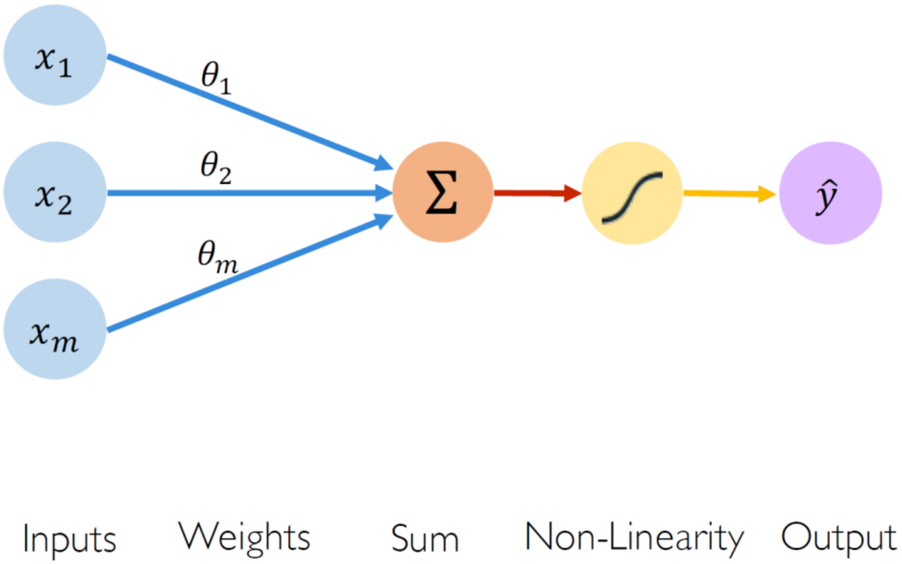
(Artificial) Neural Networks

Industrial AI Lab.
Prof. Seungchul Lee

Perceptron: Forward Propagation



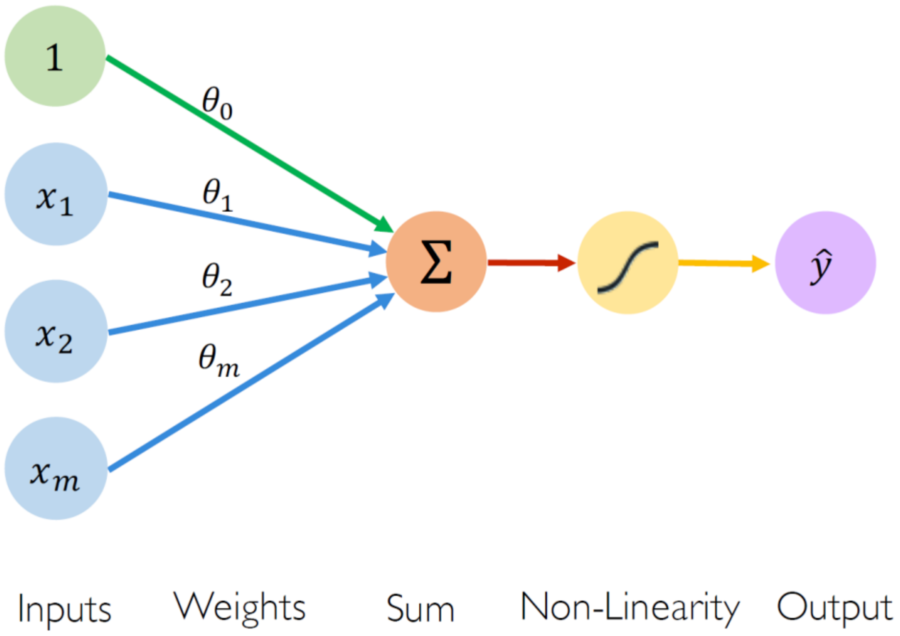
Output

Linear combination of inputs

$$\hat{y} = g\left(\sum_{i=1}^{m} x_i\,\theta_i\right)$$

Non-linear activation function

Perceptron: Forward Propagation



Output

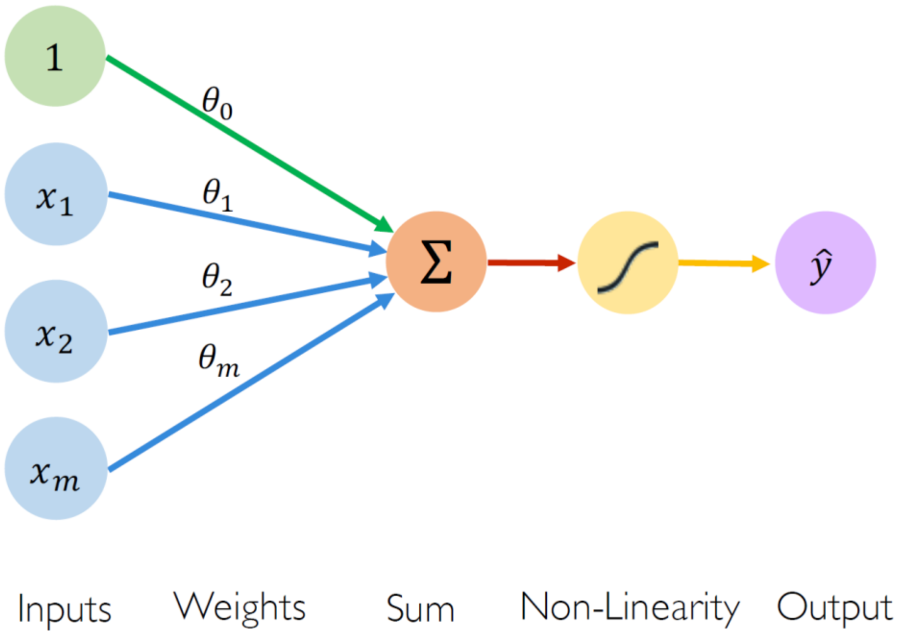
Linear combination of inputs

$$\hat{y} = g\left(\theta_0 + \sum_{i=1}^{m} x_i \, \theta_i\right)$$

Non-linear activation function

Bias

Perceptron: Forward Propagation

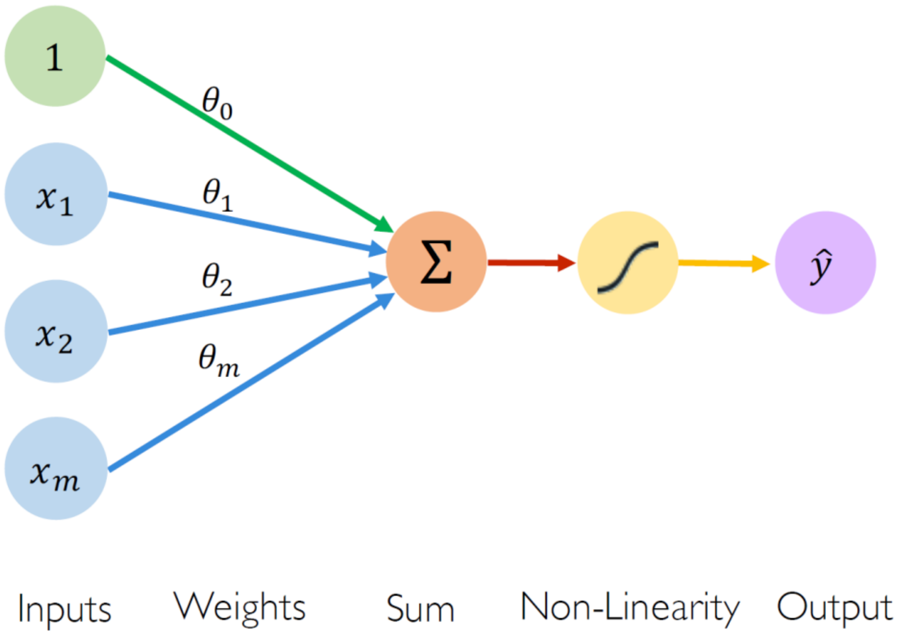


$$\hat{y} = g\left(\theta_0 + \sum_{i=1}^{m} x_i \, \theta_i\right)$$

$$\hat{y} = g\left(\theta_0 + \boldsymbol{X}^T \boldsymbol{\theta}\right)$$

where: $\boldsymbol{X} = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}$ and $\boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_m \end{bmatrix}$

Perceptron: Forward Propagation

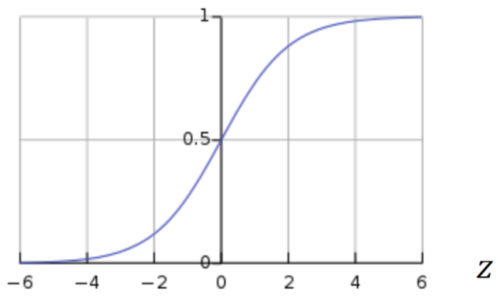


Activation Functions

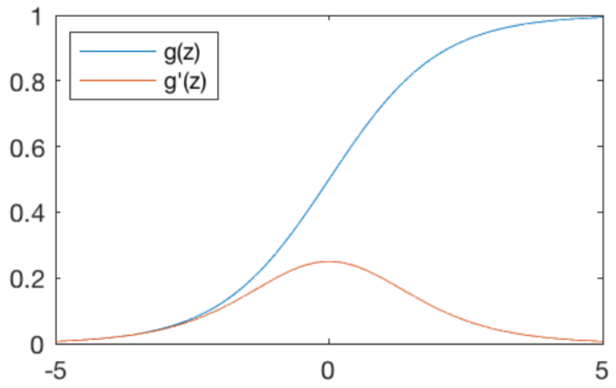
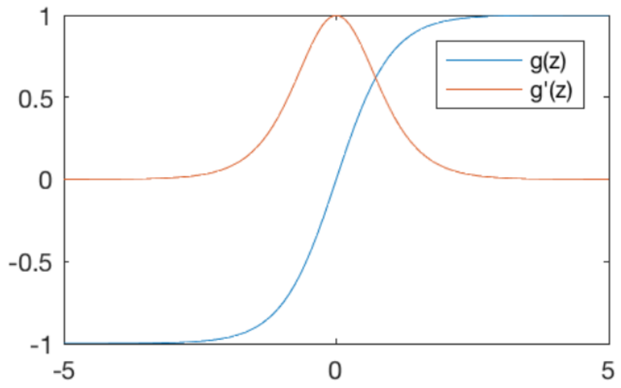
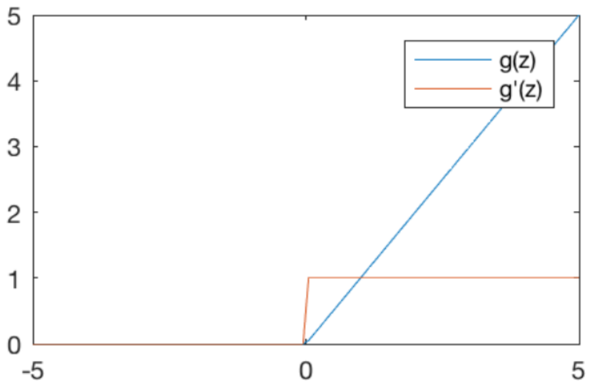
$$\hat{y} = g\left(\theta_0 + \boldsymbol{X}^T \boldsymbol{\theta}\right)$$

- Example: sigmoid function

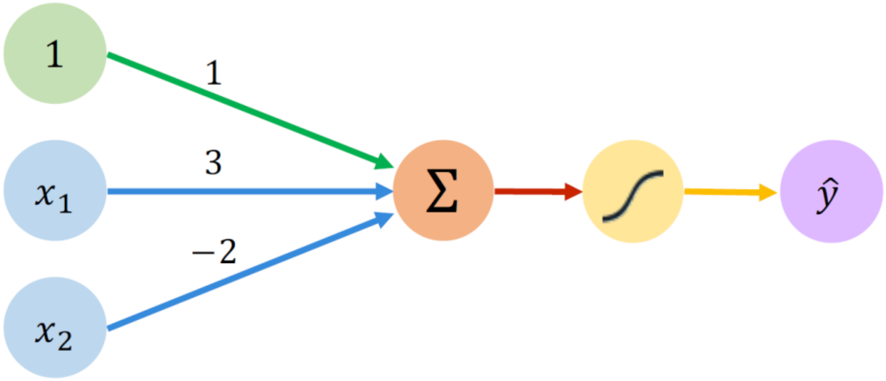
$$g(z) = \sigma(z) = \frac{1}{1 + e^{-z}}$$



Common Activation Functions

Sigmoid Function	Hyperbolic Tangent	Rectified Linear Unit (ReLU)
		
$g(z) = \dfrac{1}{1+e^{-z}}$	$g(z) = \dfrac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$	$g(z) = \max(0, z)$
$g'(z) = g(z)(1-g(z))$	$g'(z) = 1 - g(z)^2$	$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$
`tf.nn.sigmoid(z)`	`tf.nn.tanh(z)`	`tf.nn.relu(z)`

Perceptron: Example



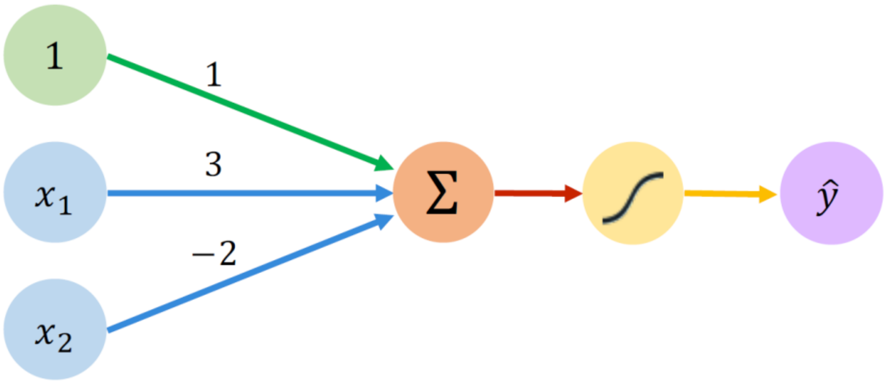
We have: $\theta_0 = 1$ and $\boldsymbol{\theta} = \begin{bmatrix} 3 \\ -2 \end{bmatrix}$

$$\hat{y} = g(\theta_0 + \boldsymbol{X}^T\boldsymbol{\theta})$$

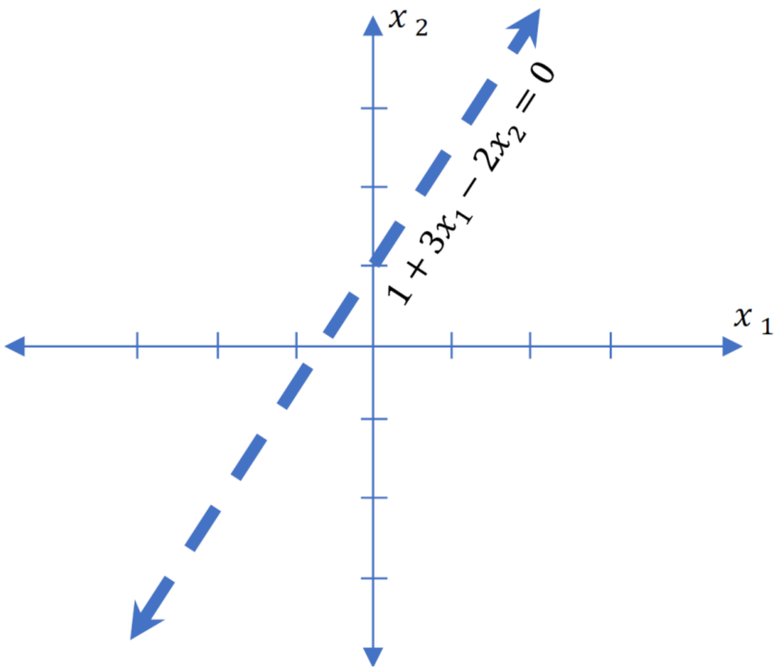
$$= g\left(1 + \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} 3 \\ -2 \end{bmatrix}\right)$$

$$\hat{y} = g(\underbrace{1 + 3x_1 - 2x_2}_{\text{This is just a line in 2D!}})$$

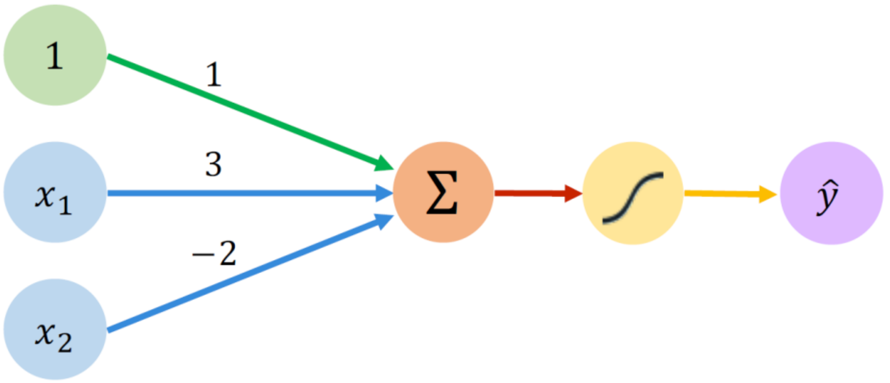
Perceptron: Example



$$\hat{y} = g(1 + 3x_1 - 2x_2)$$



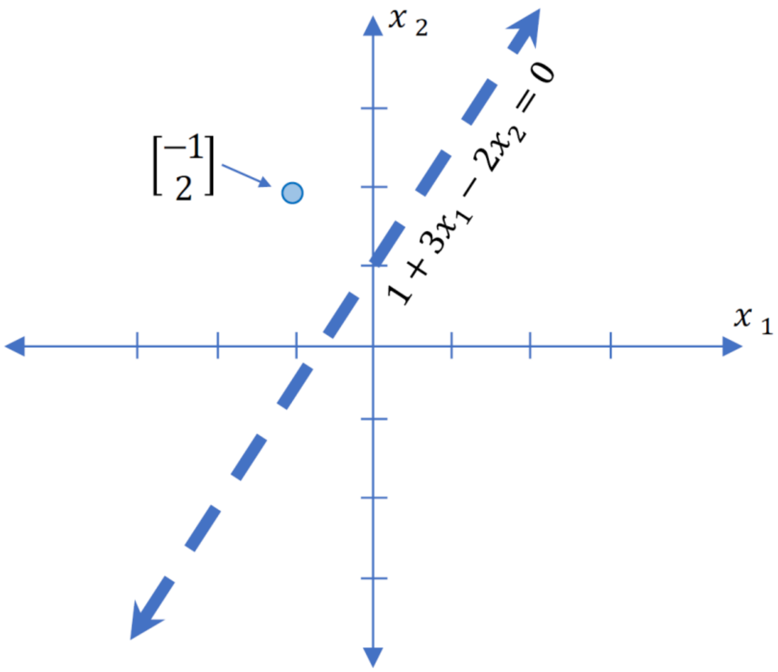
Perceptron: Example



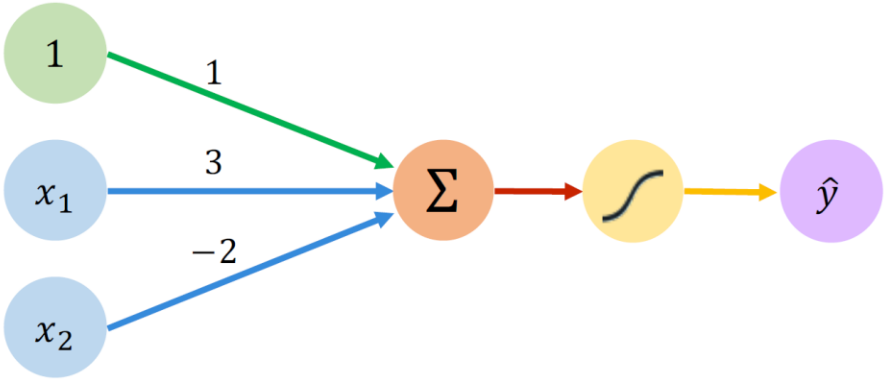
Assume we have input: $\boldsymbol{X} = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$

$$\begin{aligned} \hat{y} &= g\big(1 + (3 * -1) - (2 * 2)\big) \\ &= g(-6) \approx 0.002 \end{aligned}$$

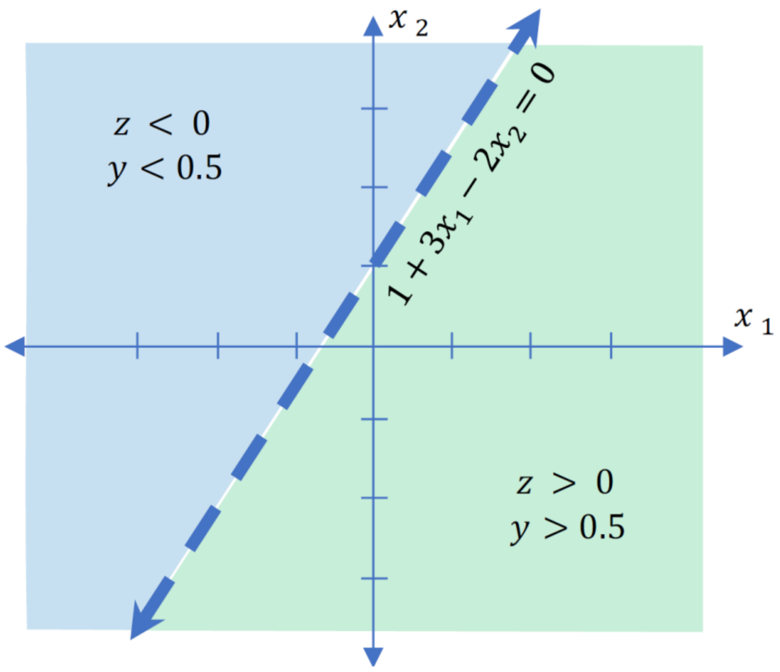
$$\hat{y} = g(1 + 3x_1 - 2x_2)$$



Perceptron: Example



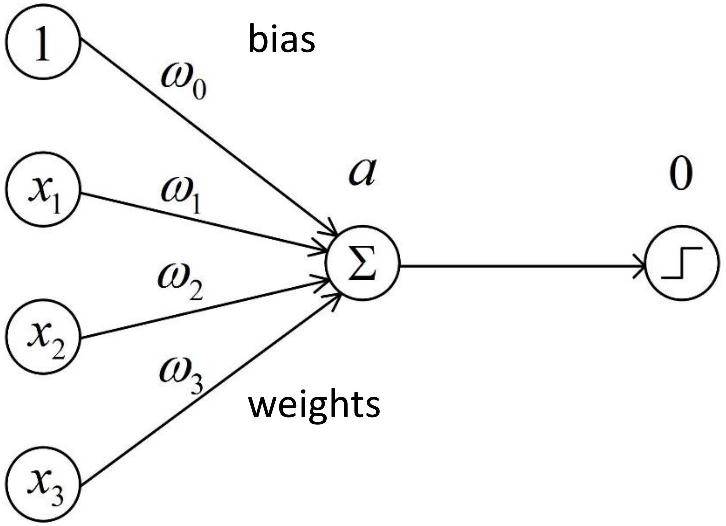
$\hat{y} = g(1 + 3x_1 - 2x_2)$



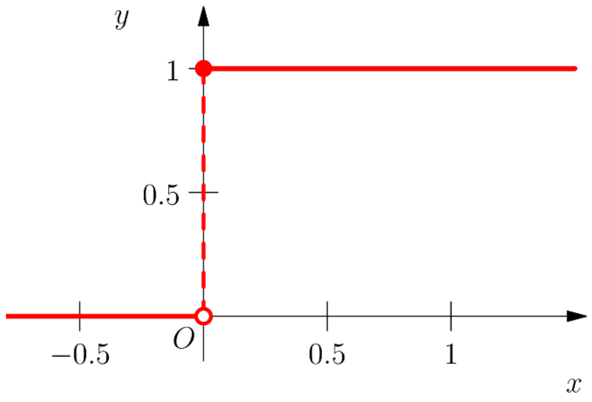
Source: 6.S191 Intro. to Deep Learning at MIT

Artificial Neural Networks: Perceptron

- Perceptron for $h(\theta)$ or $h(\omega)$
  - Neurons compute the weighted sum of their inputs
  - A neuron is activated or fired when the sum a is positive



$$a = \omega_0 + \omega_1 x_1 + \cdots$$
$$o = \sigma(\omega_0 + \omega_1 x_1 + \cdots)$$

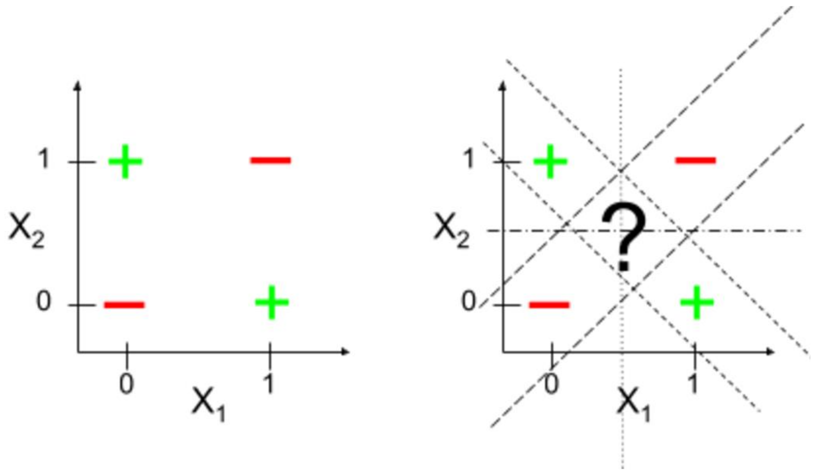


- A step function is not differentiable
- One layer is often not enough

XOR Problem

- Minsky-Papert Controversy on XOR
  - not linearly separable
  - Limitation of perceptron

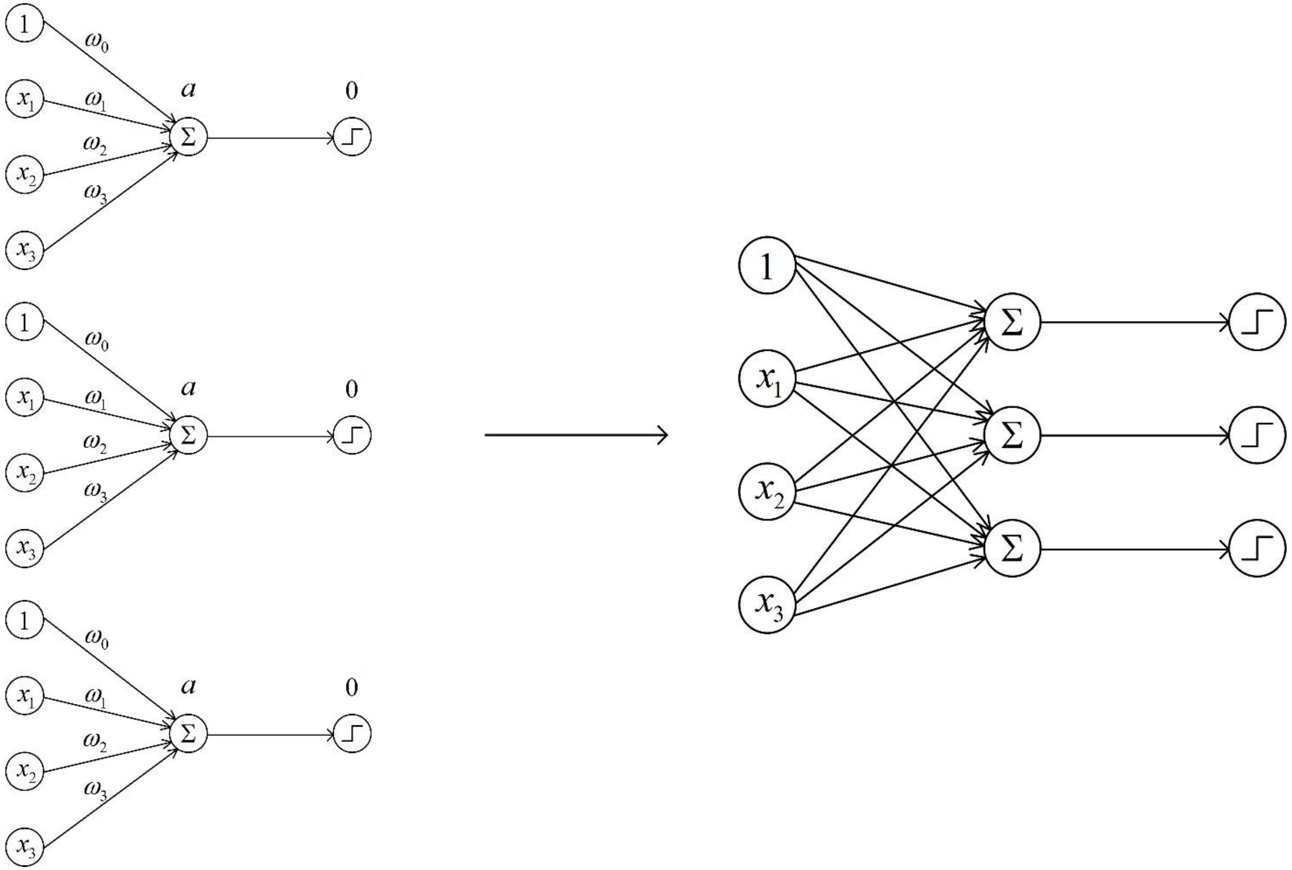
x_1	x_2	x_1 **XOR** x_2
0	0	0
0	1	1
1	0	1
1	1	0



- Single neuron = one linear classification boundary

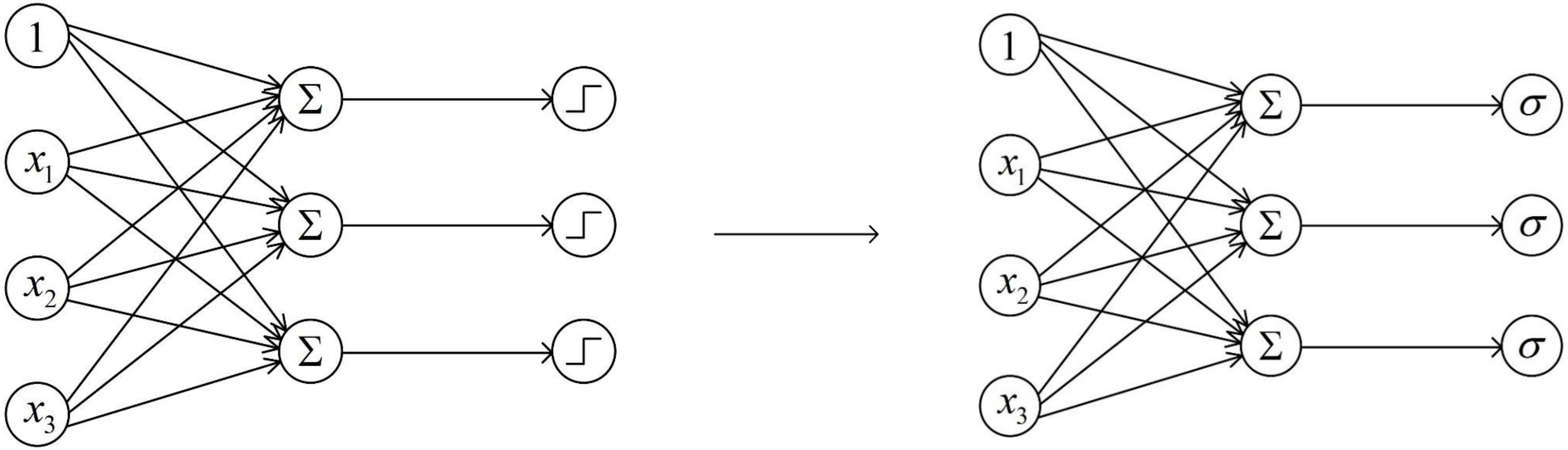
Artificial Neural Networks: MLP

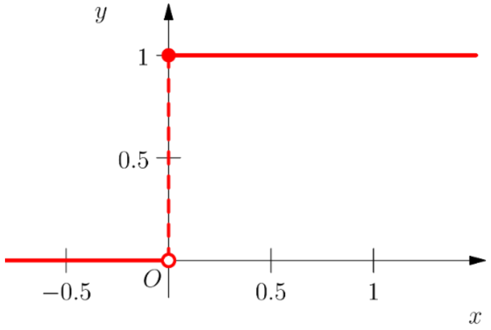
- Multi-layer Perceptron (MLP) = Artificial Neural Networks (ANN)
  - Multi neurons = multiple linear classification boundaries

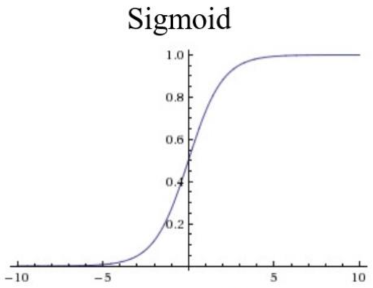


Artificial Neural Networks: Activation Func.

- Differentiable non-linear activation function



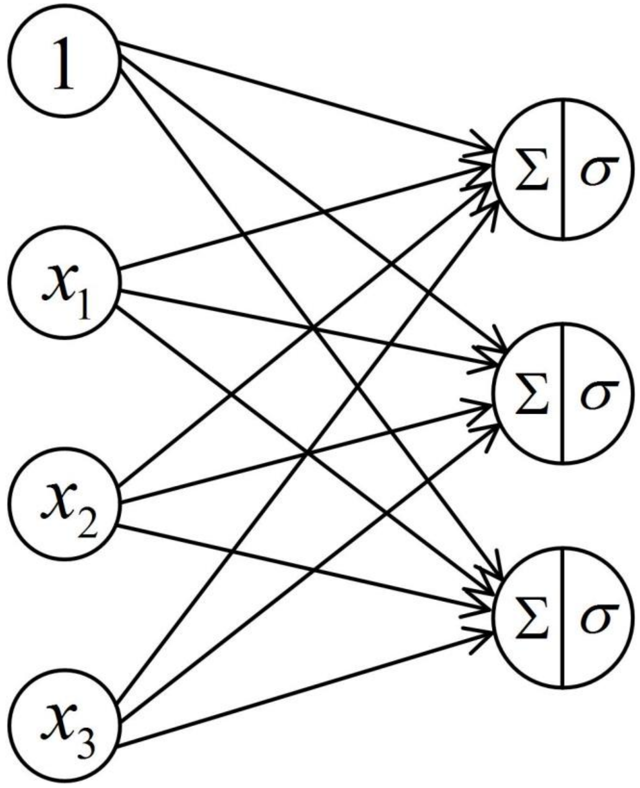




$$g(x) = \frac{1}{1 + e^{-x}}$$

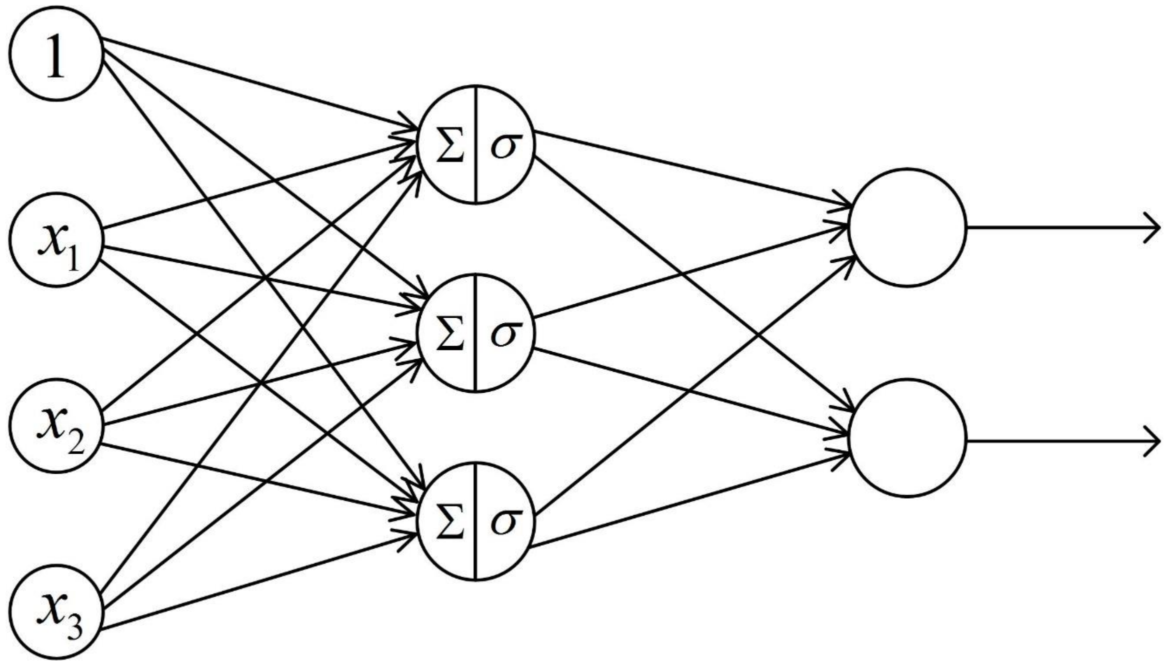
Artificial Neural Networks

- In a compact representation



Artificial Neural Networks

- Multi-layer perceptron
  - Features of features
  - Mapping of mappings

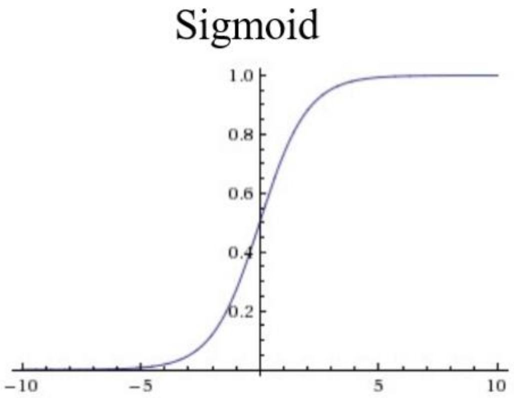


ANN: Transformation

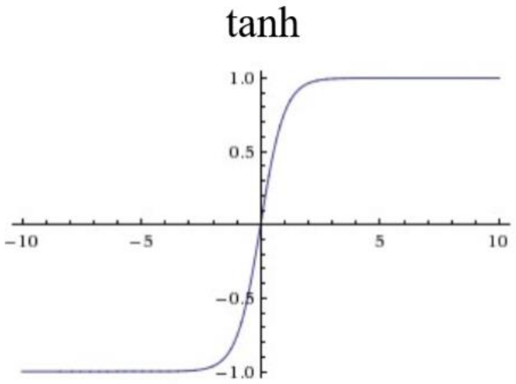
- Affine (or linear) transformation and nonlinear activation layer (notations are mixed: $g = \sigma, \omega = \theta, \omega_0 = b$)

$$o(x) = g\left(\theta^T x + b\right)$$

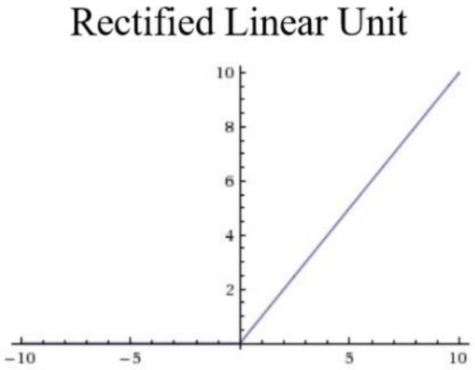
- Nonlinear activation functions ($g = \sigma$)



$$g(x) = \frac{1}{1 + e^{-x}}$$



$$g(x) = \tanh(x)$$



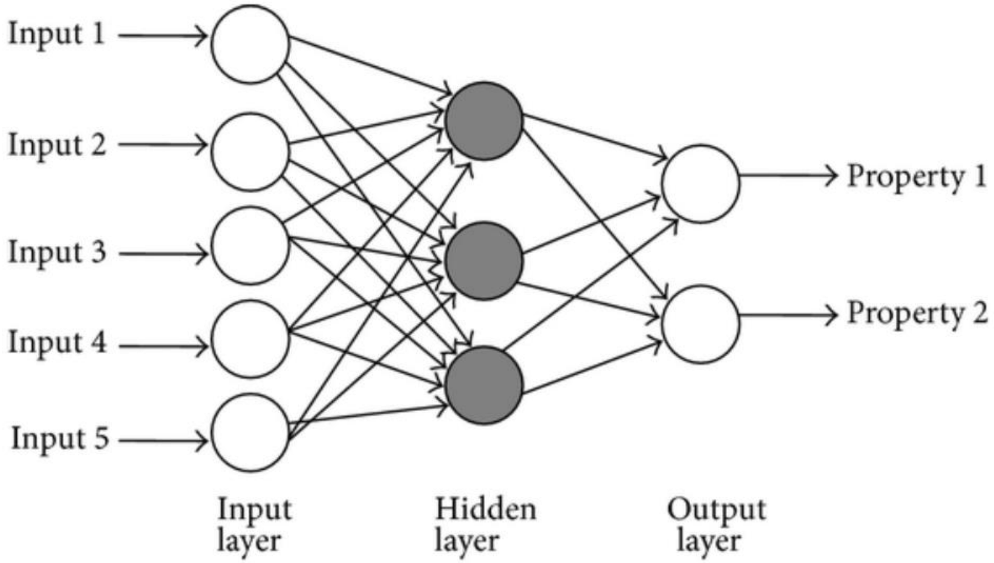
$$g(x) = \max(0, x)$$

ANN: Architecture

- A single layer is not enough to be able to represent complex relationship between input and output

$\Longrightarrow$ perceptron with many layers and units

$$o_2 = \sigma_2\left(\theta_2^T o_1 + b_2\right) = \sigma_2\left(\theta_2^T \sigma_1\left(\theta_1^T x + b_1\right) + b_2\right)$$



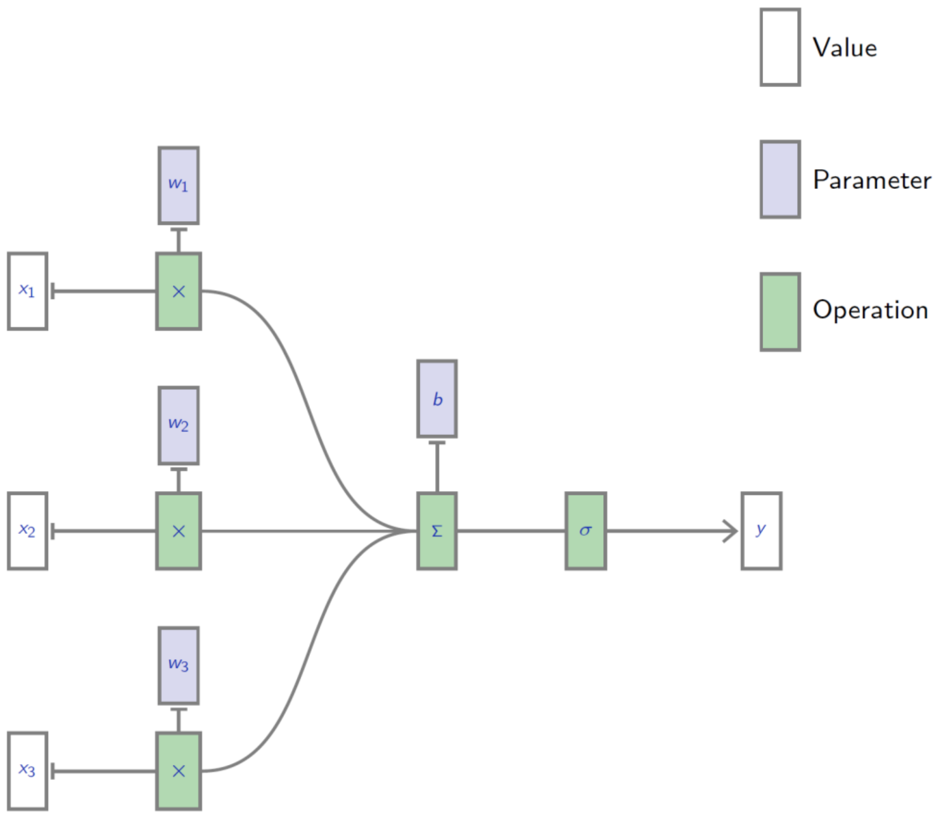
- The perceptron classification rule boils down to

$$f(x) = \sigma(w \cdot x + b).$$

- For neural networks, the function σ that follows a linear operator is called the activation function.

- We can also use tensor operations, as in

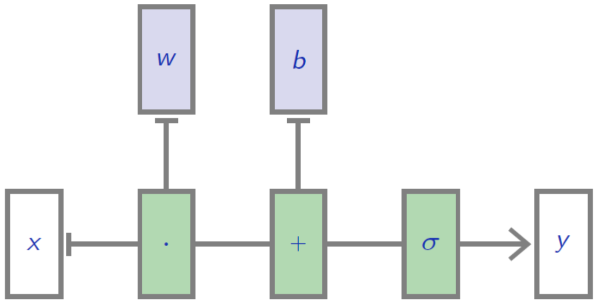
$$f(x) = \sigma(w \cdot x + b).$$



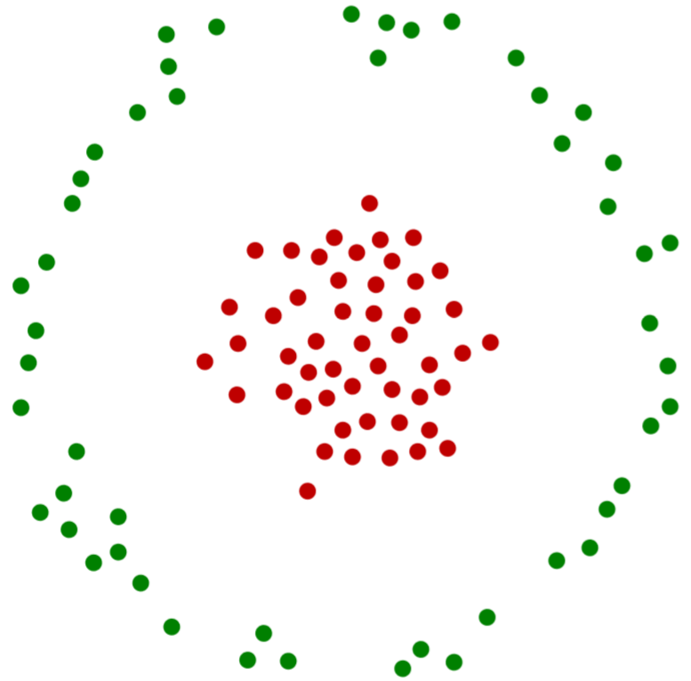
Source: Dr. Francois Fleuret at EPFL

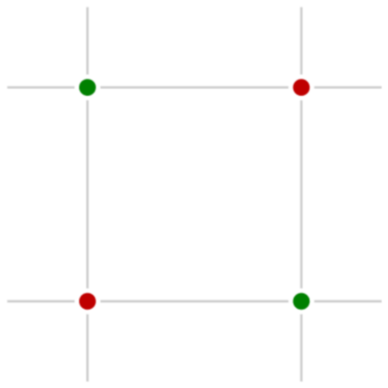
- We can represent this “neuron” as follows:

$$f(x) = \sigma(w \cdot x + b).$$



- The main weakness of linear predictors is their lack of capacity. For classification, the populations have to be linearly separable.



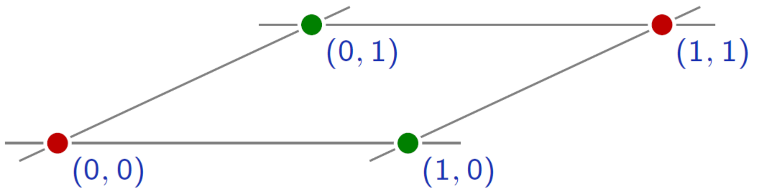


"xor"

Source: Dr. Francois Fleuret at EPFL

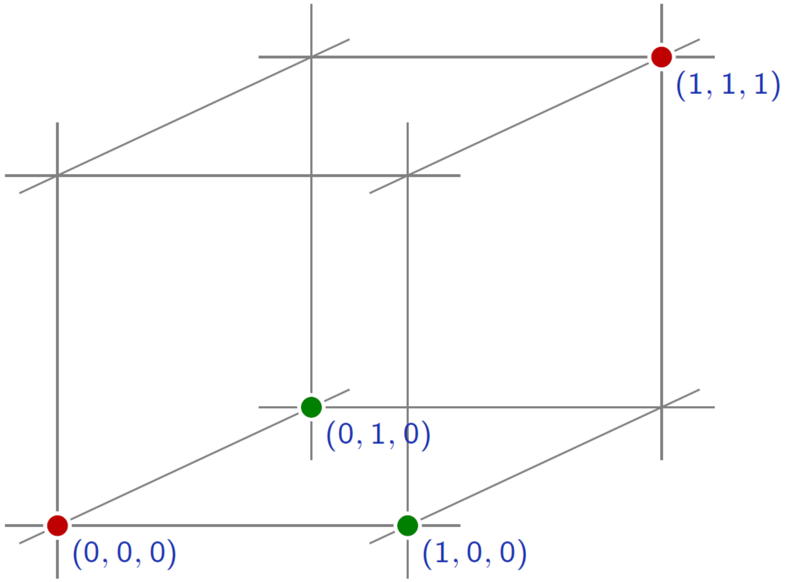
- The xor example can be solved by pre-processing the data to make the two populations linearly separable.

$$\Phi : (x_u, x_v) \mapsto (x_u, x_v, x_u x_v).$$



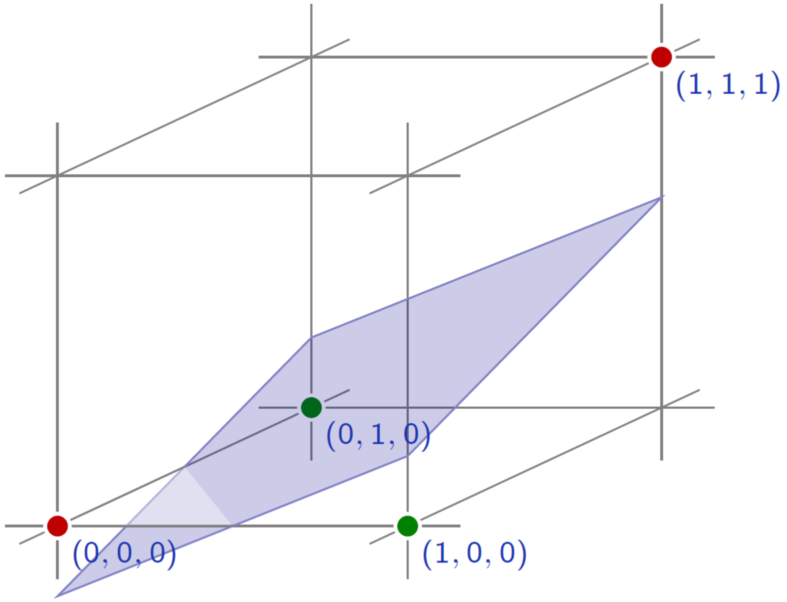
- The xor example can be solved by pre-processing the data to make the two populations linearly separable.

$$\Phi : (x_u, x_v) \mapsto (x_u, x_v, x_u x_v).$$



- The xor example can be solved by pre-processing the data to make the two populations linearly separable.

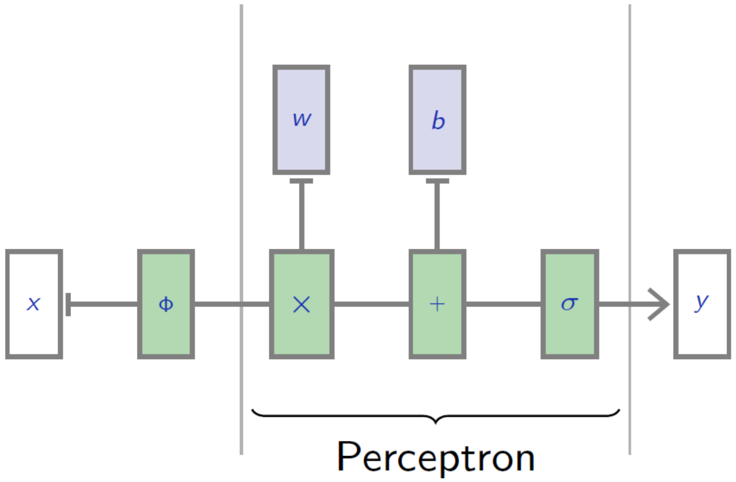
$$\Phi : (x_u, x_v) \mapsto (x_u, x_v, x_u x_v).$$



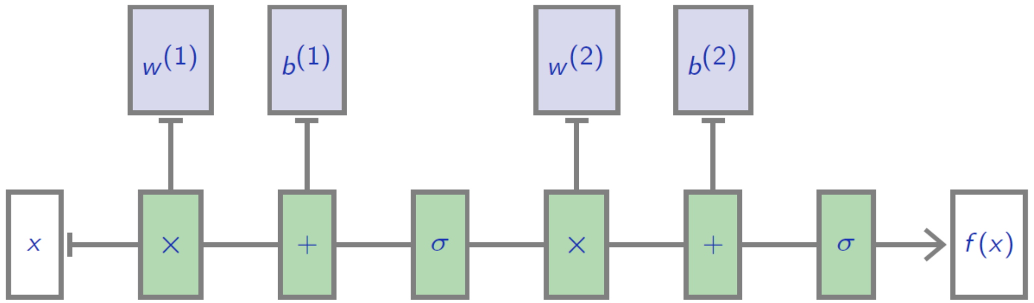
Source: Dr. Francois Fleuret at EPFL

- Nonlinear mapping + neuron

$$\Phi : (x_u, x_v) \mapsto (x_u, x_v, x_u x_v).$$

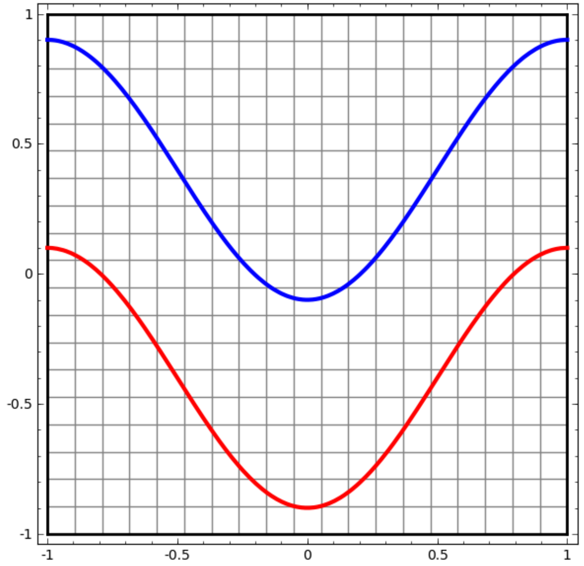


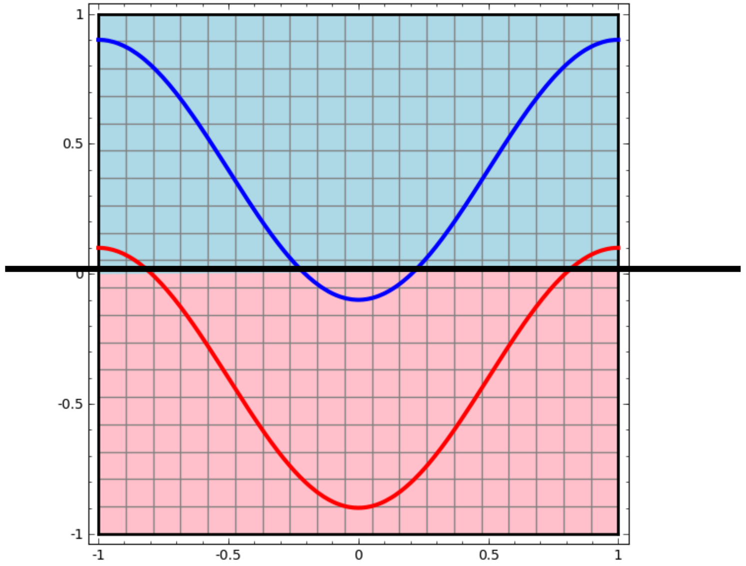
- Nonlinear mapping can be represented by another neurons
- We can generalize an MLP



Linear Classifier

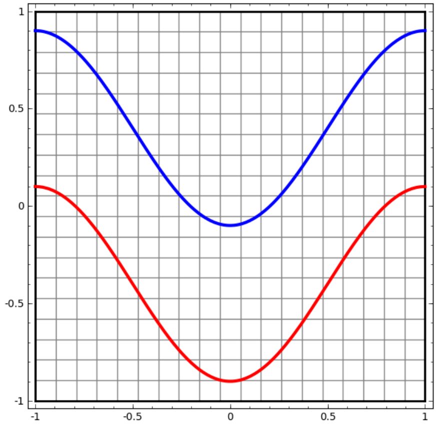
- Perceptron tries to separate the two classes of data by dividing them with a line

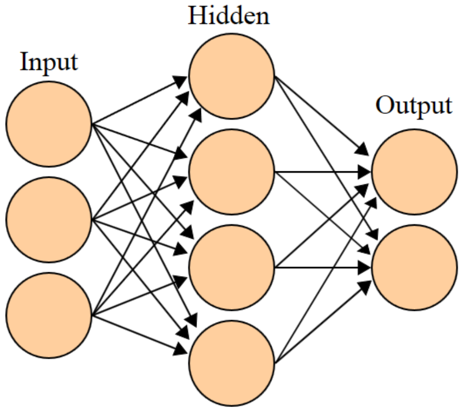


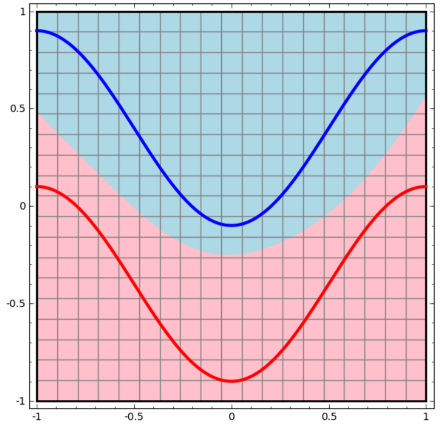


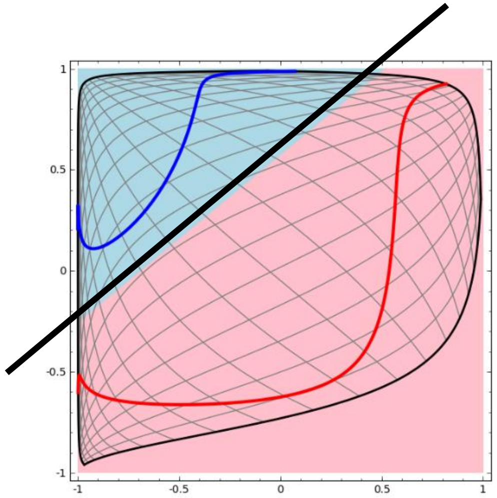
Neural Networks

- The hidden layer learns a representation so that the data gets linearly separable





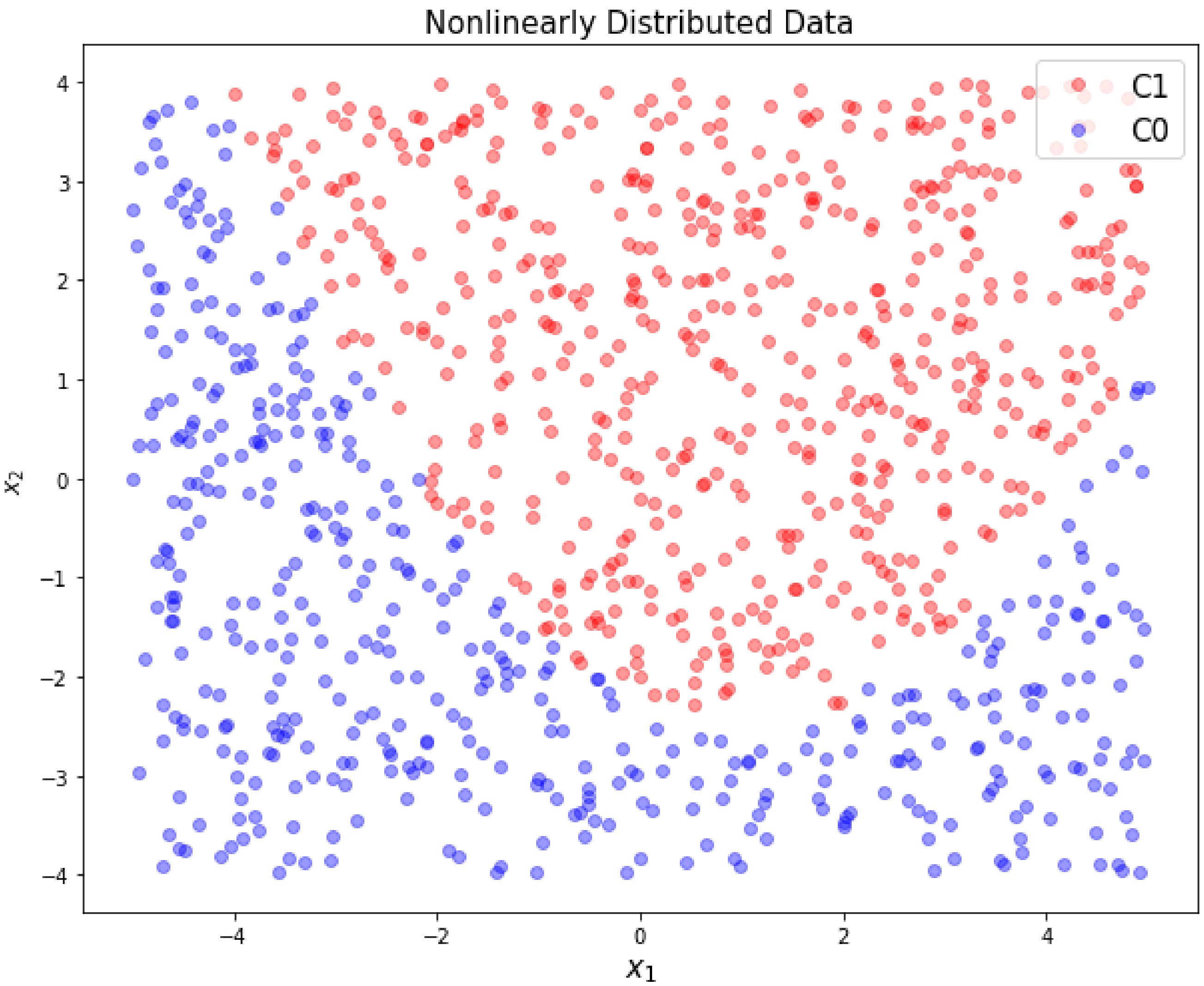




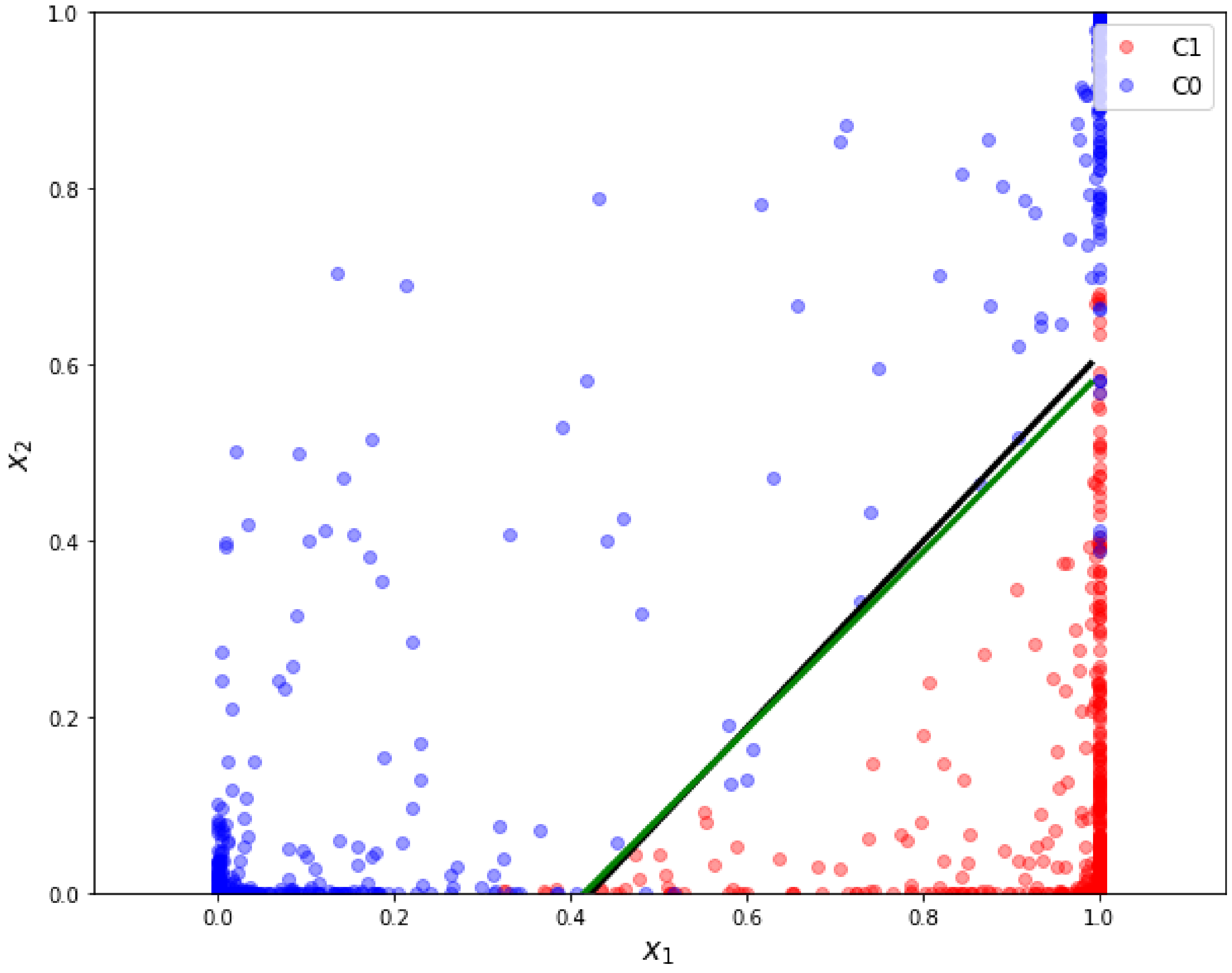
Understanding a Network's Behavior

- Understanding what is happening in a deep architectures after training is complex and the tools we have at our disposal are limited.
- We can look at
  - the network's parameters, filters as images,
  - internal activations as images,
  - distributions of activations on a population of samples,
  - derivatives of the response(s) wrt the input,
  - maximum-response synthetic samples,
  - adversarial samples.

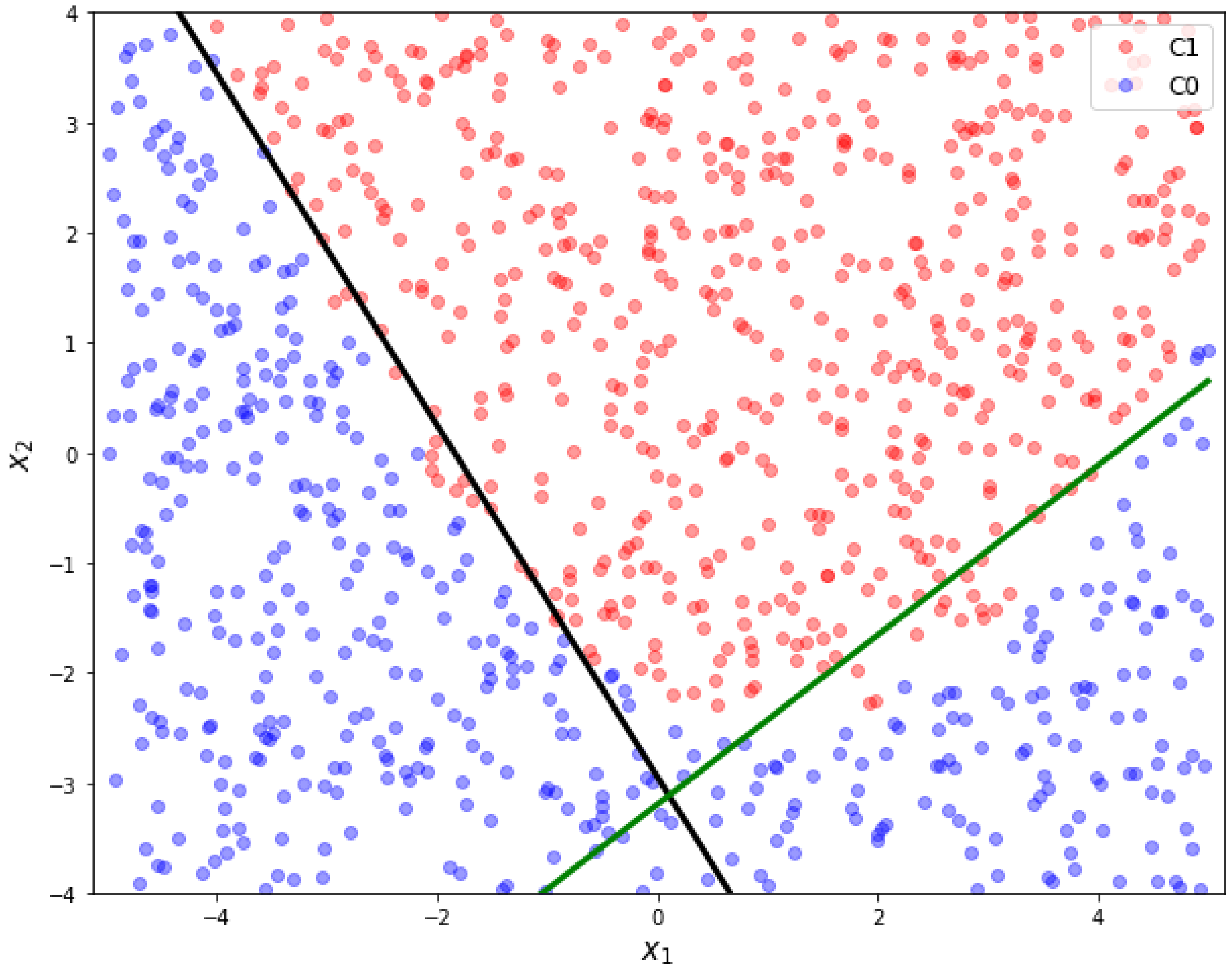
Nonlinearly Distributed Data

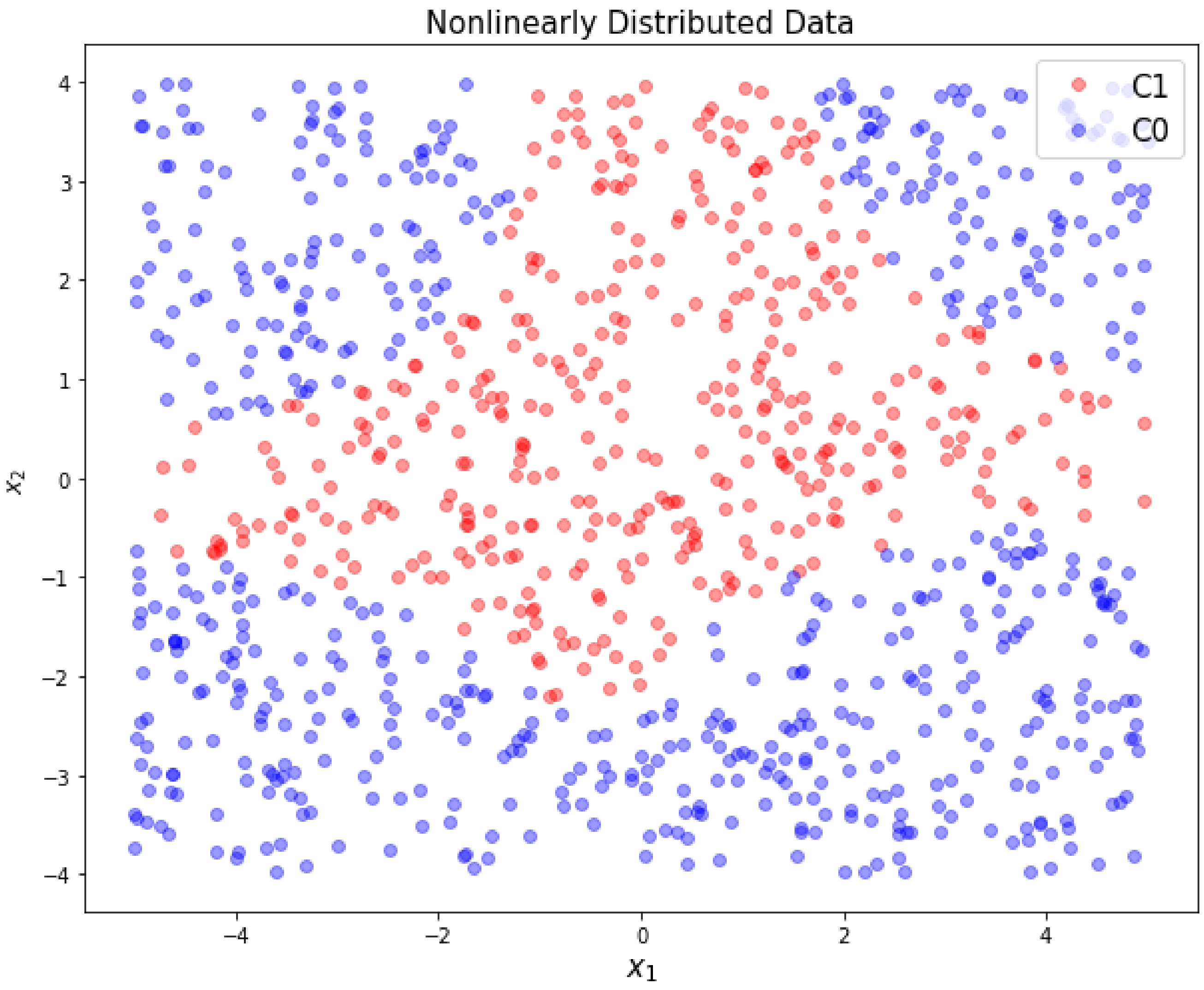


Multi Layers

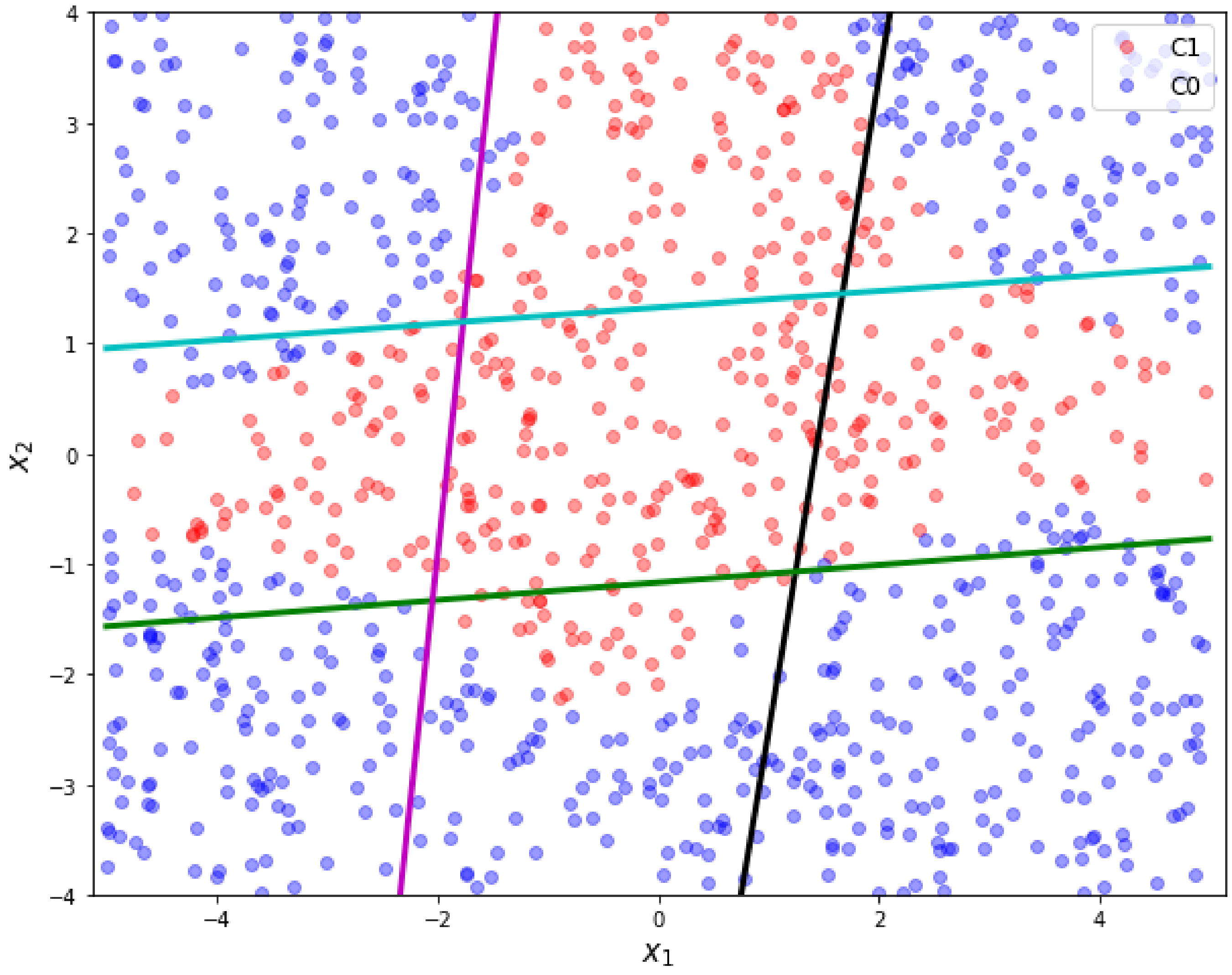


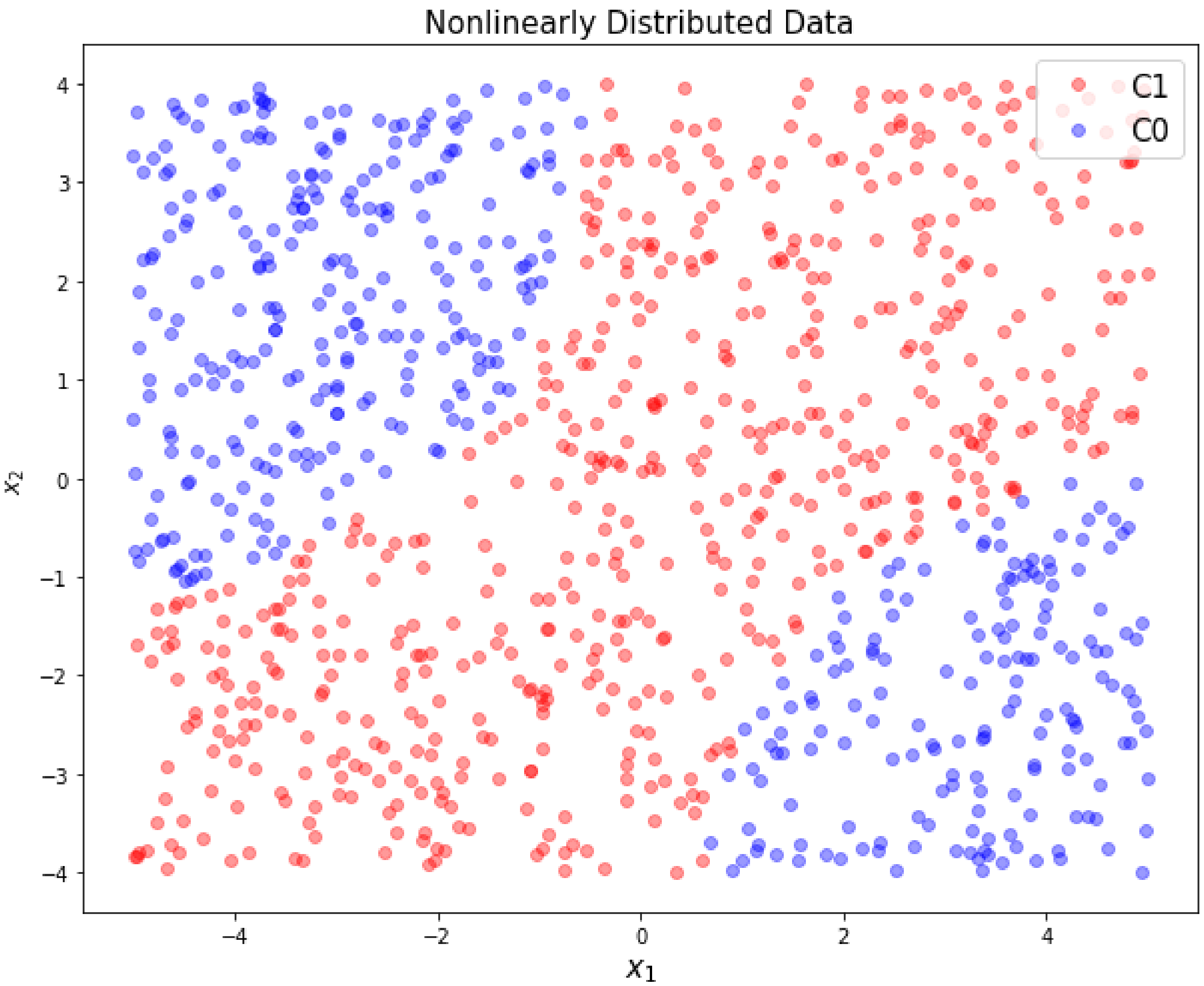
Multi Layers

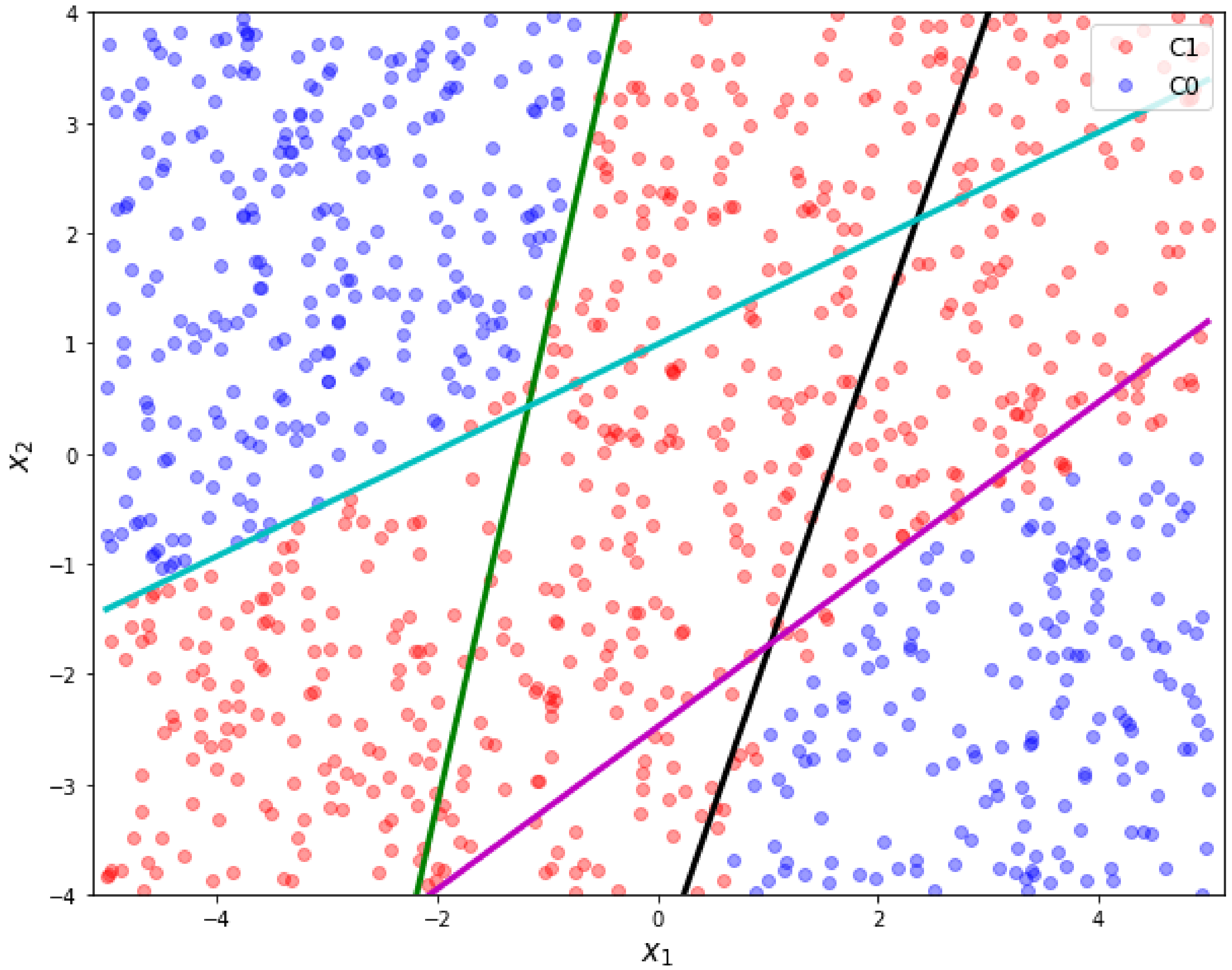




POSTECH


C1
C0
x_1
x_2
−4
−2
0
2
4
−4
−3
−2
−1
0
1
2
3
4


Nonlinearly Distributed Data
C1
C0
x_1
x_2
−4
−2
0
2
4
−4
−3
−2
−1
0
1
2
3
4


C1
C0
x_1
x_2
−4
−2
0
2
4
−3
−1
1
3