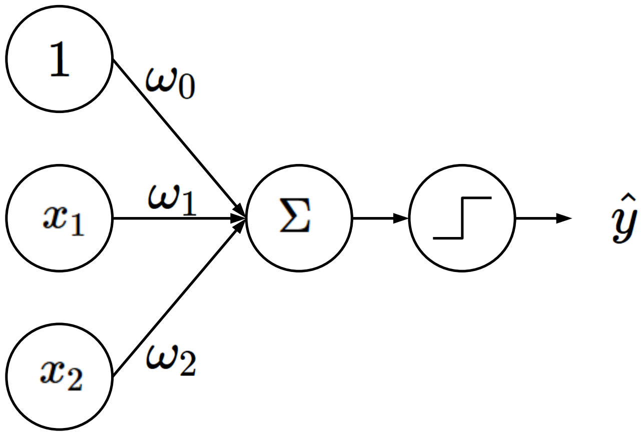
Artificial Neural Networks

Industrial AI Lab.
Prof. Seungchul Lee
Yunseob Hwang, Illjeok Kim

Artificial Neural Networks from MLP

Artificial Neural Networks: Perceptron

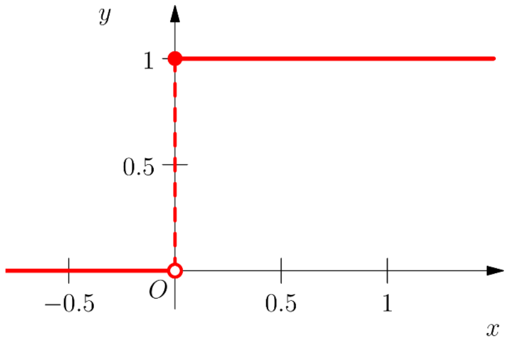
- Perceptron for $h(\theta)$ or $h(\omega)$
  - Neurons compute the weighted sum of their inputs
  - A neuron is activated or fired when the sum a is positive



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

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

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

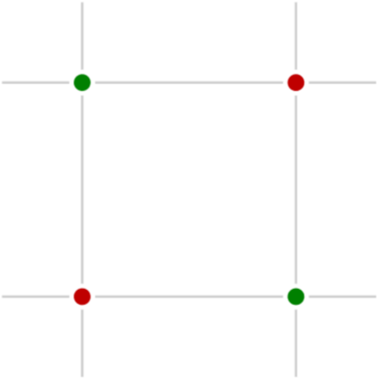


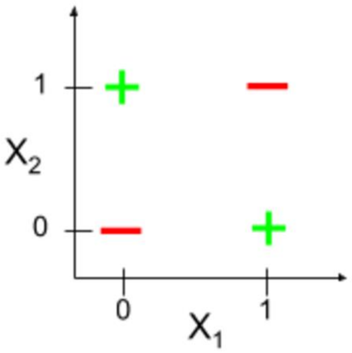
Here, a step function is illustrated instead of a sign function

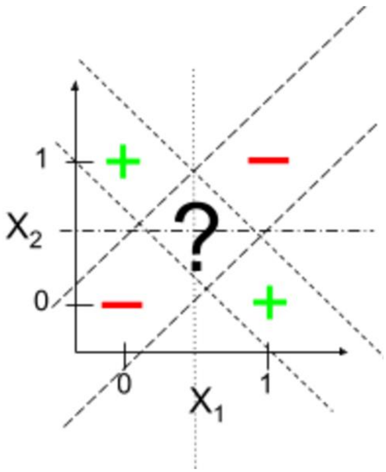
XOR Problem

- 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



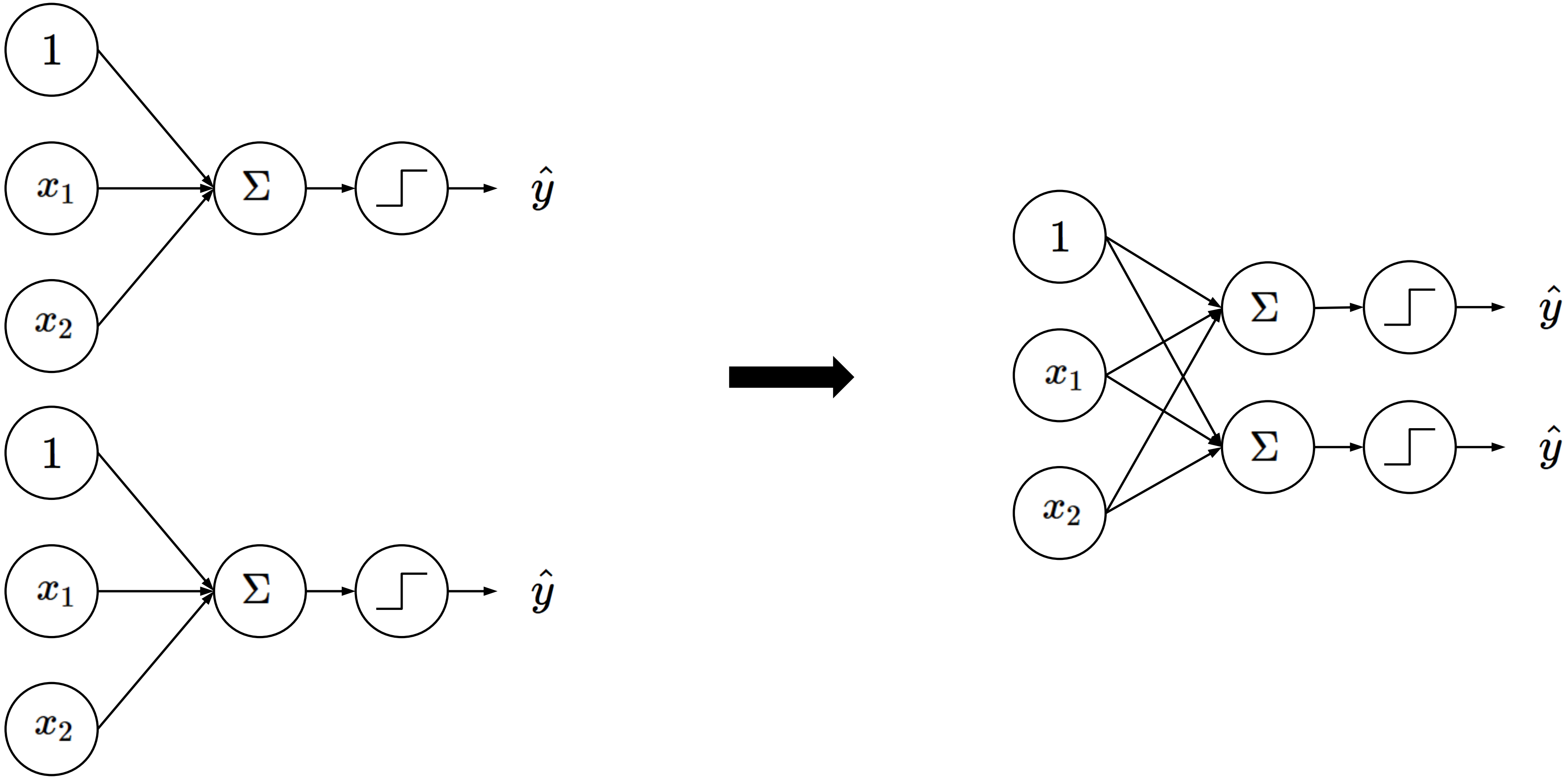




- Single neuron = one linear classification boundary

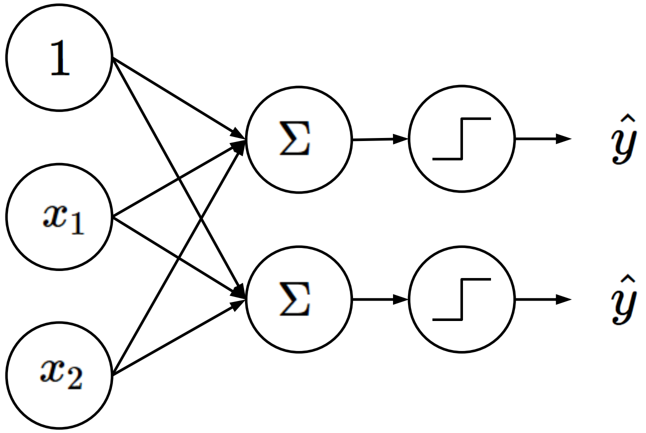
Artificial Neural Networks: MLP

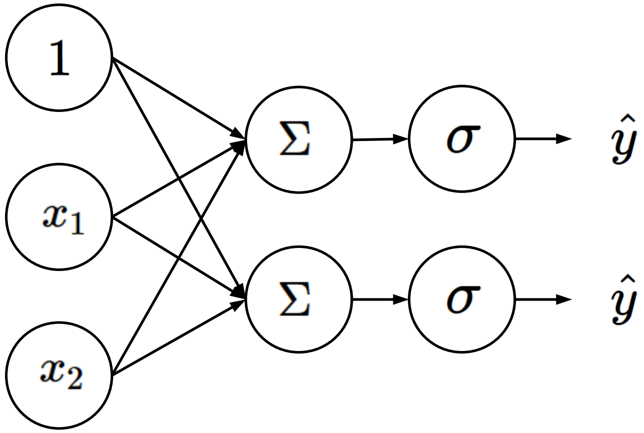
- Multi-layer Perceptron (MLP) = Artificial Neural Networks (ANN)
  - Multi neurons = multiple linear classification boundaries

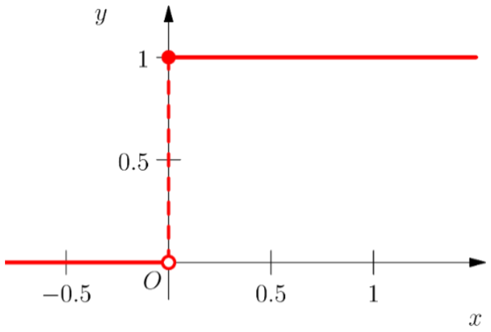


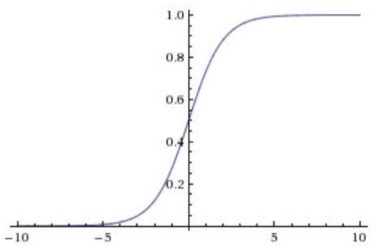
Artificial Neural Networks: Activation Function

- Differentiable nonlinear activation function







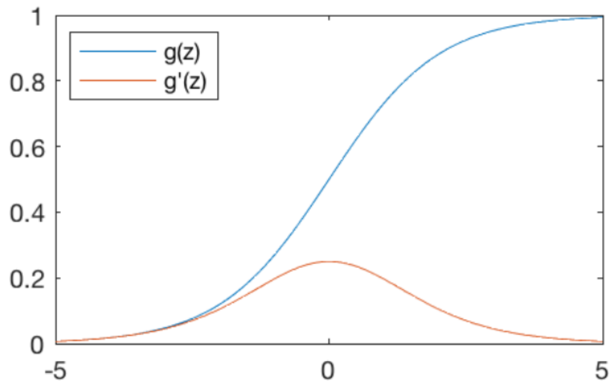


$$g(x) = \frac{1}{1 + e^{-x}}$$

Common Activation Functions

Discuss later

Sigmoid Function

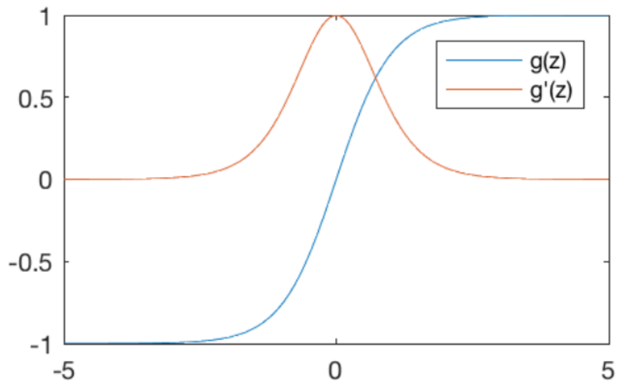


$$g(z) = \frac{1}{1 + e^{-z}}$$

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

```
tf.nn.sigmoid(z)
```

Hyperbolic Tangent

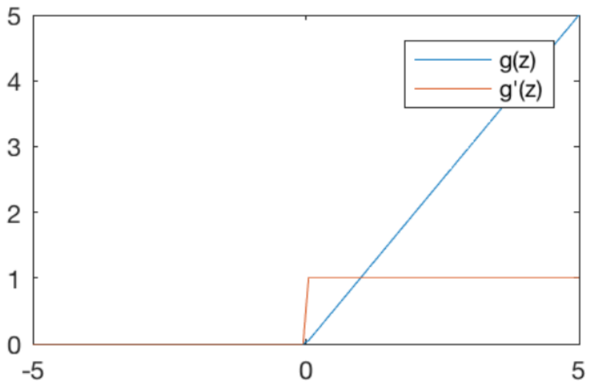


$$g(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

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

```
tf.nn.tanh(z)
```

Rectified Linear Unit (ReLU)



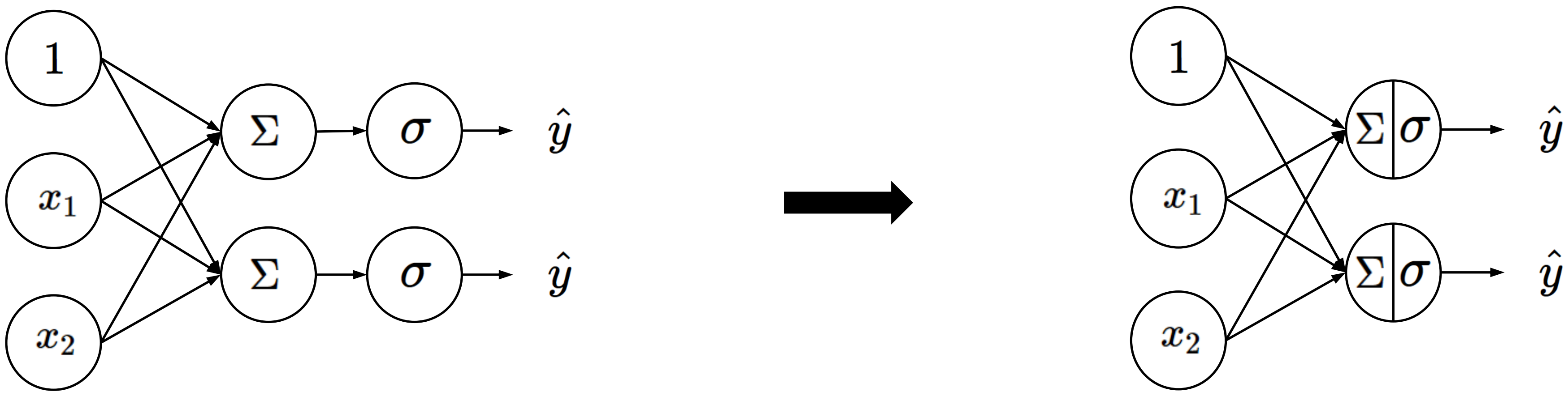
$$g(z) = \max(0, z)$$

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

```
tf.nn.relu(z)
```

Artificial Neural Networks

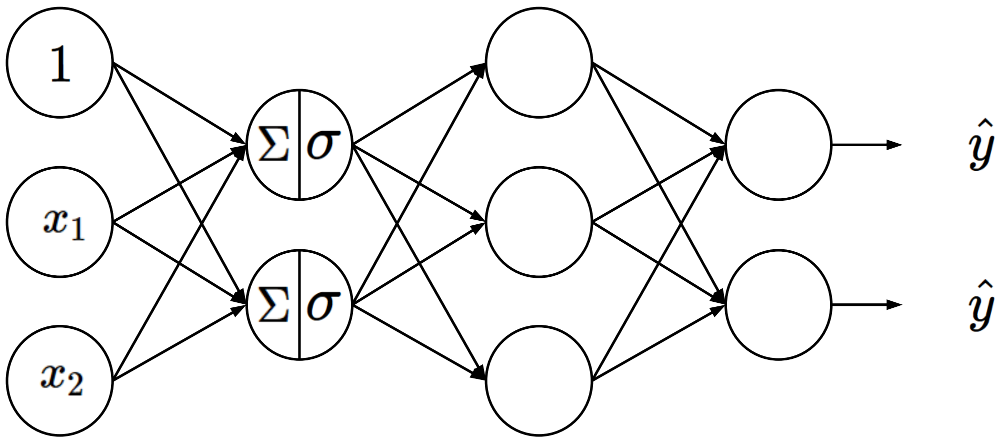
- In a compact representation



Artificial Neural Networks

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

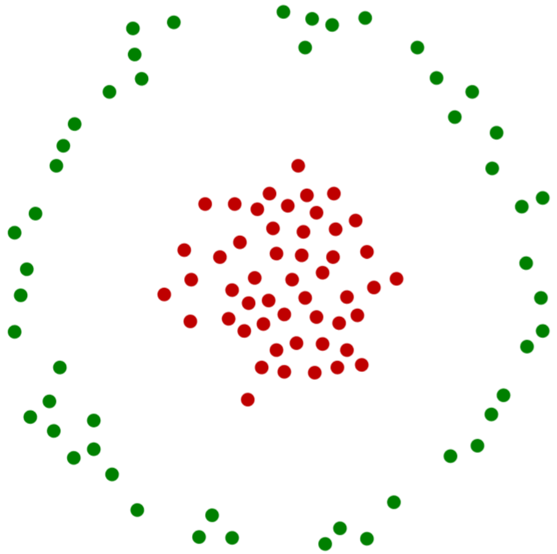
  $\Longrightarrow$ perceptron with many layers and units

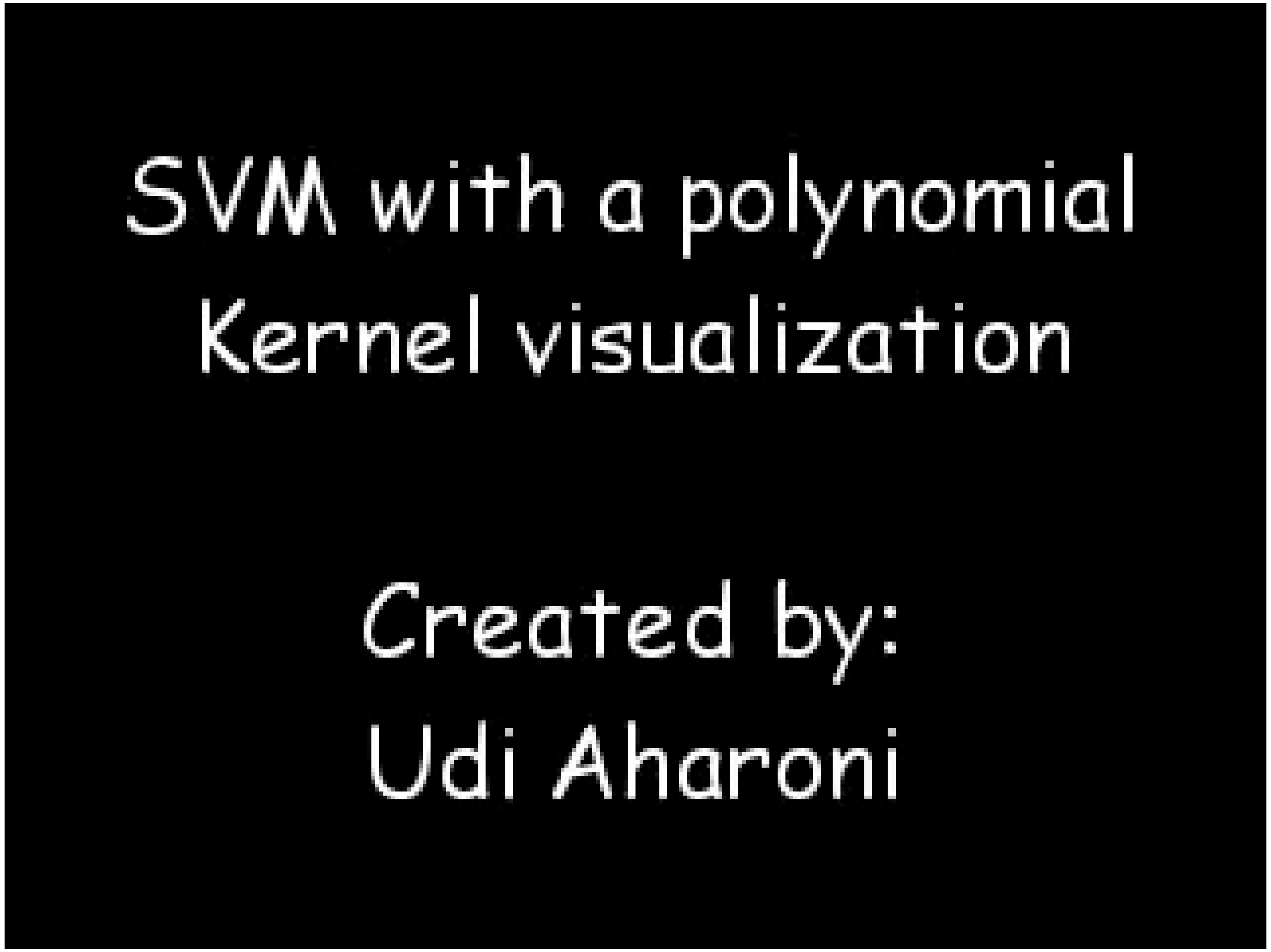


- Multi-layer perceptron
  - Features of features
  - Mapping of mappings

ANN as Kernel Learning

Nonlinear Classification



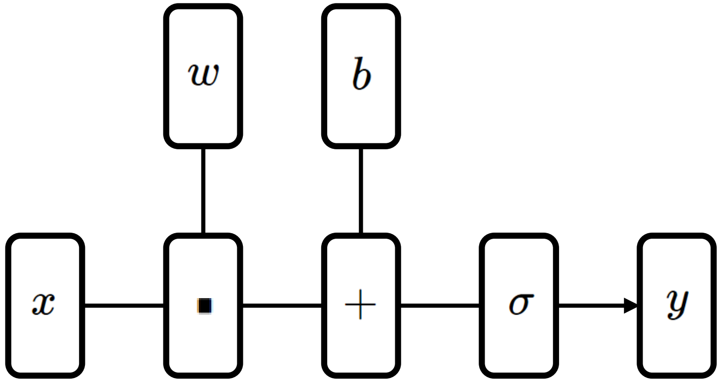


https://www.youtube.com/watch?v=3liCbRZPrZA

Neuron

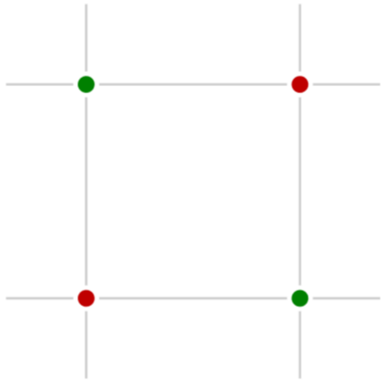
- We can represent this “neuron” as follows:

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



XOR Problem

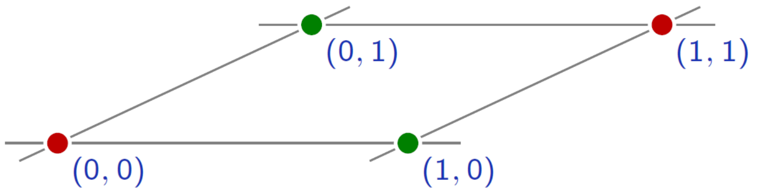
- The main weakness of linear predictors is their lack of capacity.
- For classification, the populations have to be linearly separable.



"xor"

Nonlinear Mapping

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

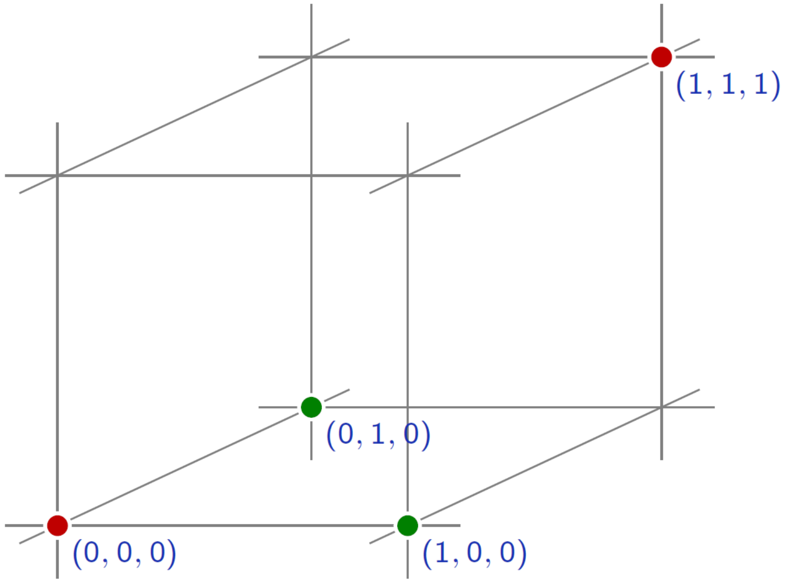


Source: Dr. Francois Fleuret at EPFL

Nonlinear Mapping

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

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

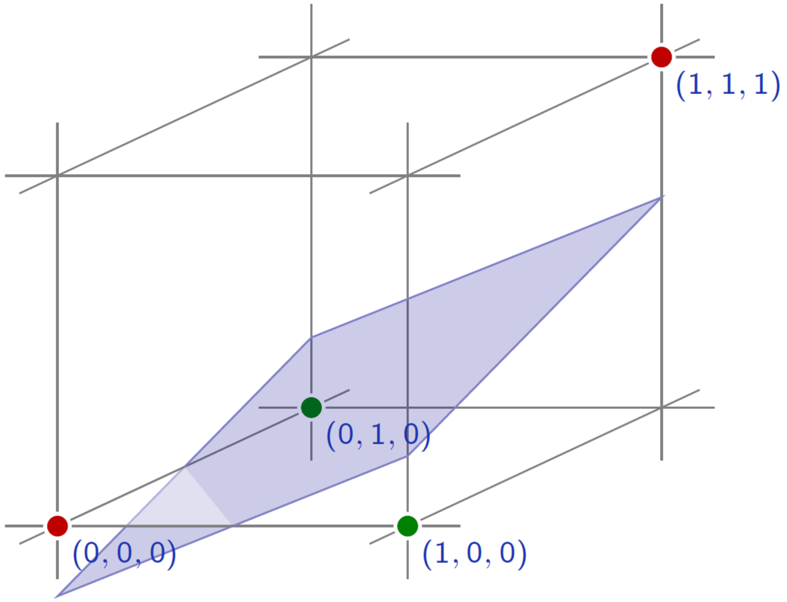


Source: Dr. Francois Fleuret at EPFL

Nonlinear Mapping

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

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



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 not be separable by a linear boundary

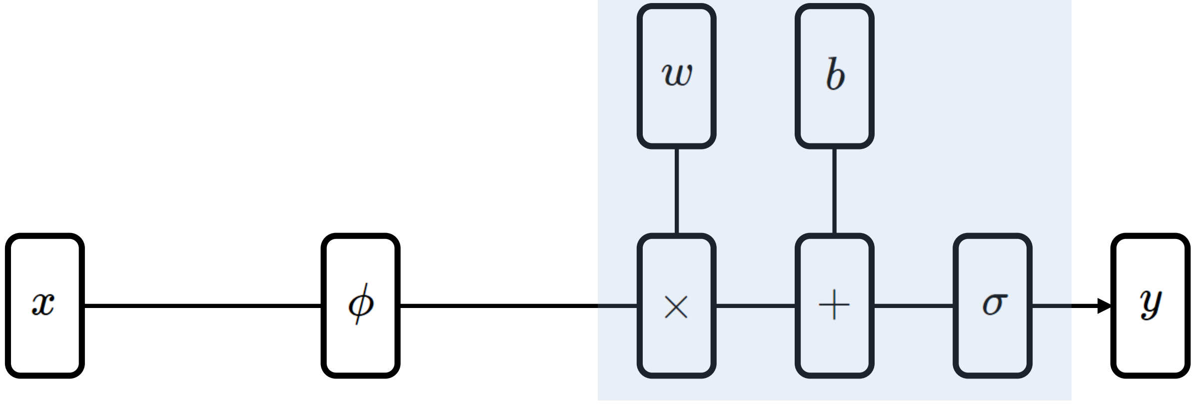
- Linear models (e.g. linear regression, linear SVM) are not just rich enough
  - by mapping data to higher dimensions where it exhibits linear patterns
  - apply the linear model in the new input feature space
  - mapping = changing the feature representation

- Kernels: make linear model work in nonlinear settings

Kernel + Neuron

- Nonlinear mapping + neuron

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

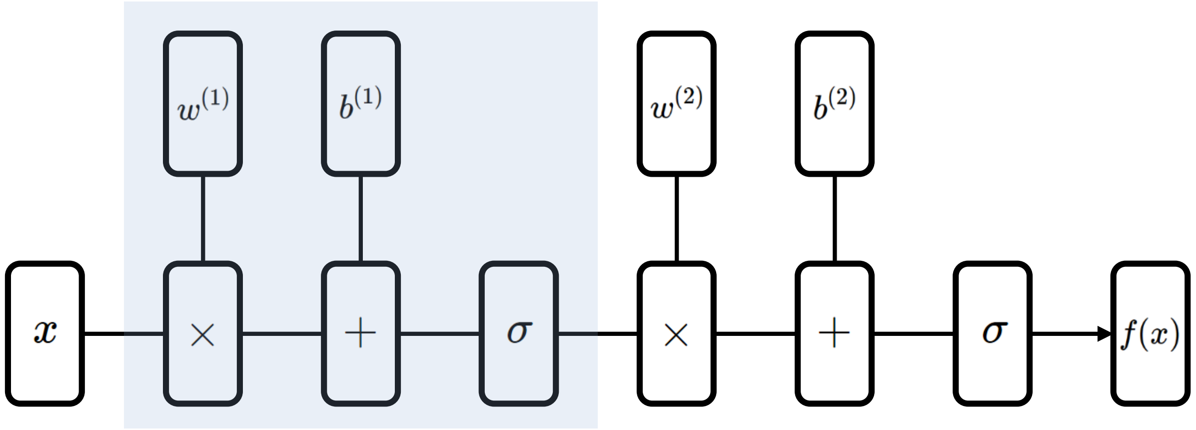


ϕ: Kernel

Perceptron

Neuron + Neuron

- Nonlinear mapping can be represented by another neurons



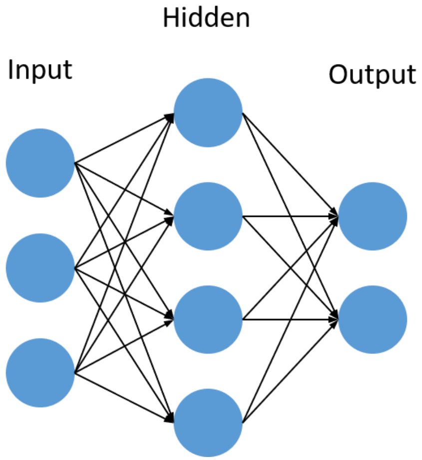
ϕ: Kernel

- Nonlinear Kernel
  - Nonlinear activation functions

Summary

- Universal function approximator
- Universal function classifier

- Parameterized

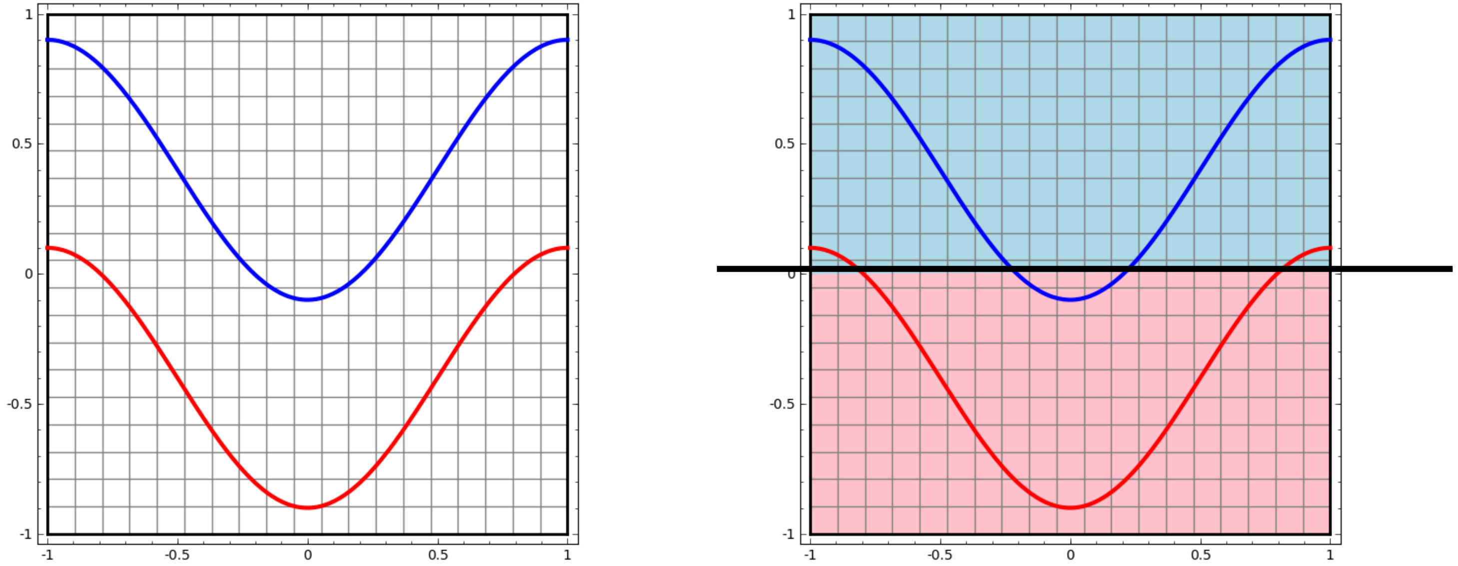


$$x \longrightarrow \boxed{\hat{y} = f_{\omega_1,\cdots,\omega_k}(x)} \longrightarrow y$$

Looking at Hidden Layers

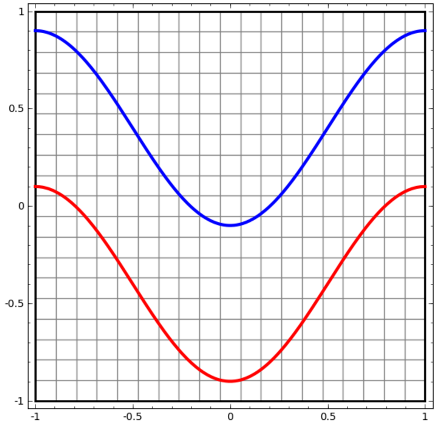
Example: Linear Classifier

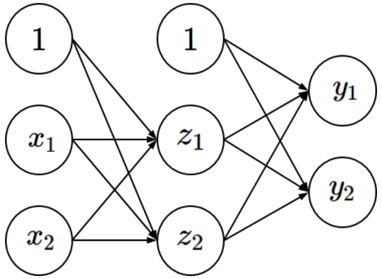
- Perceptron tries to separate the two classes of data by dividing them with a line

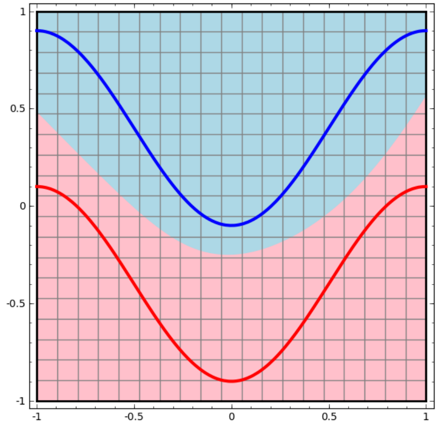


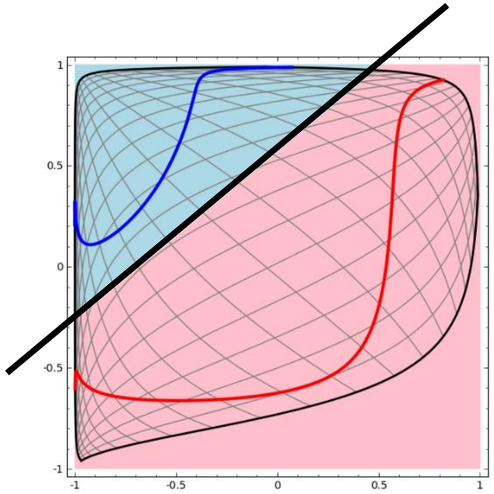
Example: Neural Networks

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

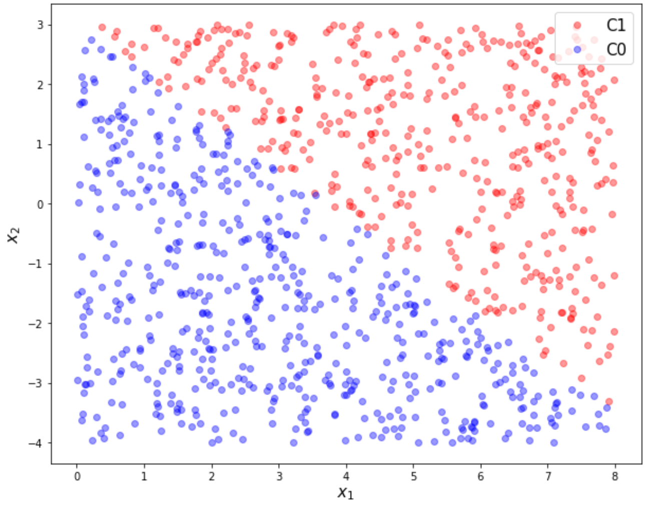




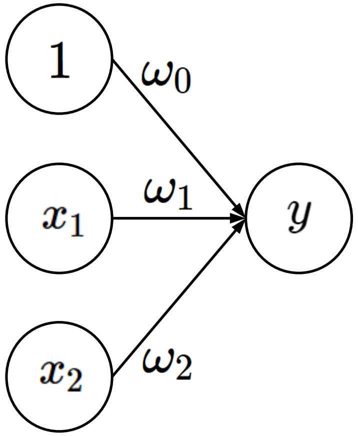


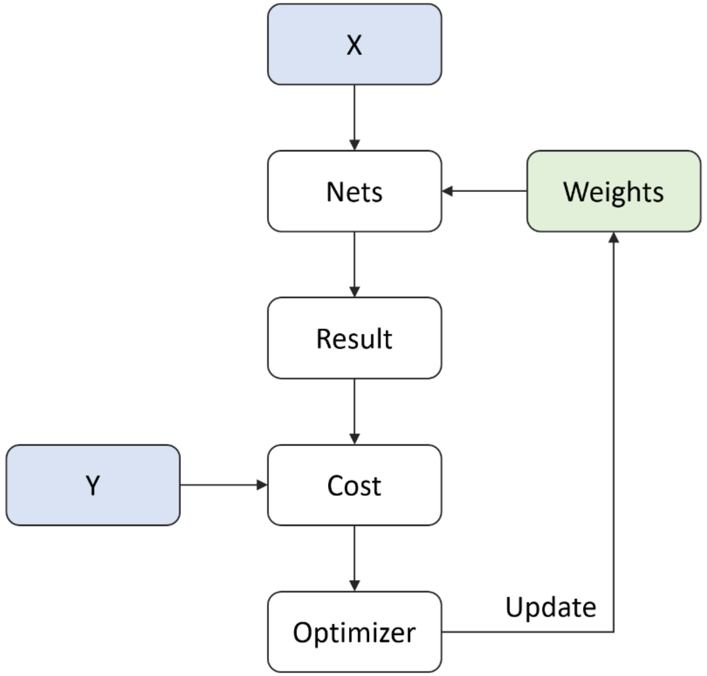


Logistic Regression in a Form of Neural Network



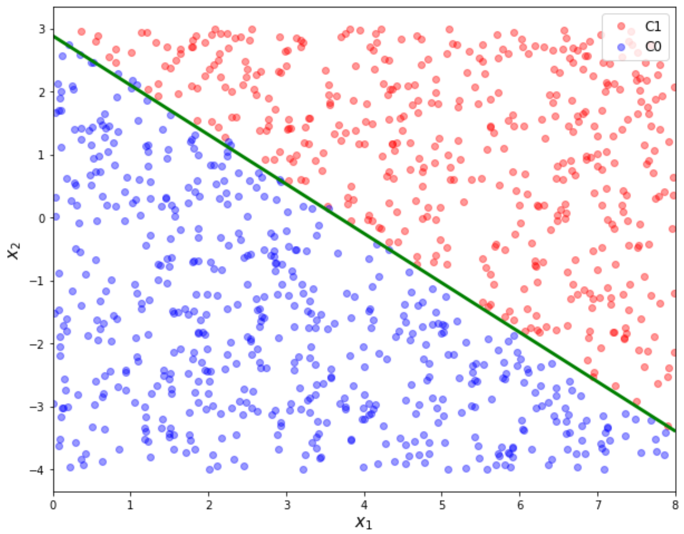
$$y = \sigma\left(\omega_0 + \omega_1 x_1 + \omega_2 x_2\right)$$





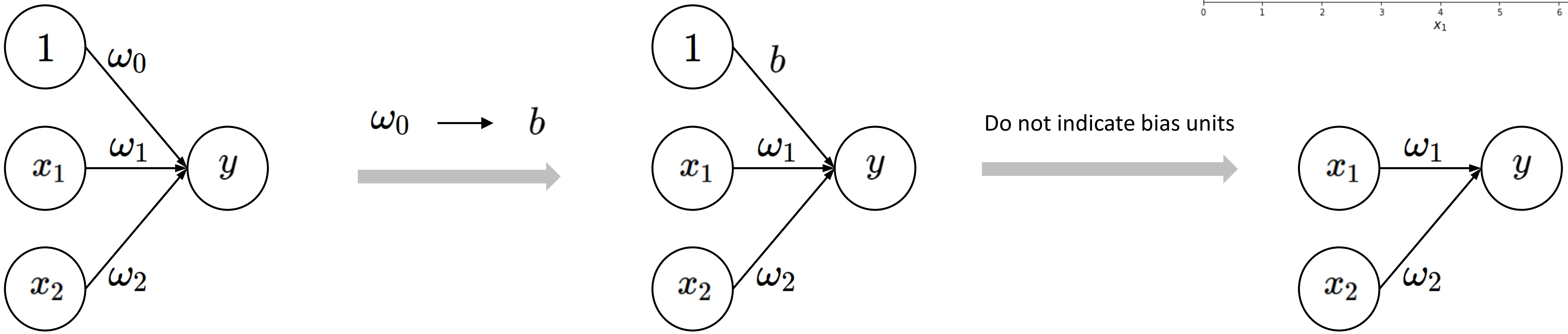
Logistic Regression in a Form of Neural Network

- Neural network convention



$$y = \sigma\left(\omega_0 + \omega_1 x_1 + \omega_2 x_2\right)$$

$$y = \sigma\left(b + \omega_1 x_1 + \omega_2 x_2\right)$$

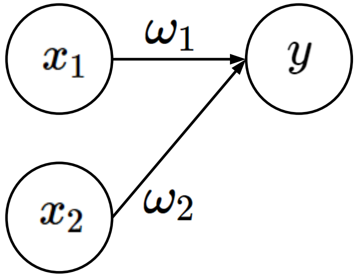



```
n_input = 2
n_output = 1
```

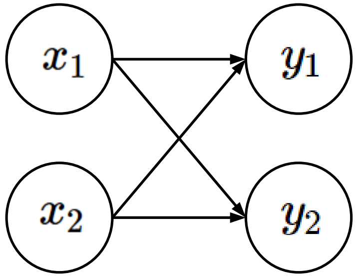
Logistic Regression in a Form of Neural Network

- One-hot encoding
  - One-hot encoding is a conventional practice for a multi-class classification

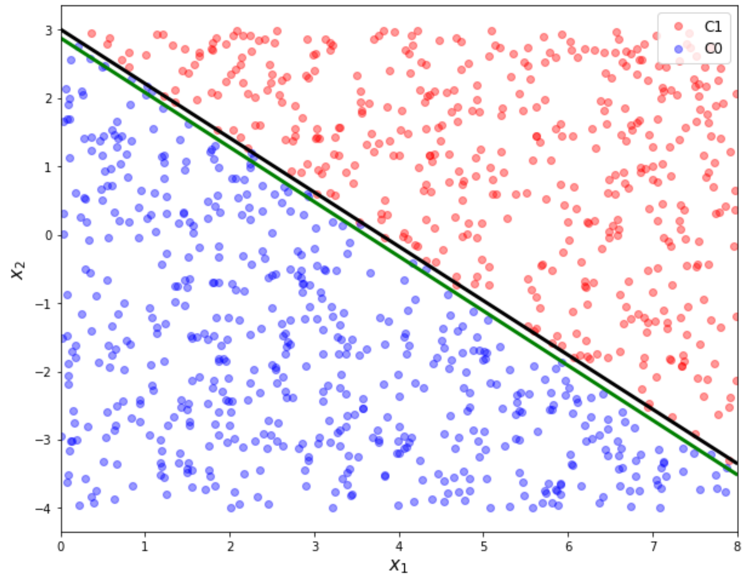
$$y^{(i)} \in \{1, 0\} \quad \Longrightarrow \quad y^{(i)} \in \{[0,1], [1,0]\}$$



One-hot encoding

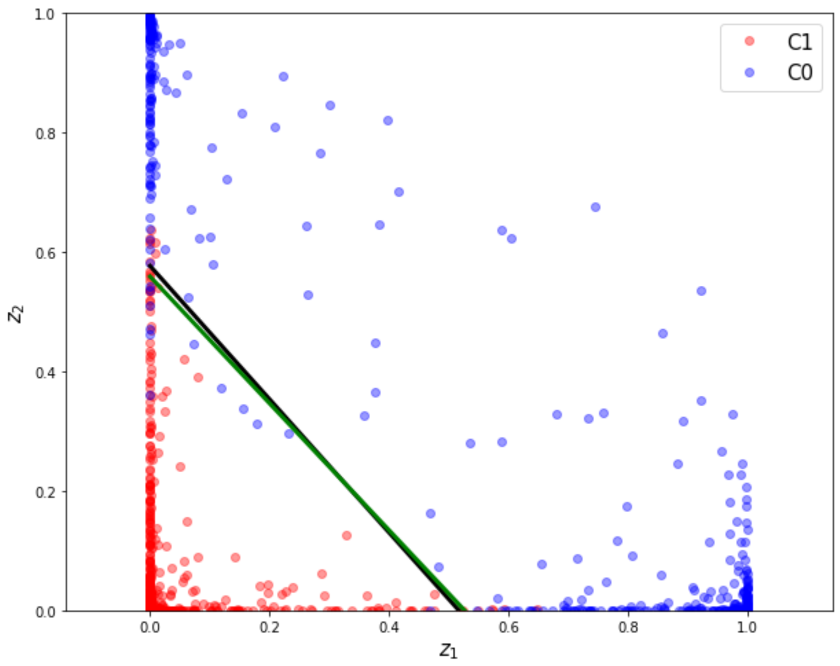


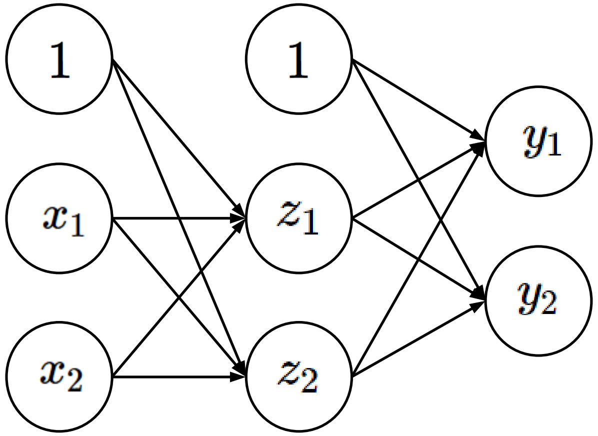

```
n_input = 2
n_output = 2
```



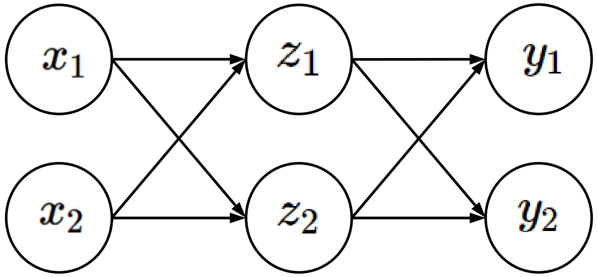
Multi Layers

- z space





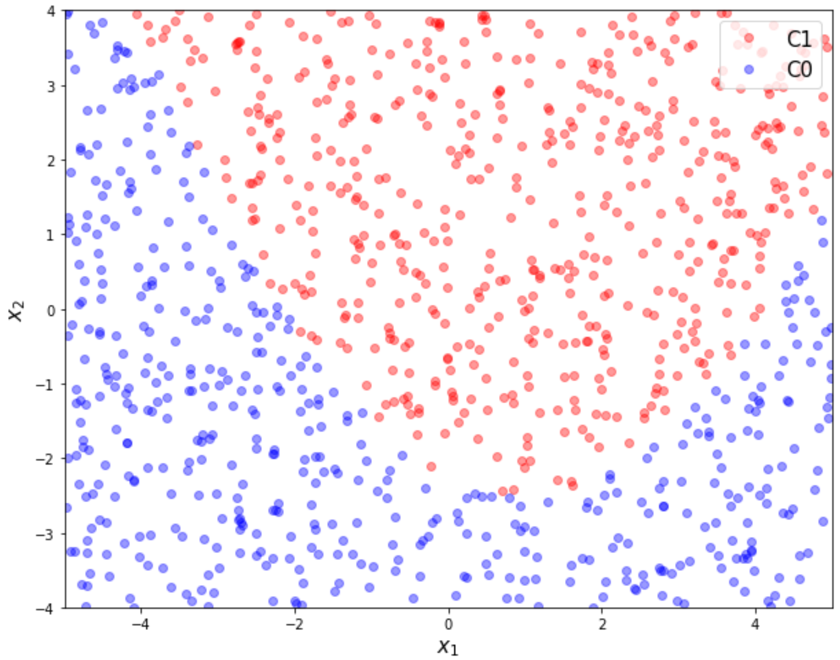
Do not include bias units

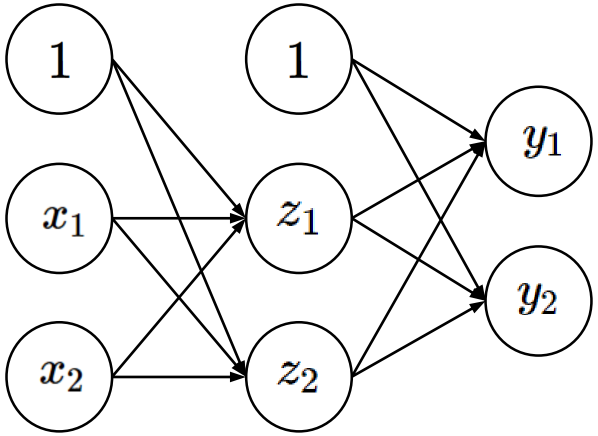



```
n_input = 2
n_hidden = 2
n_output = 2
```

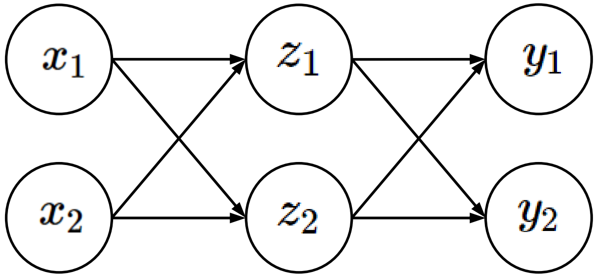
Nonlinearly Distributed Data

- Example to understand network's behavior
  - Include a hidden layer





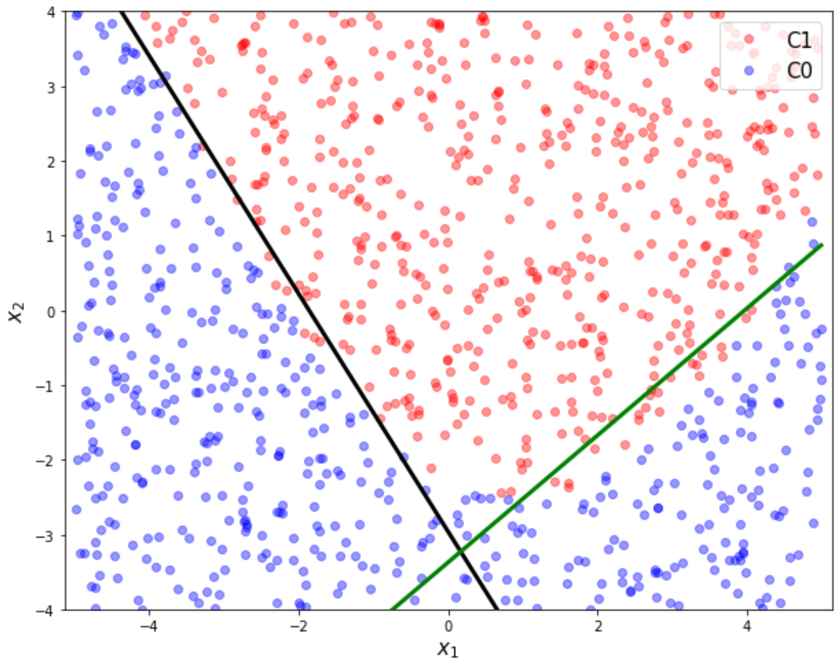
Do not include bias units

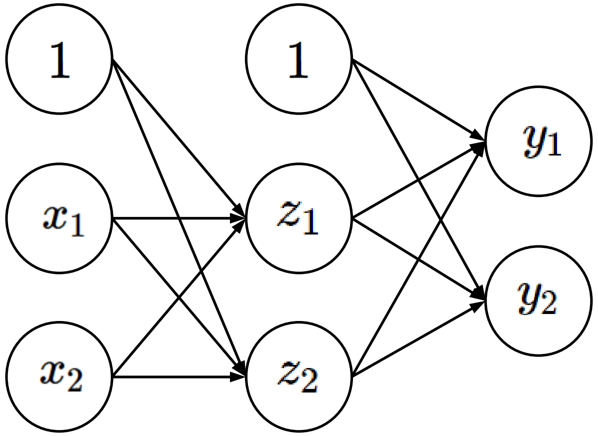



```
n_input  = 2
n_hidden  = 2
n_output  = 2
```

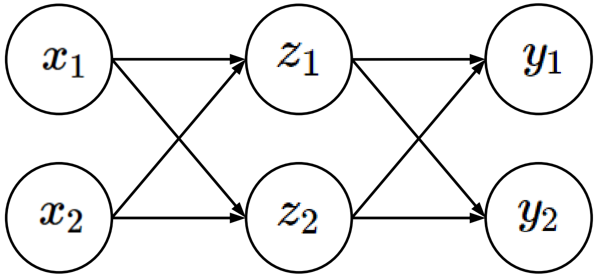
Multi Layers

- x space





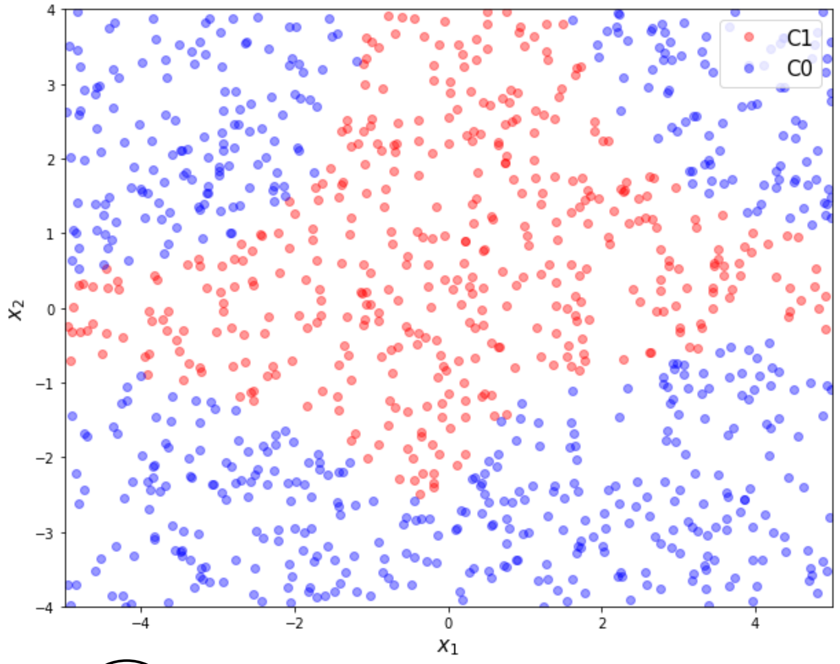
Do not include bias units

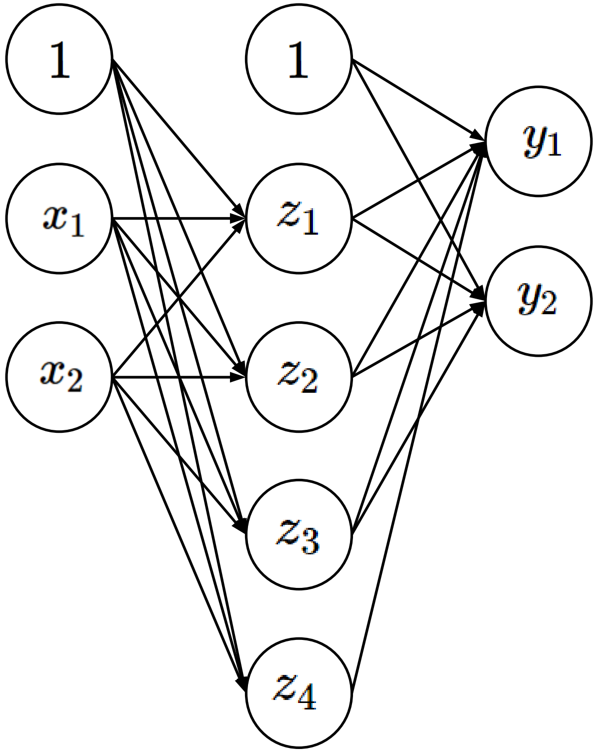



```
n_input  = 2
n_hidden  = 2
n_output  = 2
```

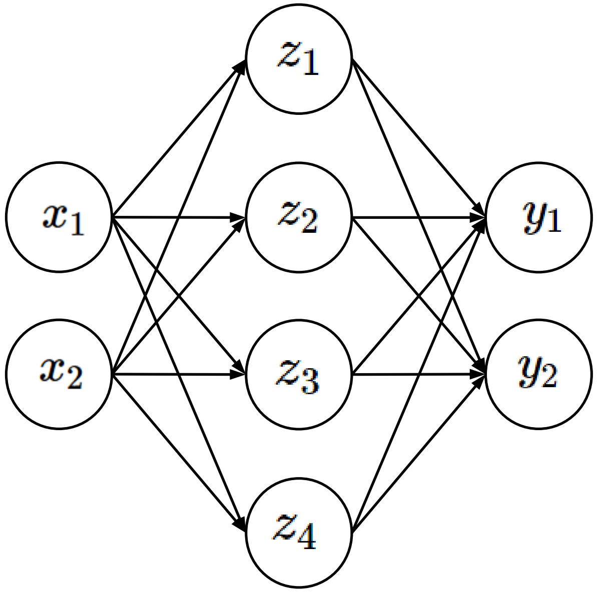
Nonlinearly Distributed Data

- More neurons in hidden layer





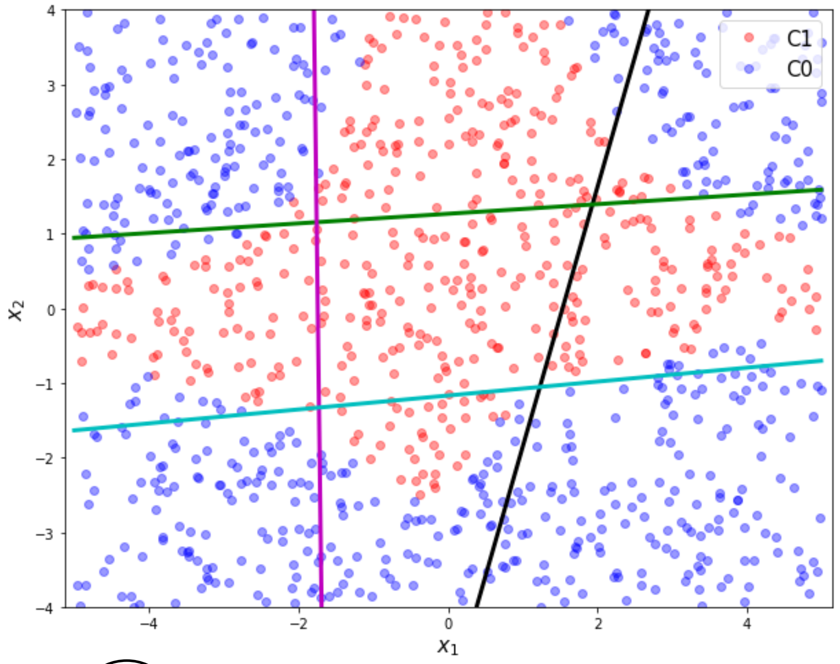
Do not include bias units

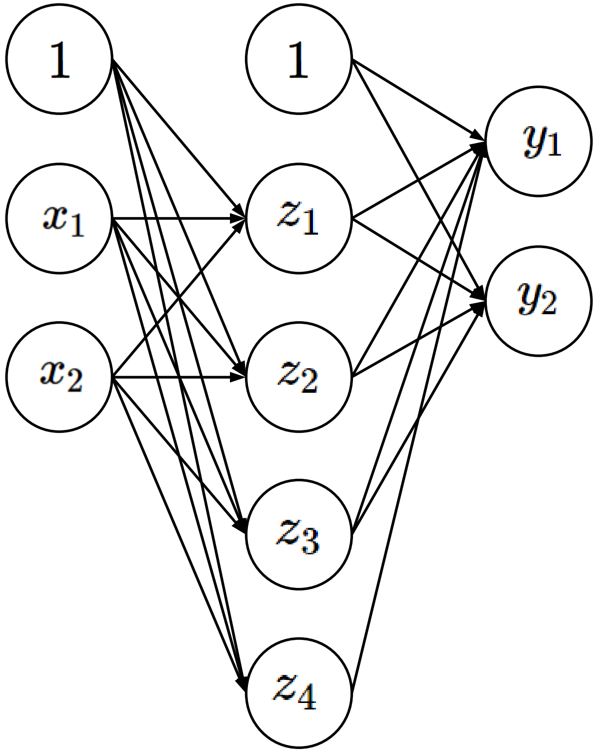



```
n_input = 2
n_hidden = 4
n_output = 2
```

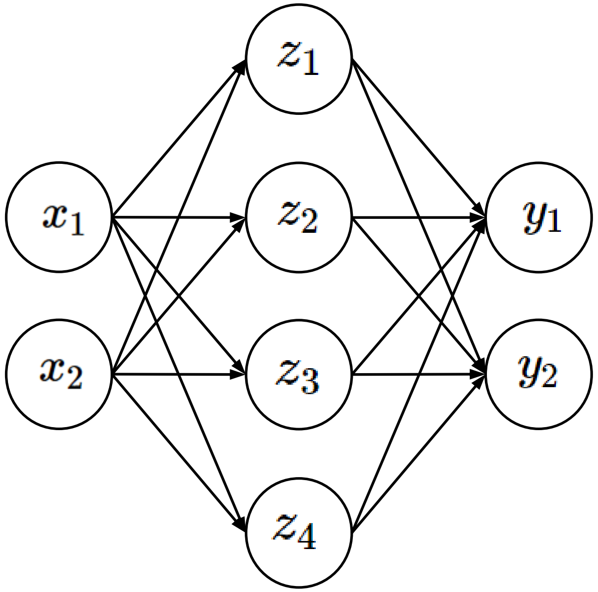
Multi Layers

- Multiple linear classification boundaries





Do not include bias units




```
n_input = 2
n_hidden = 4
n_output = 2
```