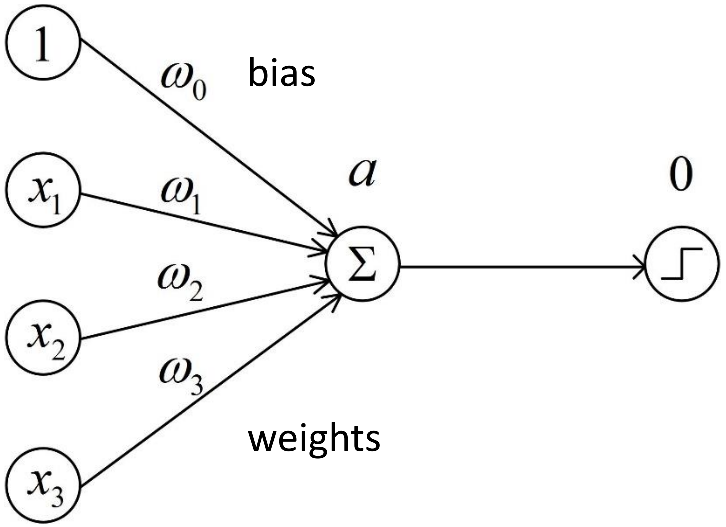
(Artificial) Neural Networks

Industrial AI Lab.

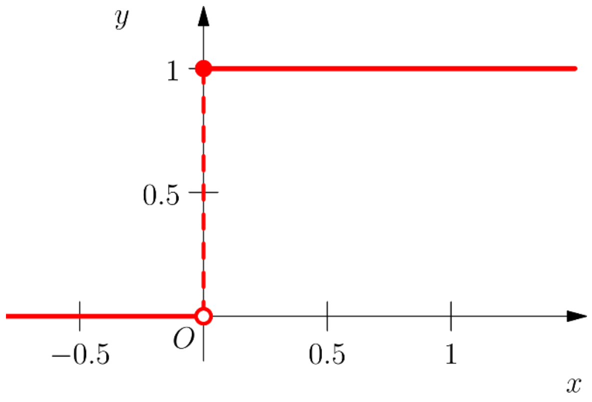
Prof. Seungchul Lee

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



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

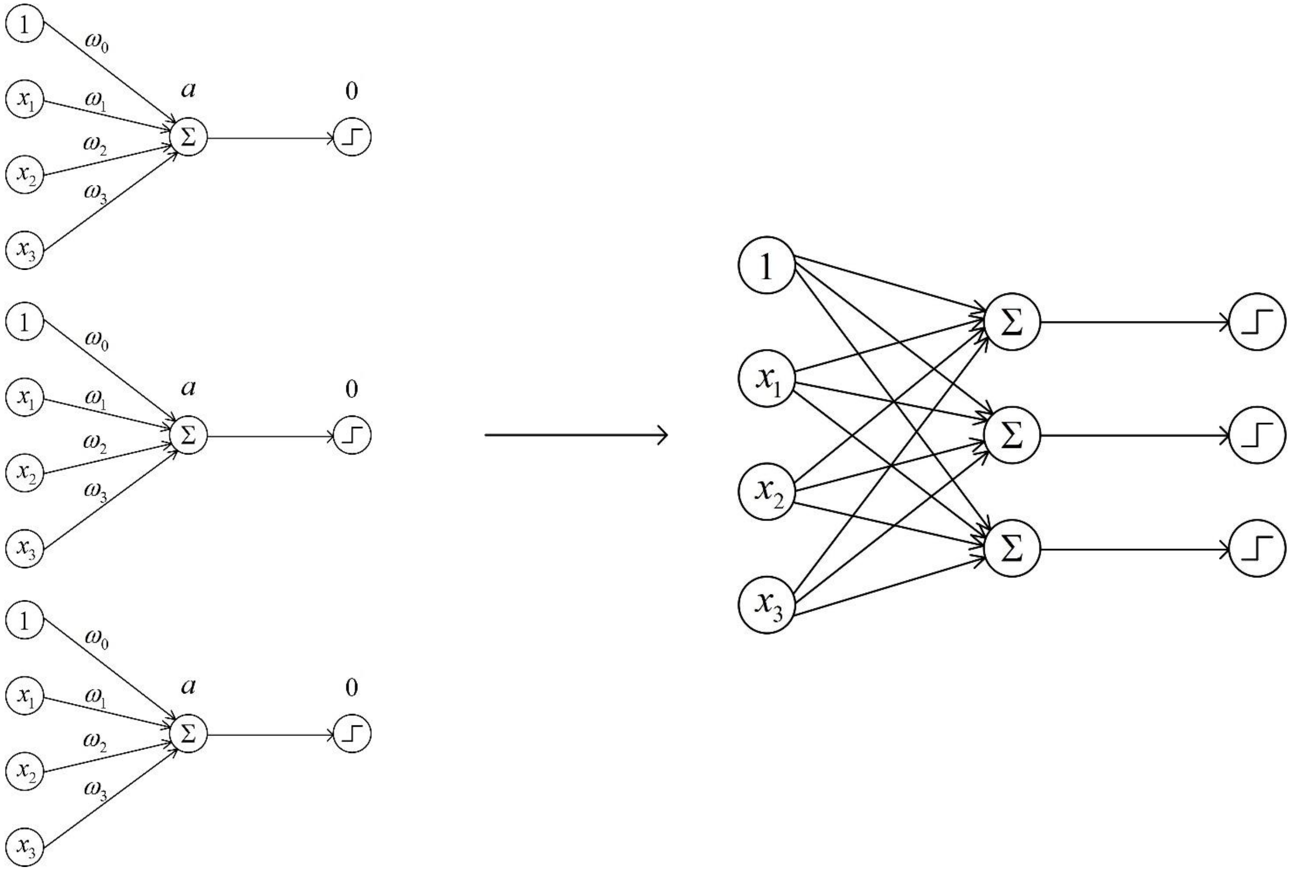


- A step function is not differentiable
- One layer is often not enough

Artificial Neural Networks: MLP

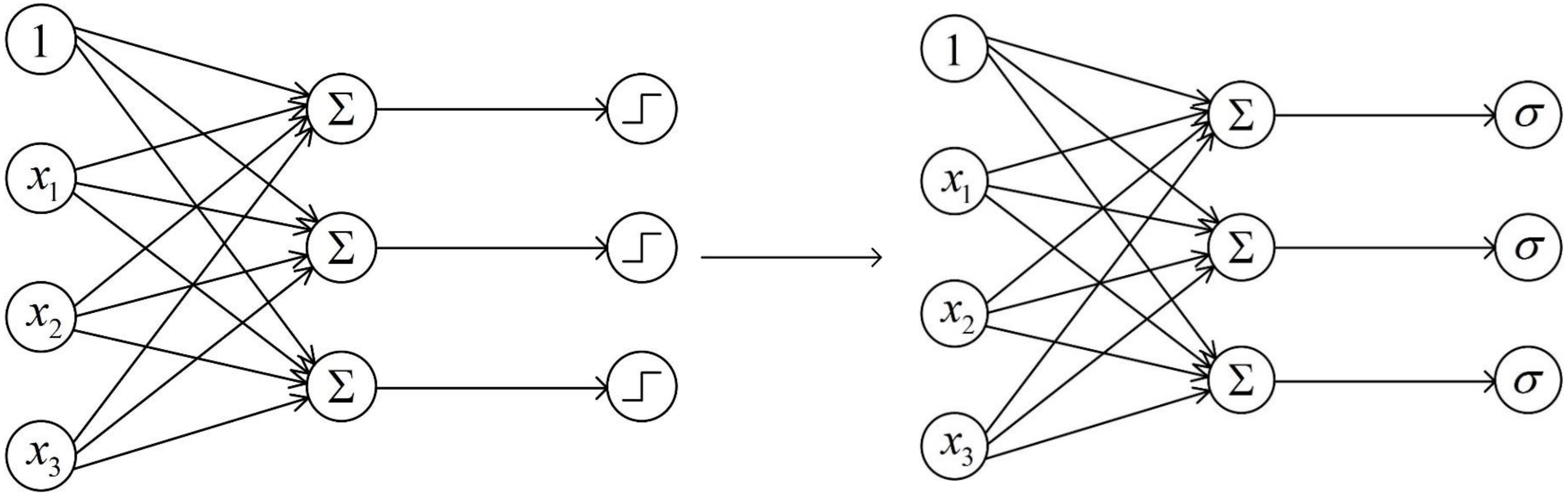
- Multi-layer Perceptron (MLP) = Artificial Neural Networks (ANN)

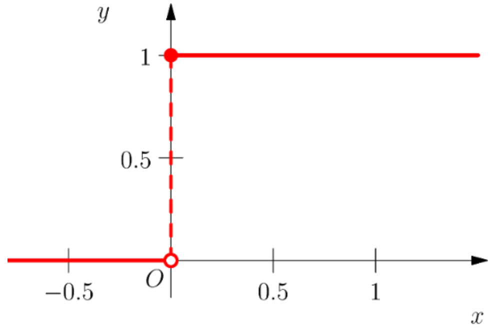
  multi-neurons

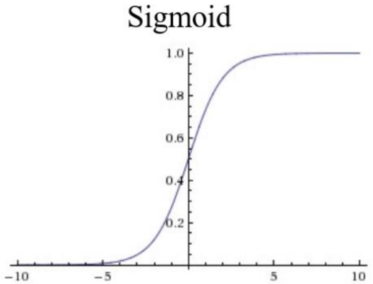


Artificial Neural Networks: Activation Func.

- differentiable non-linear activation function



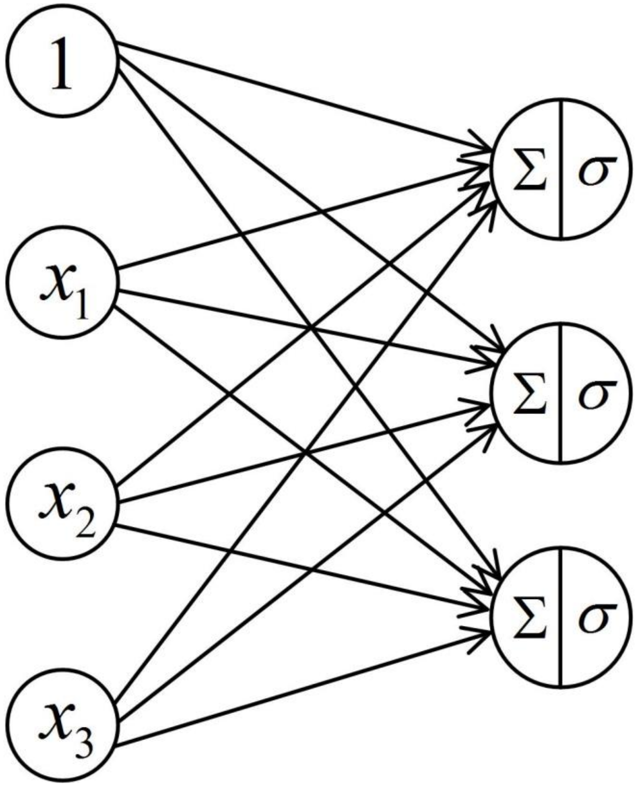




$$g(x) = \frac{1}{1+e^{-x}}$$

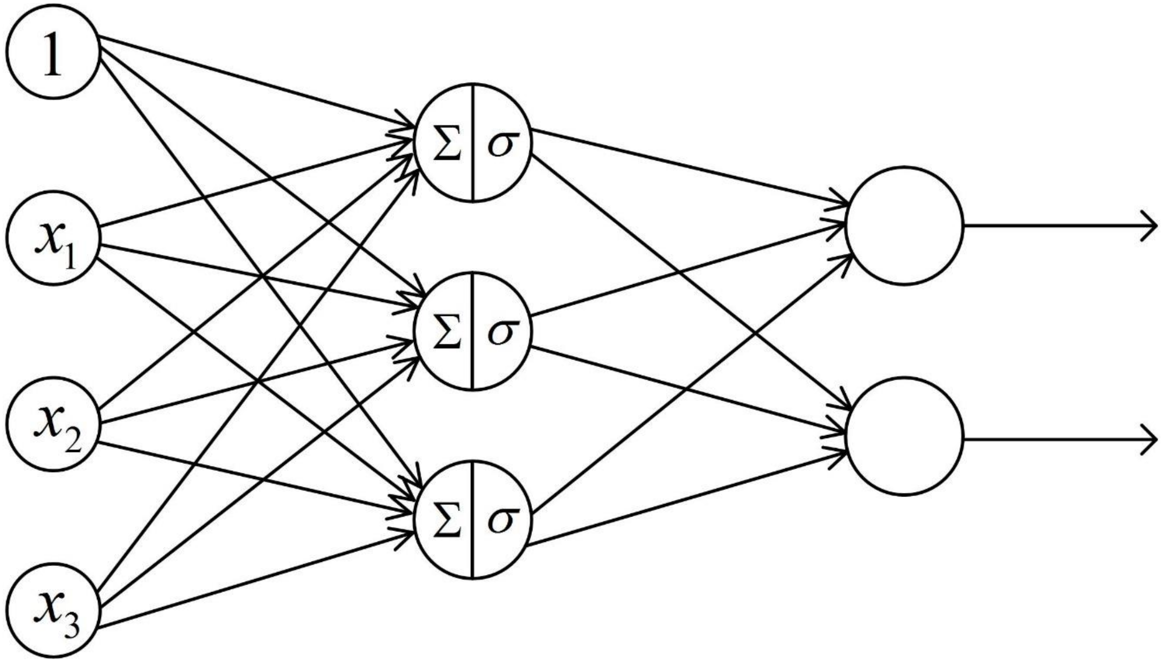
Artificial Neural Networks

- in a compact representation



Artificial Neural Networks

- multi-layer perceptron

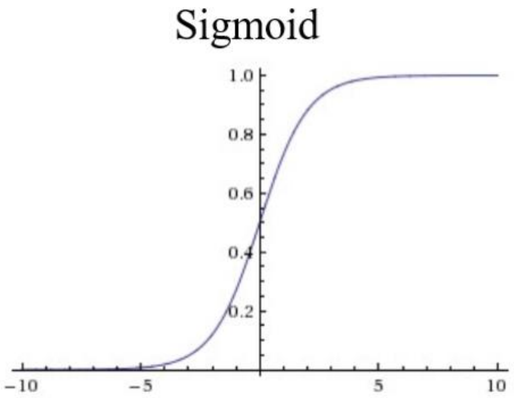


ANN: Transformation

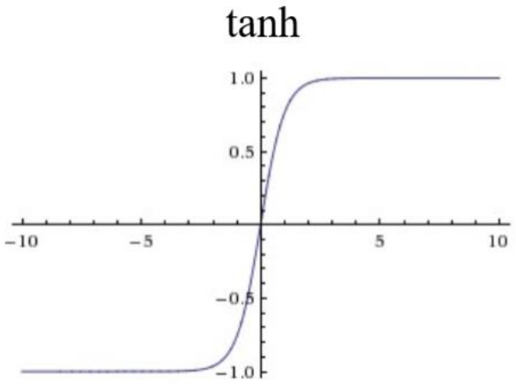
- Affine (or linear) transformation and nonlinear activation layer (notations are mixed: $g = \sigma, \omega = \theta, \omega_0 = b$)

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

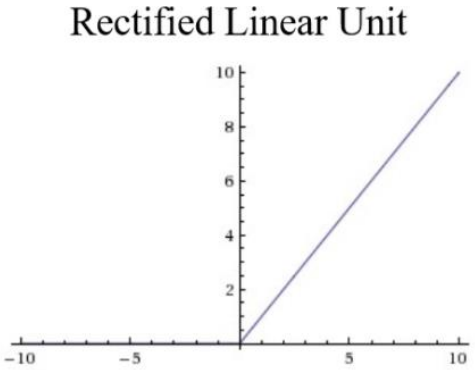
- Nonlinear activation functions ($g = \sigma$)



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



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



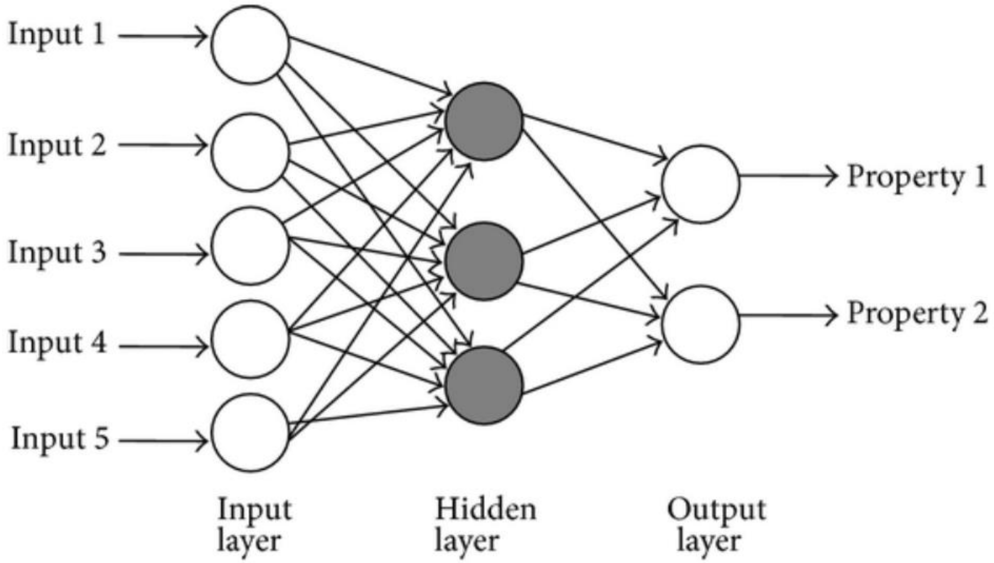
$$g(x) = \max(0, x)$$

ANN: Structure

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

  $\Longrightarrow$ perceptron with many layers and units

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



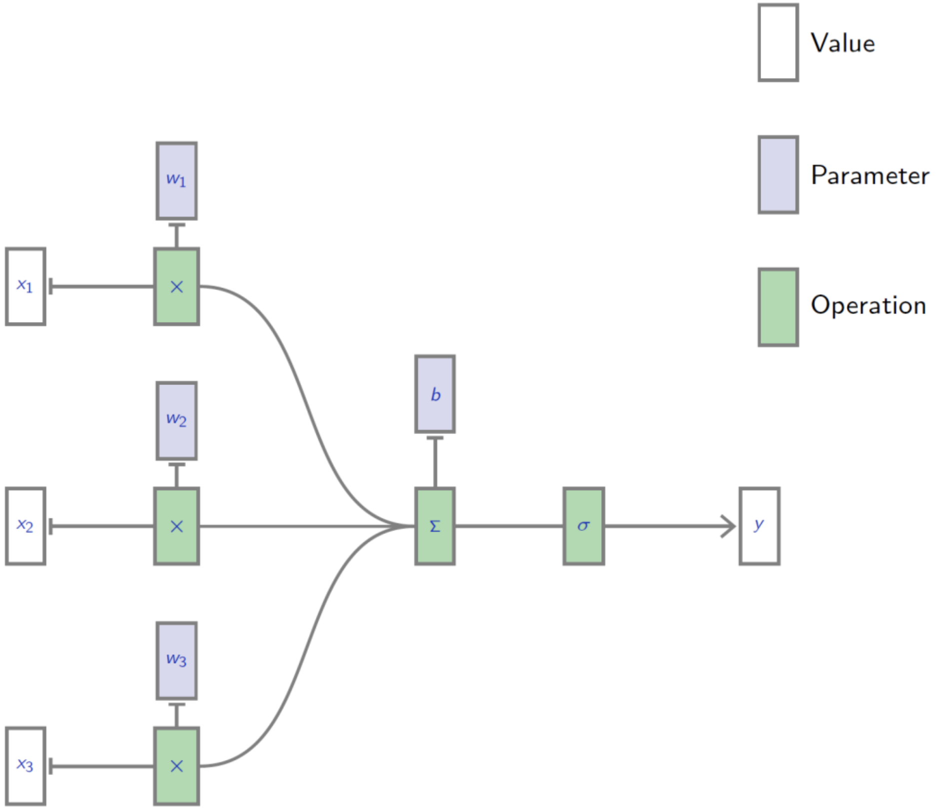
- The perceptron classification rule boils down to

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

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

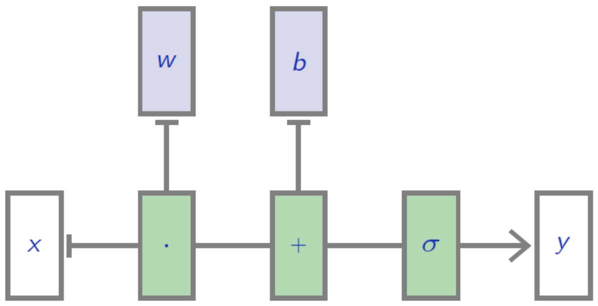
- We can also use tensor operations, as in

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

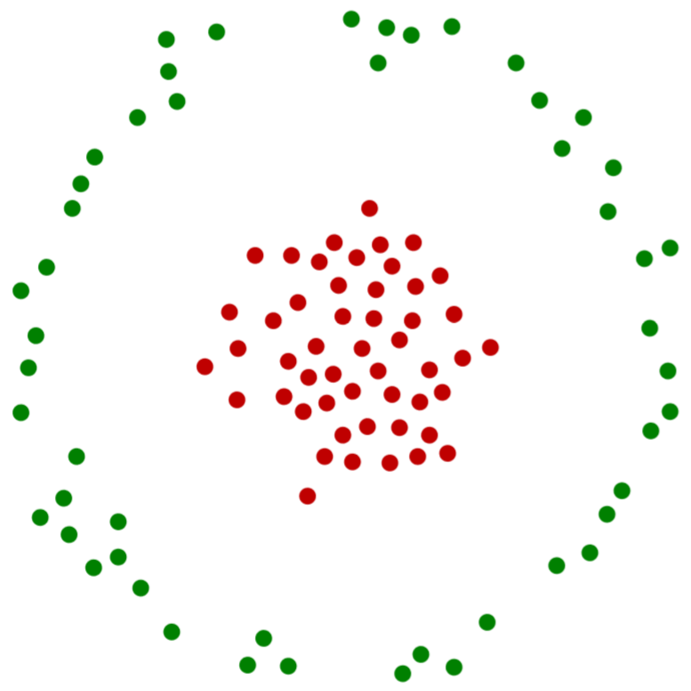


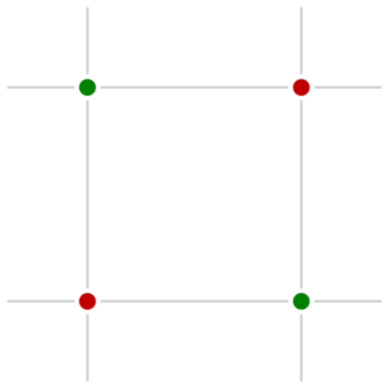
- We can represent this “neuron” as follows:

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



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

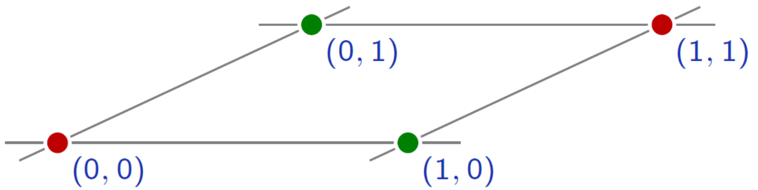




"xor"

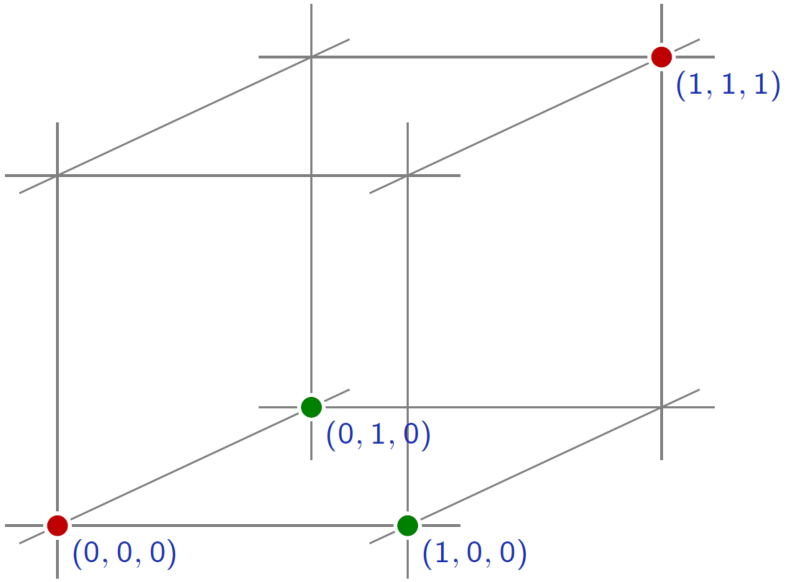
- The xor example can be solved by pre-processing the data to make the two populations linearly separable.

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



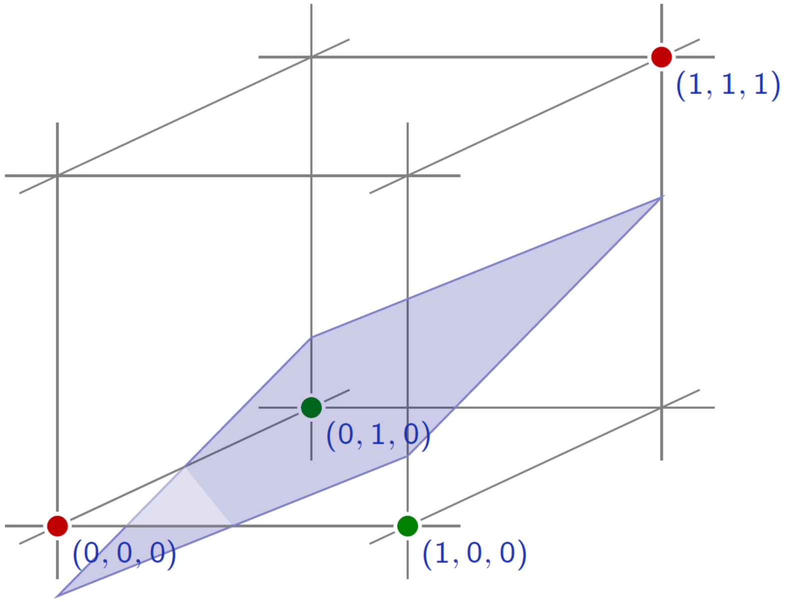
- The xor example can be solved by pre-processing the data to make the two populations linearly separable.

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



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

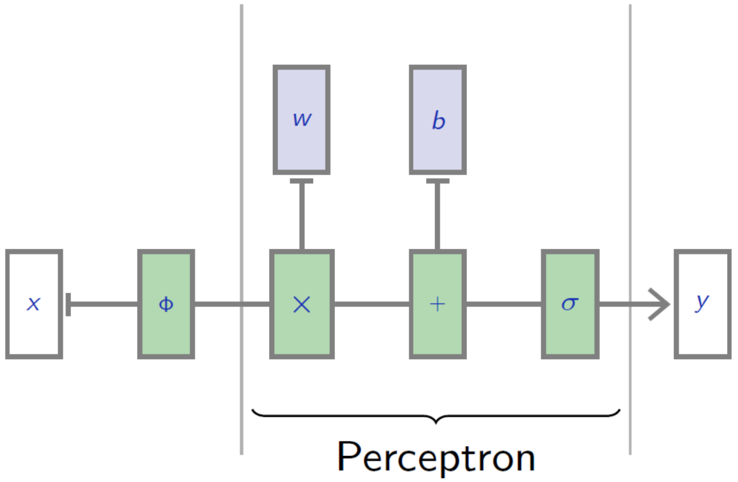
$$\Phi : (x_u, x_v) \mapsto (x_u, x_v, x_u x_v).$$



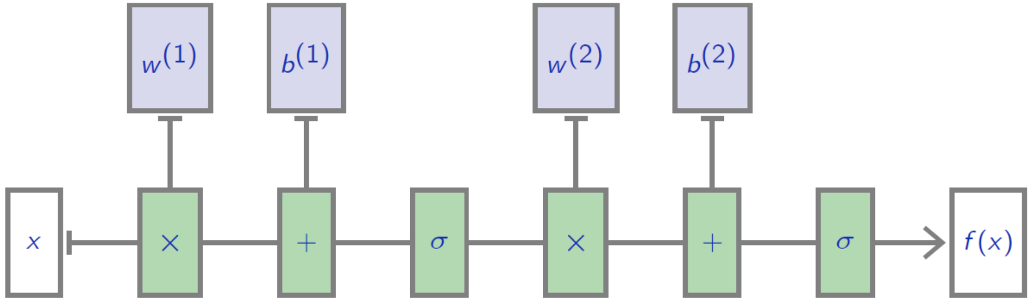
POSTECH

- Nonlinear mapping + neuron

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

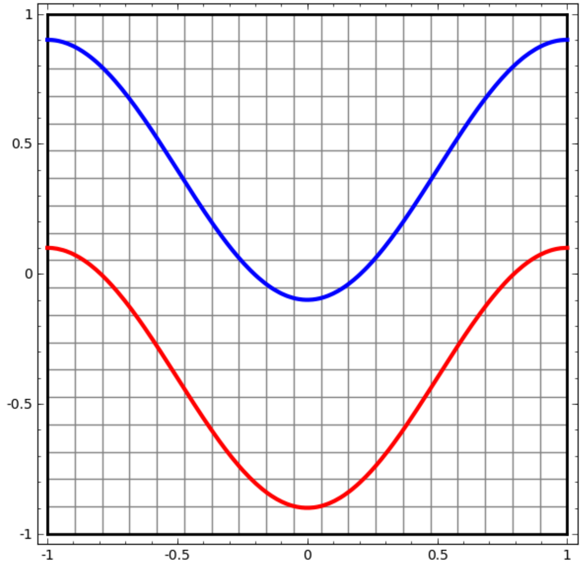


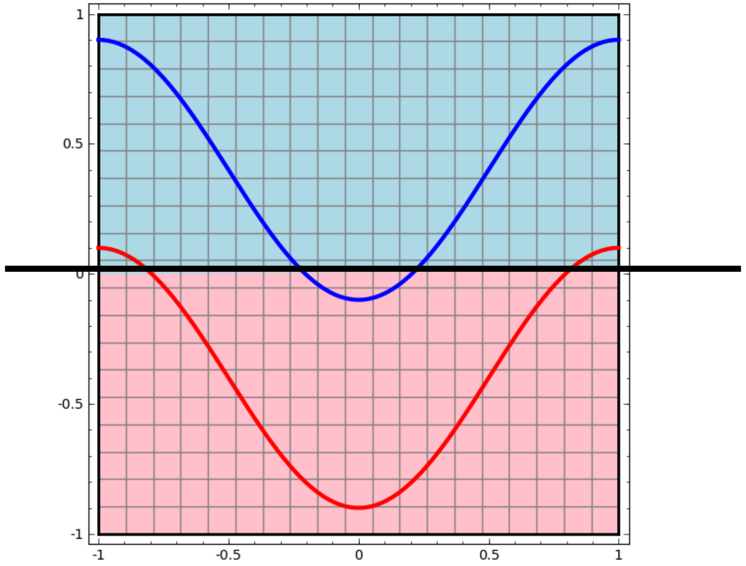
- Nonlinear mapping can be represented by another neurons
- We can generalize an MLP



Linear Classifier

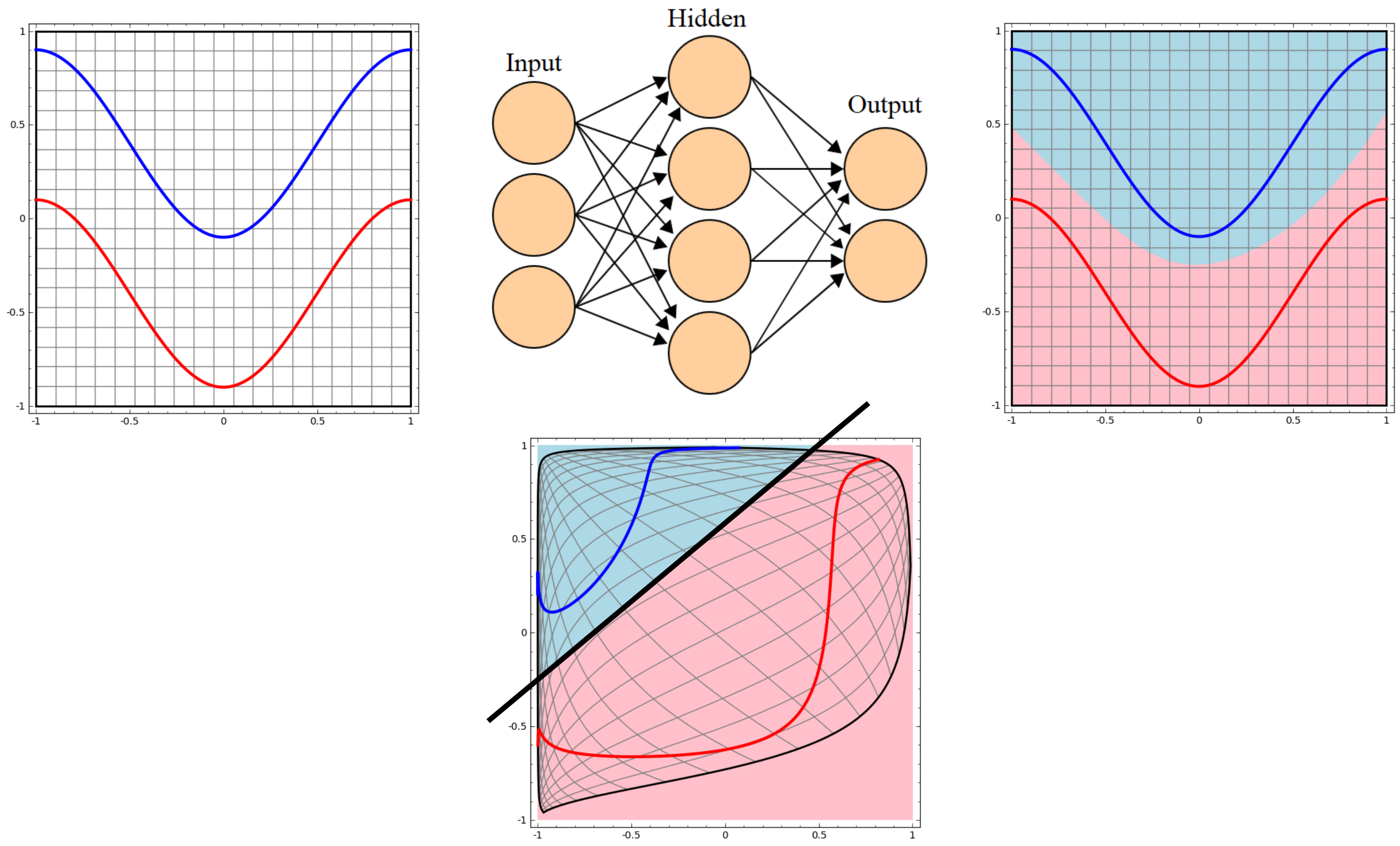
- Perceptron tries to separate the two classes of data by dividing them with a line





Neural Networks

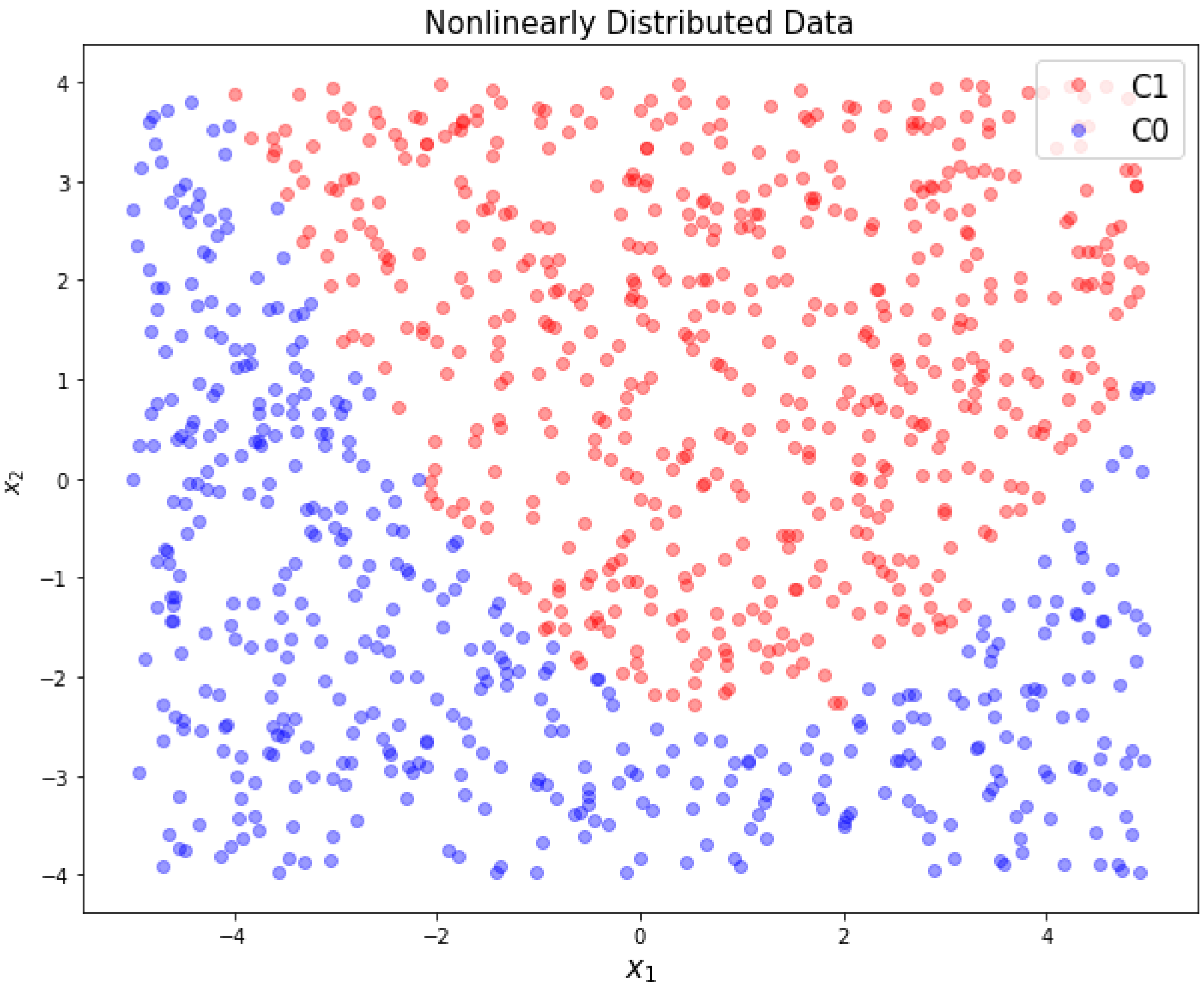
- The hidden layer learns a representation so that the data is linearly separable



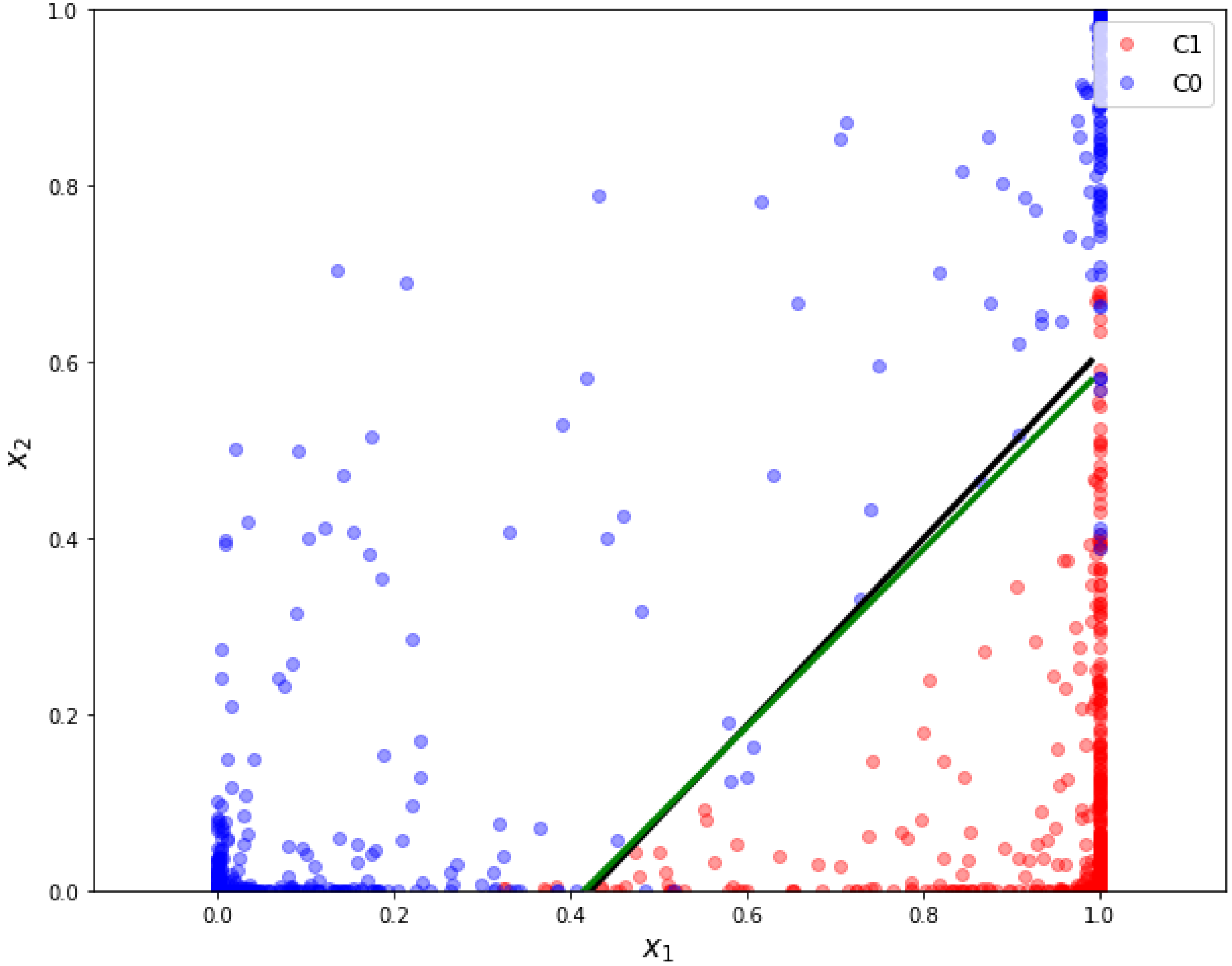
Understanding a Network's Behavior

- 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.

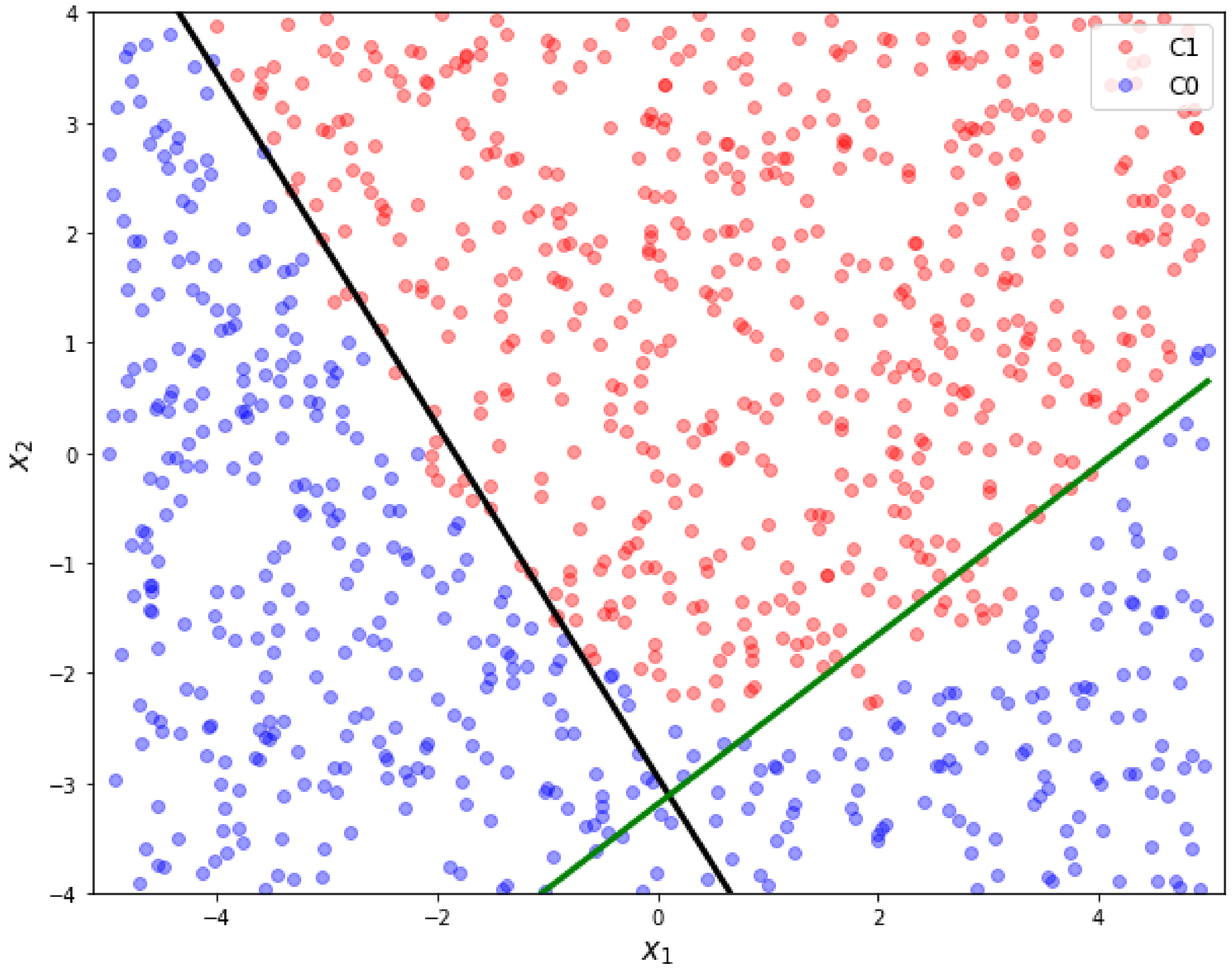
Nonlinearly Distributed Data



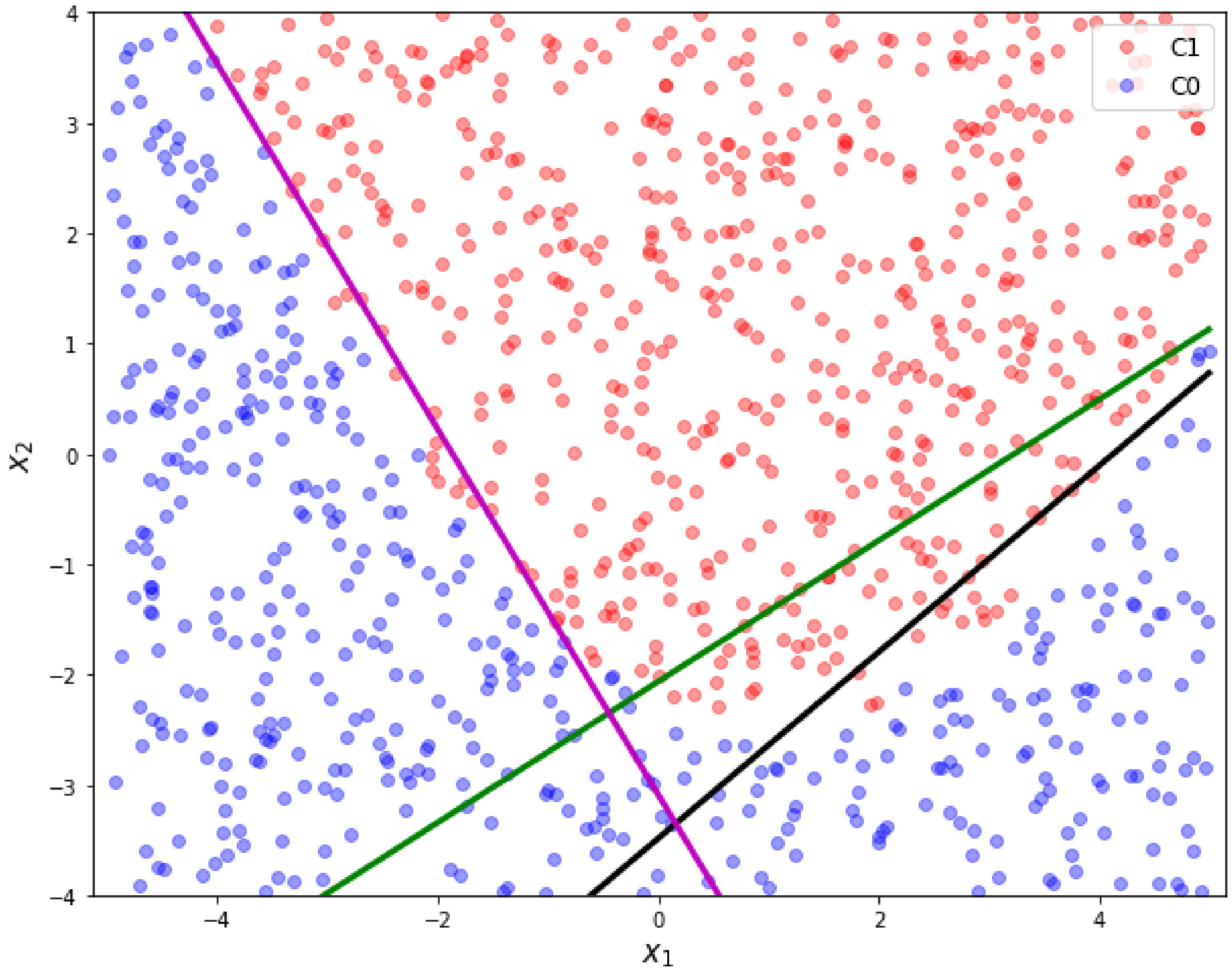
Multi Layers

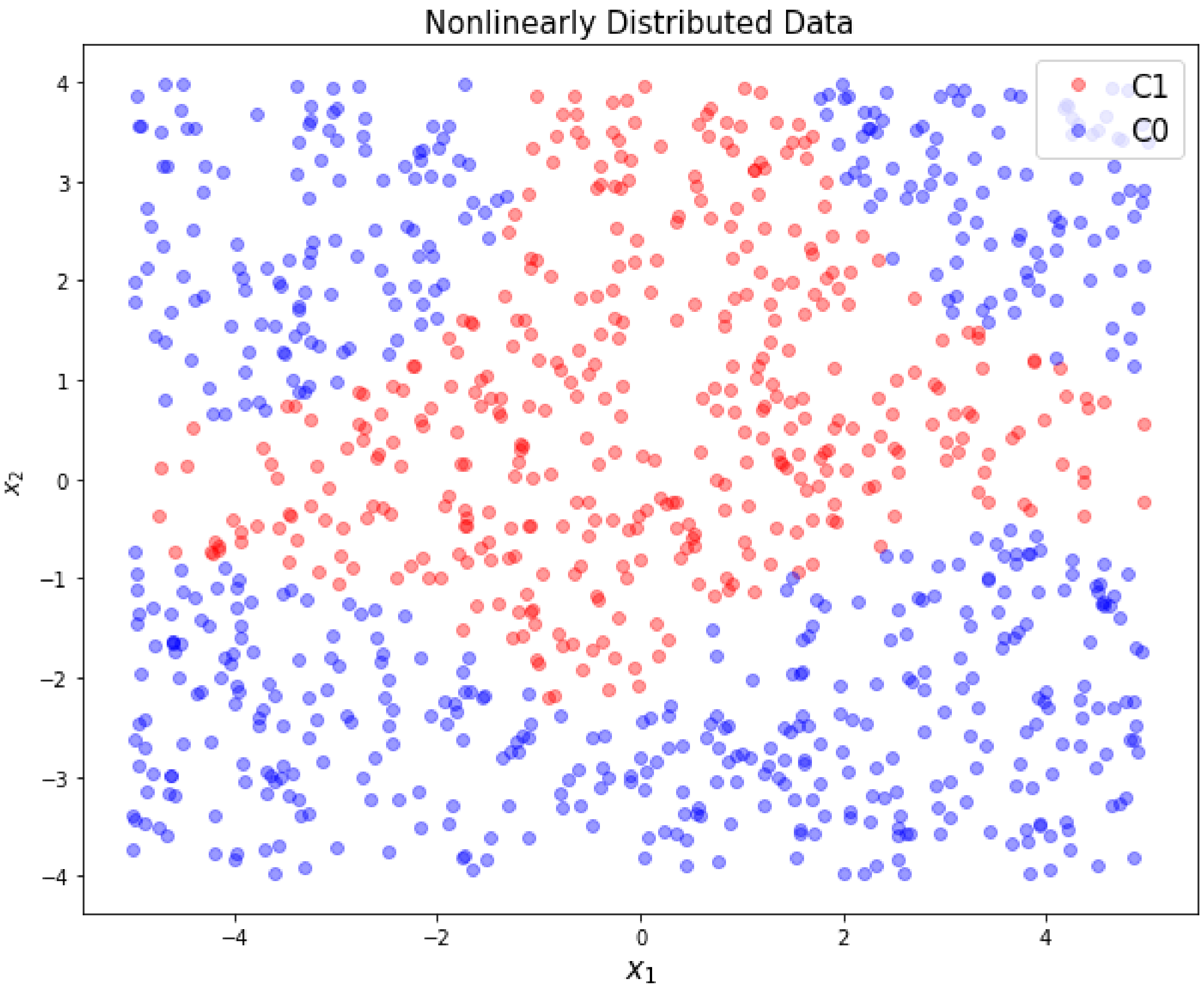


Multi Layers

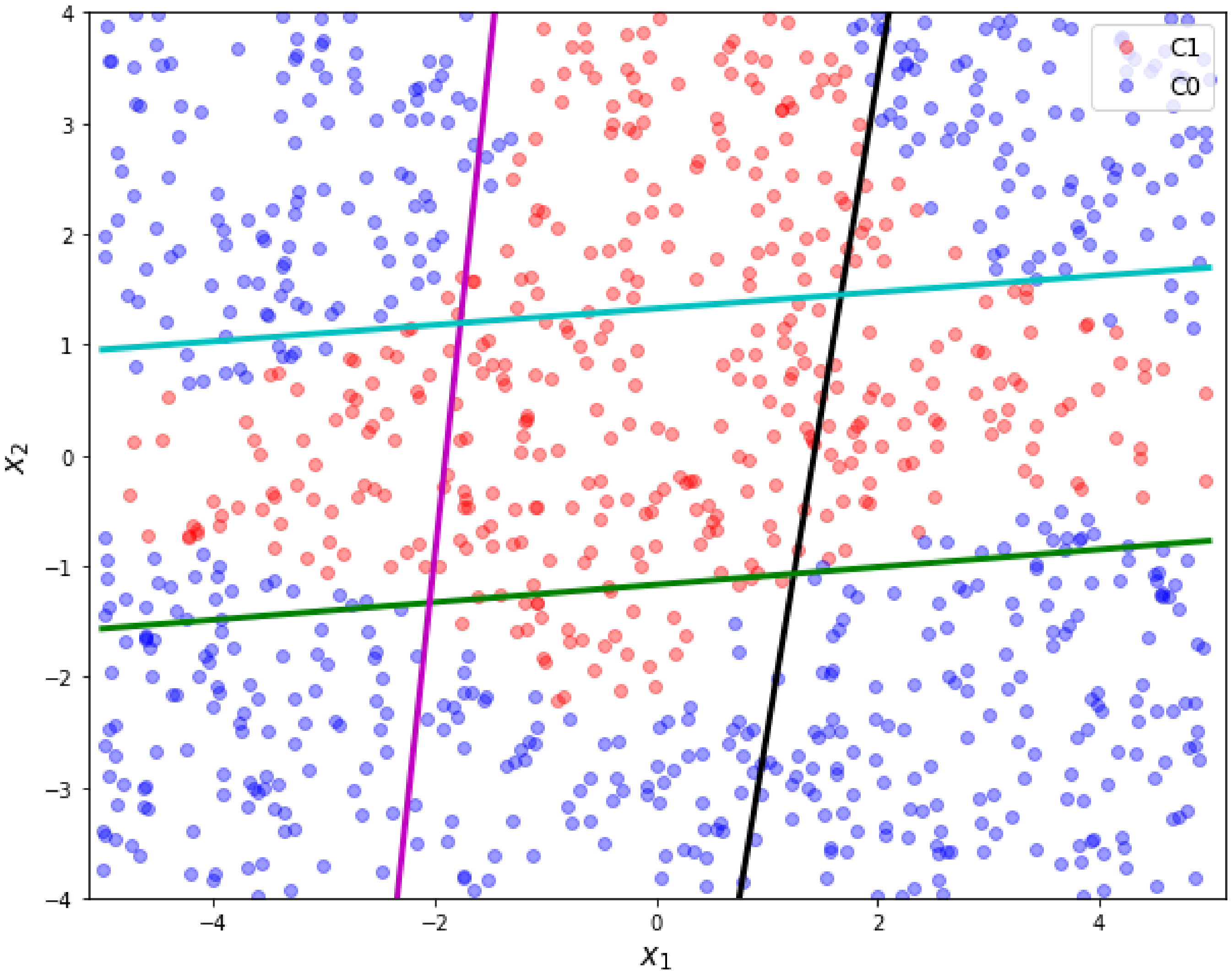


Multi Neurons

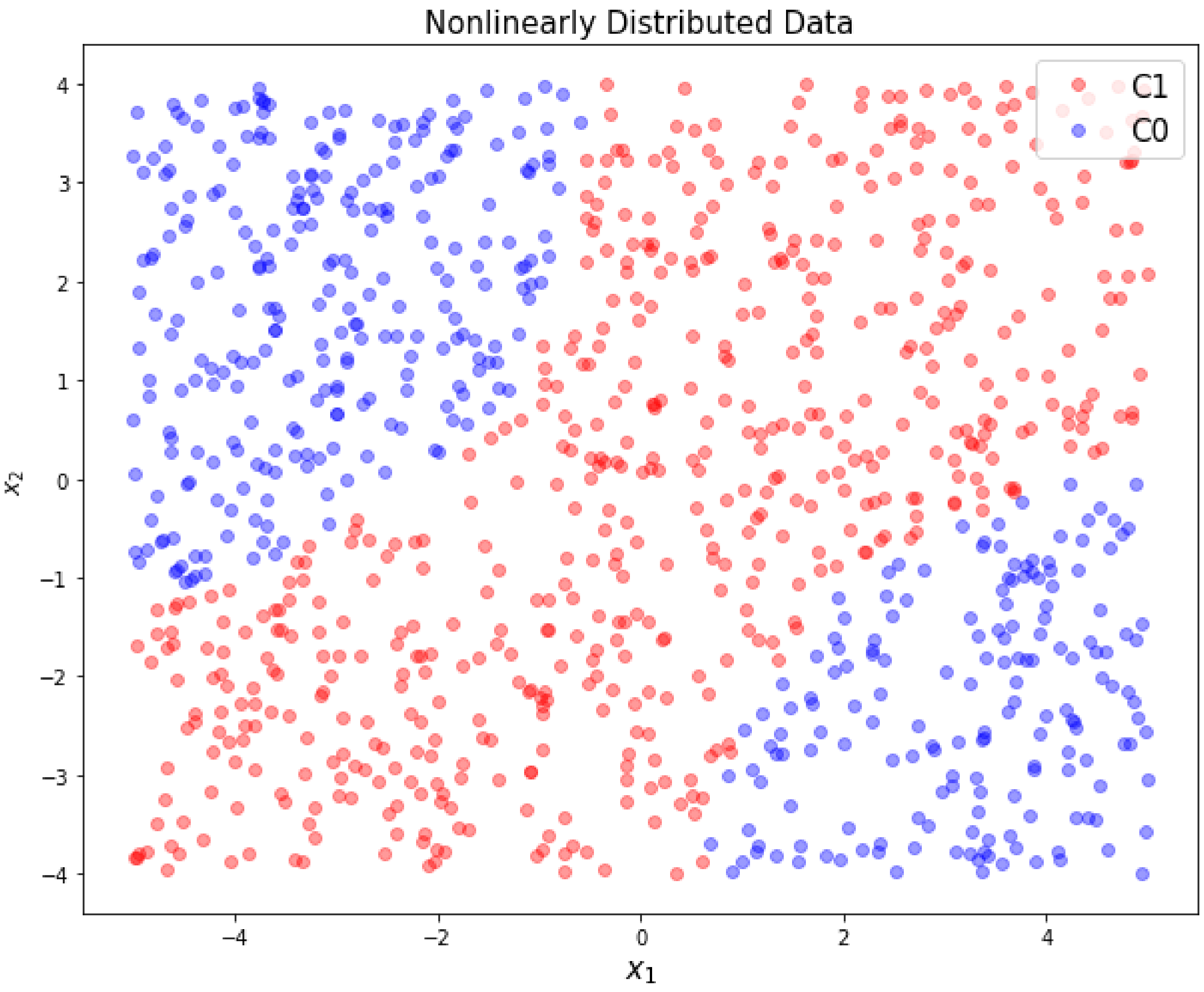




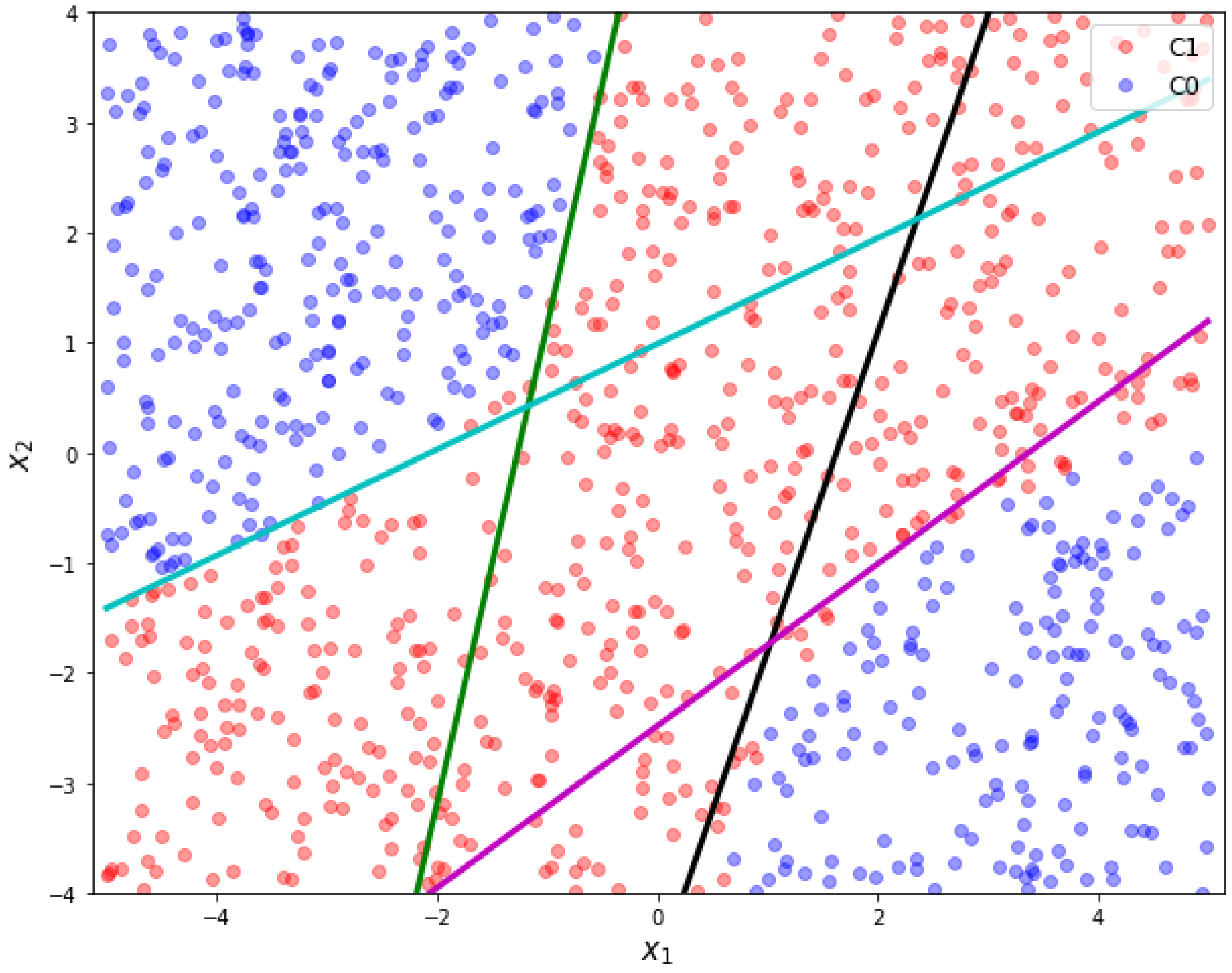
POSTECH



C1
C0
x_1
x_2

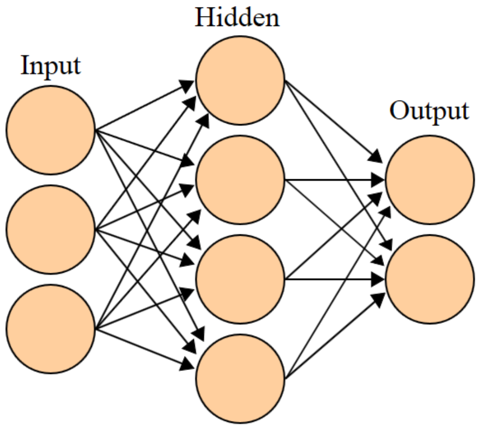


Nonlinearly Distributed Data
C1
C0
4
3
2
1
0
−1
−2
−3
−4
x_2
−4
−2
0
2
4
x_1


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

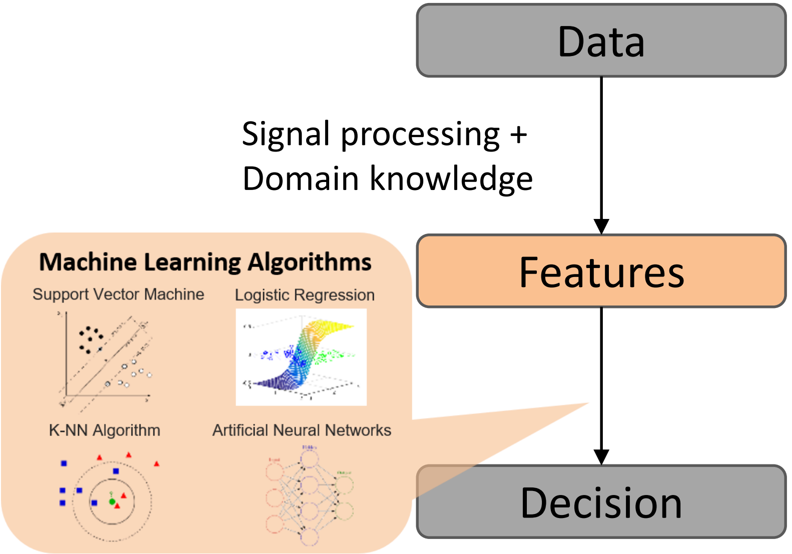
Summary

- Learning weights and biases from data using gradient descent



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

Machine Learning

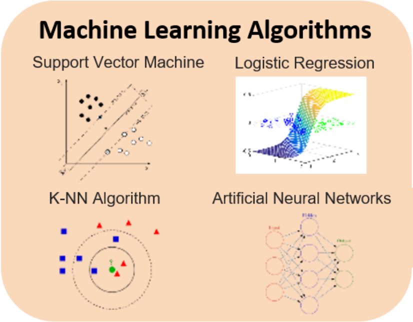


Machine Learning

Data

Signal processing +
Domain knowledge

Features



Decision

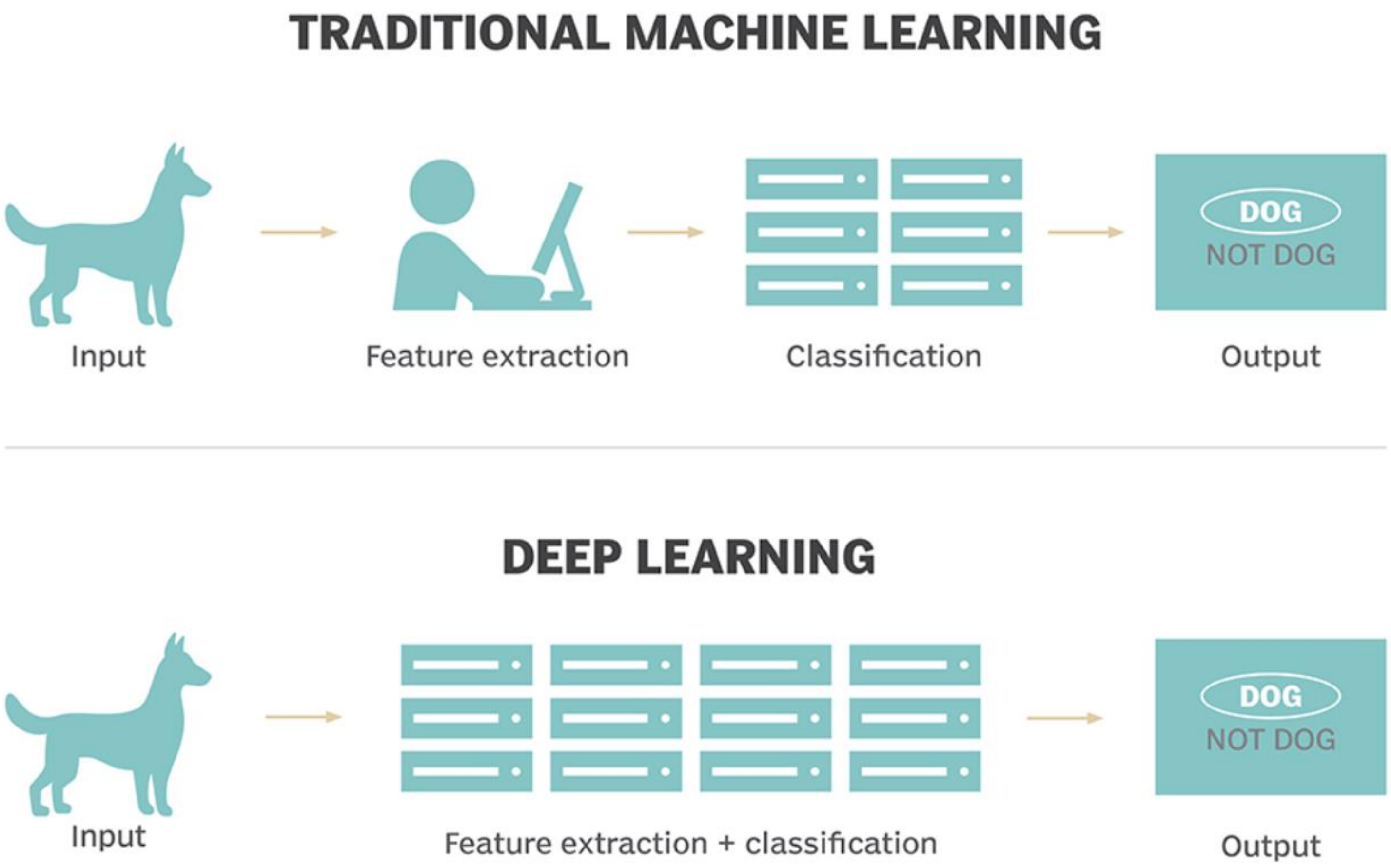
Deep Learning

Data

Less depends on
domain knowledge

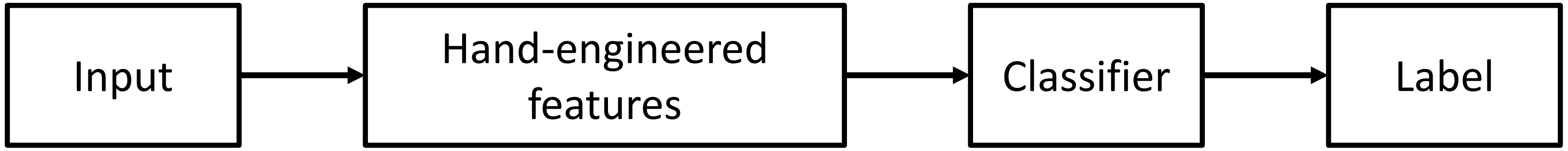
Decision

Recall Supervised Learning Setup

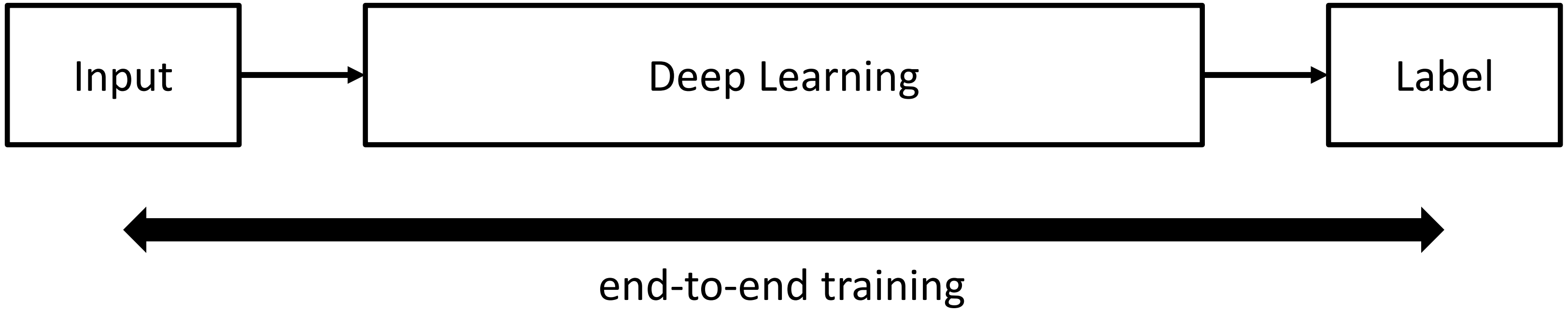


Machine Learning and Deep Learning

- Machine Learning



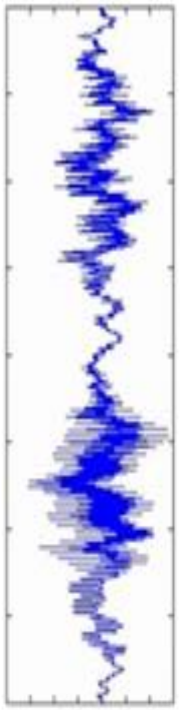
- Deep supervised learning



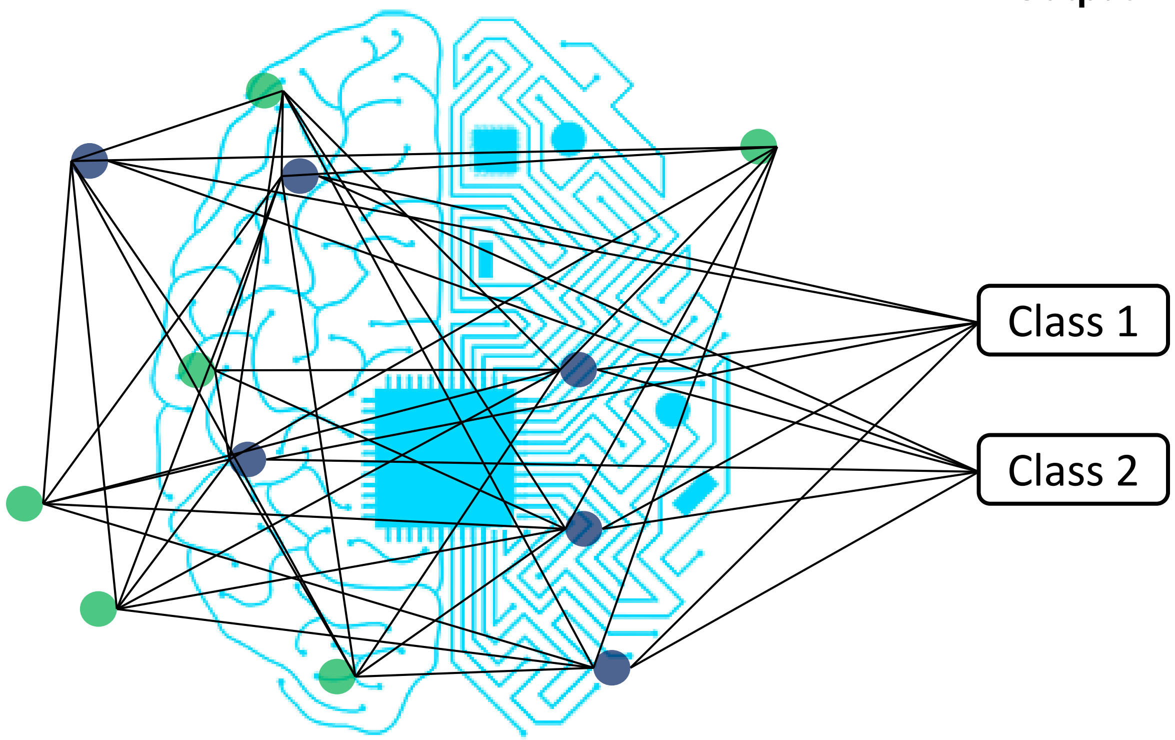
Artificial Neural Networks

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

Input



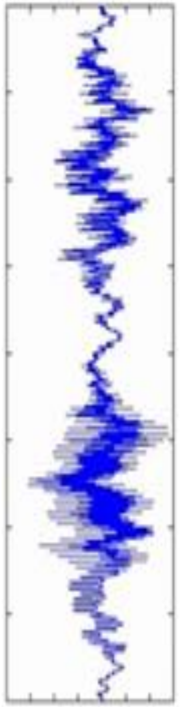
Output



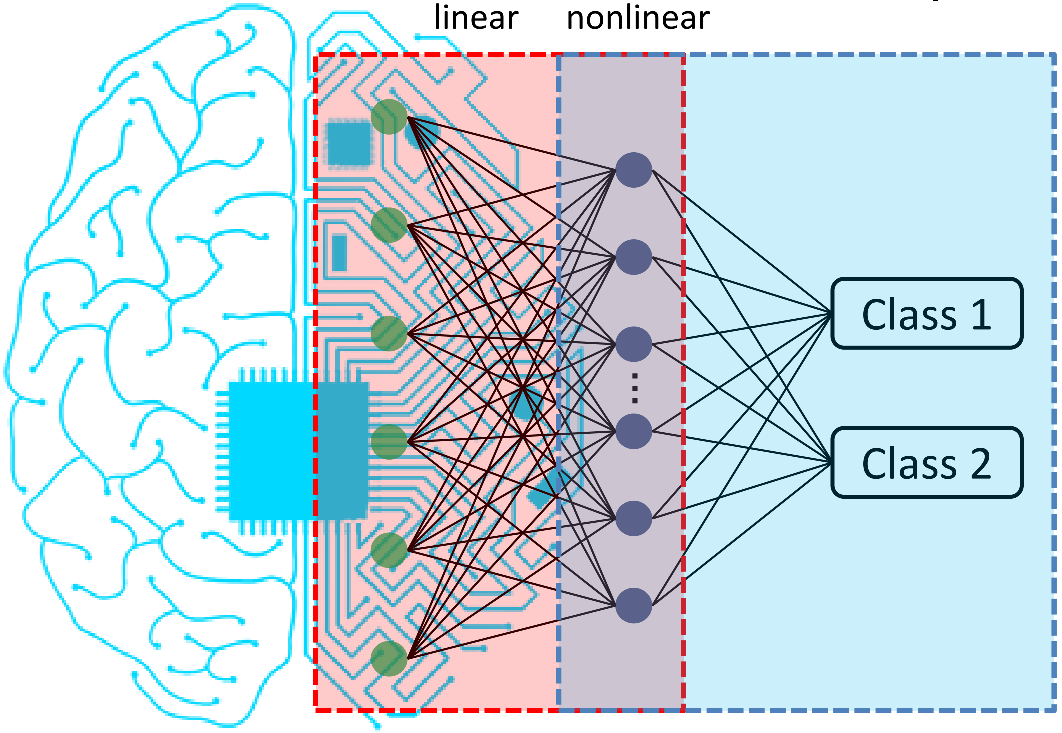
Artificial Neural Networks

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

Input



Output



Feature learning **Classification**

Deep Artificial Neural Networks

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