UNIVERSIDADE CATÓLICA PORTUGUESA

BRAGA

Deep Learning

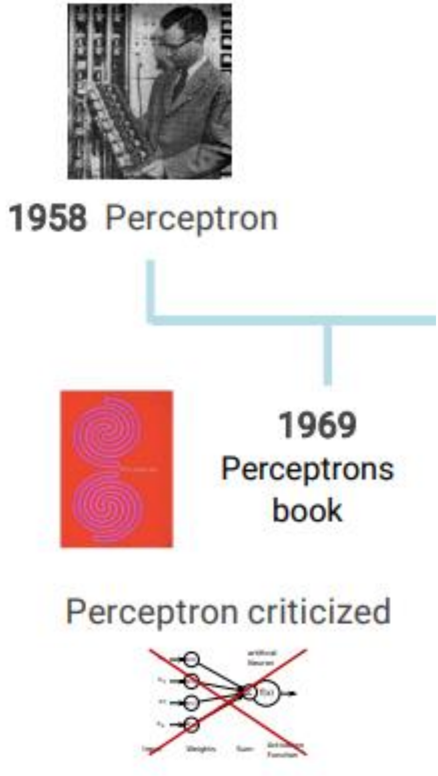
Session 3

The Perceptron

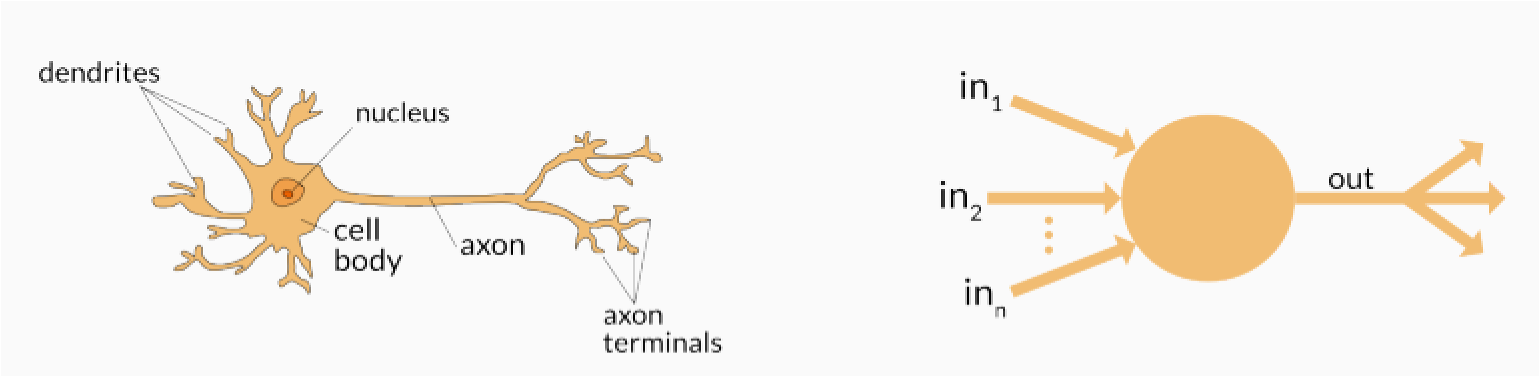
Applied Data Science

2024/2025

Perceptron



1958 Perceptron
1969
Perceptrons
book
Perceptron criticized



dendrites
nucleus
cell
body
axon
axon
terminals
in$_1$
in$_2$
in$_n$
out

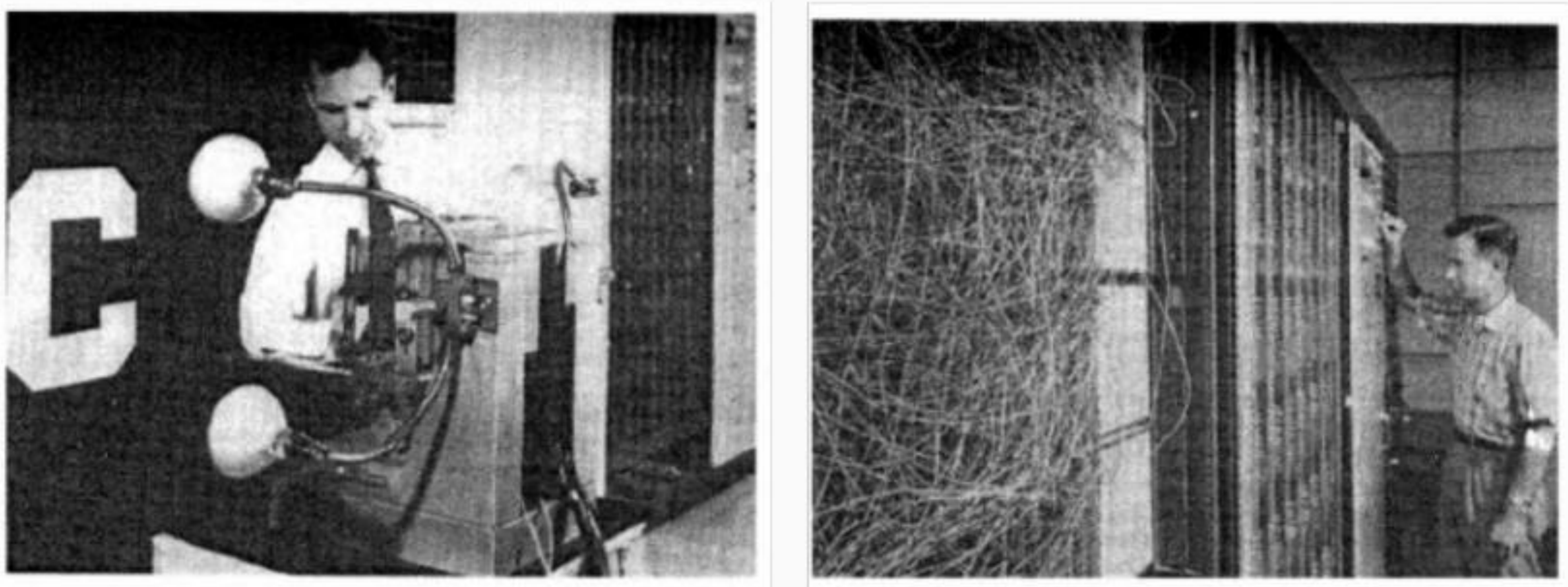
Perceptron

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

- First binary classifier based on supervised learning;

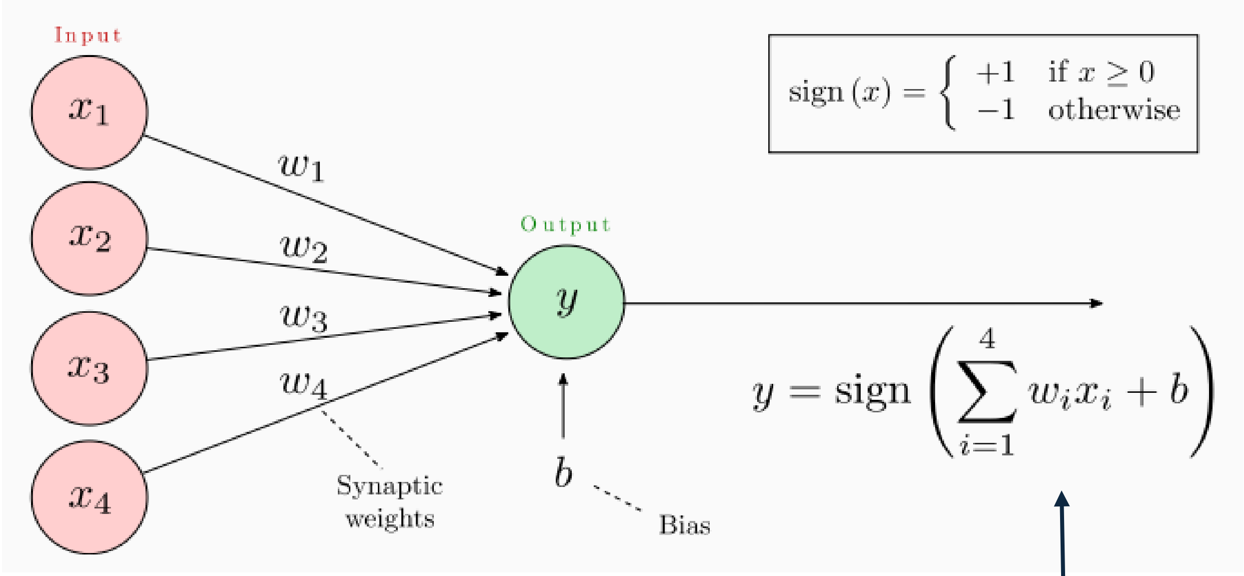
- Foundation of modern artificial neural networks;

Perceptron (Frank Rosenblatt, 1958)



Representation of the Perceptron

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

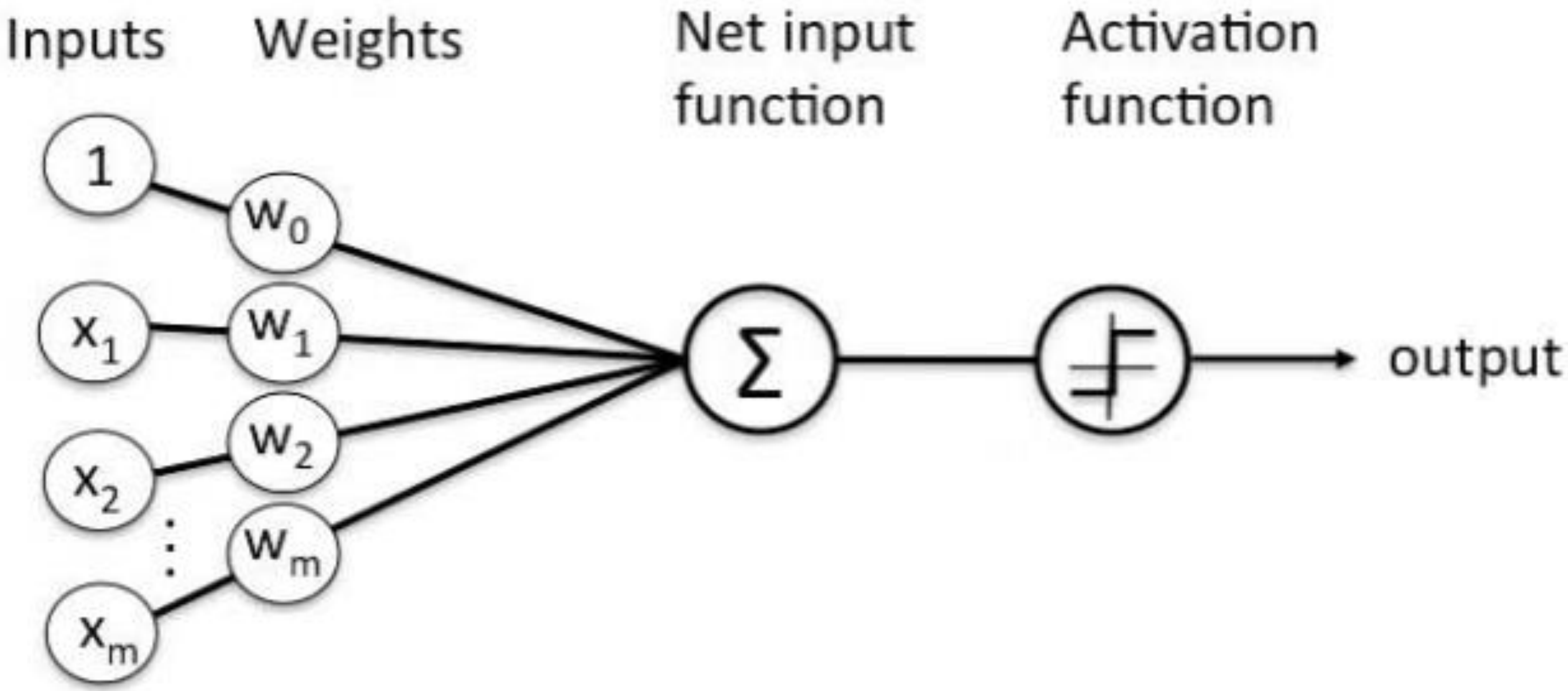


Activation function

- Parameters of the perceptron:
  - w_k: weights
  - b: bias

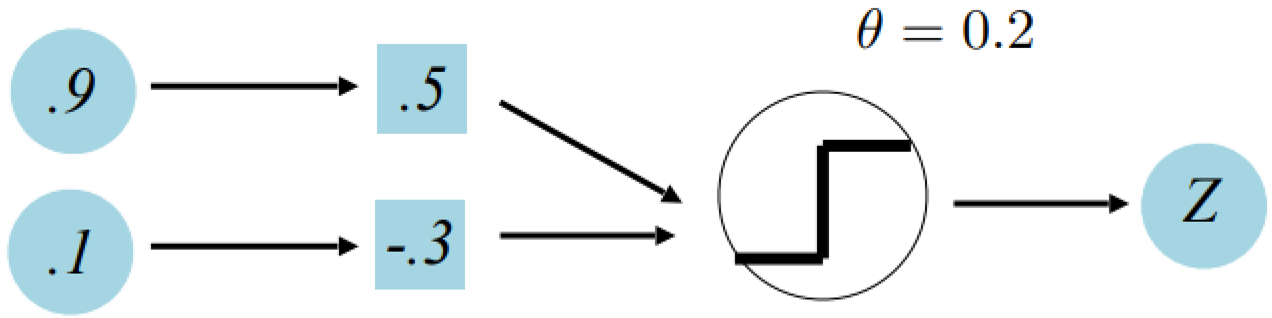
- Training → adjusting the weights and bias.

Alternative Representation of the Perceptron



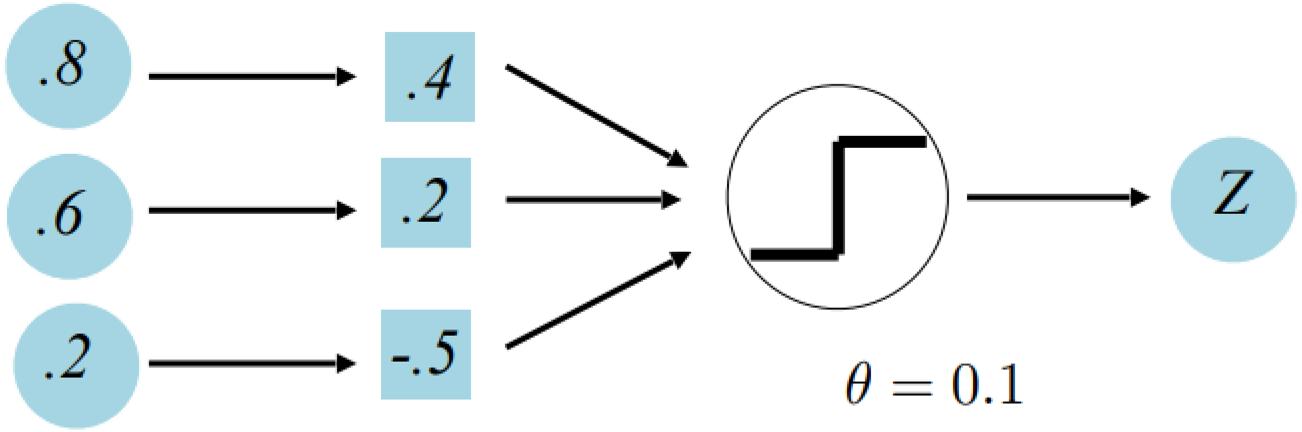
Perceptron Examples

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA



$z = ?$

$$z = \begin{cases} 1, if \ w \cdot x > \theta \\ 0, \ w \cdot x <= \theta \end{cases}$$



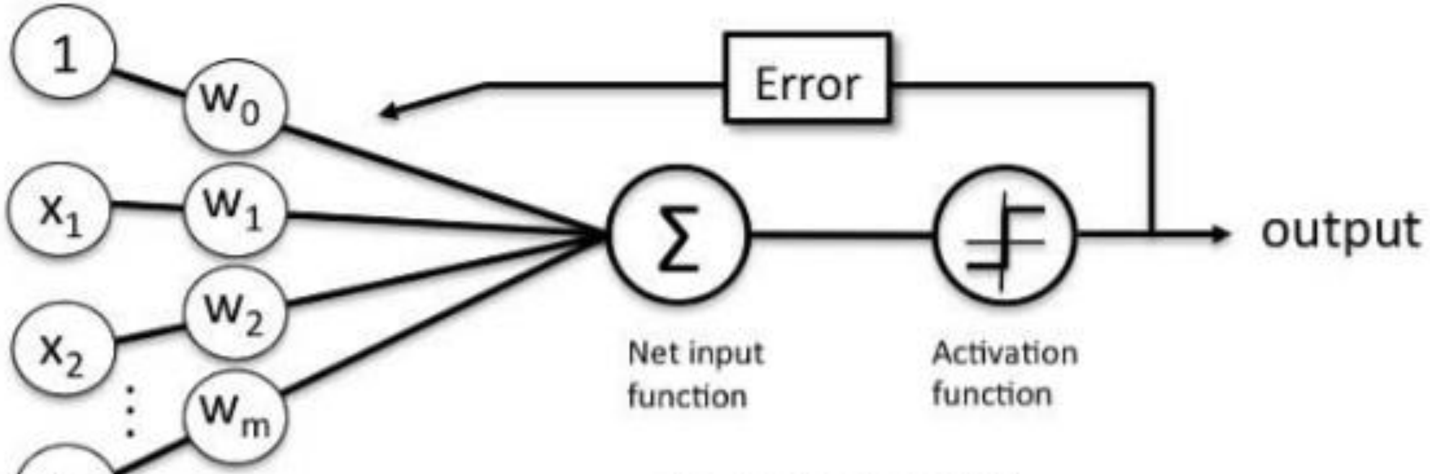
$z = ?$

Perceptron Learning Rule

- Suppose that x is a feature vector, y is the correct class label, and y' is the predicted class label computed using the current weights.
  - If y' = y, do nothing
  - Otherwise $w_i^{new} = w_i^{old} + \Delta w_i^{old}$, $\Delta w_i = \alpha\,(t - o)x_i$

perceptron prediction

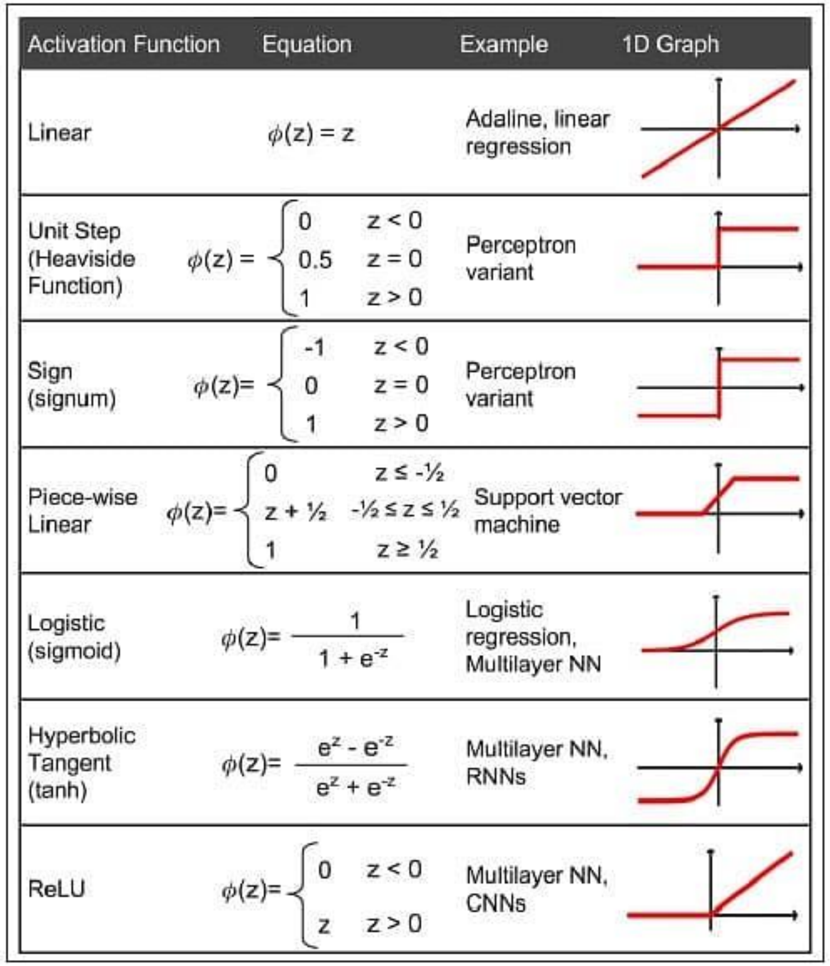
learning rate real target value



Perceptron rule.

Activation Functions

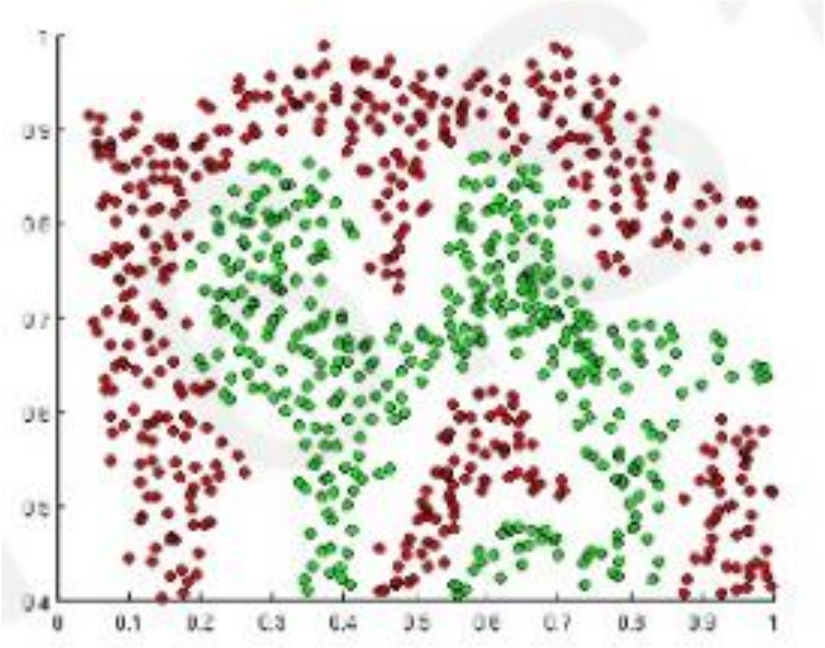
- The purpose of activation functions is to **introduce non-linearities** into the network.



Activation Function	Equation	Example	1D Graph
Linear	$\phi(z) = z$	Adaline, linear regression	
Unit Step (Heaviside Function)	$\phi(z) = \begin{cases} 0 & z < 0 \\ 0.5 & z = 0 \\ 1 & z > 0 \end{cases}$	Perceptron variant	
Sign (signum)	$\phi(z) = \begin{cases} -1 & z < 0 \\ 0 & z = 0 \\ 1 & z > 0 \end{cases}$	Perceptron variant	
Piece-wise Linear	$\phi(z) = \begin{cases} 0 & z \le -\frac{1}{2} \\ z + \frac{1}{2} & -\frac{1}{2} \le z \le \frac{1}{2} \\ 1 & z \ge \frac{1}{2} \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multilayer NN	
Hyperbolic Tangent (tanh)	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multilayer NN, RNNs	
ReLU	$\phi(z) = \begin{cases} 0 & z < 0 \\ z & z > 0 \end{cases}$	Multilayer NN, CNNs	

Activation Functions

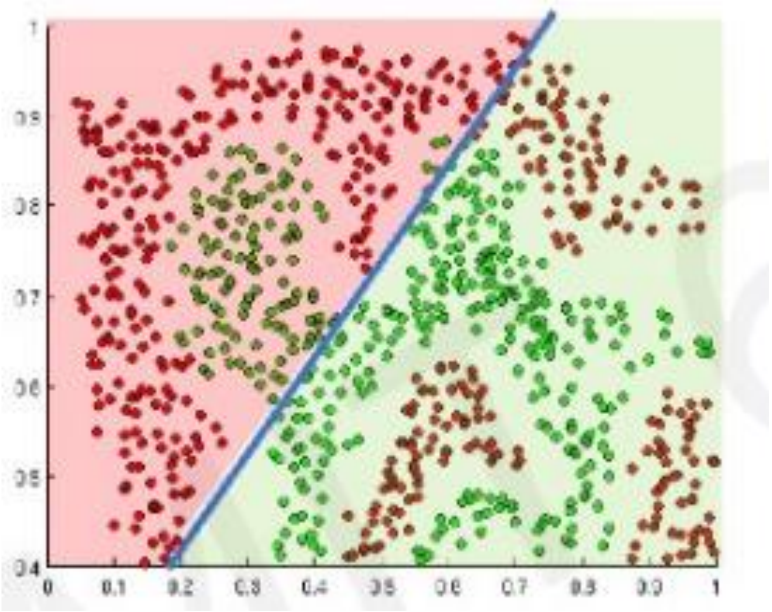
- The purpose of activation functions is to **introduce non-linearities** into the network.



What if we wanted to build a neural network to distinguish green vs red points?

Activation Functions

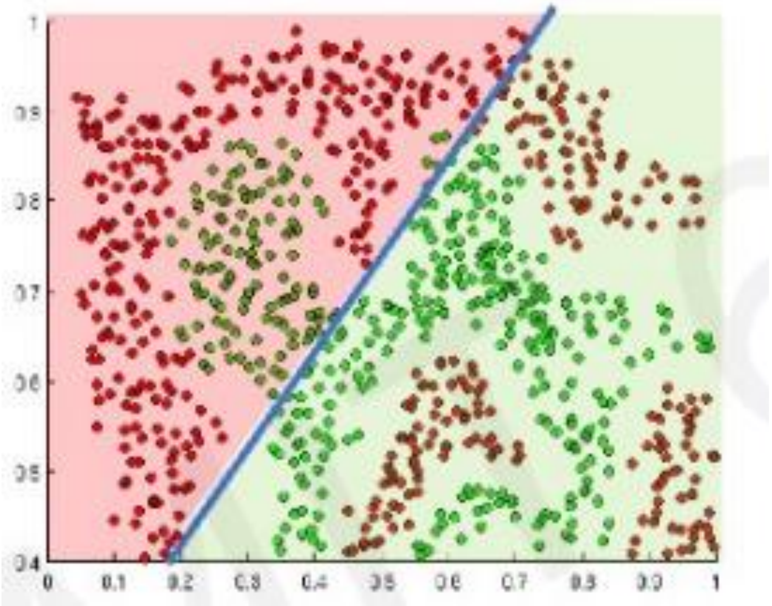
- The purpose of activation functions is to **introduce non-linearities** into the network.



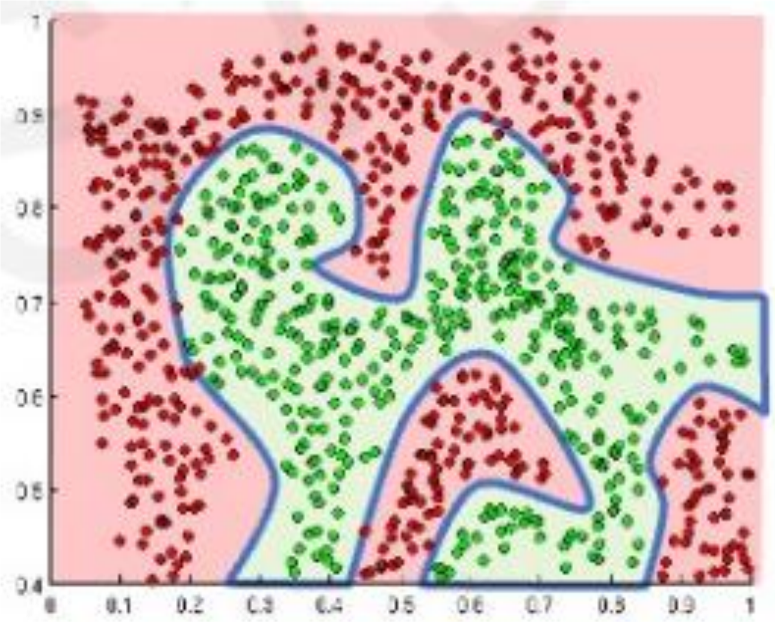
Linear activations produce linear decisions
no matter the network size.

Activation Functions

- The purpose of activation functions is to **introduce non-linearities** into the network.



Linear activations produce linear decisions no matter the network size.



Non-linearities allow us to approximate arbitrarily complex functions.

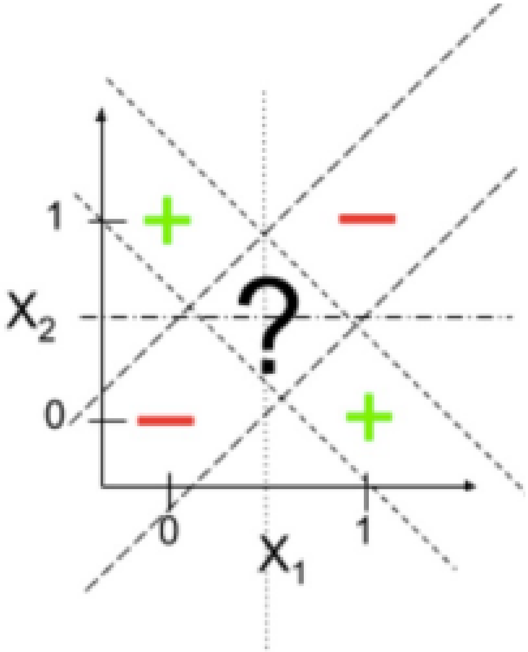
Perceptron Limitation: XOR Problem

- XOR = "Exclusive Or"
  - Input: two binary values x_1 and x_2
  - Output:
    - 1, when exactly one input equals 1
    - 0, otherwise

x_1	x_2	x_1 XOR x_2
0	0	?
0	1	?
1	0	?
1	1	?

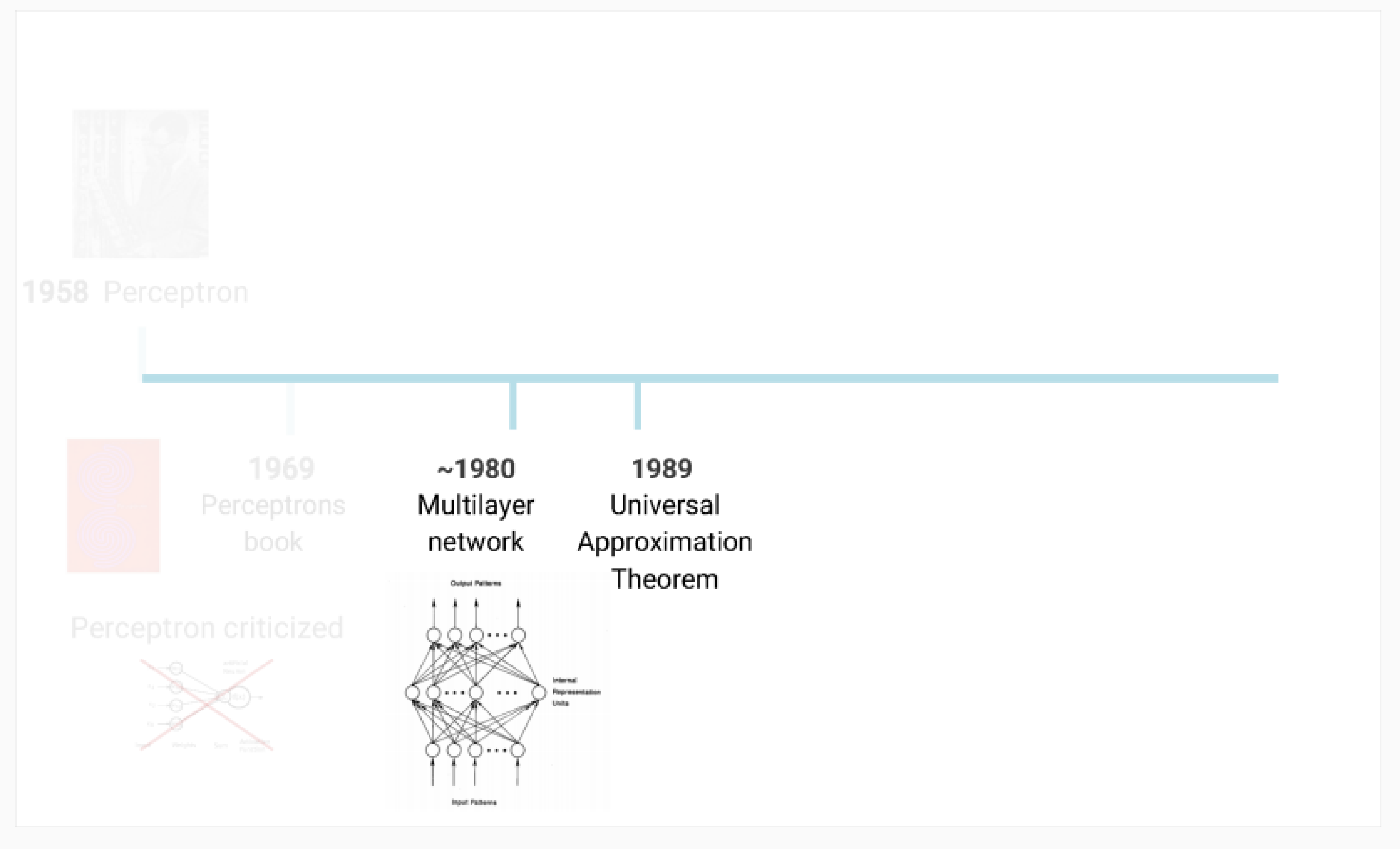
Perceptron Limitation: XOR Problem

- Cannot solve XOR problem and so separate 1s from 0s with a perceptron (linear function):



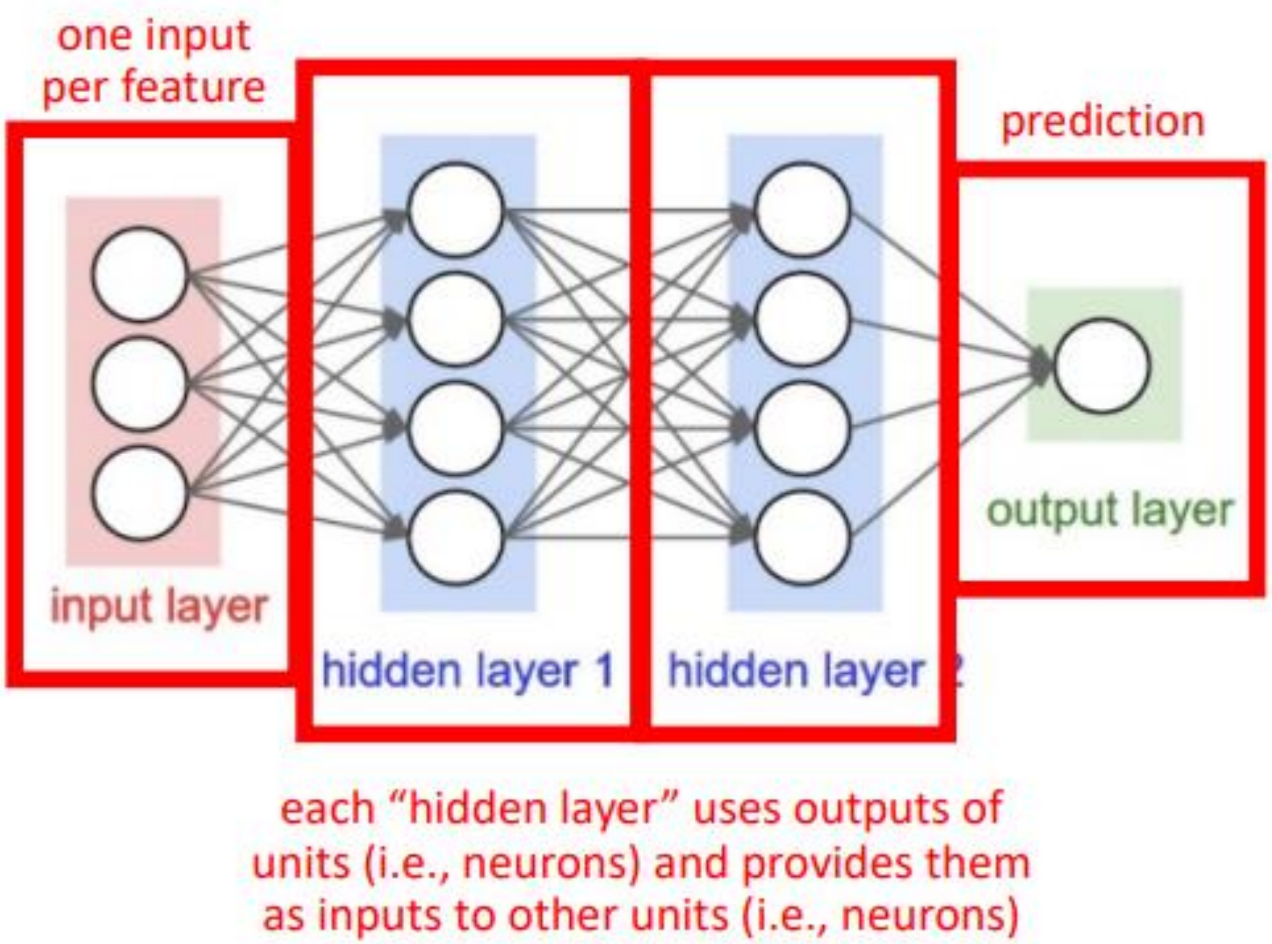
x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0

Multilayer Perceptron

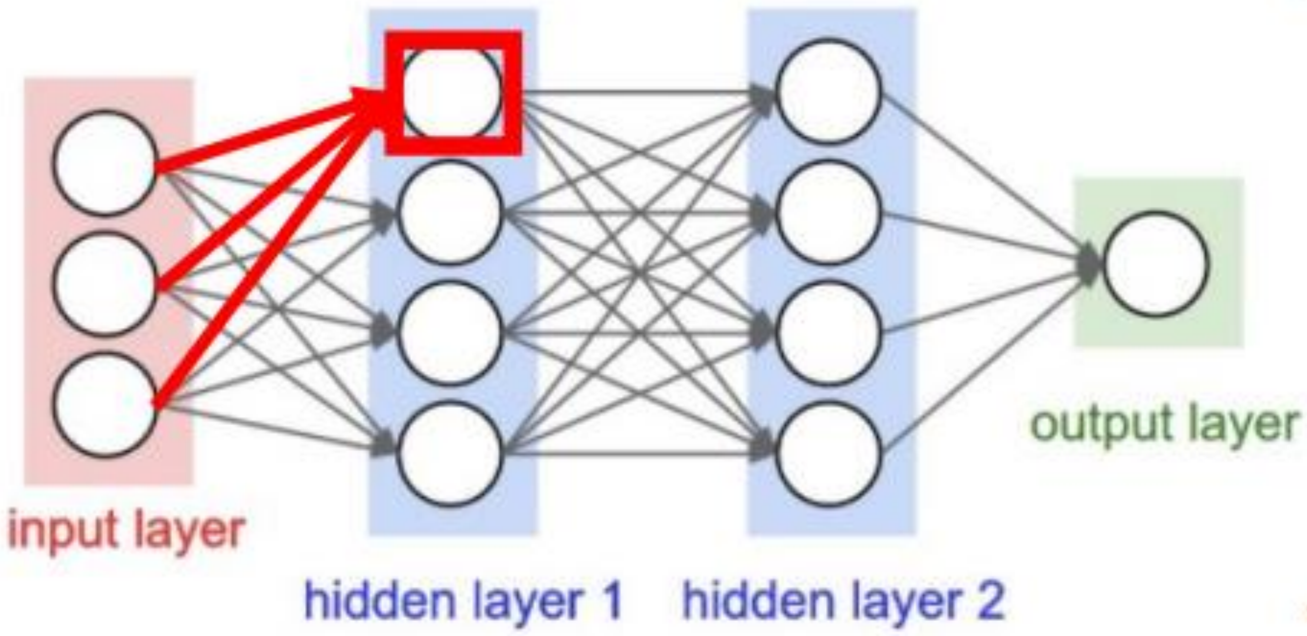


Mulilayer Perceptron

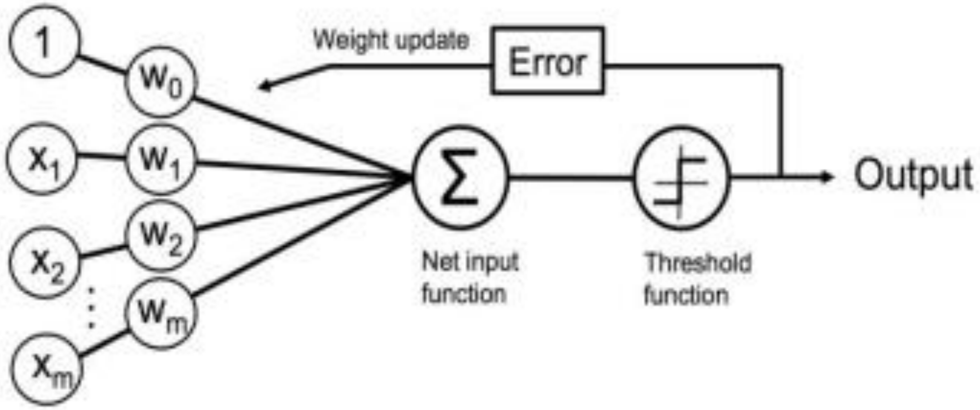
- AKA Artificial Neural Networks



Mulilayer Perceptron

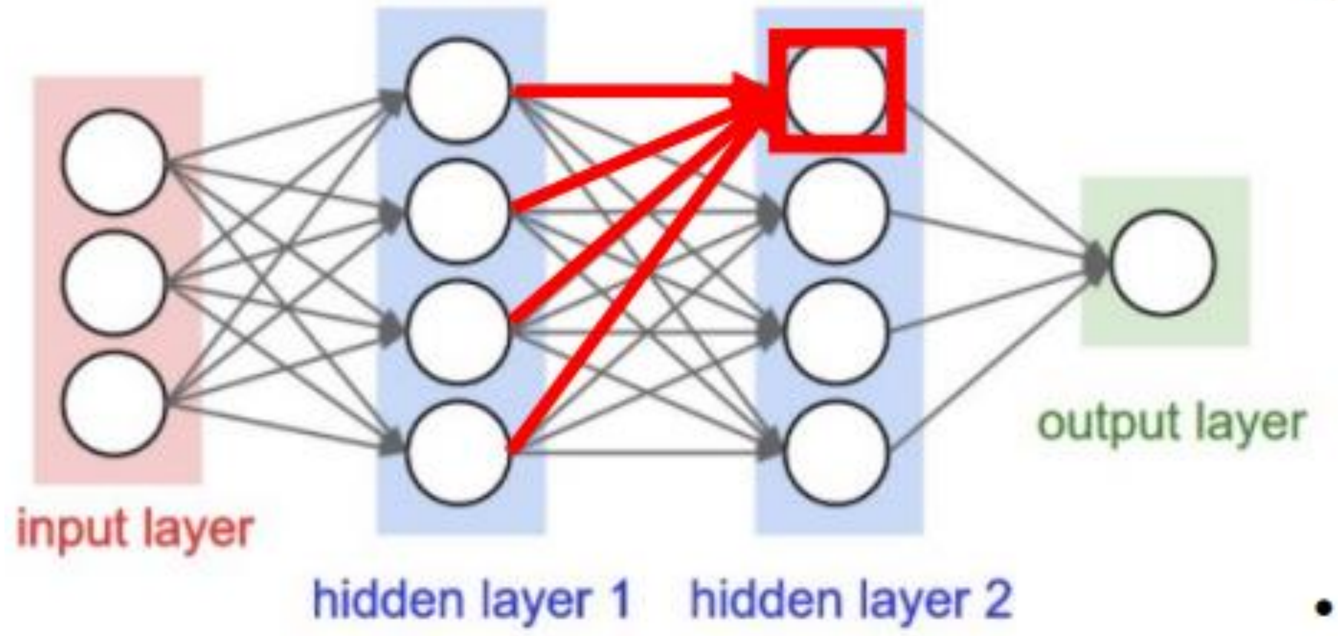


- How does this relate to a perceptron?

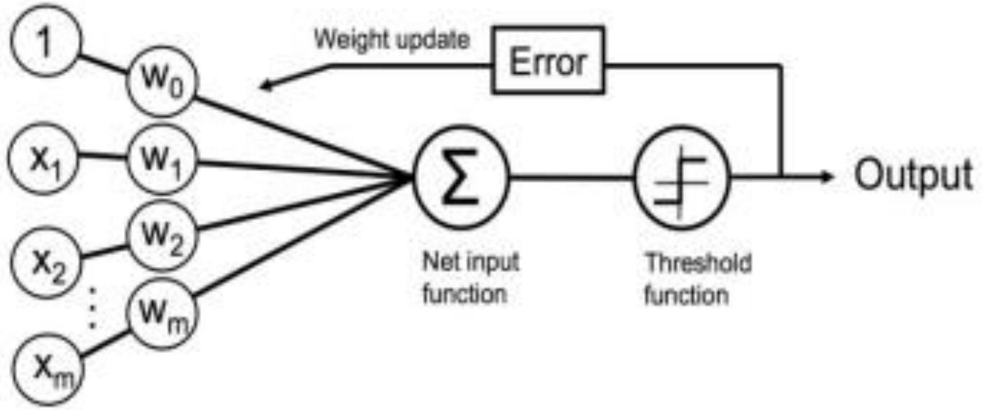


- Unit: takes as input a weighted sum and applies an activation function

Mulilayer Perceptron

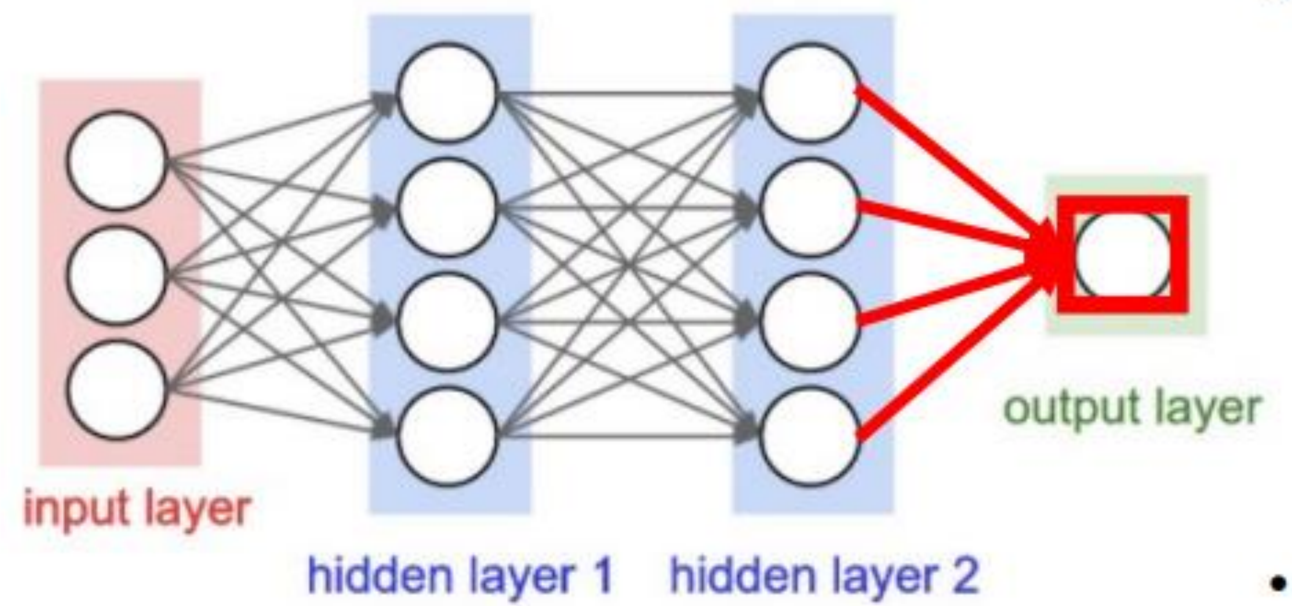


- How does this relate to a perceptron?

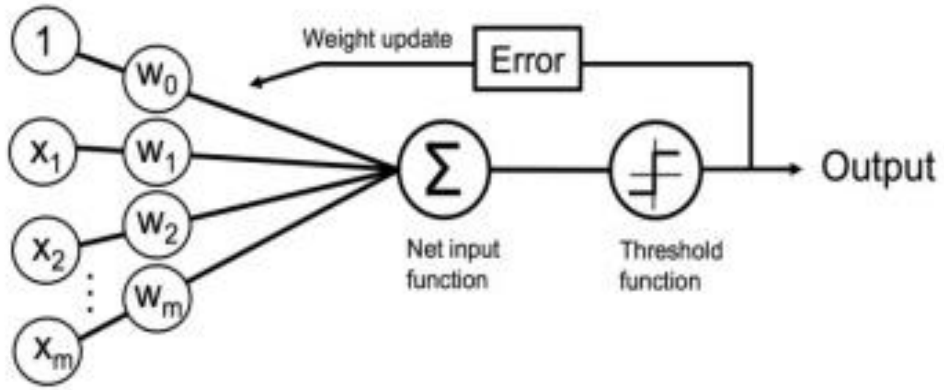


- Unit: takes as input a weighted sum and applies an activation function

Mulilayer Perceptron



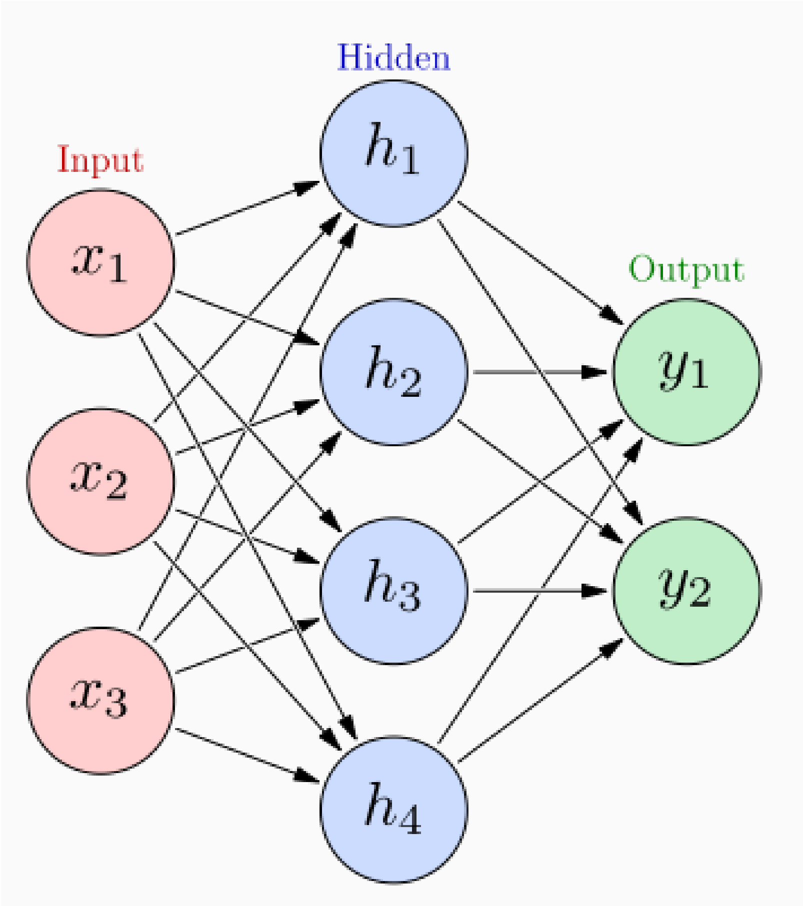
- How does this relate to a perceptron?



- Unit: takes as input a weighted sum and applies an activation function

Artificial Neural Networks

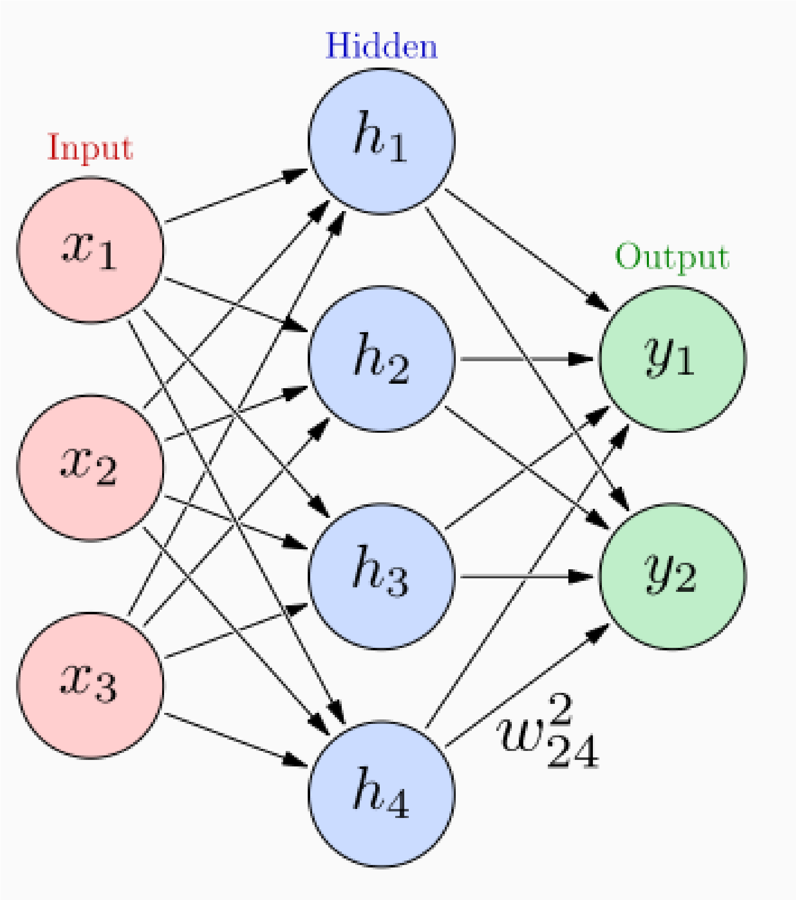
- Inter-connection of several artificial neurons (also called nodes or units);
- Each "level" in the graph is called a lauer:
  - Input layer;
  - Hidden layer(s);
  - Output layer.
- Each neuron in the hidden layers acts as a classifier / feature detector;
- Fedforward neural network (no cycles):
  - First na simplest type of neural network;
  - Information moves in one direction.



Artificial Neural Networks

$$h_1 = g_1\left(w^1_{11}x_1 + w^1_{12}x_2 + w^1_{13}x_3 + b^1_1\right)$$
$$h_2 = g_1\left(w^1_{21}x_1 + w^1_{22}x_2 + w^1_{23}x_3 + b^1_2\right)$$
$$h_3 = g_1\left(w^1_{31}x_1 + w^1_{32}x_2 + w^1_{33}x_3 + b^1_3\right)$$
$$h_4 = g_1\left(w^1_{41}x_1 + w^1_{42}x_2 + w^1_{43}x_3 + b^1_4\right)$$

$$y_1 = g_2\left(w^2_{11}h_1 + w^2_{12}h_2 + w^2_{13}h_3 + w^2_{14}h_4 + b^2_1\right)$$
$$y_2 = g_2\left(w^2_{21}h_1 + w^2_{22}h_2 + w^2_{23}h_3 + w^2_{24}h_4 + b^2_2\right)$$

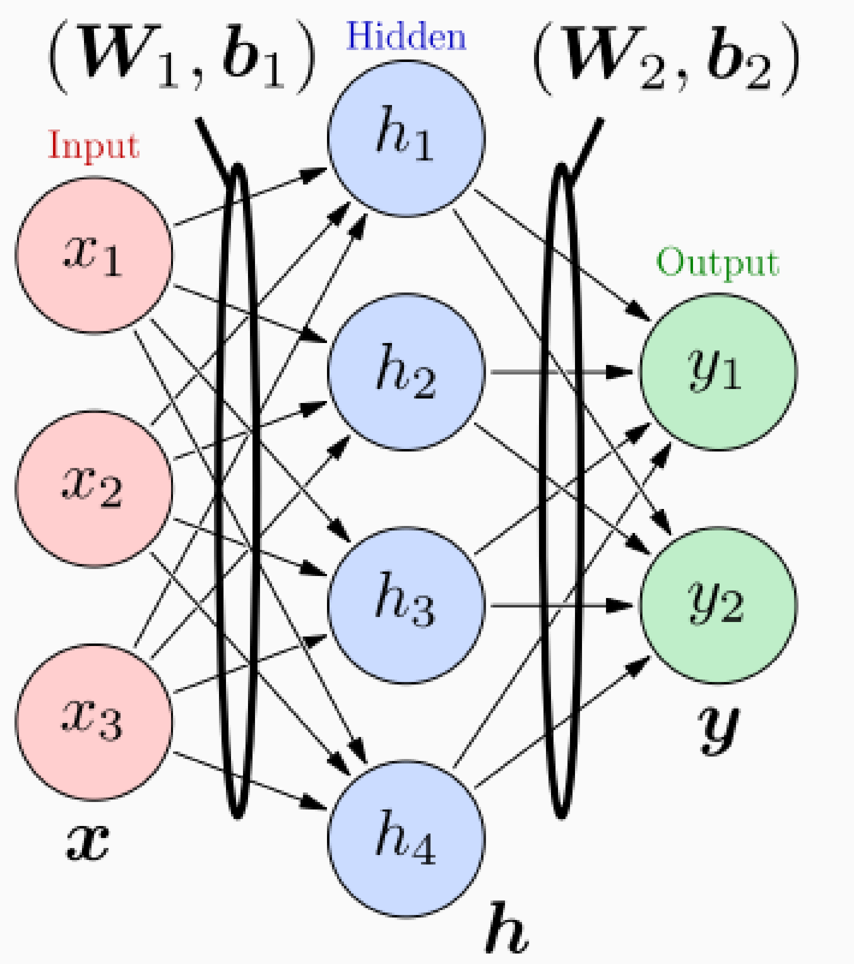


- w^k_{ij} weight between previous node j and next node i at layer k;
- g_k is any activation function applied to each its input vector

Artificial Neural Networks

$$h_1 = g_1\left(w_{11}^1 x_1 + w_{12}^1 x_2 + w_{13}^1 x_3 + b_1^1\right)$$
$$h_2 = g_1\left(w_{21}^1 x_1 + w_{22}^1 x_2 + w_{23}^1 x_3 + b_2^1\right)$$
$$h_3 = g_1\left(w_{31}^1 x_1 + w_{32}^1 x_2 + w_{33}^1 x_3 + b_3^1\right)$$
$$h_4 = g_1\left(w_{41}^1 x_1 + w_{42}^1 x_2 + w_{43}^1 x_3 + b_4^1\right)$$
$$\boldsymbol{h} = g_1\left(\boldsymbol{W}_1 \boldsymbol{x} + \boldsymbol{b}_1\right)$$

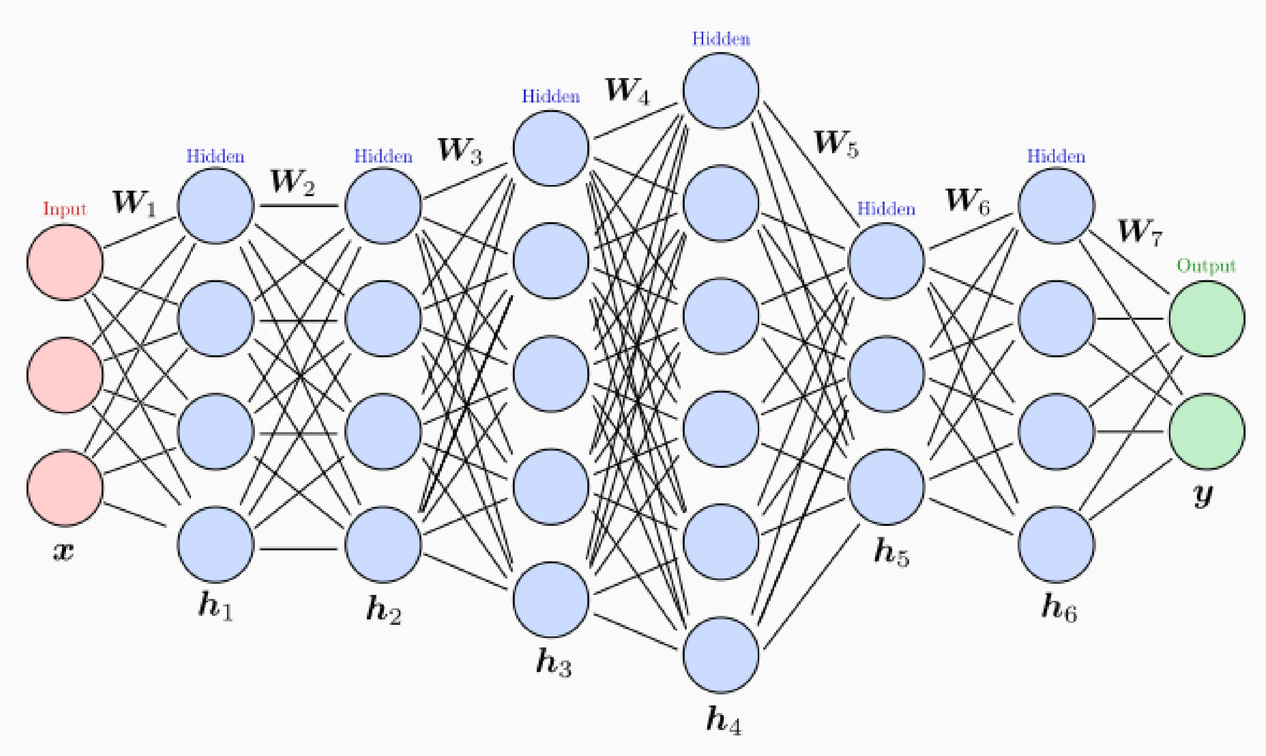
$$y_1 = g_2\left(w_{11}^2 h_1 + w_{12}^2 h_2 + w_{13}^2 h_3 + w_{14}^2 h_4 + b_1^2\right)$$
$$y_2 = g_2\left(w_{21}^2 h_1 + w_{22}^2 h_2 + w_{23}^2 h_3 + w_{24}^2 h_4 + b_2^2\right)$$
$$\boldsymbol{y} = g_2\left(\boldsymbol{W}_2 \boldsymbol{h} + \boldsymbol{b}_2\right)$$



- The matrices W_k and biases b_k are learned from labeled training data.

Artificial Neural Networks

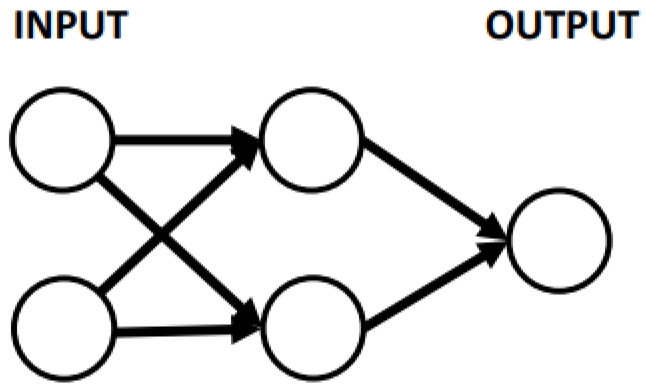
UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

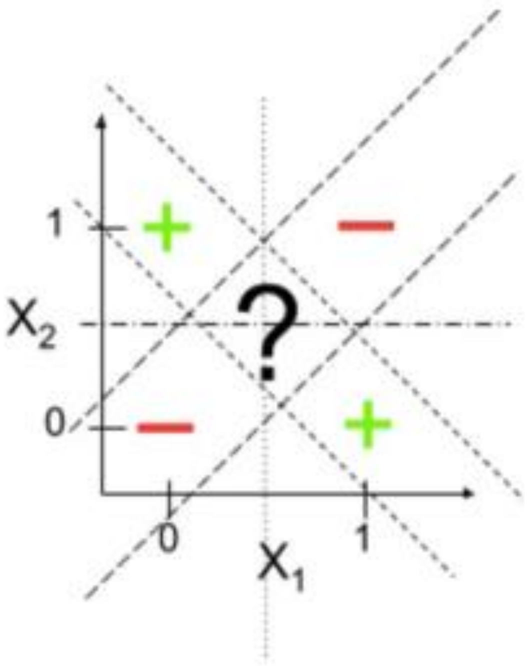


- It can have 1 hidden layer only (shallow network);
- It can have more than 1 hidden layer (deep network);
- Each layer can have a different size, and hidden and output layers often have different activation functions.

Revisiting the XOR Problem

- Non-linear function: separate 1s from 0s



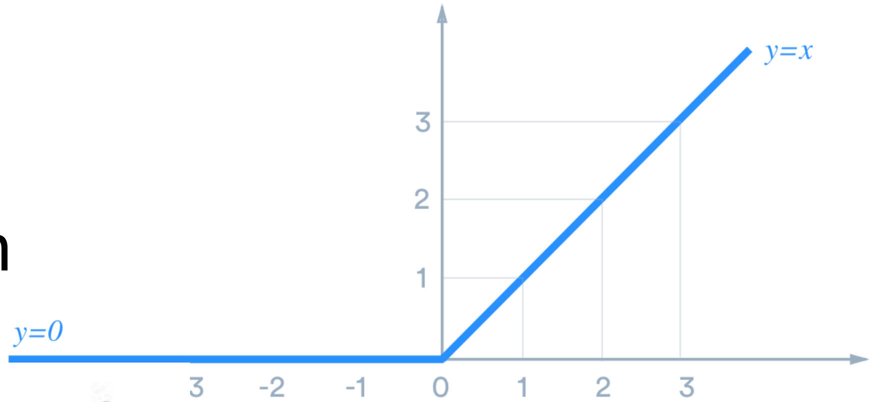


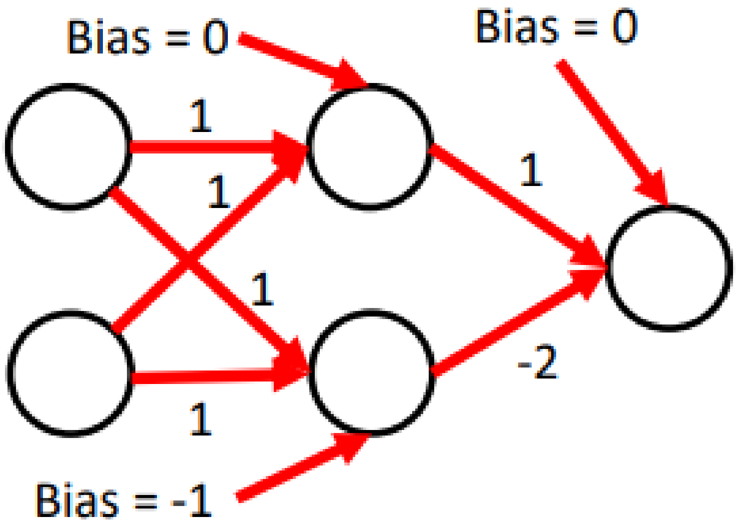
x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0

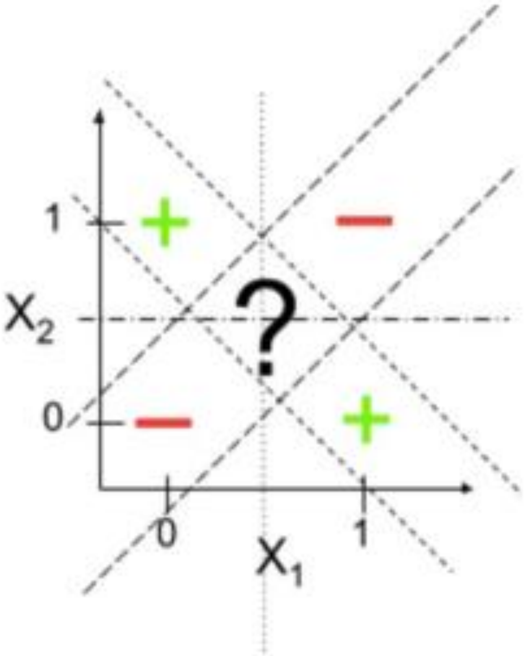
Revisiting the XOR Problem

- Non-linear function: separate 1s from 0s

- Approach: use the ReLU activation function







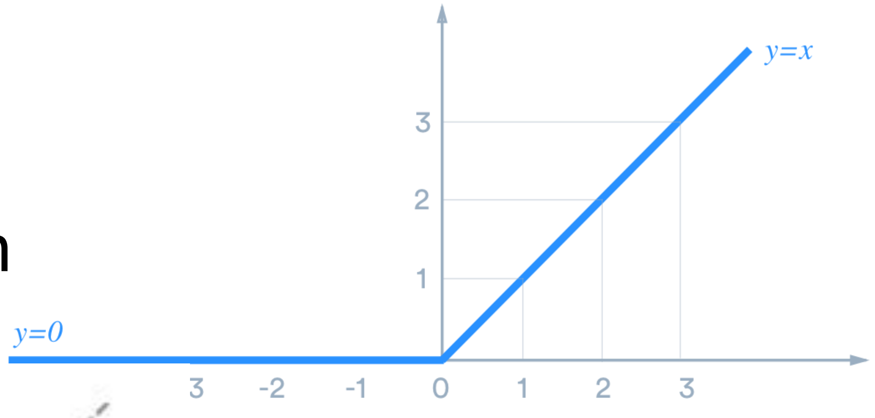
x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0

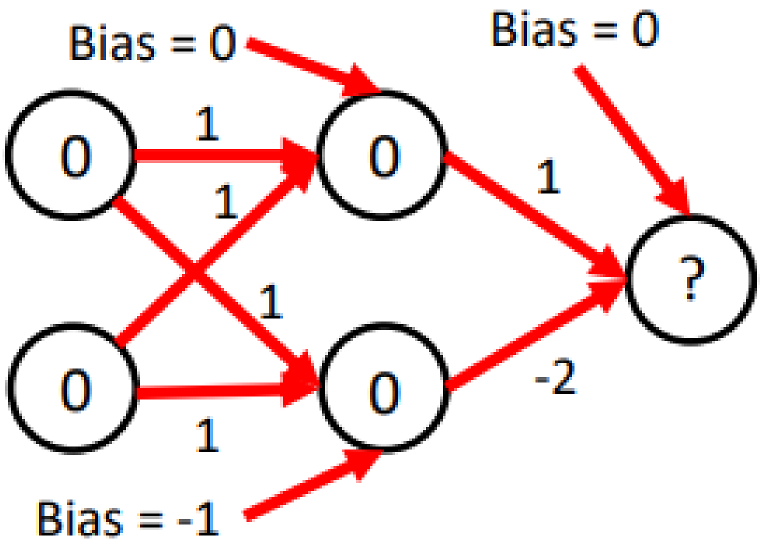
Revisiting the XOR Problem

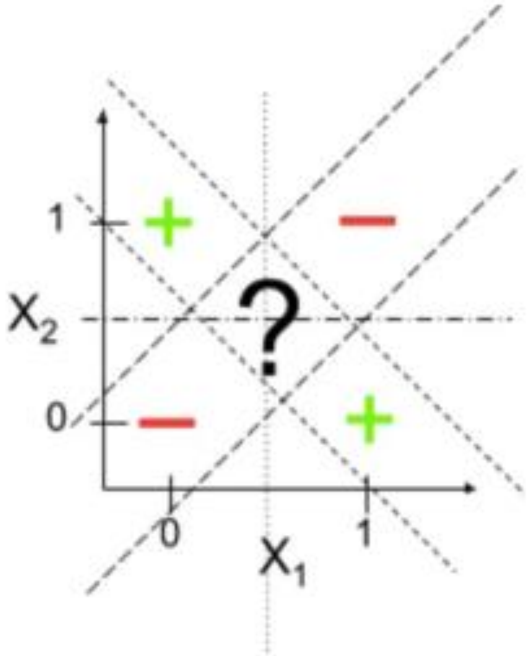
UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

- Non-linear function: separate 1s from 0s

- Approach: use the ReLU activation function





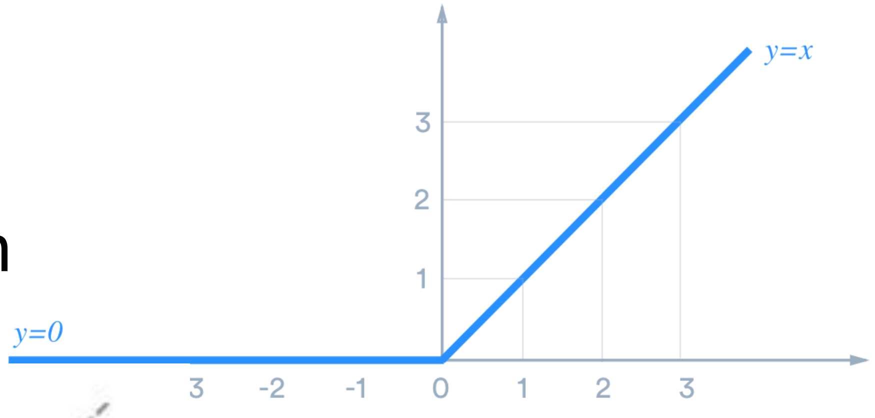


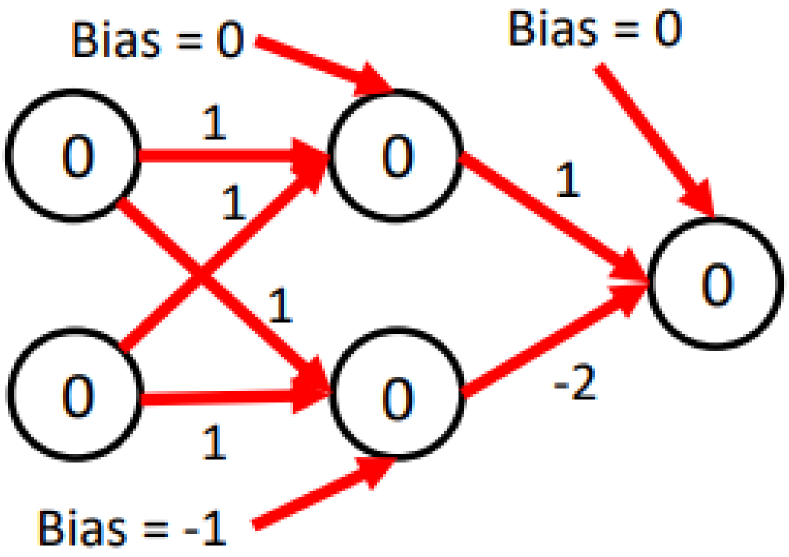
x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0

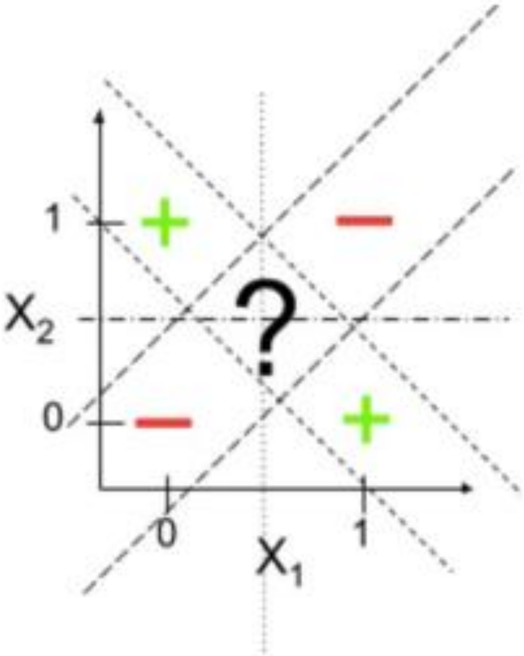
Revisiting the XOR Problem

- Non-linear function: separate 1s from 0s

- Approach: use the ReLU activation function



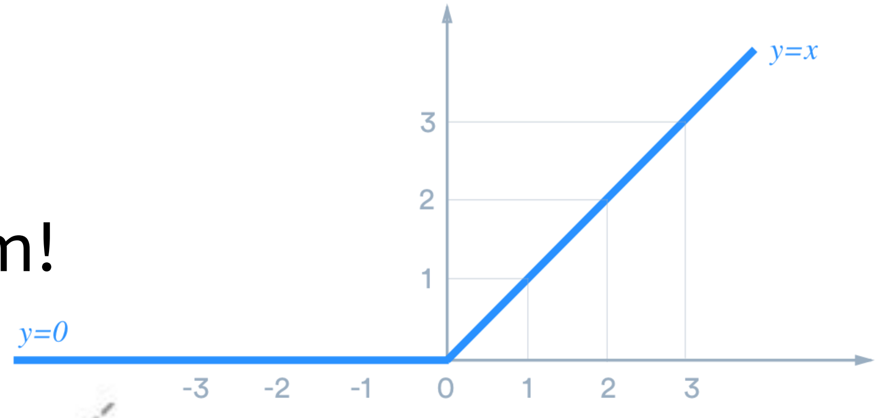




x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0

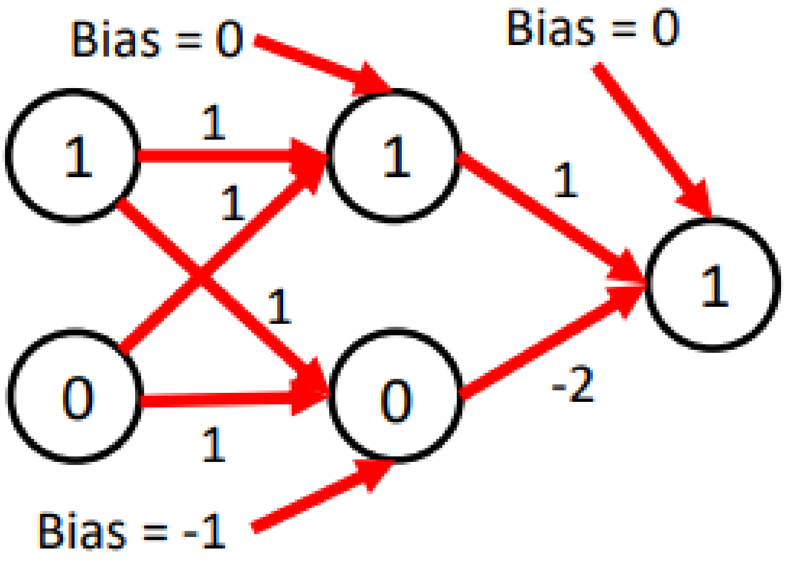
Revisiting the XOR Problem

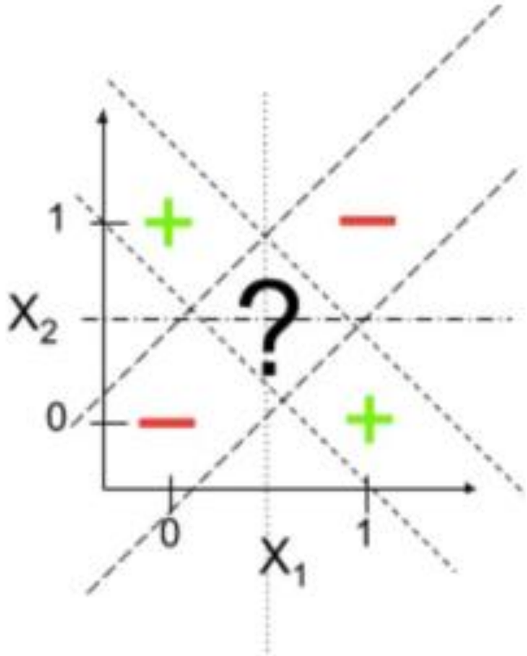
- Non-linear function: separate 1s from 0s



- Neural networks can solve the XOR problem!

And so model non-linear functions!

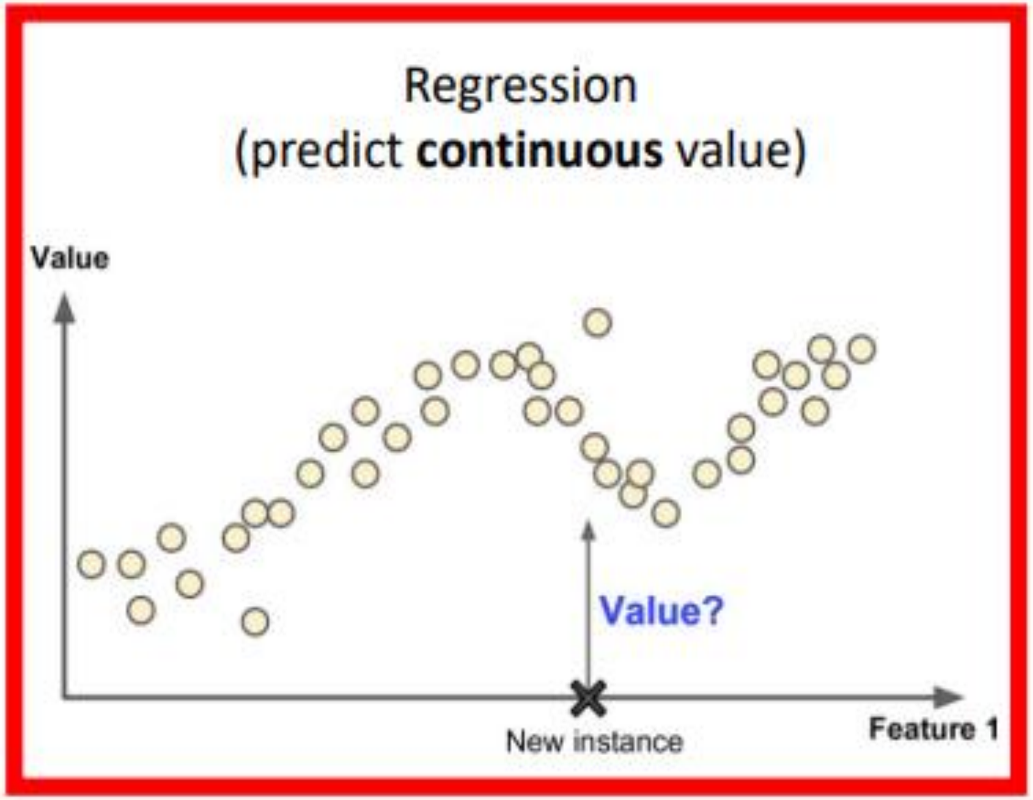


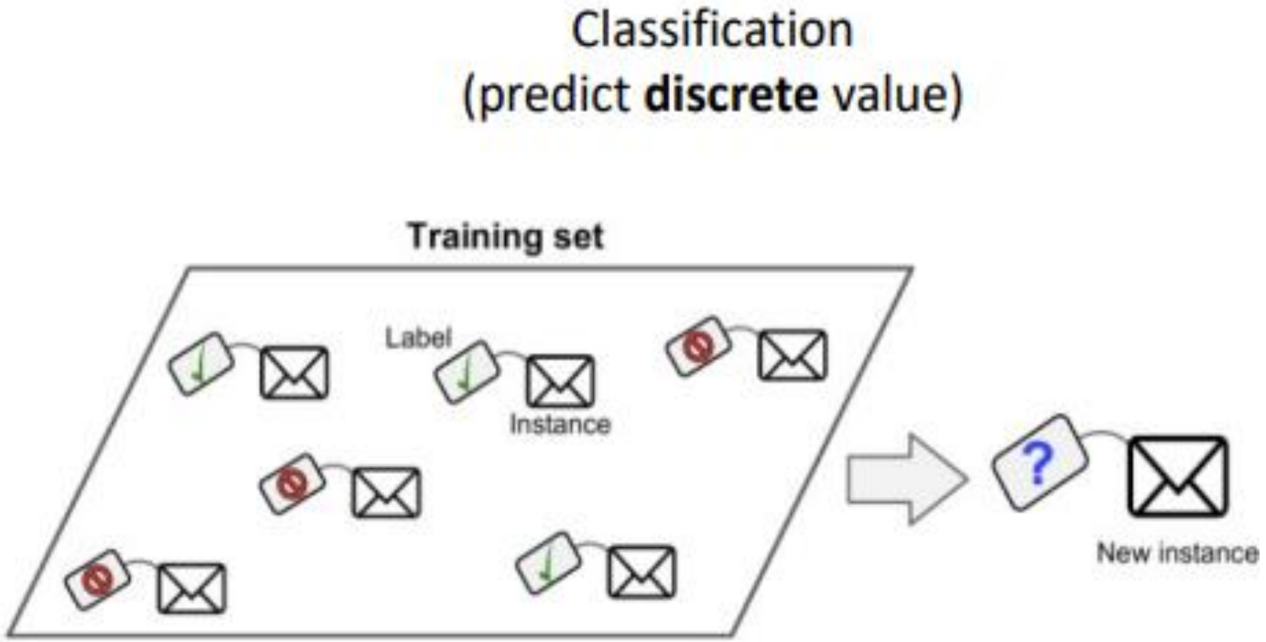


x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0

Output Activations

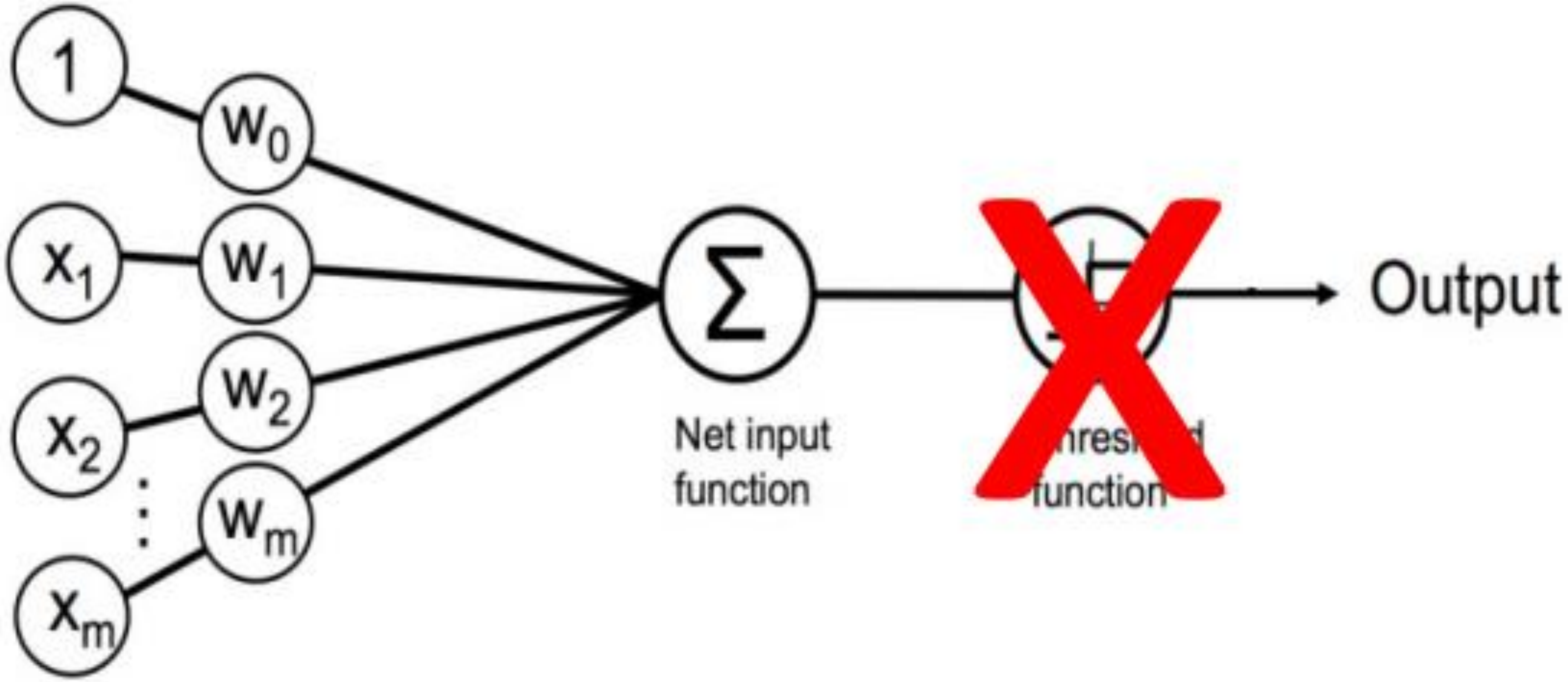
- Desired output driven by task





Output Activations

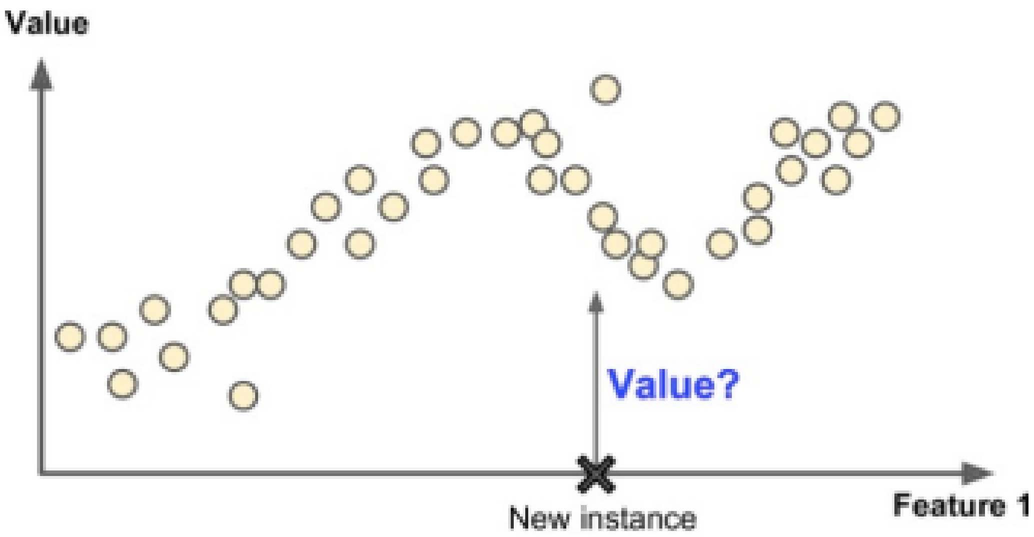
- Linear (No Activation Function)



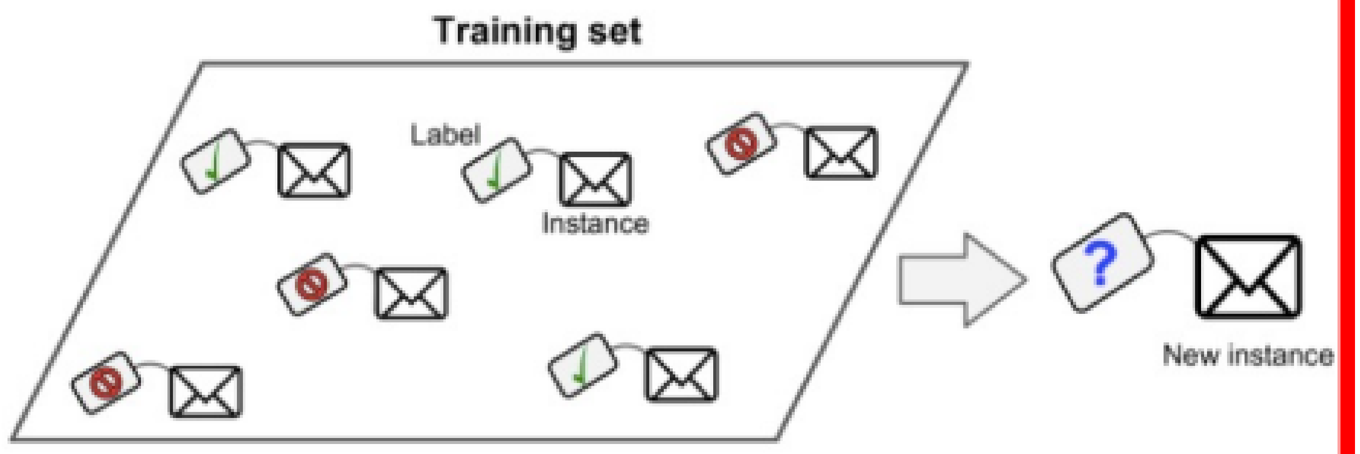
Output Activations

- Desired output driven by task

Regression
(predict **continuous** value)

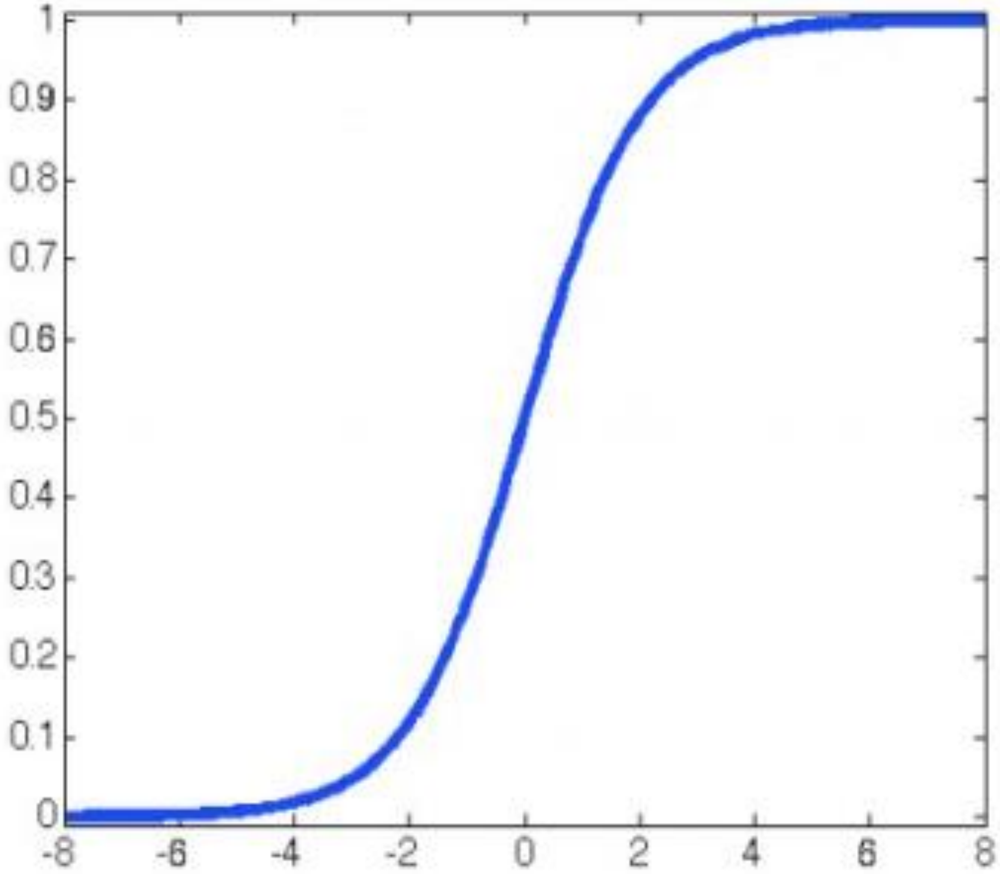


Classification
(predict **discrete** value)



Output Activations

- Sigmoid for Binary Classification



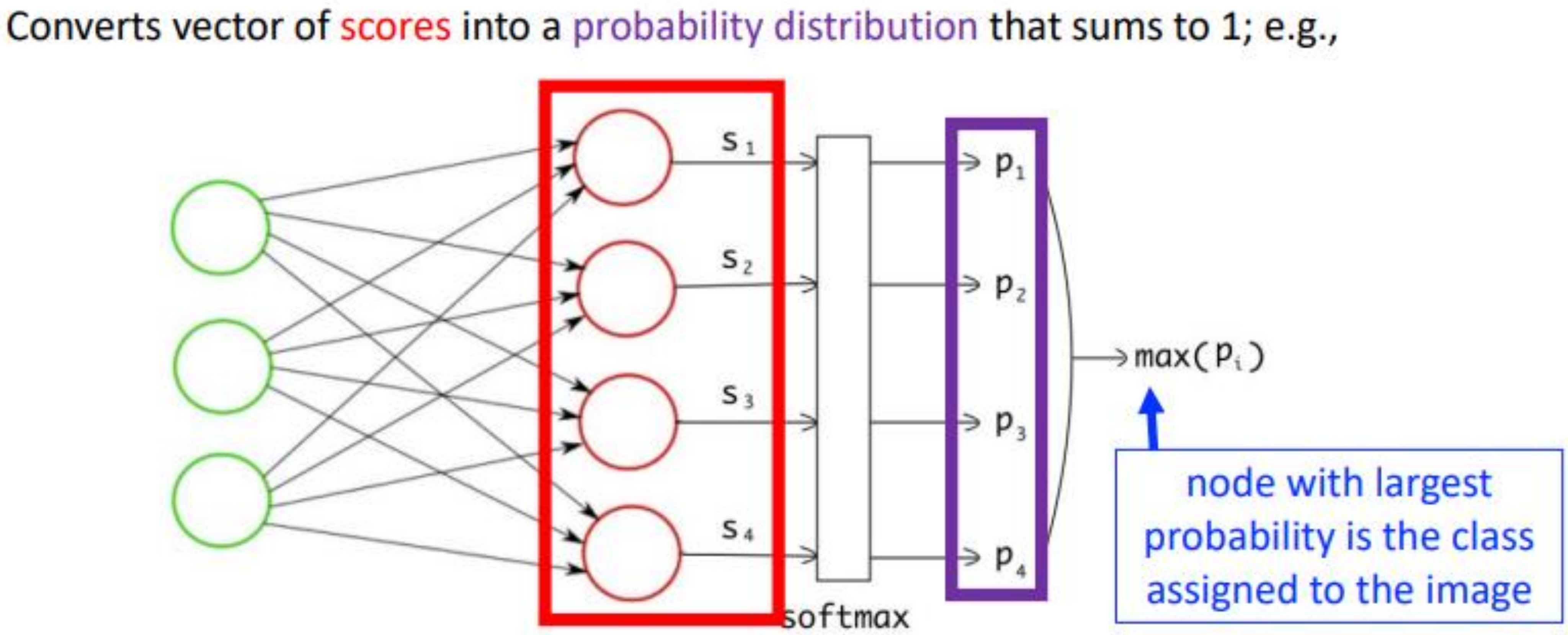
$$\sigma(z) = \frac{1}{1+\exp(-z)}$$

If >= 0.5, output 1;

Else, outputs 0

Output Activations

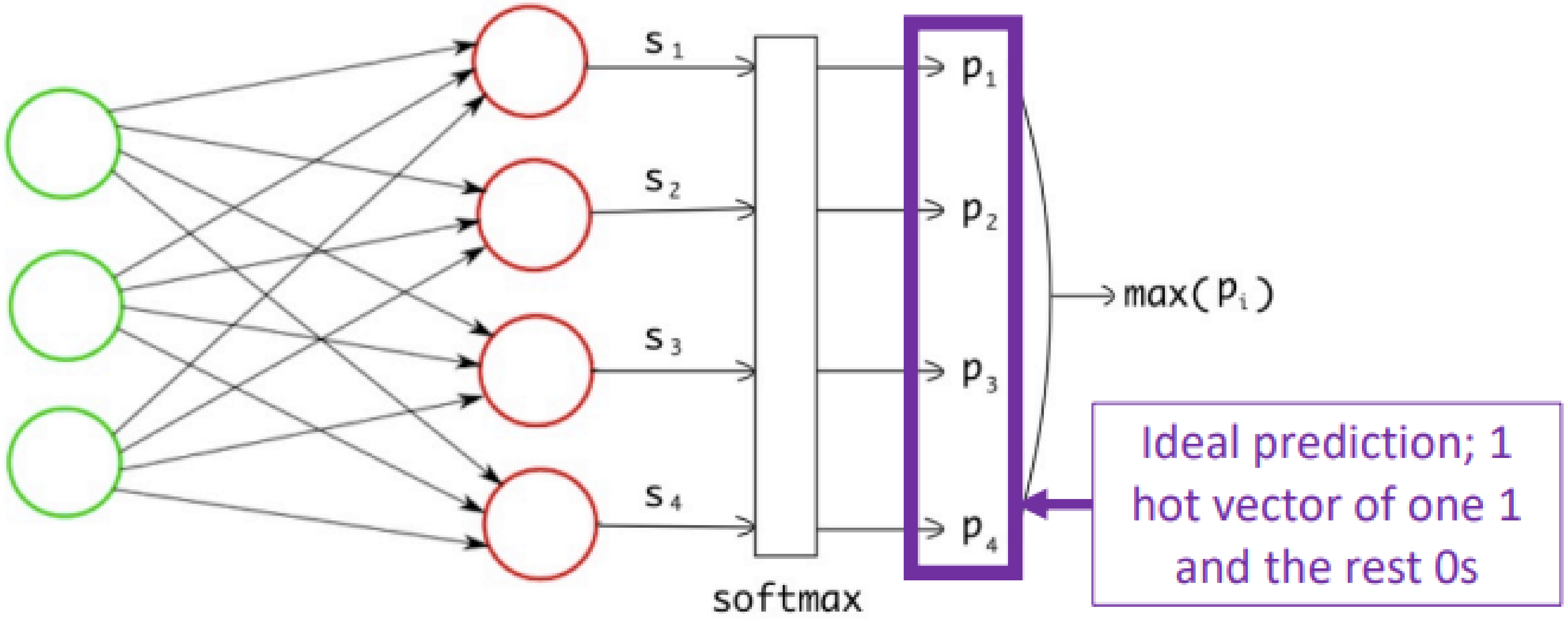
- Softmax for Multiclass Classification



Output Activations

- Softmax for Multiclass Classification

Converts vector of scores into a probability distribution that sums to 1; e.g.,



Output Activations

- Softmax for Multiclass Classification

Converts vector of scores into a probability distribution that sums to 1

Get rid of negative values while preserving original order of scores; e causes negative scores to become slightly larger than 0 while positive values grow exponentially (choosing *e* rather than another exponent base simplifies math during training)

$$\sigma(\mathbf{z})_i = \frac{\boxed{e}^{\boxed{z_i}}}{\boxed{\sum_{j=1}^{K} \boxed{e}^{z_j}}}$$

i = 1, ..., K

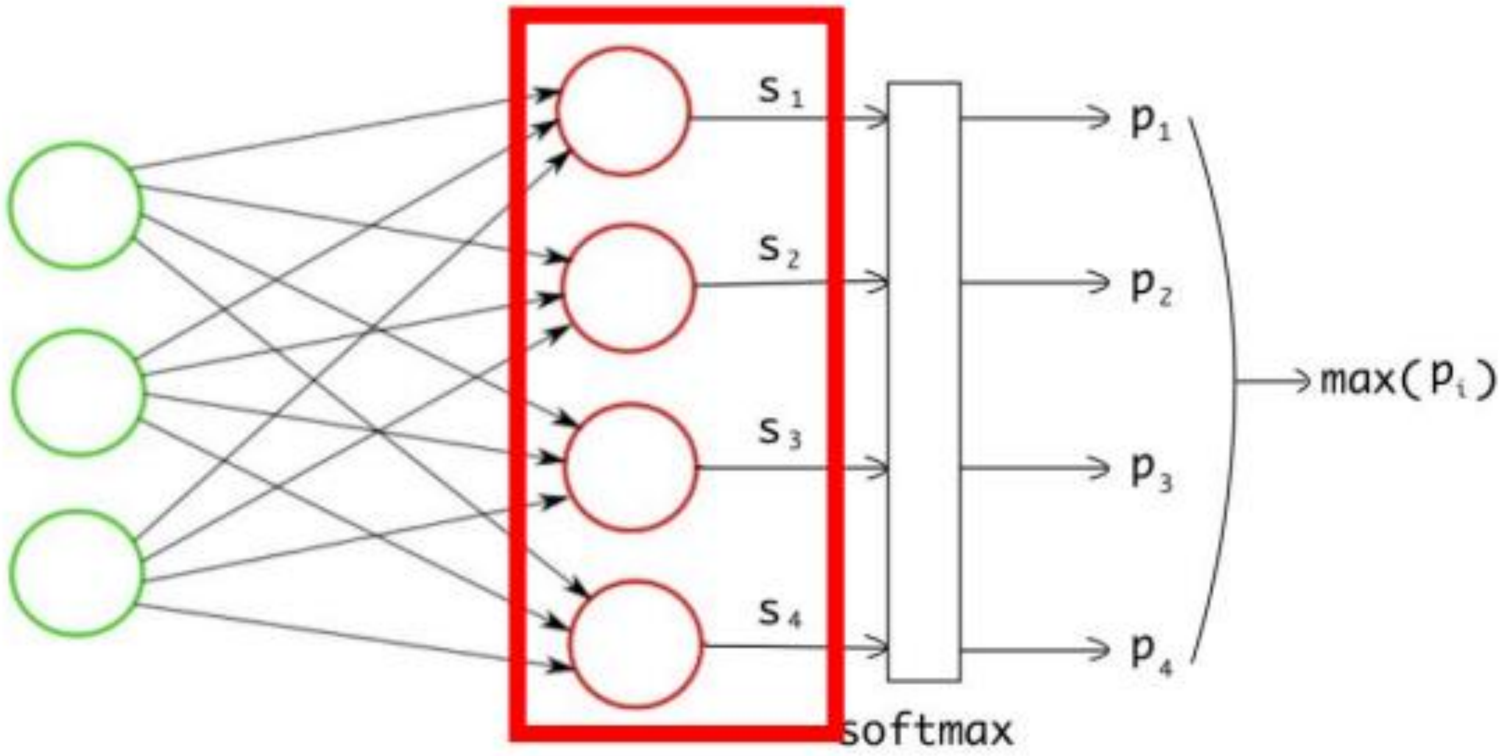
Number of classes

Want to divide each node's score by sum of all entries to make them sum to 1 (normalization)

Output Activations

- Softmax for Multiclass Classification

Converts vector of scores into a probability distribution that sums to 1; e.g.,



Output Activations

- Softmax for Multiclass Classification

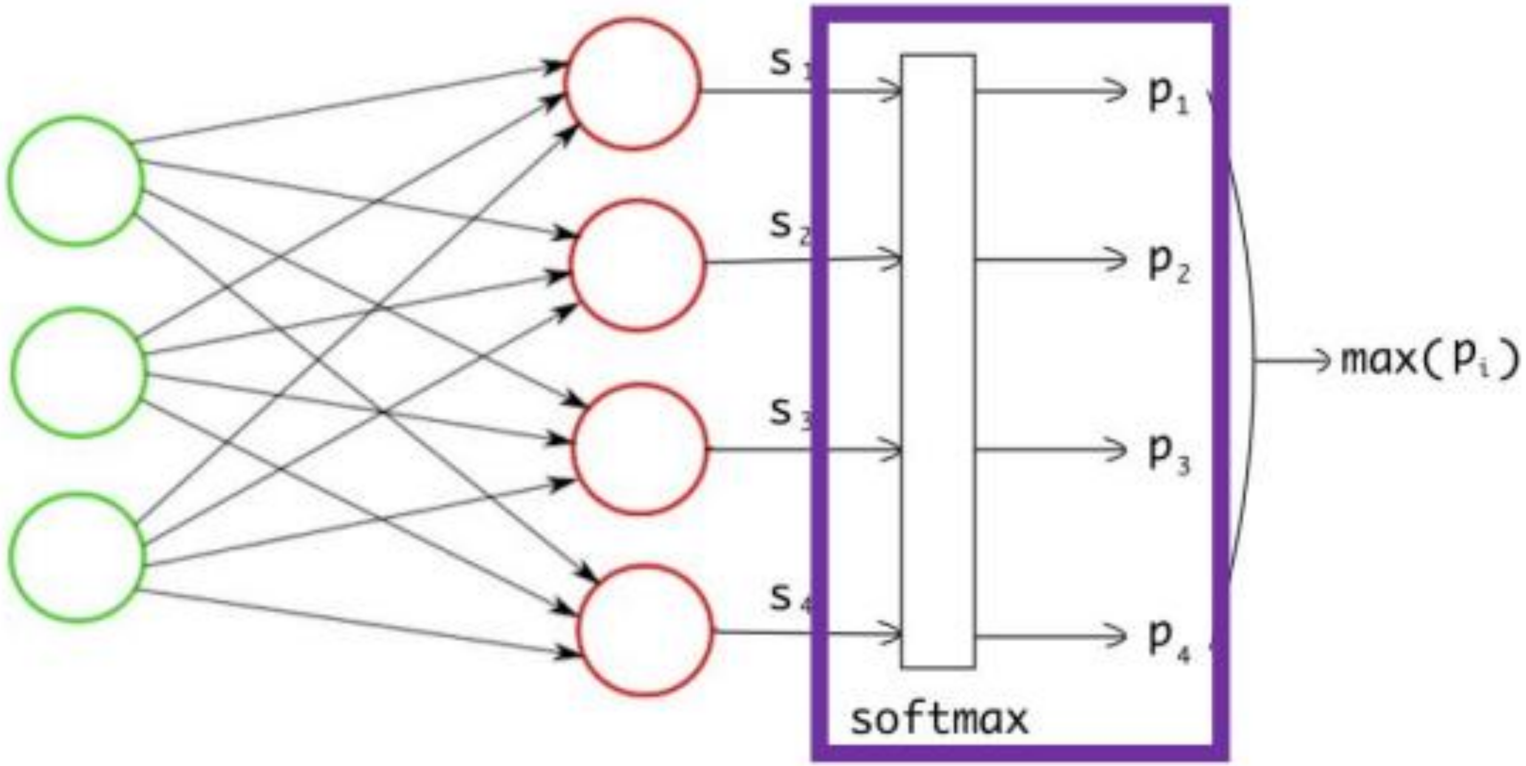
Converts vector of scores into a probability distribution that sums to 1; e.g.,

	Scoring Function
Dog	-3.44
Cat	1.16
Boat	-0.81
Airplane	3.91

Output Activations

- Softmax for Multiclass Classification

Converts vector of scores into a probability distribution that sums to 1; e.g.,



Output Activations

- Softmax for Multiclass Classification

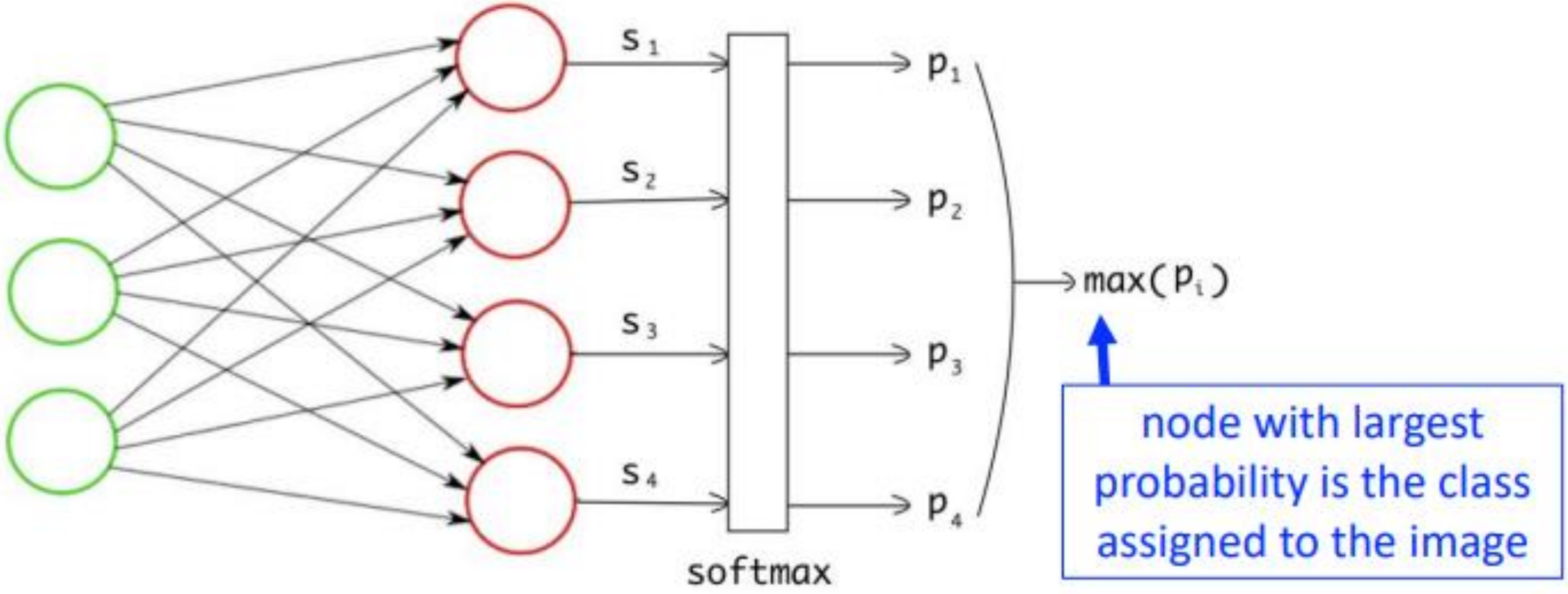
Converts vector of scores into a probability distribution that sums to 1; e.g.,

	Scoring Function	Unnormalized Probabilities e^{z_i}	Normalized Probabilities $\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$
Dog	-3.44	0.0321	0.0006
Cat	1.16	3.1899	0.0596
Boat	-0.81	0.4449	0.0083
Airplane	3.91	49.8990	0.9315

Output Activations

- Softmax for Multiclass Classification

Converts vector of scores into a probability distribution that sums to 1; e.g.,



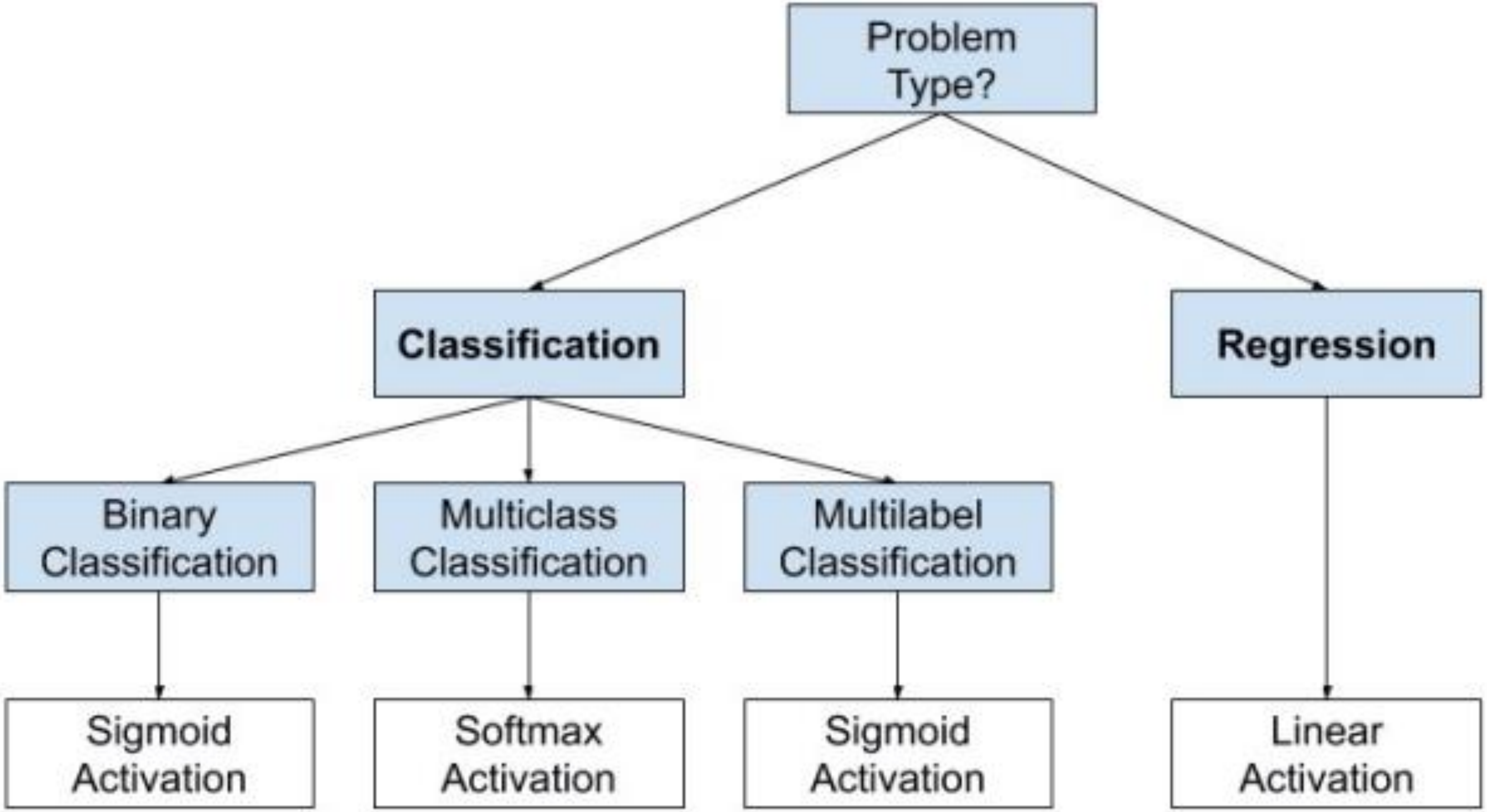
Output Activations

- Softmax for Multiclass Classification

Converts vector of scores into a probability distribution that sums to 1; e.g.,

	Scoring Function	Unnormalized Probabilities	Normalized Probabilities
Dog	-3.44	0.0321	0.0006
Cat	1.16	3.1899	0.0596
Boat	-0.81	0.4449	0.0083
Airplane	3.91	49.8990	0.9315

Output Activations



Example Problem

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

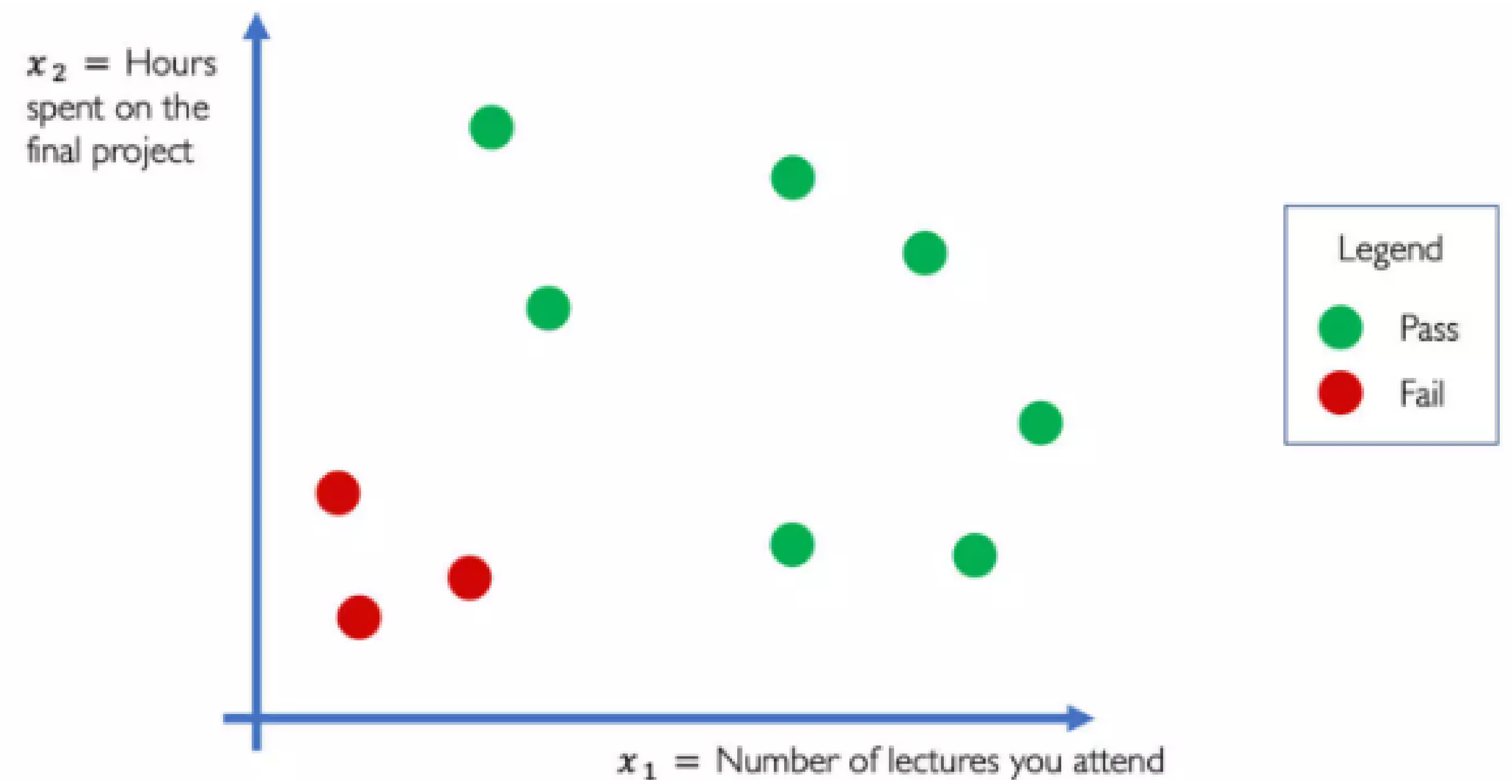
Will I pass this class?

Let's start with a simple two feature model

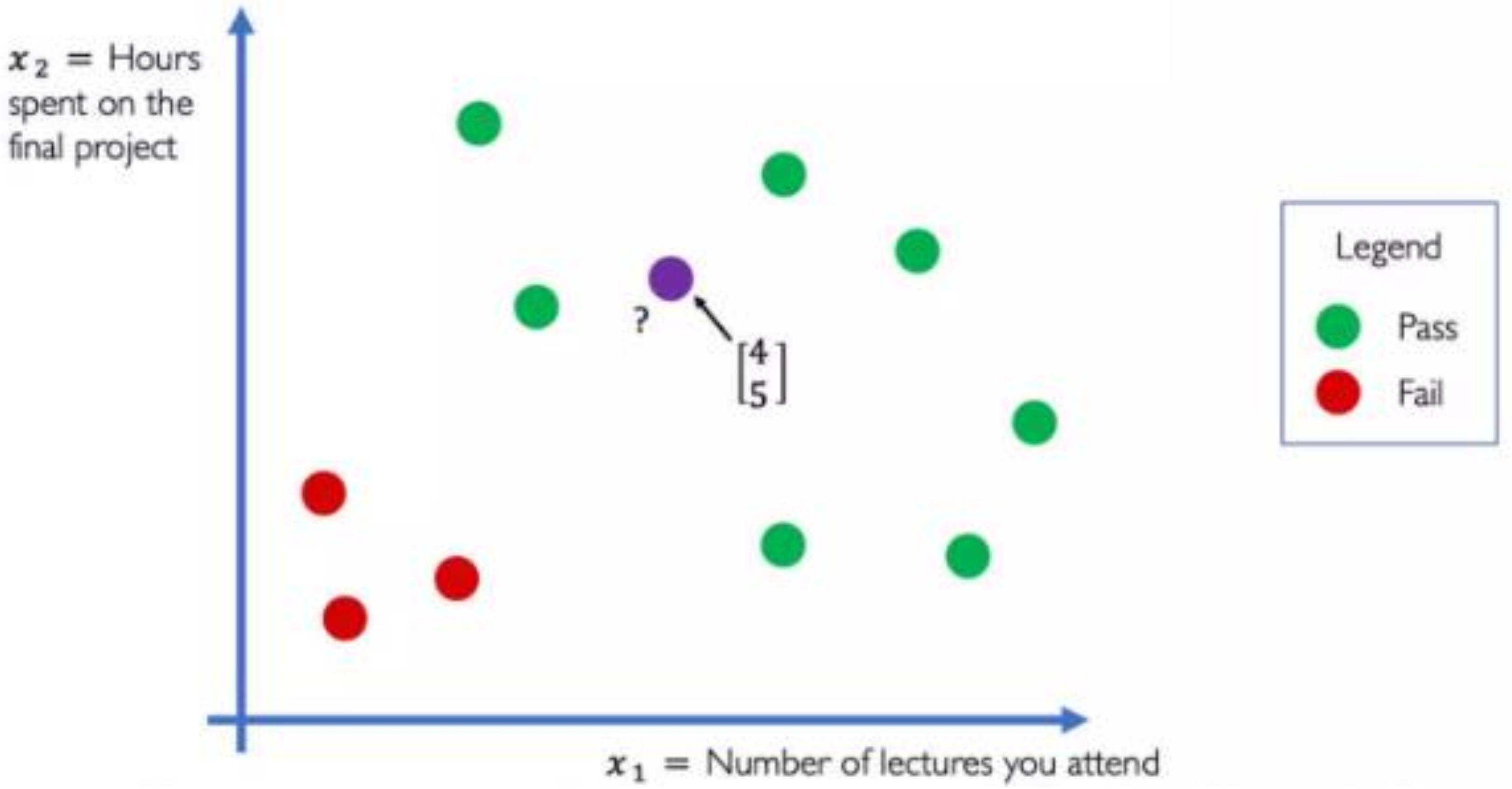
x_1 = Numbers of lectures you attend

x_2 = Hours spent on the final project

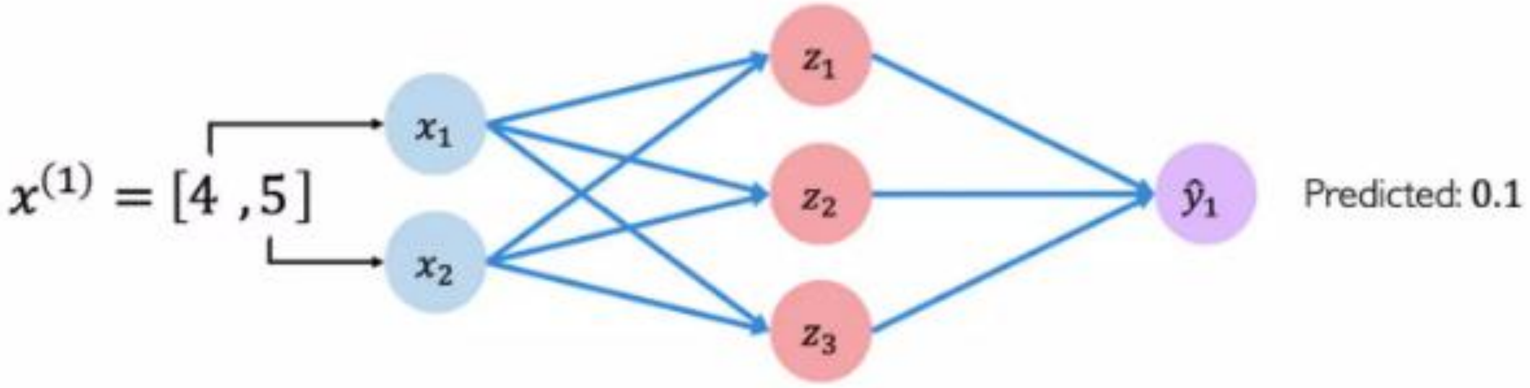
Example Problem: Will I pass this class?



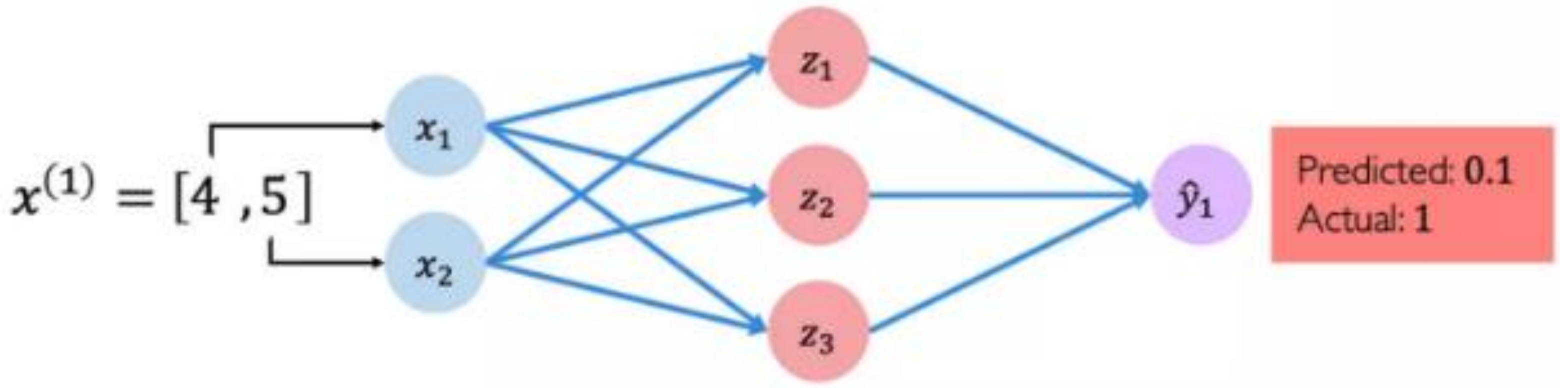
Example Problem: Will I pass this class?



Example Problem: Will I pass this class?

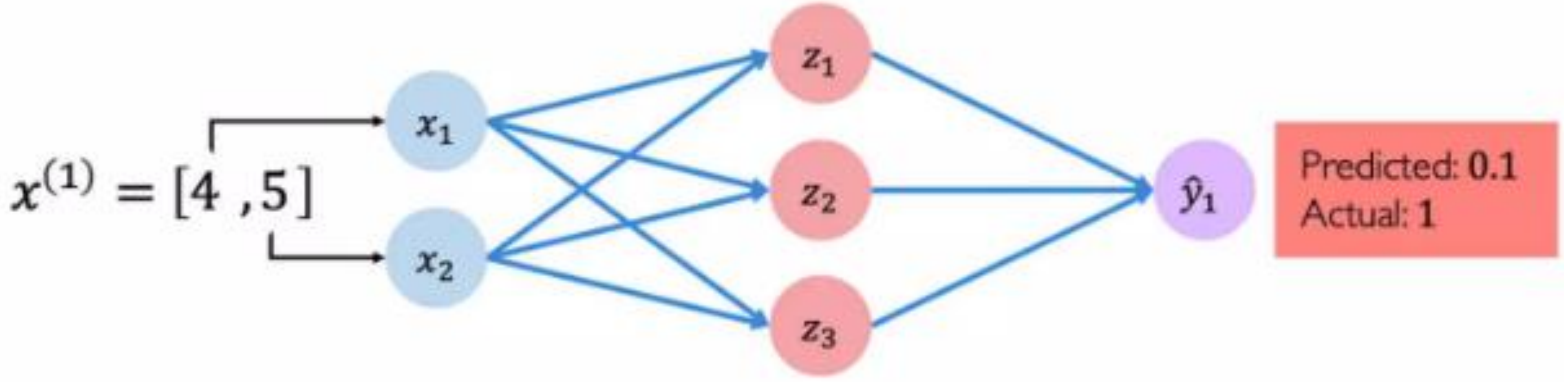


Example Problem: Will I pass this class?



Quantifying Loss

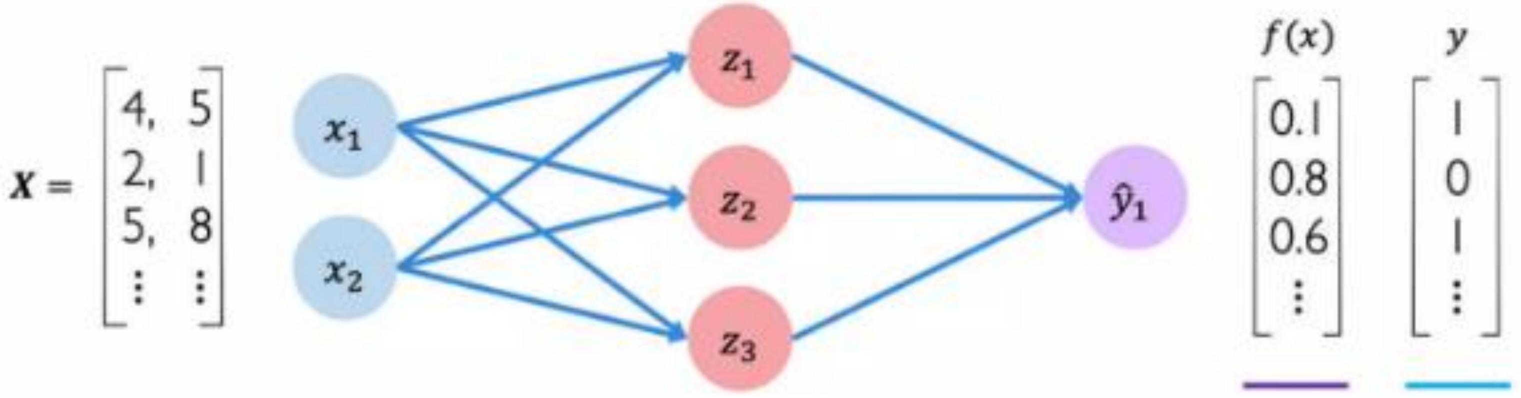
*The **loss** of our network measures the cost incurred from incorrect predictions*



$$\mathcal{L}\big(\underbrace{f\left(x^{(i)}; \boldsymbol{W}\right)}_{\text{Predicted}}, \underbrace{y^{(i)}}_{\text{Actual}}\big)$$

Empirical Loss

The **empirical loss** measures the total loss over our entire dataset

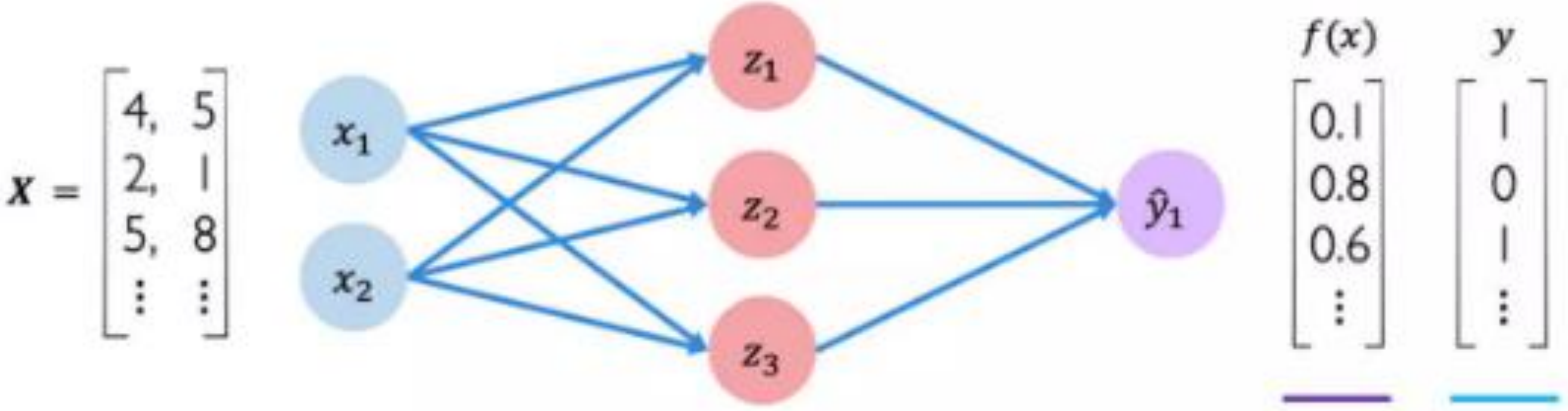


$$J(W) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}\Big(\underbrace{f\big(x^{(i)};W\big)}_{\text{Predicted}},\underbrace{y^{(i)}}_{\text{Actual}}\Big)$$

Also known as:
- Objective function
- Cost function
- Empirical Risk

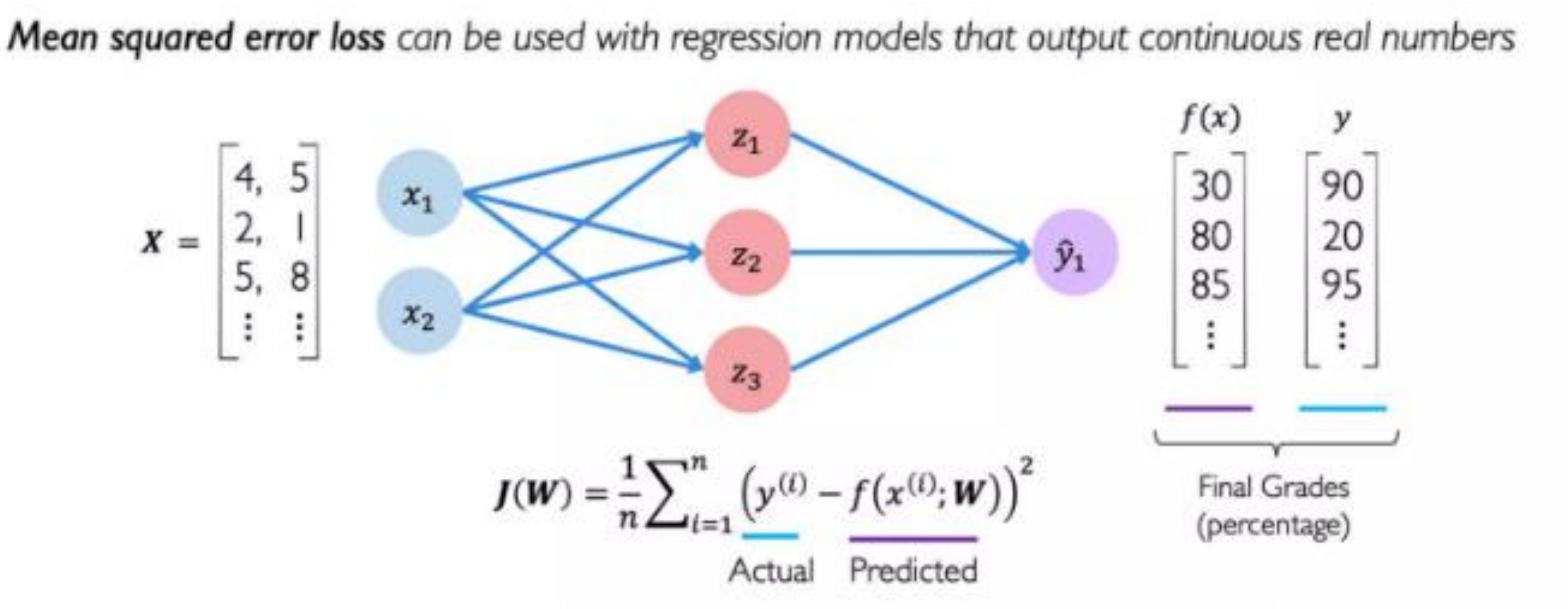
Binary Cross Entropy Loss

Cross entropy loss *can be used with models that output a probability between 0 and 1*



$$J(\boldsymbol{W}) = \frac{1}{n}\sum_{i=1}^{n} \underbrace{y^{(i)}}_{\text{Actual}} \log\Big(\underbrace{f\big(x^{(i)};\boldsymbol{W}\big)}_{\text{Predicted}}\Big) + (1 - \underbrace{y^{(i)}}_{\text{Actual}}) \log\Big(1 - \underbrace{f\big(x^{(i)};\boldsymbol{W}\big)}_{\text{Predicted}}\Big)$$

Mean Squared Error Loss



The Perceptron from Scratch

- Let's implement the perceptron architecture from scratch using only numpy.

- Use the *perceptron.py* script as a starting point.