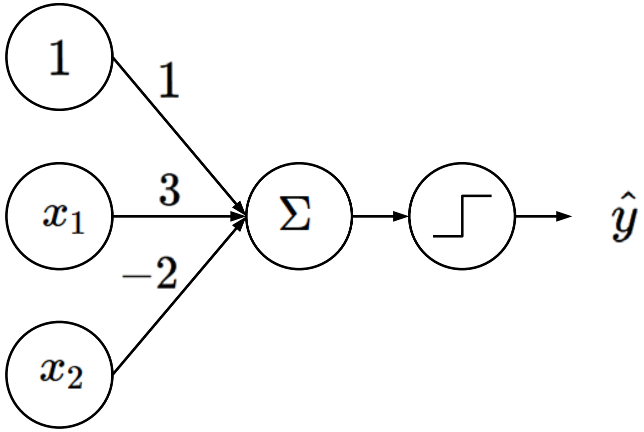
(Artificial) Neural Networks: From Perceptron to MLP

Industrial AI Lab.

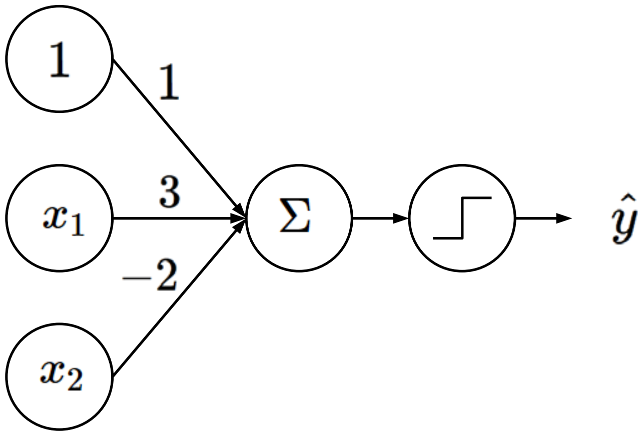
Prof. Seungchul Lee

Perceptron: Example

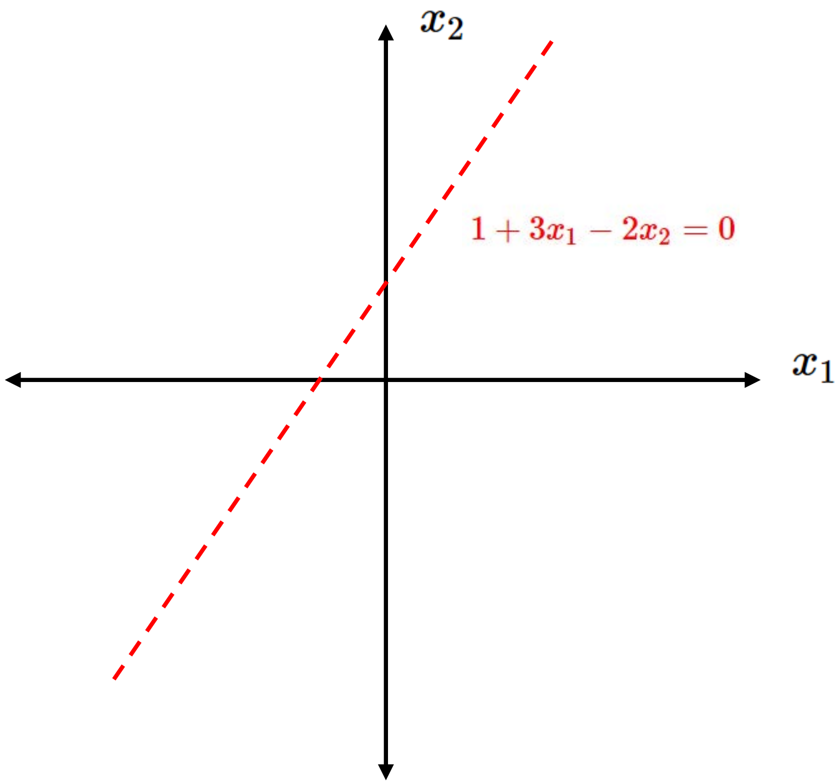


$$
\begin{aligned}
\hat{y} &= g\left(\omega_0 + X^T\omega\right) \\
&= g\left(1 + \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} 3 \\ -2 \end{bmatrix}\right) \\
&= g\left(1 + 3x_1 - 2x_2\right)
\end{aligned}
$$

Perceptron: Example

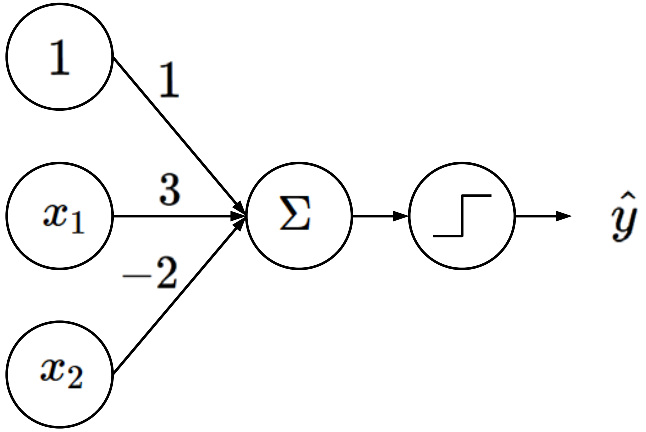


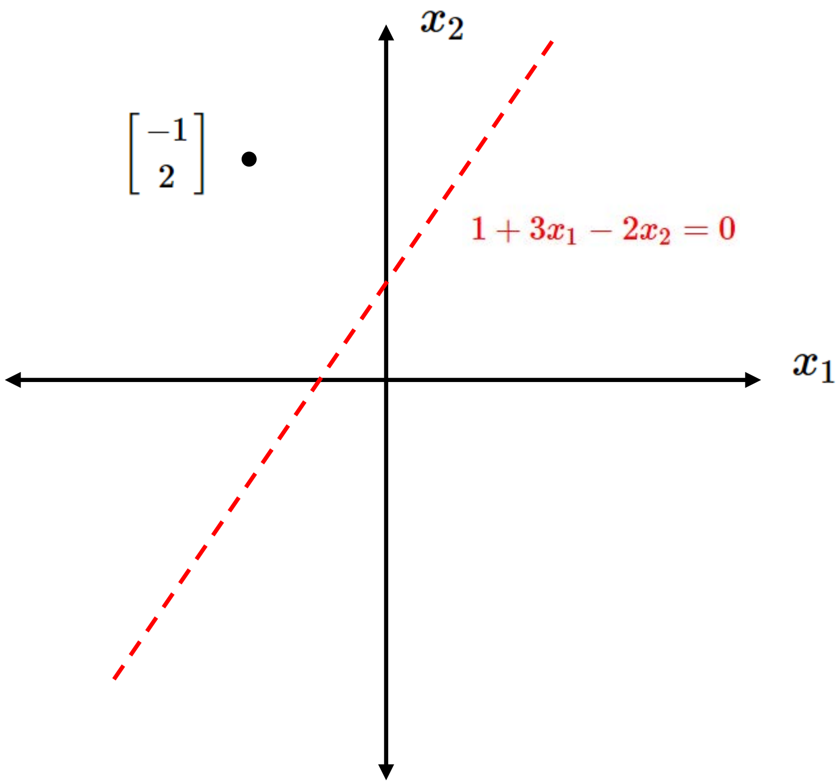
$$\hat{y} = g\left(1 + 3x_1 - 2x_2\right)$$



Perceptron: Example

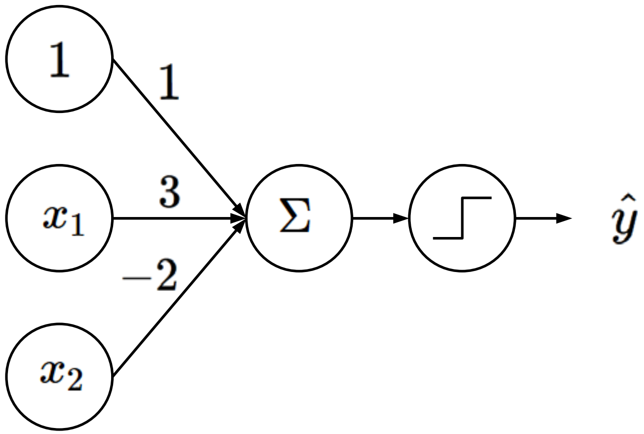
$$\hat{y} = g\left(1 + 3x_1 - 2x_2\right)$$



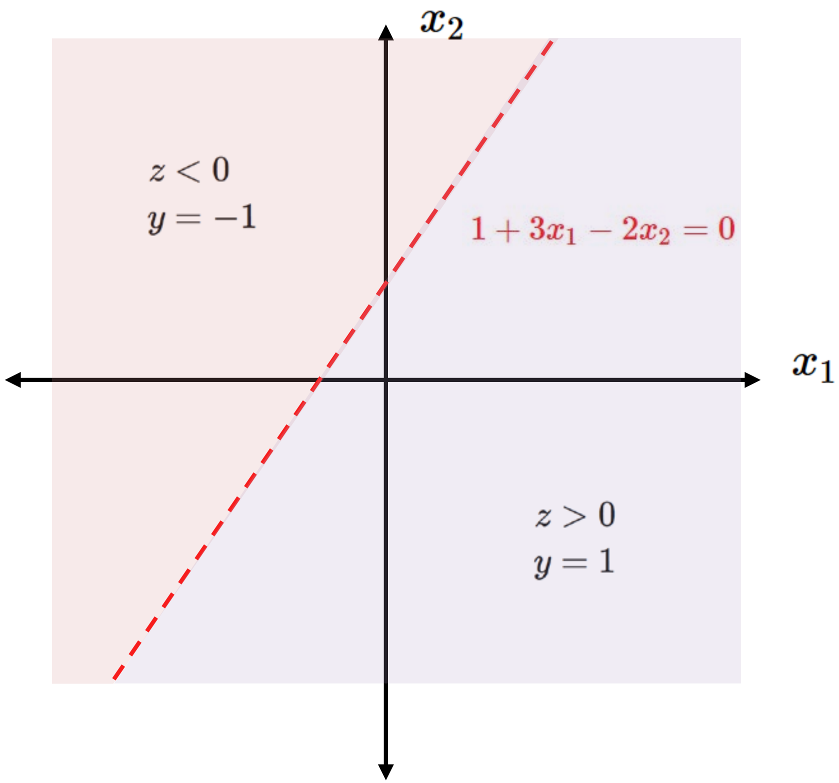


$$\hat{y} = g\left(1 + 3 \times (-1) - 2 \times 2\right) = g(-6) = -1$$

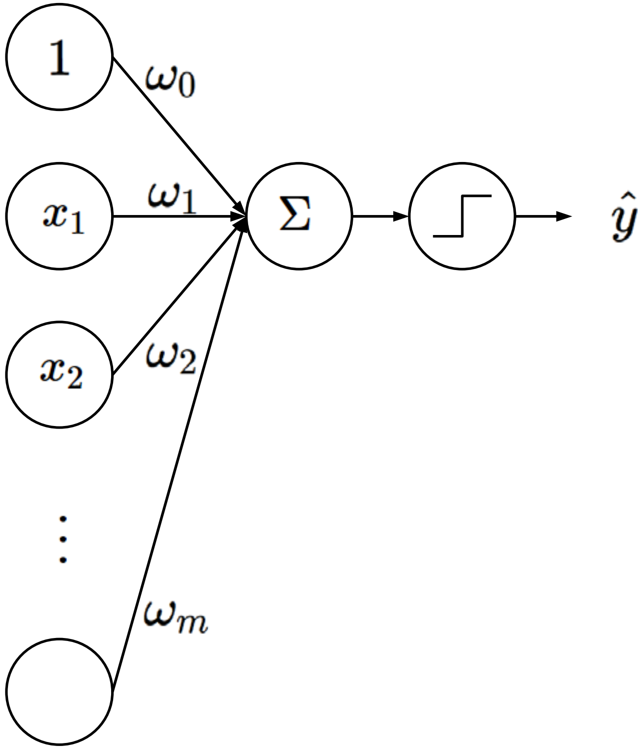
Perceptron: Example



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



Perceptron: Forward Propagation



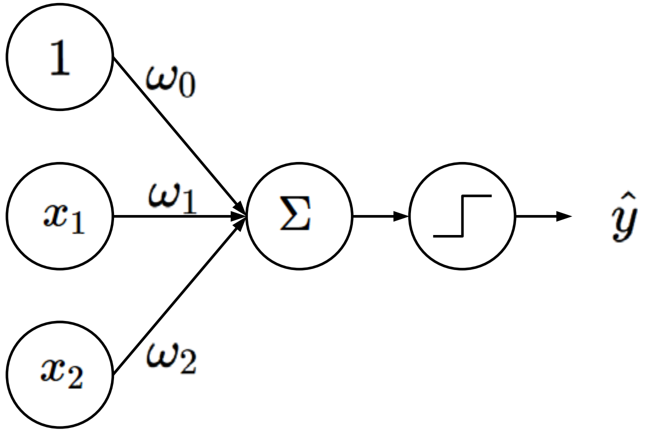
$$\hat{y} = g\left(\omega_0 + X^T \omega\right)$$

$$= g\left(\omega_0 + \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}^T \begin{bmatrix} \omega_1 \\ \vdots \\ \omega_m \end{bmatrix}\right)$$

From Perceptron to MLP

Artificial Neural Networks: Perceptron

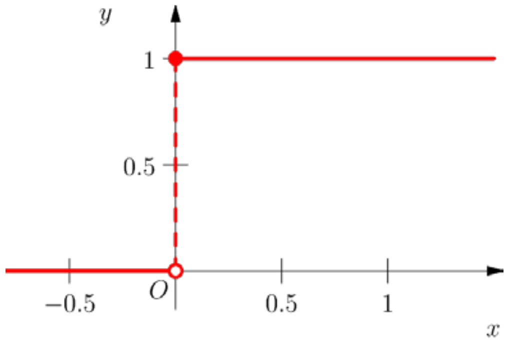
- Perceptron for $h(\theta)$ or $h(\omega)$
  - Neurons compute the weighted sum of their inputs
  - A neuron is activated or fired when the sum a is positive



$$a = \omega_0 + \omega_1 x_1 + \omega_2 x_2$$

$$\hat{y} = g(a) = \begin{cases} 1 & a > 0 \\ 0 & \text{otherwise} \end{cases}$$

- A step function is not differentiable
- One neuron is often not enough
  - One hyperplane

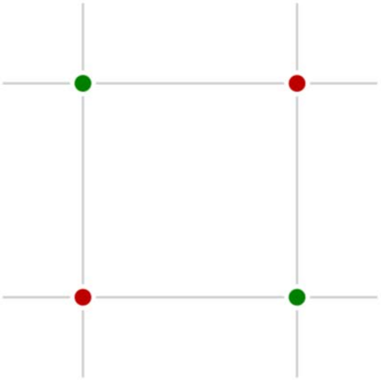


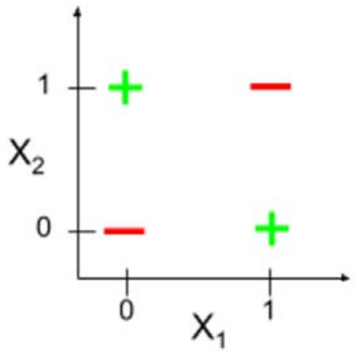
Here, a step function is illustrated instead of a sign function

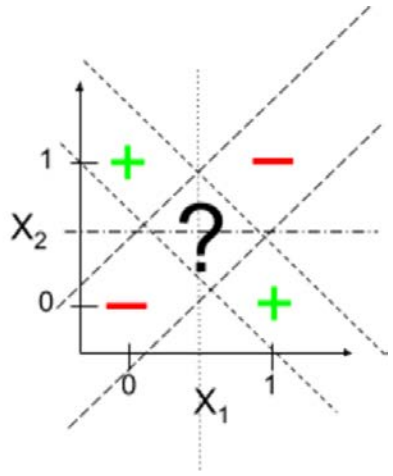
XOR Problem

- 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



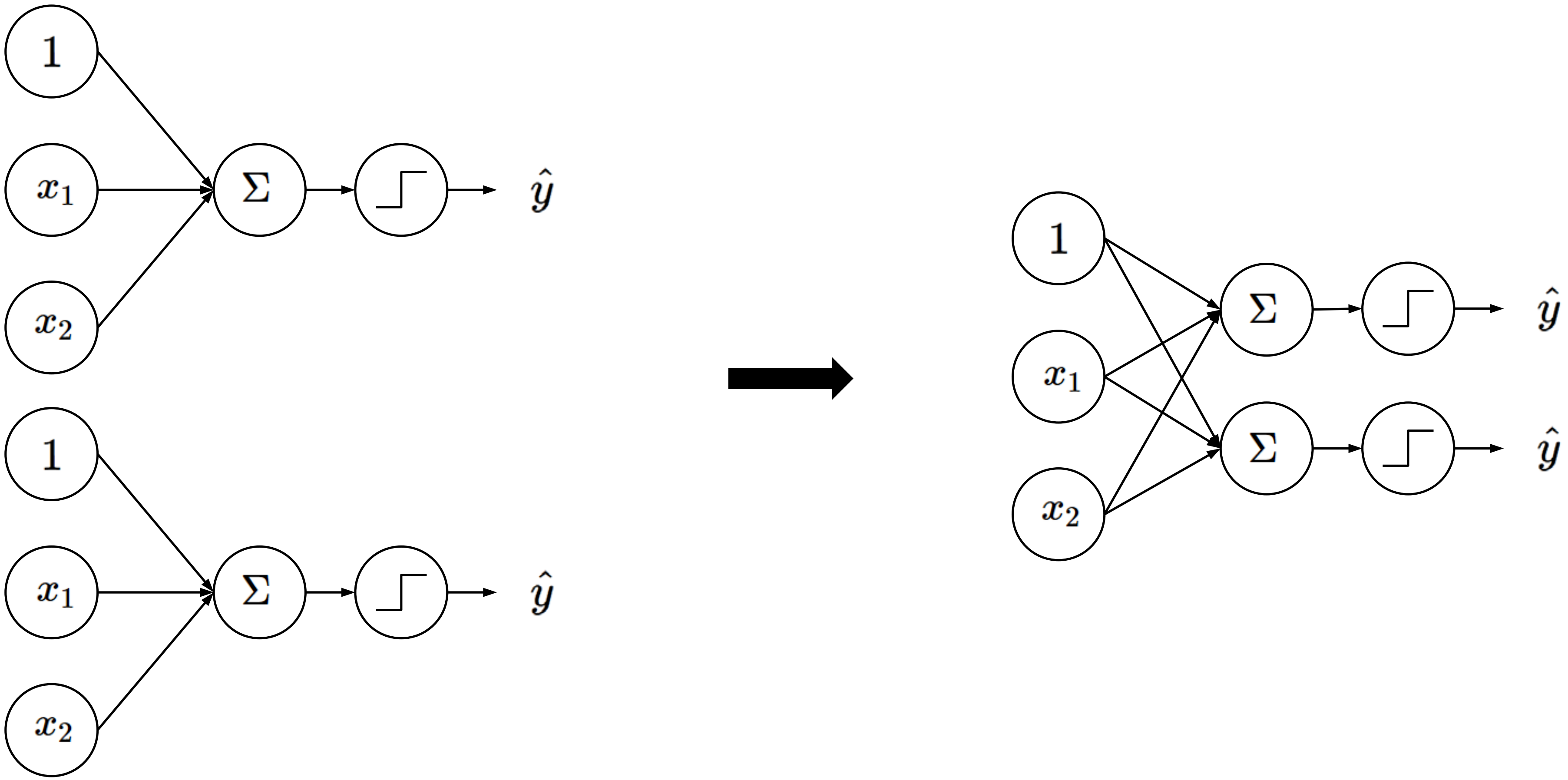




- Single neuron = one linear classification boundary

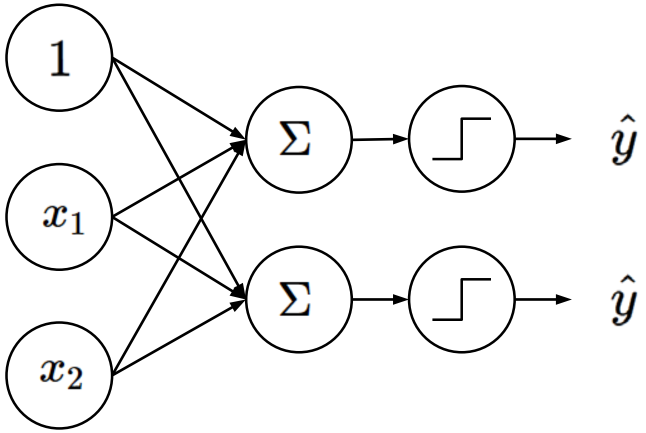
Artificial Neural Networks: MLP

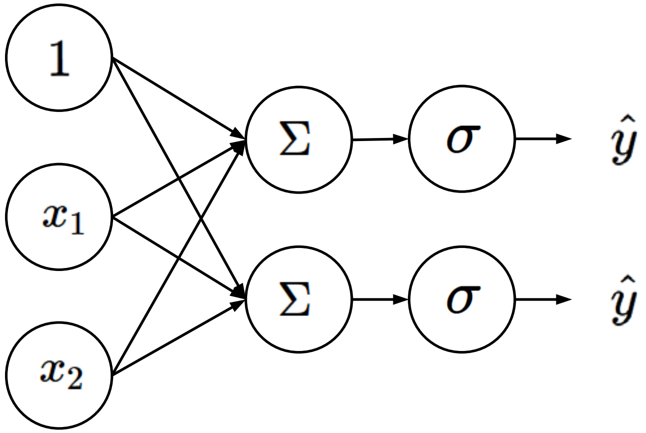
- Multi-layer Perceptron (MLP) = Artificial Neural Networks (ANN)
  - Multi neurons = multiple linear classification boundaries

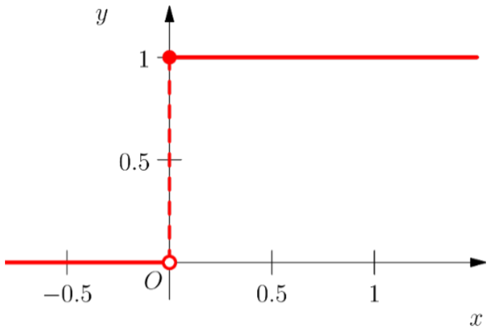


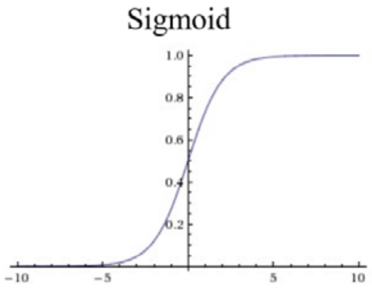
Artificial Neural Networks: Activation Function

- Differentiable nonlinear activation function







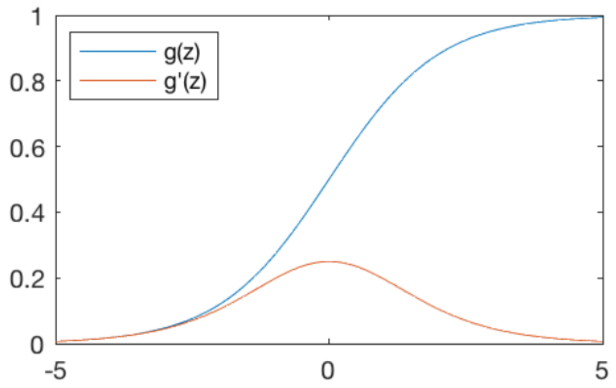


$$g(x) = \frac{1}{1 + e^{-x}}$$

Common Activation Functions

Discuss later

Sigmoid Function

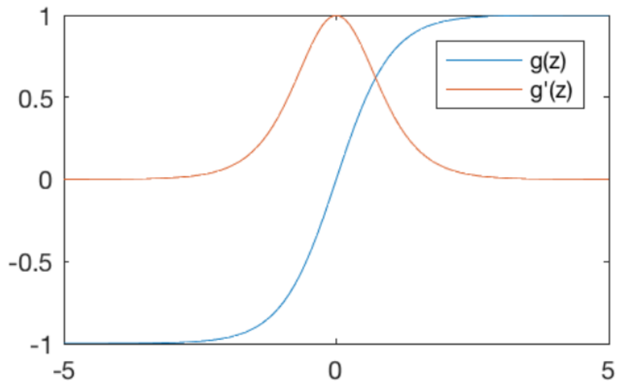


$$g(z) = \frac{1}{1 + e^{-z}}$$

$$g'(z) = g(z)(1 - g(z))$$

tf.nn.sigmoid(z)

Hyperbolic Tangent

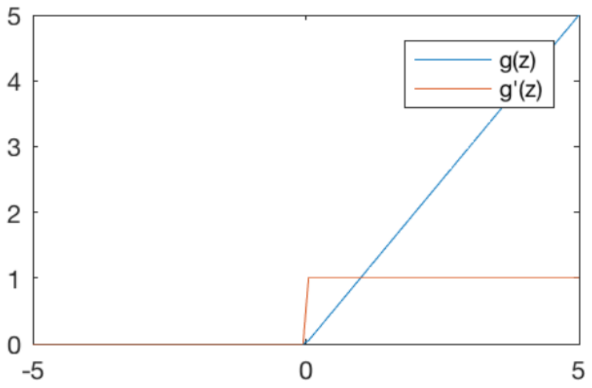


$$g(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

$$g'(z) = 1 - g(z)^2$$

tf.nn.tanh(z)

Rectified Linear Unit (ReLU)



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

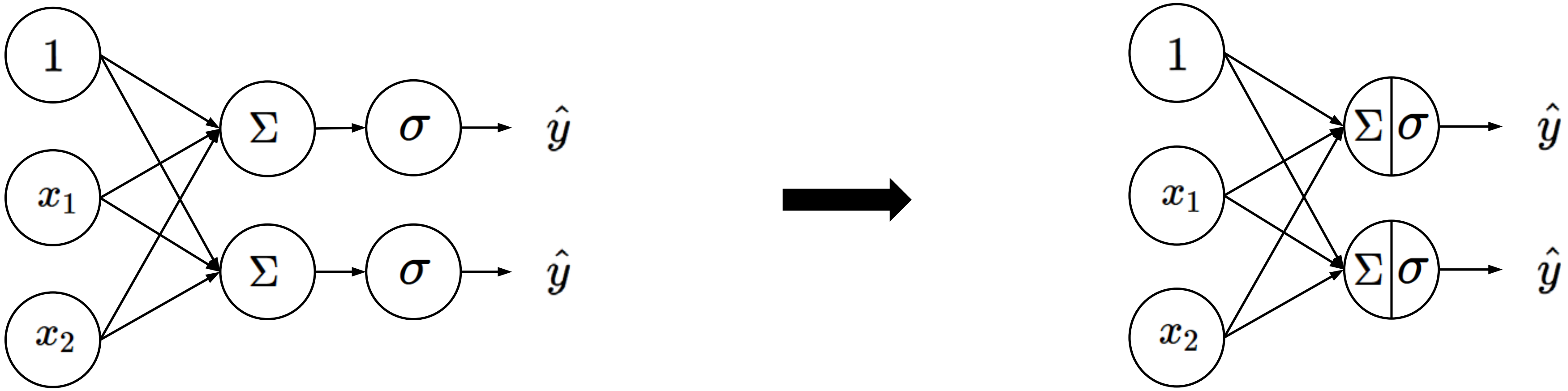
$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

tf.nn.relu(z)

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

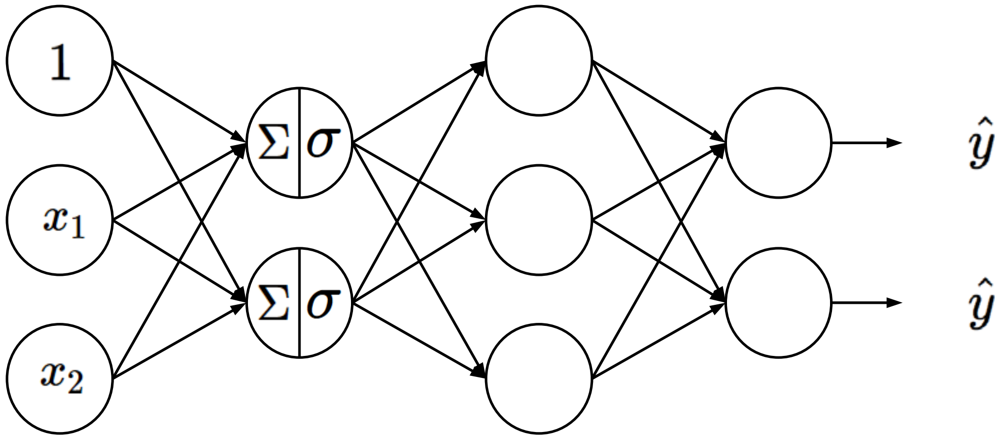
Artificial Neural Networks

- In a compact representation



Artificial Neural Networks

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

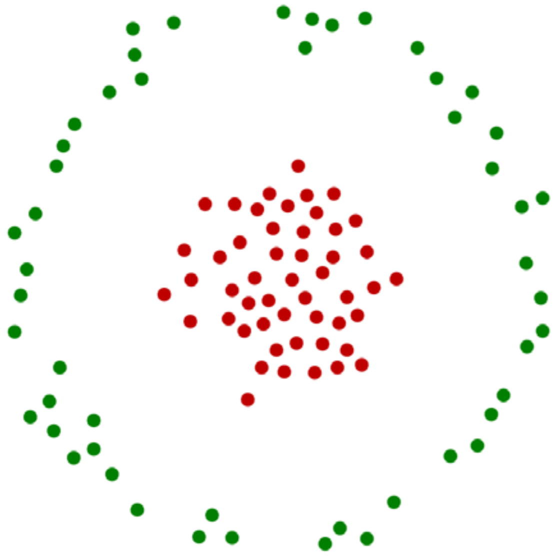


- A single layer is not enough to be able to represent complex relationship between input and output

$\Longrightarrow$ perceptron with many layers and units

Another Perspective:
ANN as Kernel Learning

Nonlinear Classification

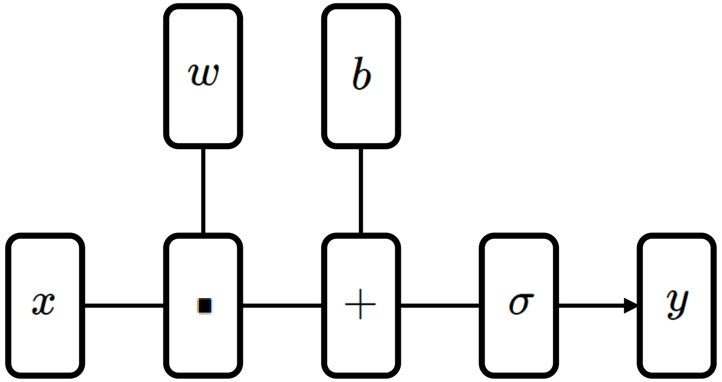


https://www.youtube.com/watch?v=3liCbRZPrZA

Neuron

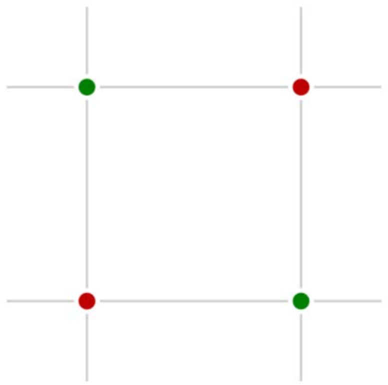
- We can represent this "neuron" as follows:

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



XOR Problem

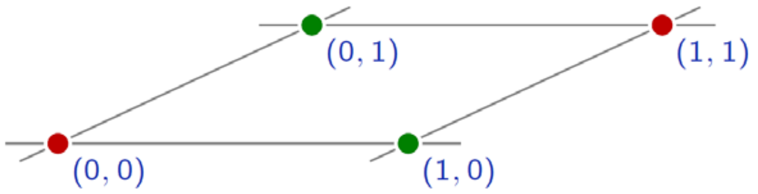
- The main weakness of linear predictors is their lack of capacity. For classification, the populations have to be linearly separable.



“xor”

Nonlinear Mapping

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

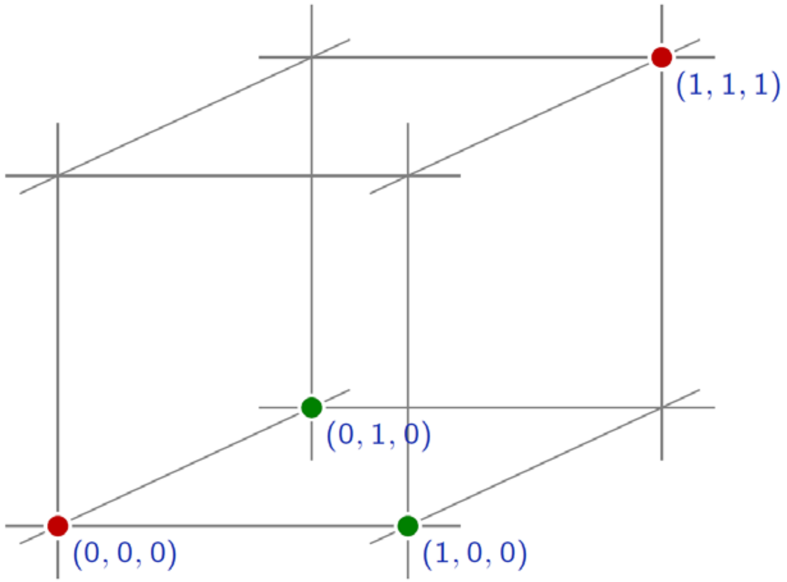


Source: Dr. Francois Fleuret at EPFL

Nonlinear Mapping

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

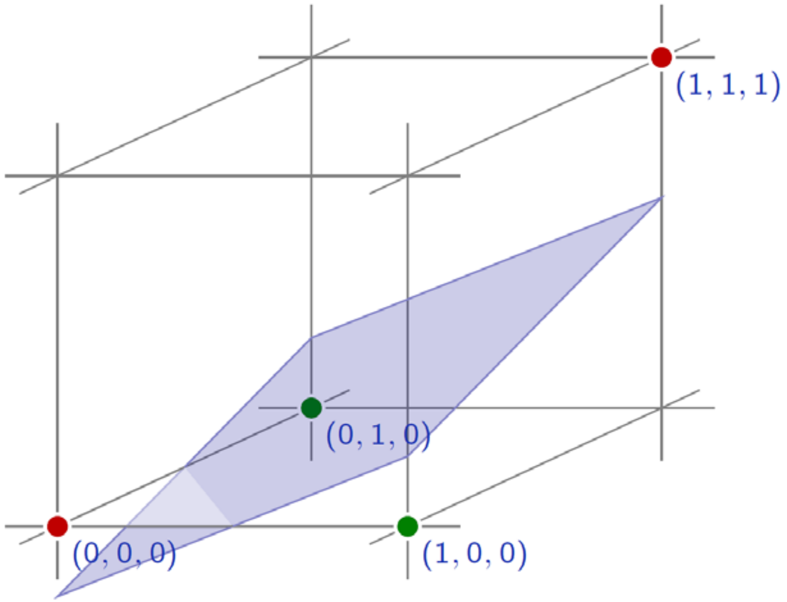
$$\phi : (x_u, x_v) \rightarrow (x_u, x_v, x_u x_v)$$



Nonlinear Mapping

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

$$\phi : (x_u, x_v) \rightarrow (x_u, x_v, x_u x_v)$$



Source: Dr. Francois Fleuret at EPFL

Kernel

- Often we want to capture nonlinear patterns in the data
  - nonlinear regression: input and output relationship may not be linear
  - nonlinear classification: classes may note be separable by a linear boundary

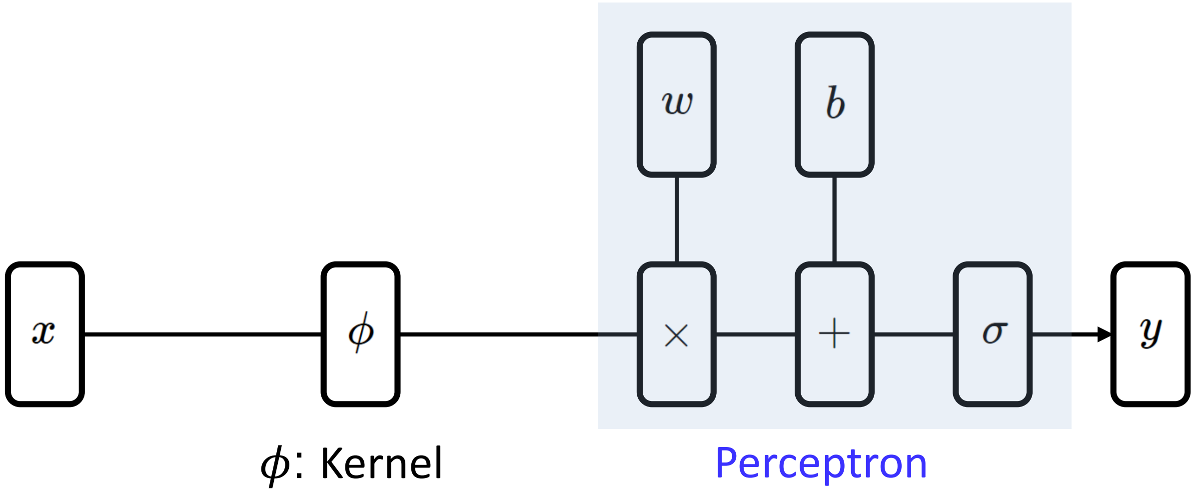
- Linear models (e.g. linear regression, linear SVM) are not just rich enough
  - by mapping data to higher dimensions where it exhibits linear patterns
  - apply the linear model in the new input feature space
  - mapping = changing the feature representation

- Kernels: make linear model work in nonlinear settings

Kernel + Neuron

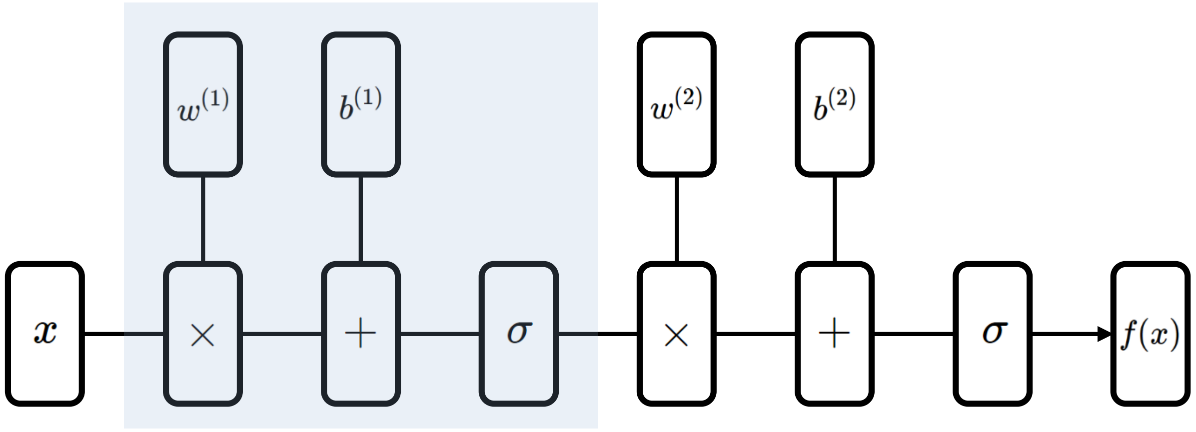
- Nonlinear mapping + neuron

$$\phi : (x_u, x_v) \to (x_u, x_v, x_u x_v)$$



Neuron + Neuron

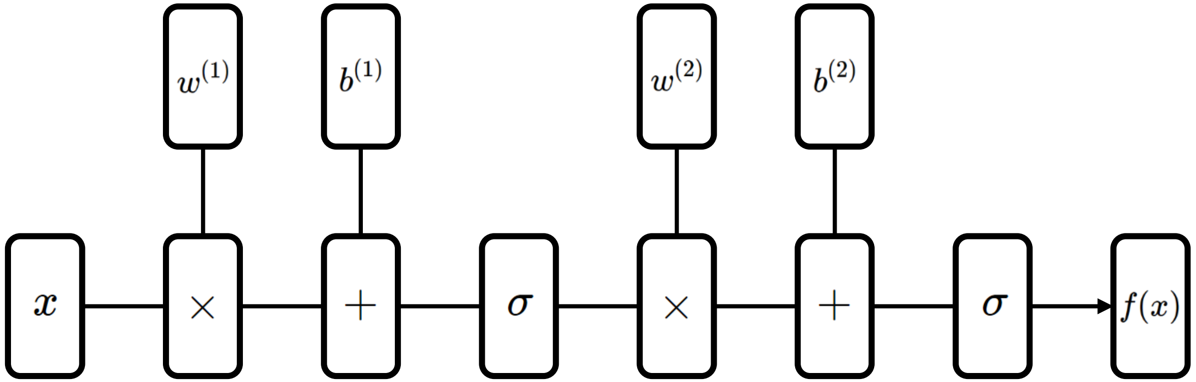
- Nonlinear mapping can be represented by another neurons



ϕ: Kernel

- Nonlinear Kernel
  - Nonlinear activation functions

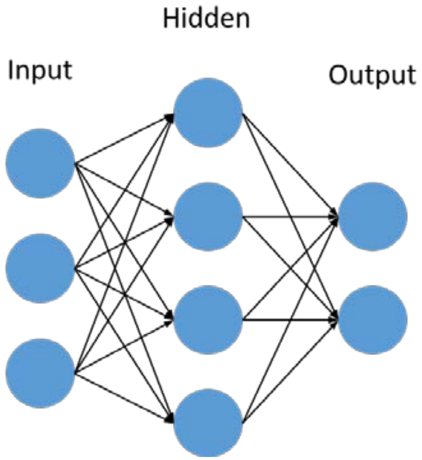
- Nonlinear mapping can be represented by another neurons
- We can generalize an MLP



Summary

- Universal function approximator
- Universal function classifier

- Parameterized

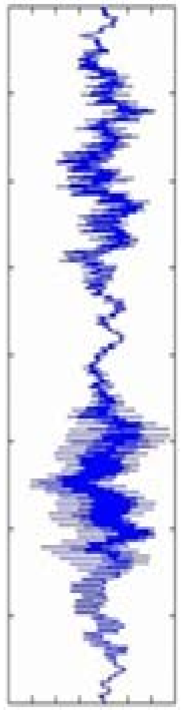


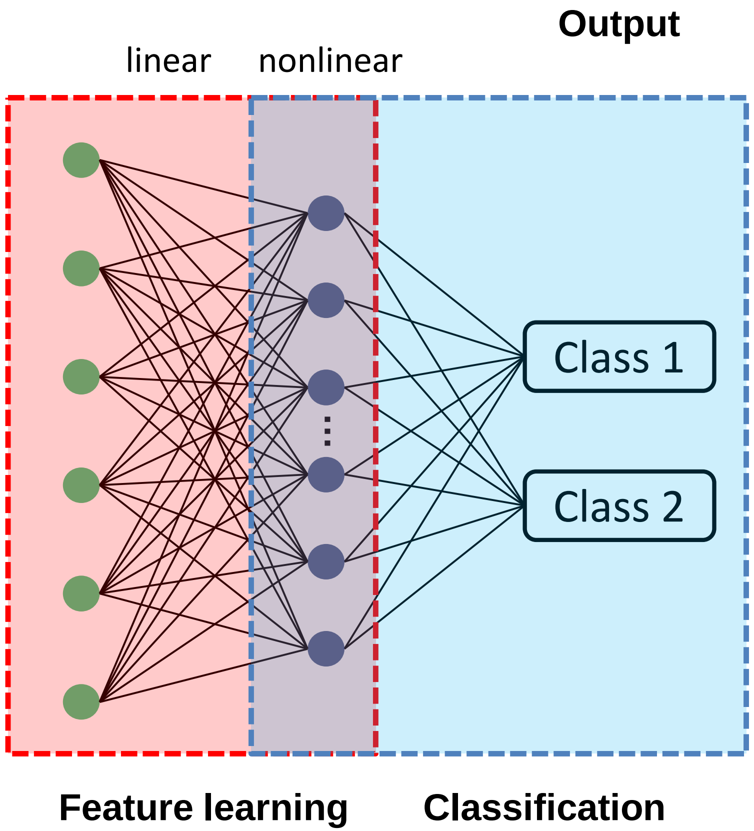
$$x \longrightarrow \boxed{\hat{y} = f_{\omega_1, \cdots, \omega_k}(x)} \longrightarrow y$$

Artificial Neural Networks

- Complex/Nonlinear universal function approximator
  - Linearly connected networks
  - Simple nonlinear neurons

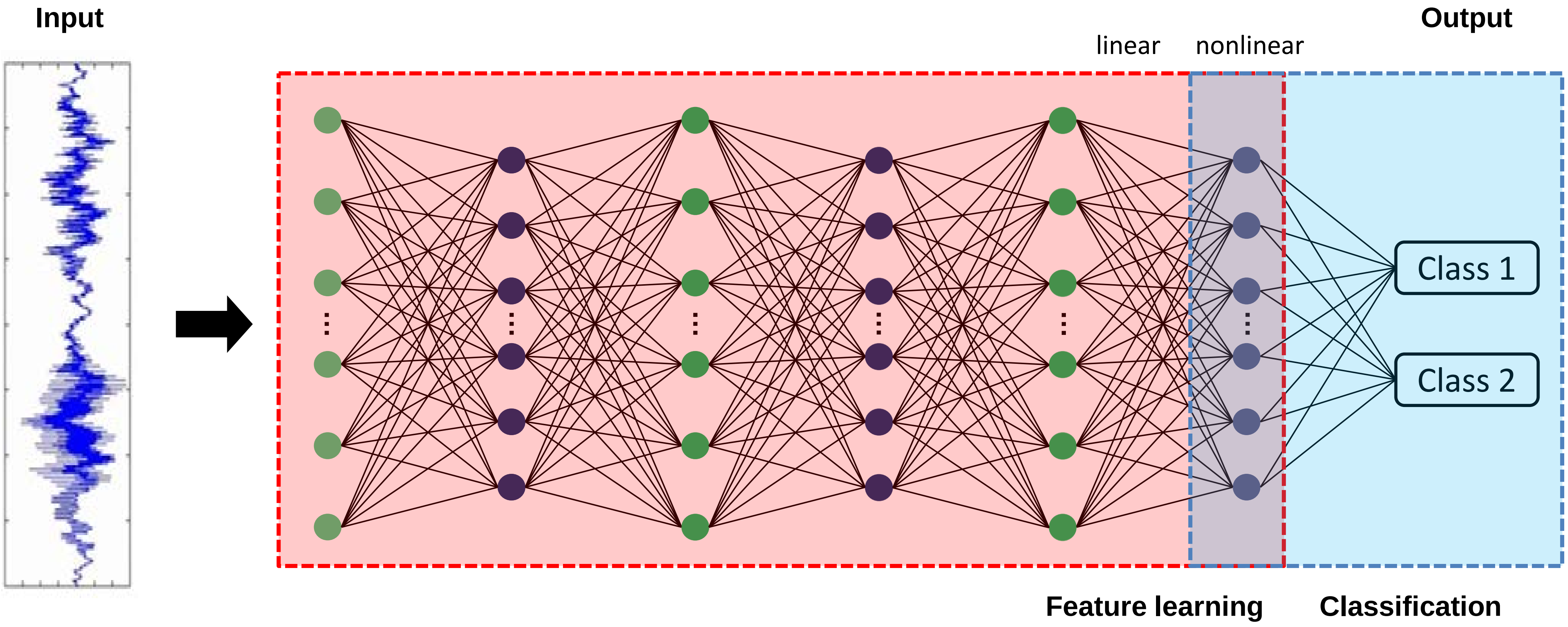
Input





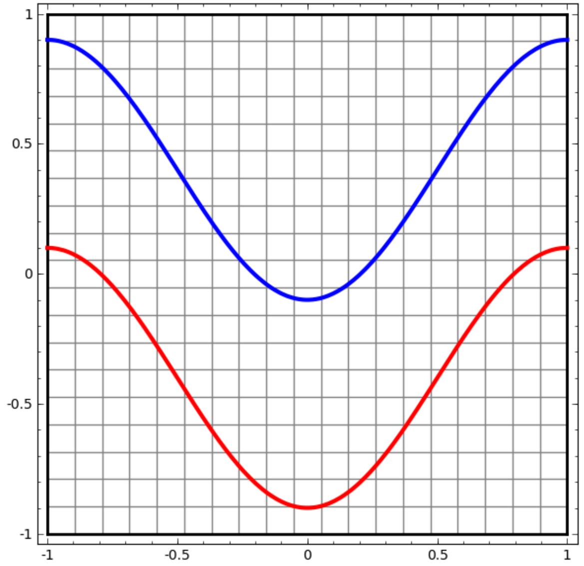
Deep Artificial Neural Networks

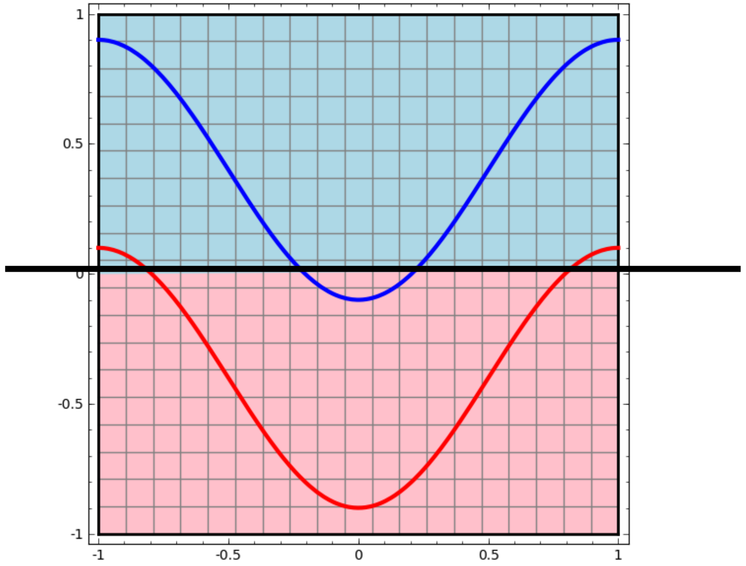
- Complex/Nonlinear universal function approximator
  - Linearly connected networks
  - Simple nonlinear neurons



Example: Linear Classifier

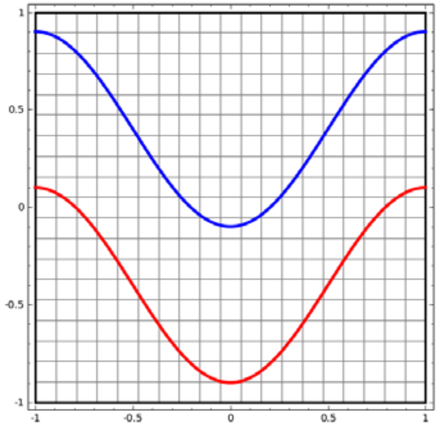
- Perceptron tries to separate the two classes of data by dividing them with a line

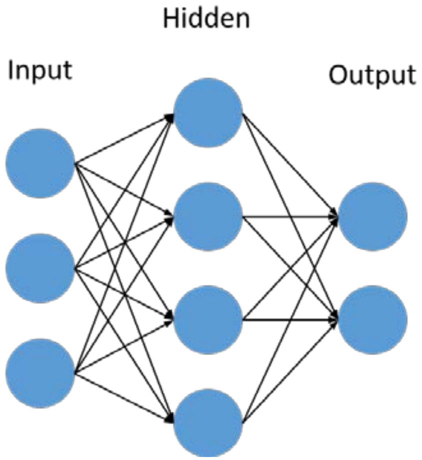


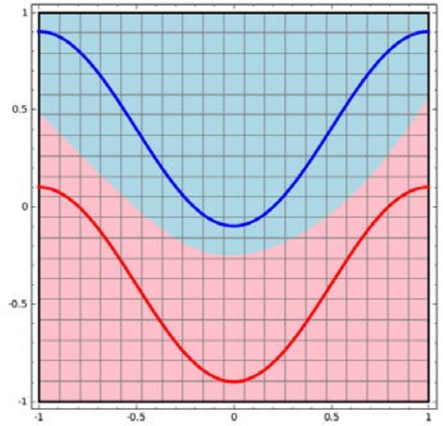


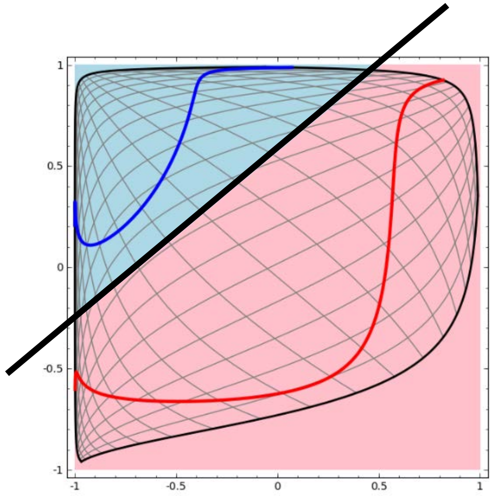
Example: Neural Networks

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

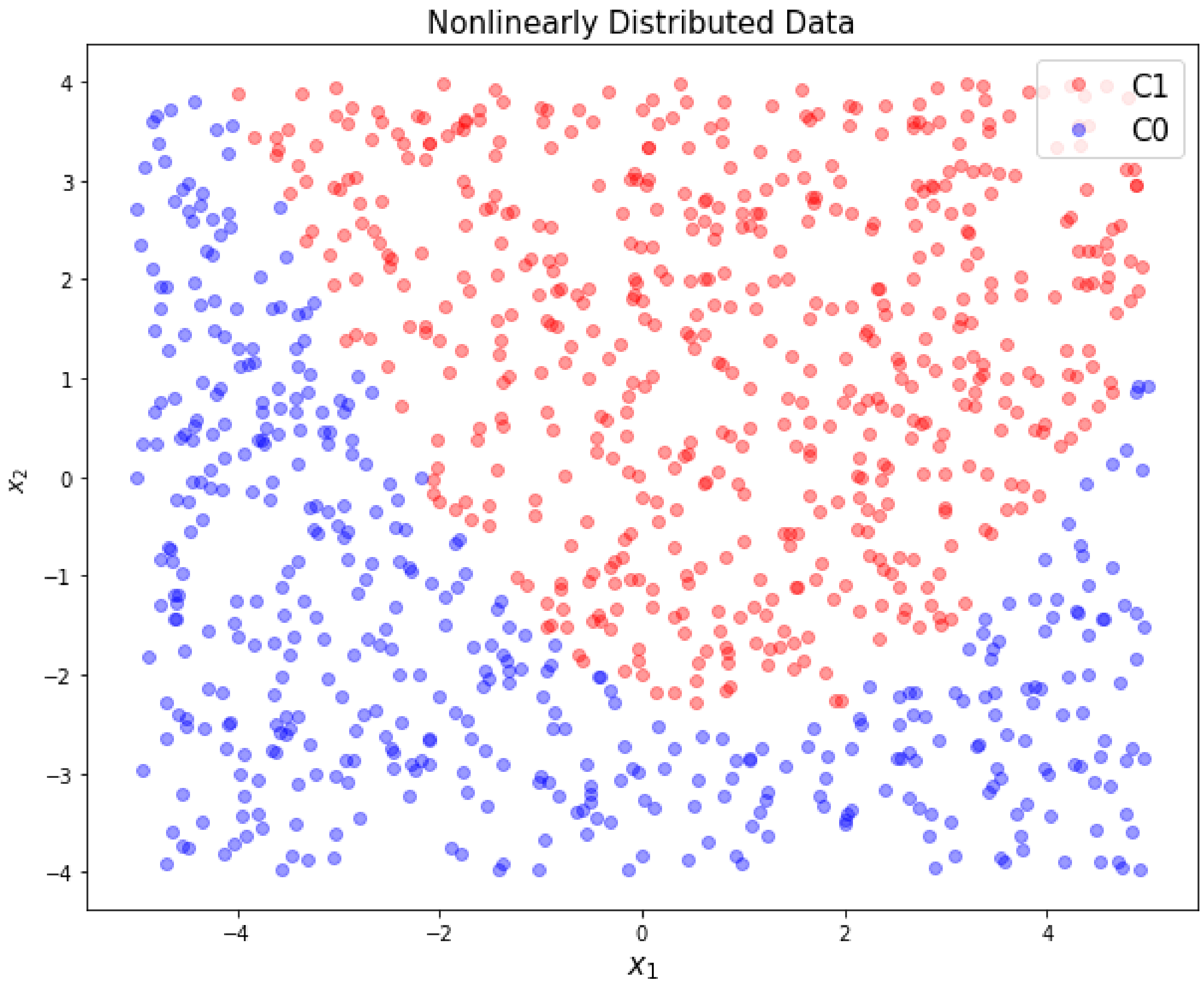




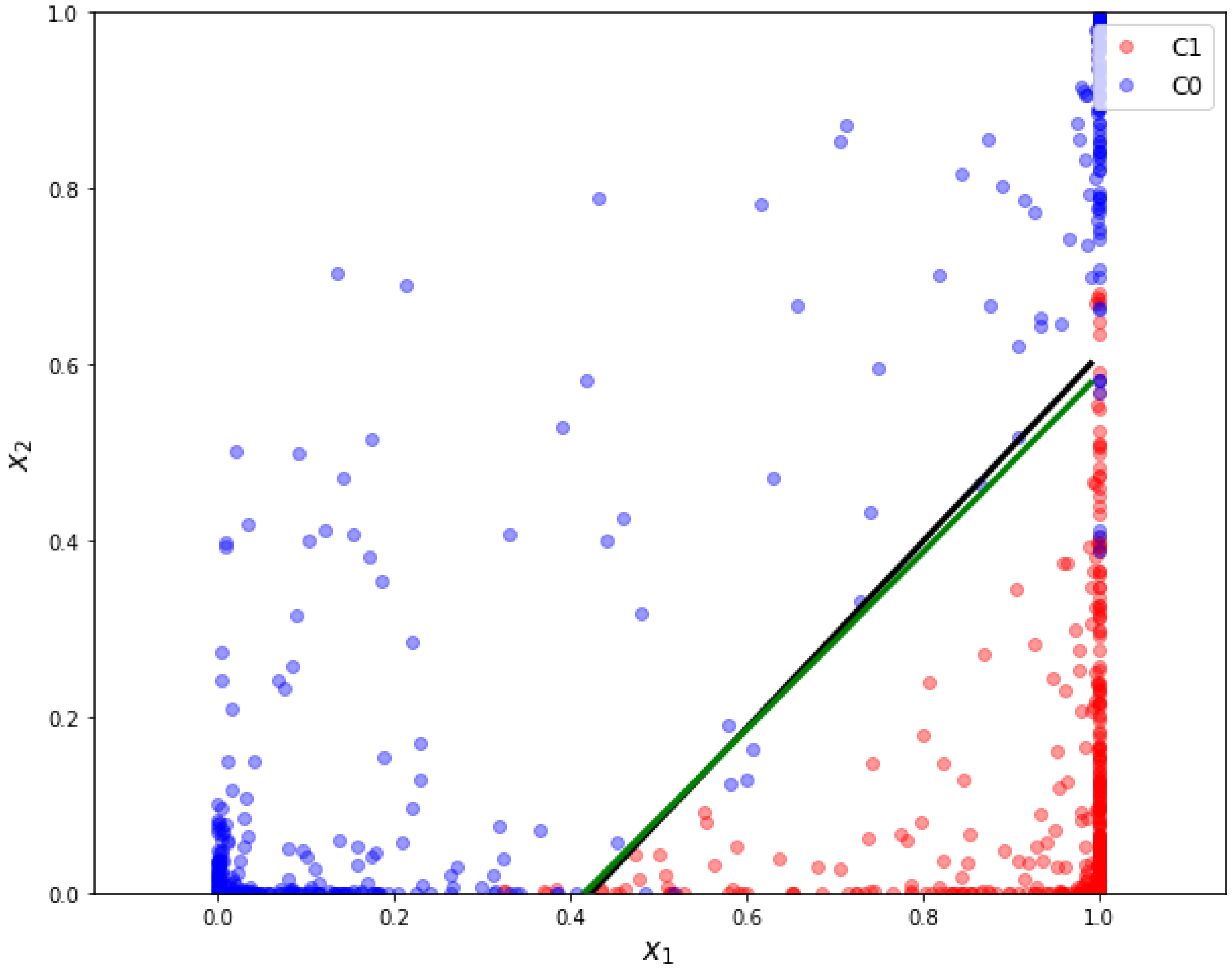




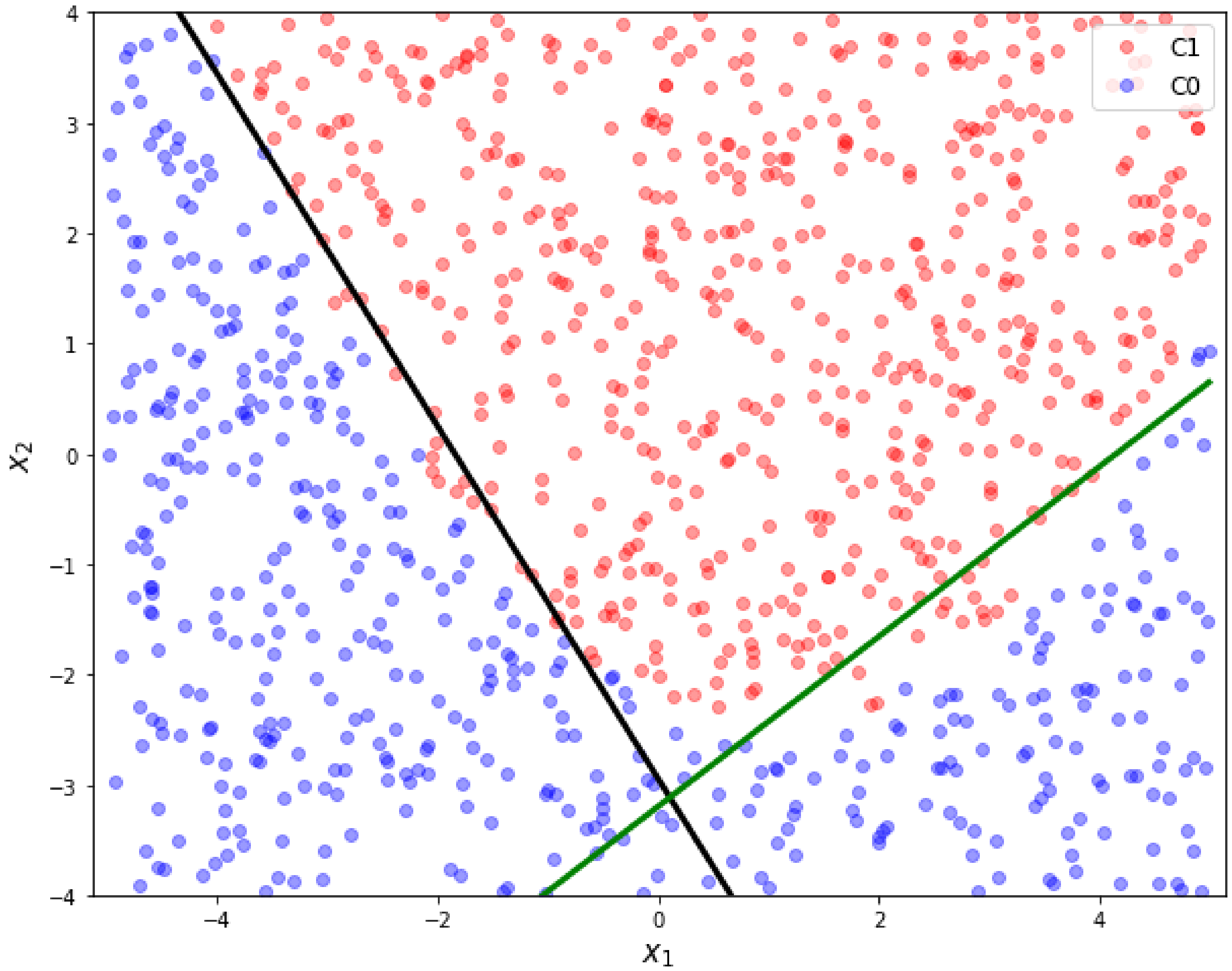
Nonlinearly Distributed Data



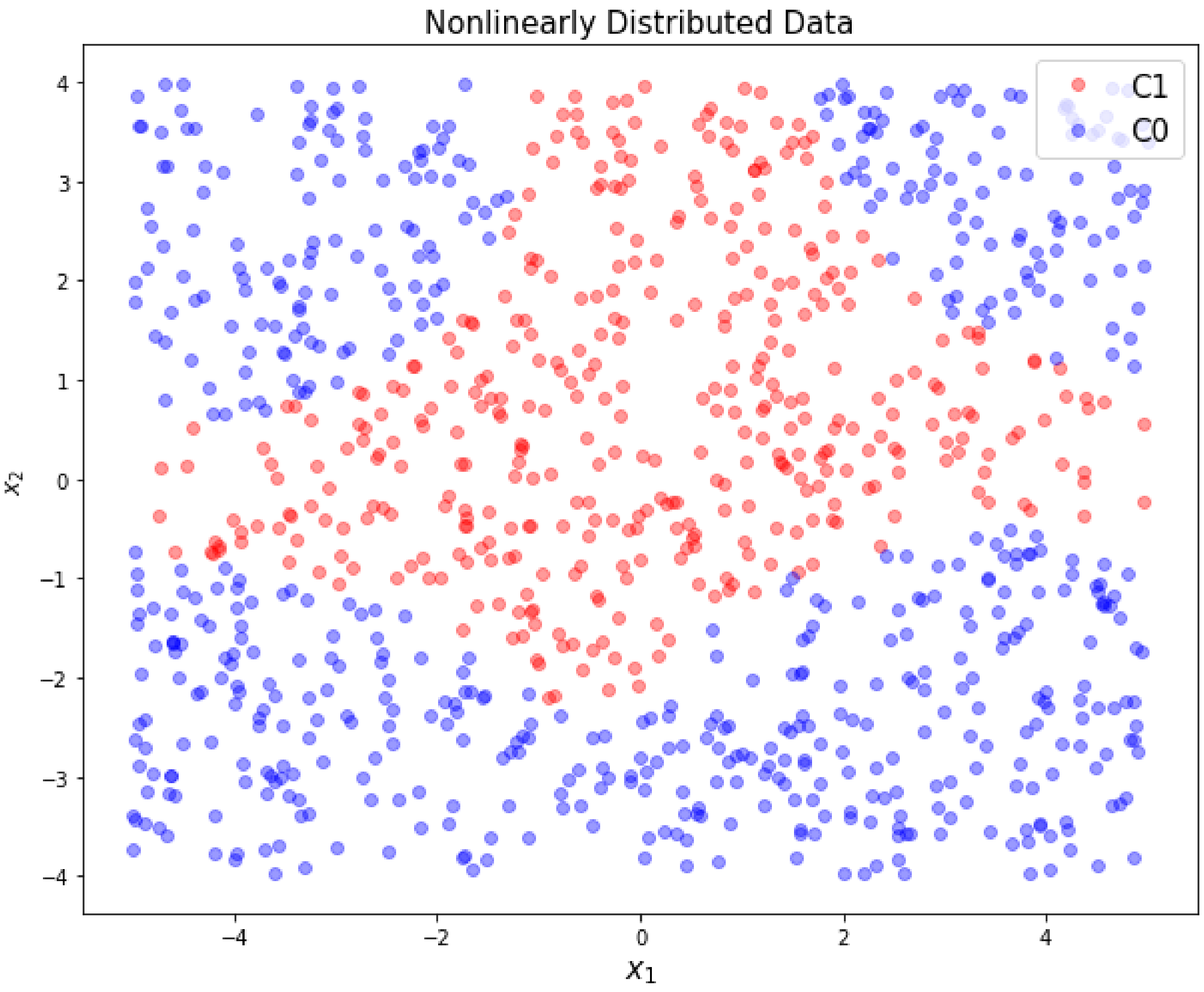
Multi Layers



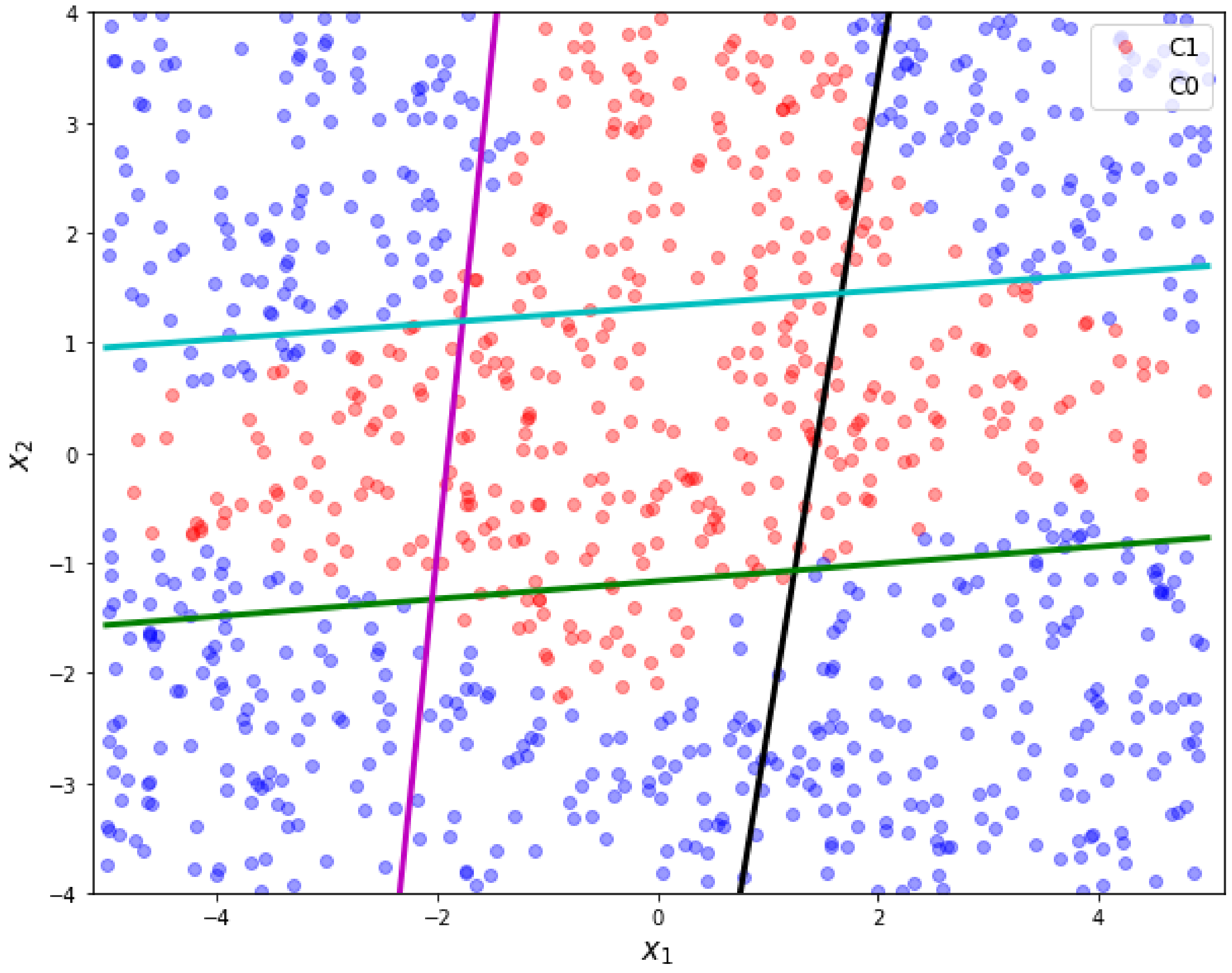
Multi Layers



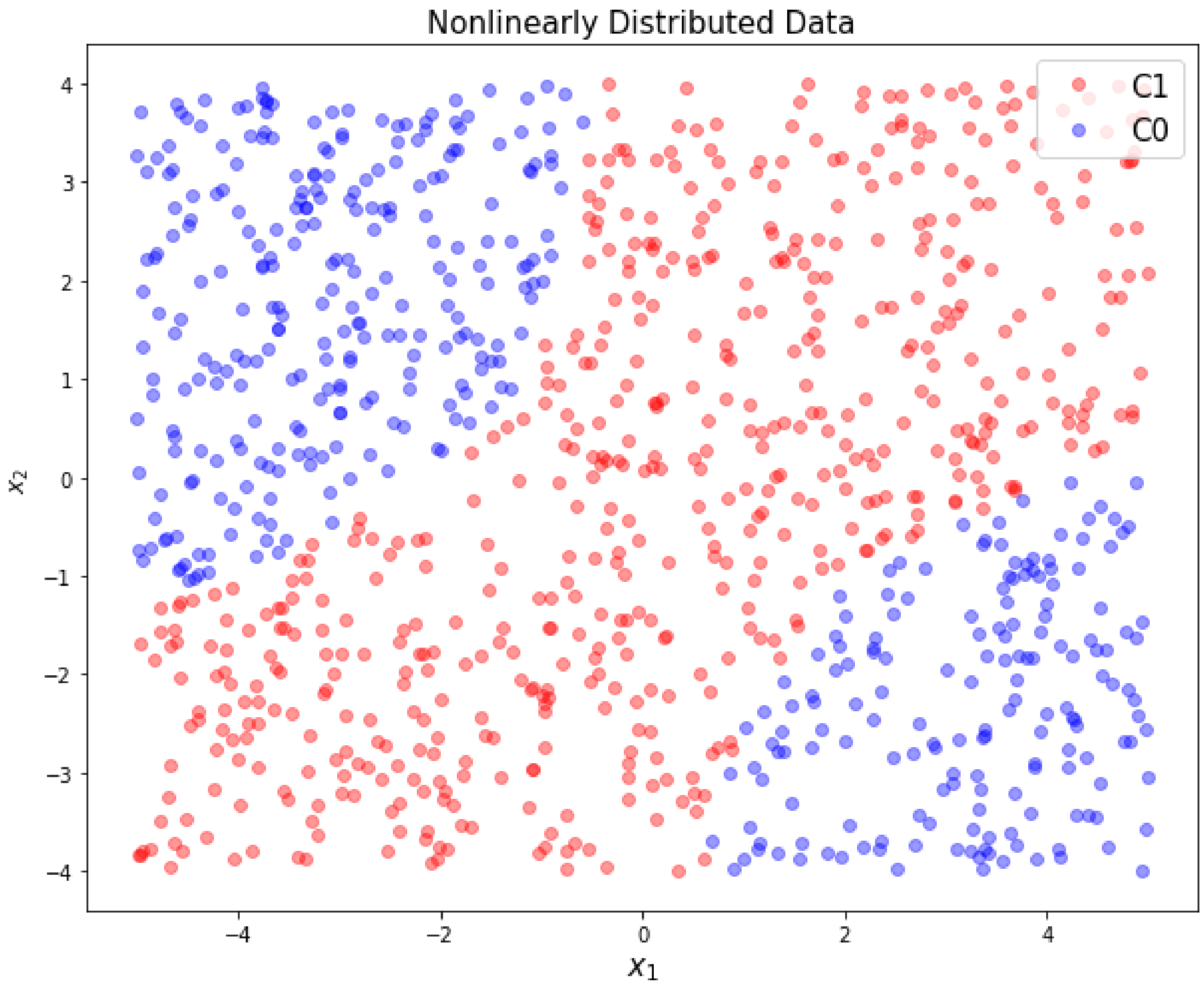
Nonlinearly Distributed Data



Multi Layers



Nonlinearly Distributed Data



Multi Layers