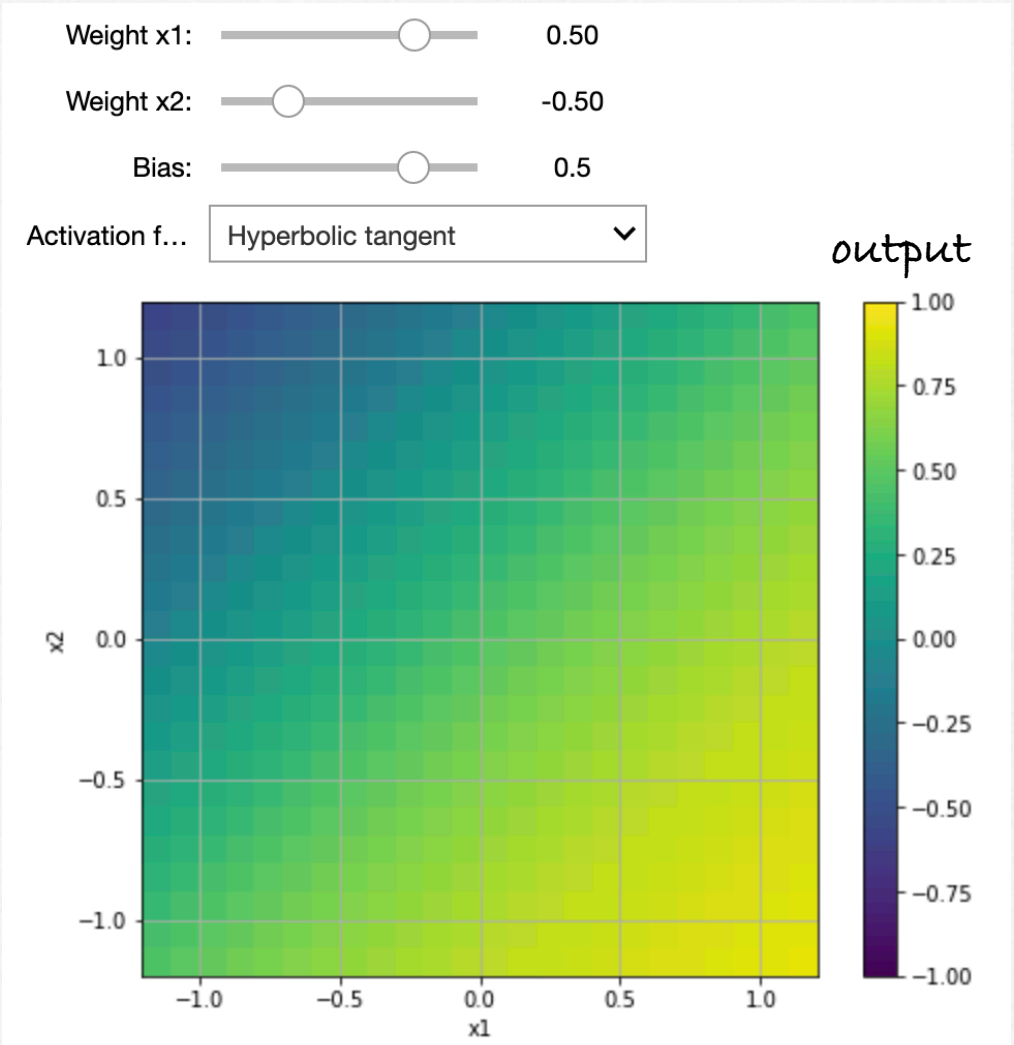
HE
IG

APE 2025

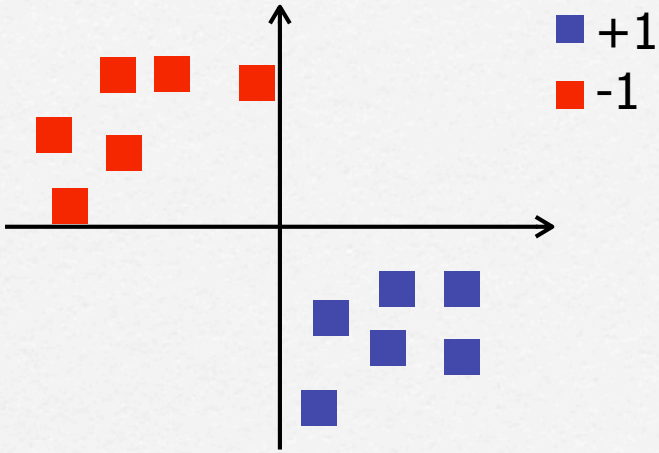
A two-class classifier example (1)

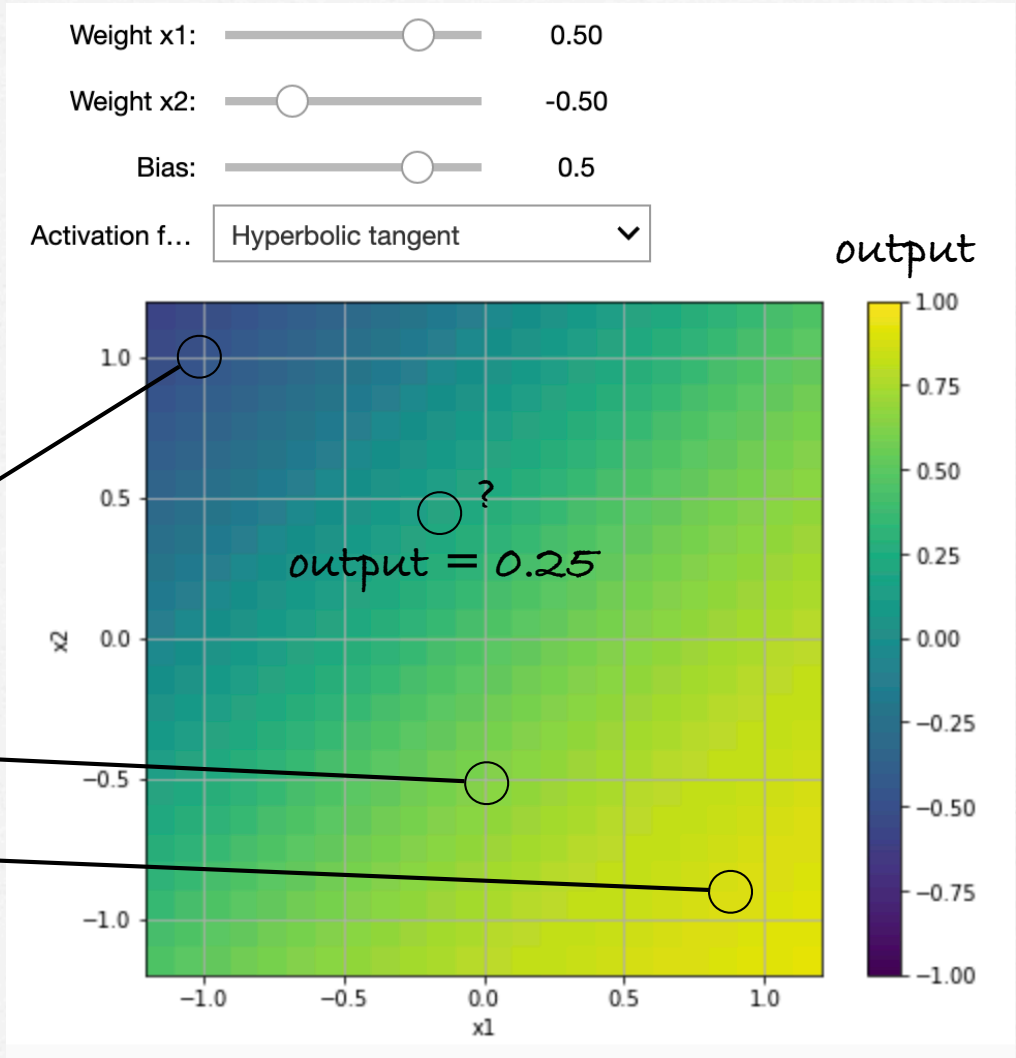




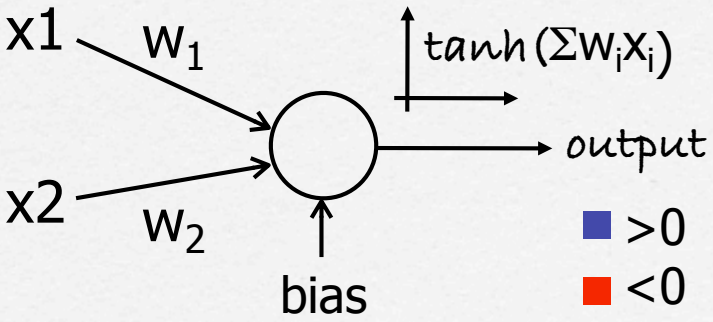


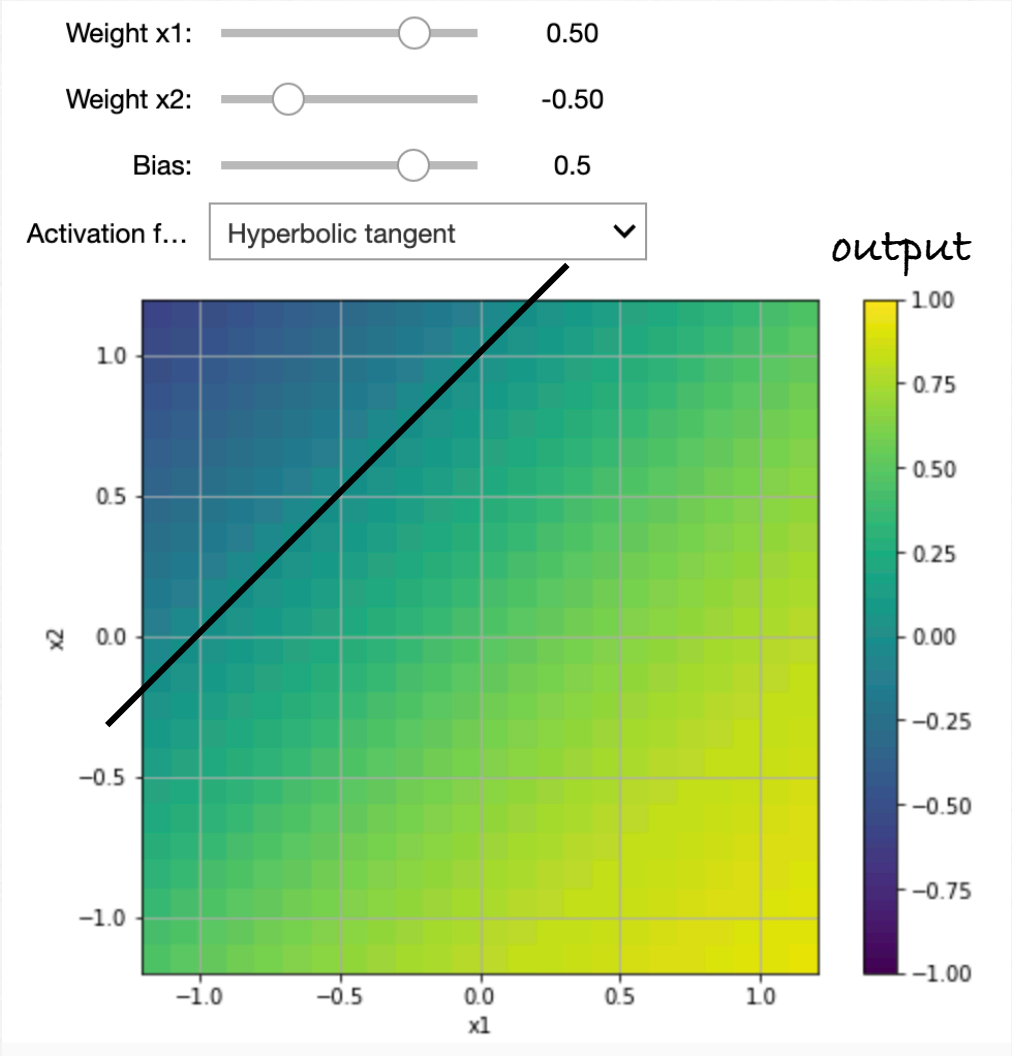
A two-class classifier example (2)





A two-class classifier example (3)

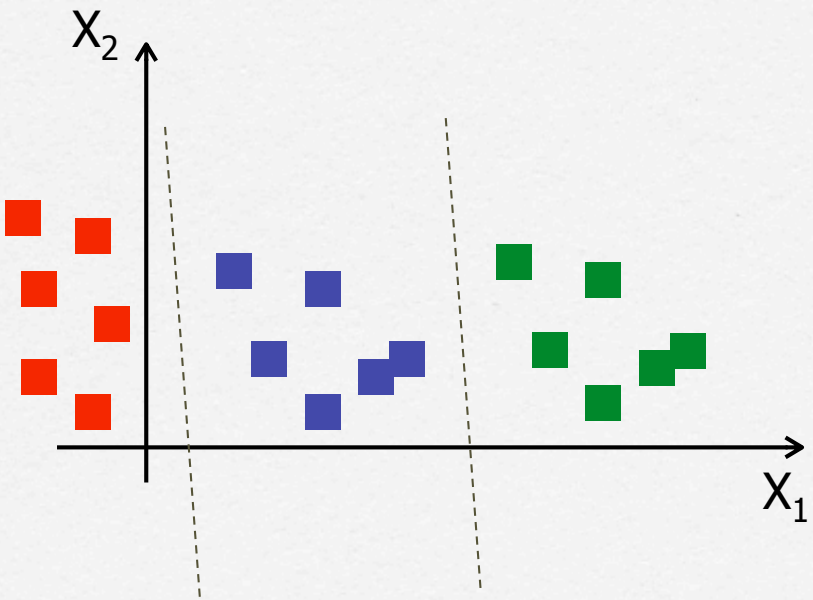


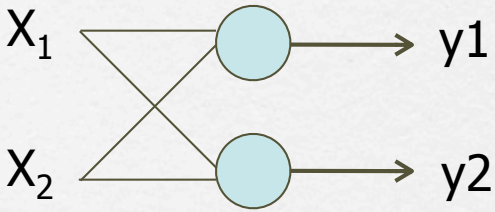


Two-class classification threshold

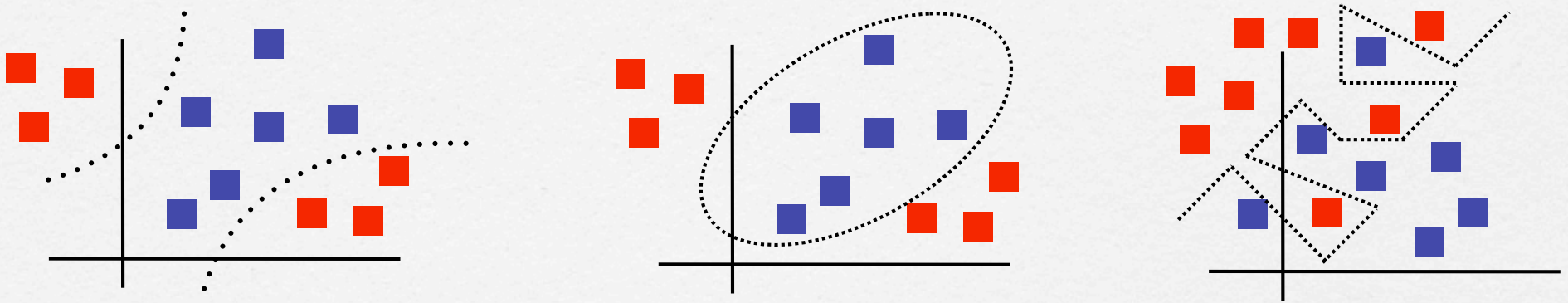
- A threshold has to be fixed by the engineer
- If we have a validation dataset, we can evaluate the effect of such a threshold and decide the best value
- In general, there is a trade-off between the true positive rate (sensitivity) and the false positive rate (1 - specificity) of such classifier.

A linearly separable three-class problem





Non-linearly separable classes



- It is not possible to draw a line to separate these two classes, e.g., red points from blue points.

Made with Whimsical