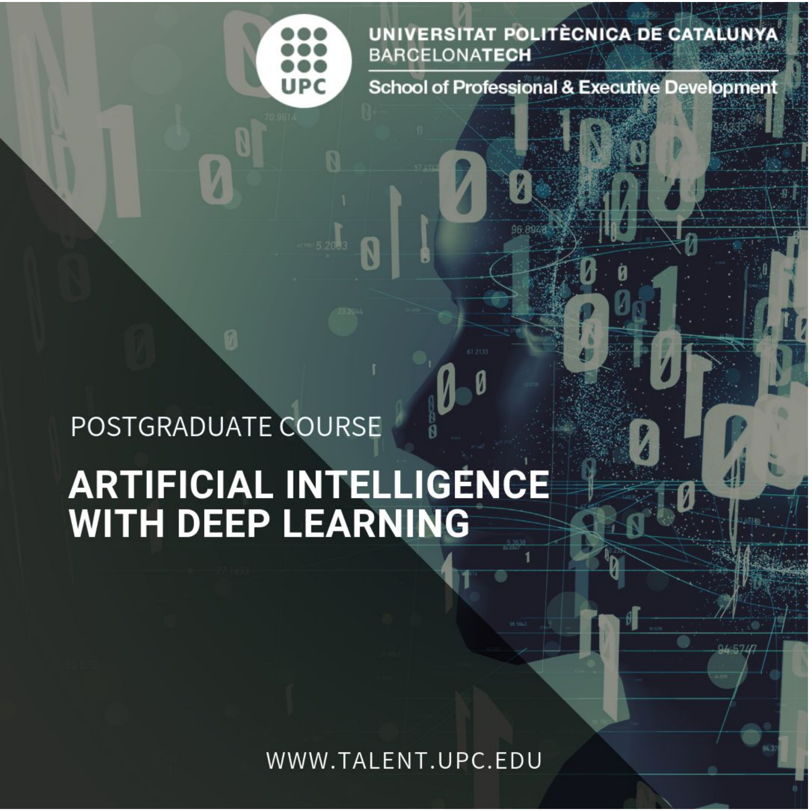


#DLUPC

Day 1 Lecture 3

The Perceptron

Xavier Giro-i-Nieto
xavier.giro@upc.edu

Associate Professor
Universitat Politècnica de Catalunya
Technical University of Catalonia

Acknowledgements

Santiago Pascual

Kevin McGuinness

kevin.mcguinness@dcu.ie

Research Fellow

Insight Centre for Data Analytics

Dublin City University

Video lectures

Santiago Pascual, [DLSL 2017]

[Xavier Giro-i-Nieto], [DLAI 2017]

Outline

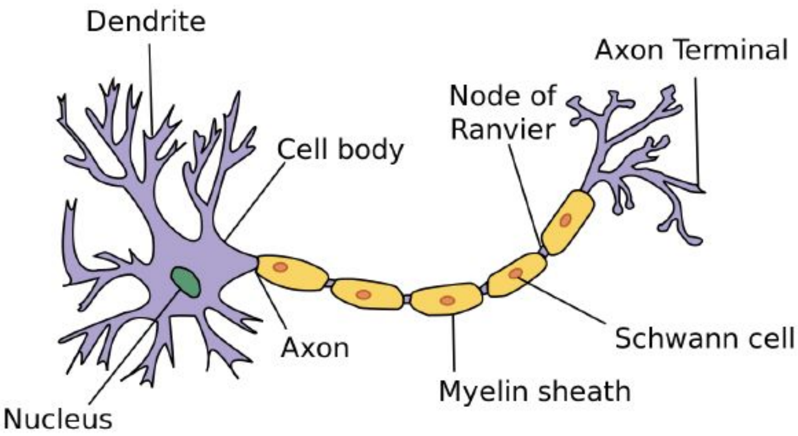
1. **Single neuron models (perceptrons)**
    a. Linear regression
    b. Logistic regression
    c. Multiple outputs and softmax regression
2. Limitations of the perceptron

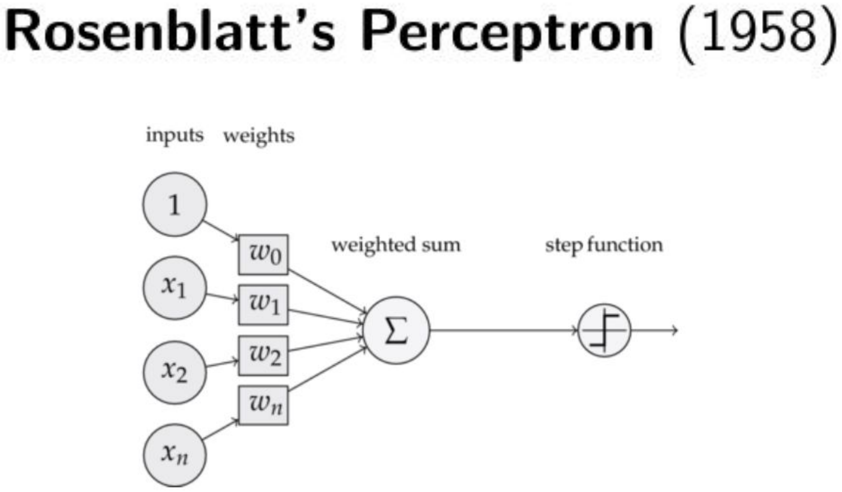
Single neuron model (perceptron)

The Perceptron is seen as an **analogy** to a biological neuron.

Biological neurons fire an impulse once the sum of all inputs is over a threshold.

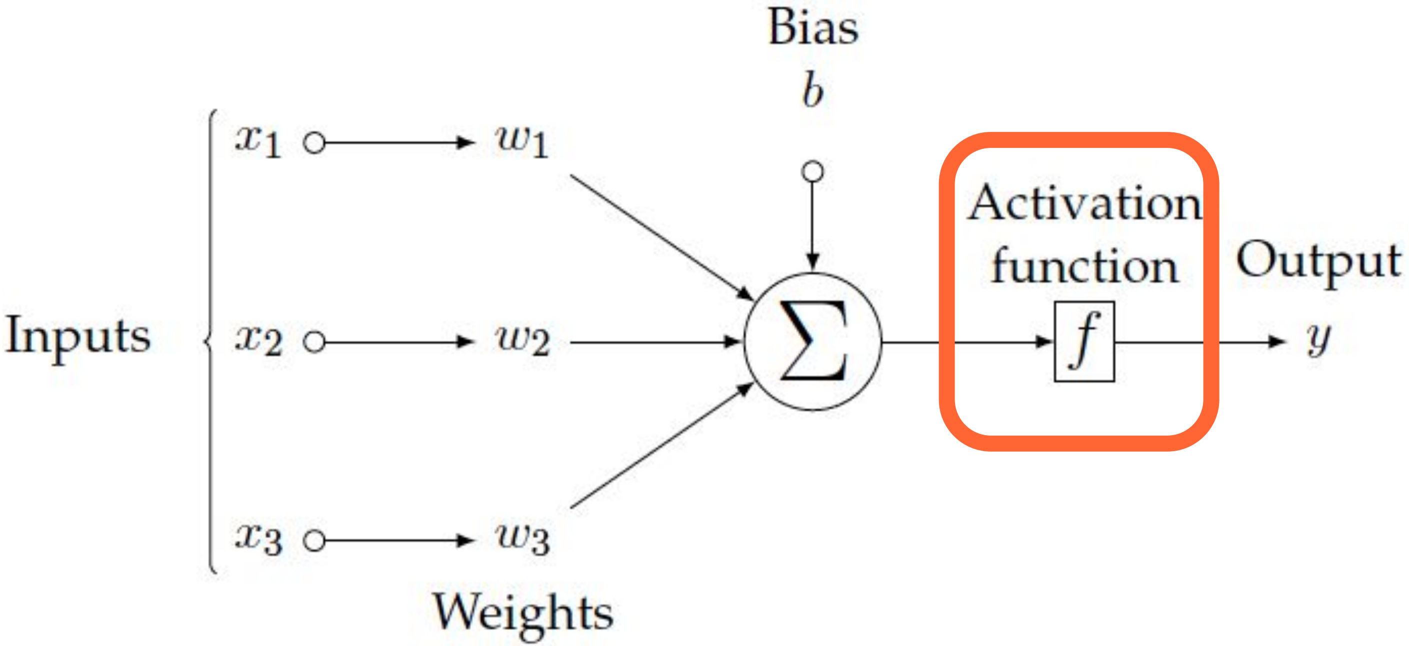
The perceptron acts like a switch (learn how in the next slides...).



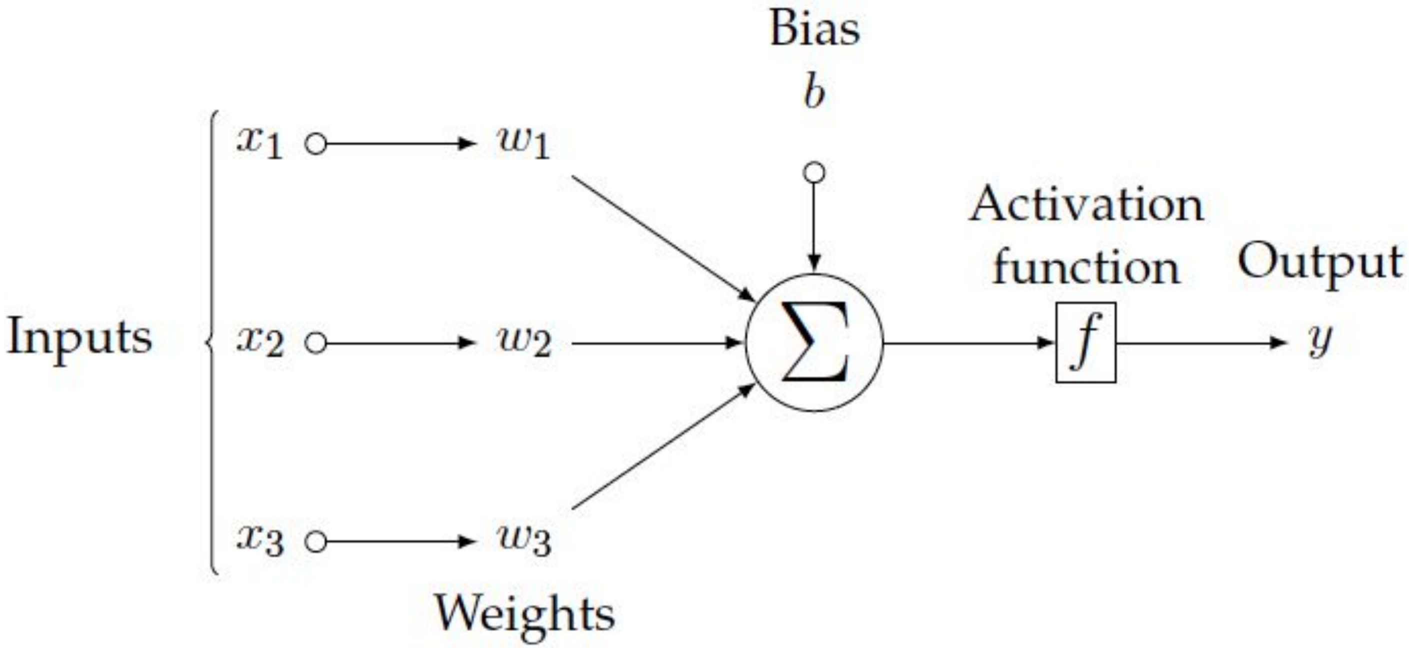


Single Neuron Model (Perceptron)

The perceptron is suitable for both regression or classification problems, depending on the chosen **activation function**.

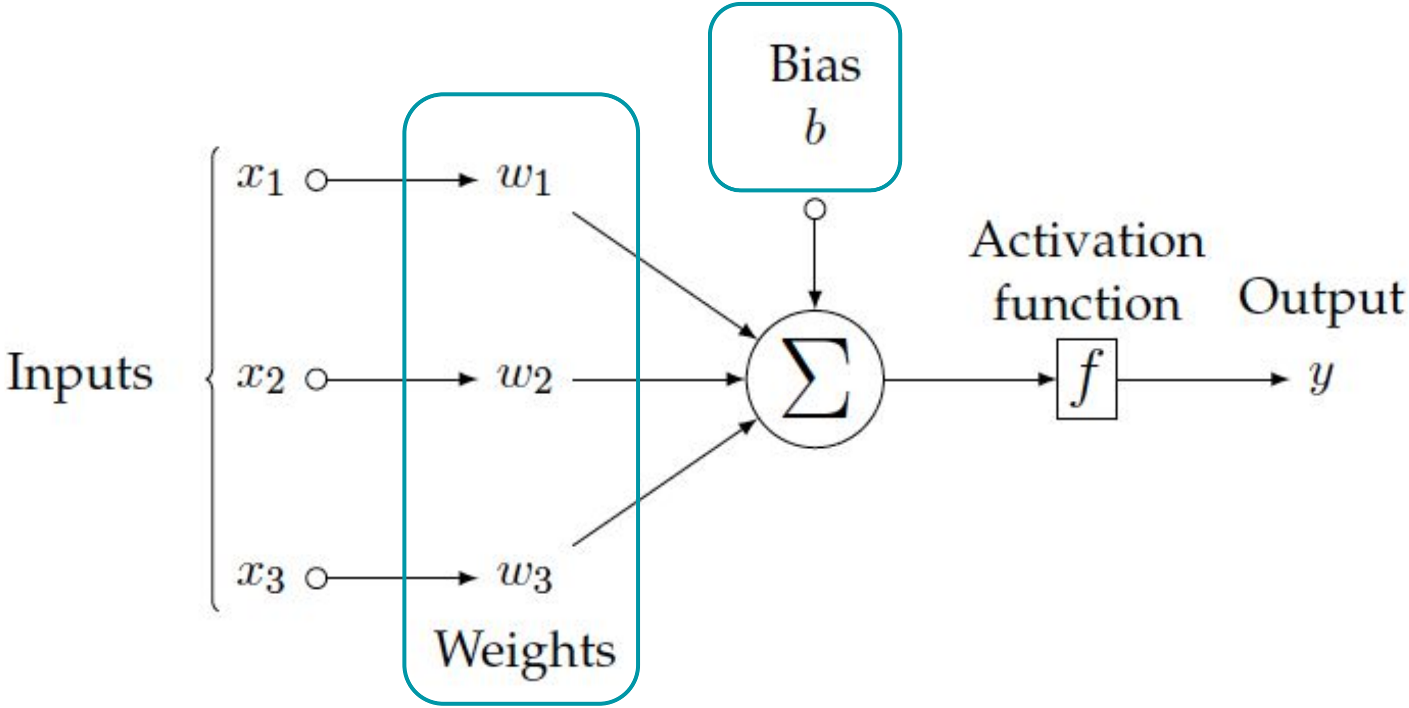


Single neuron model (perceptron)



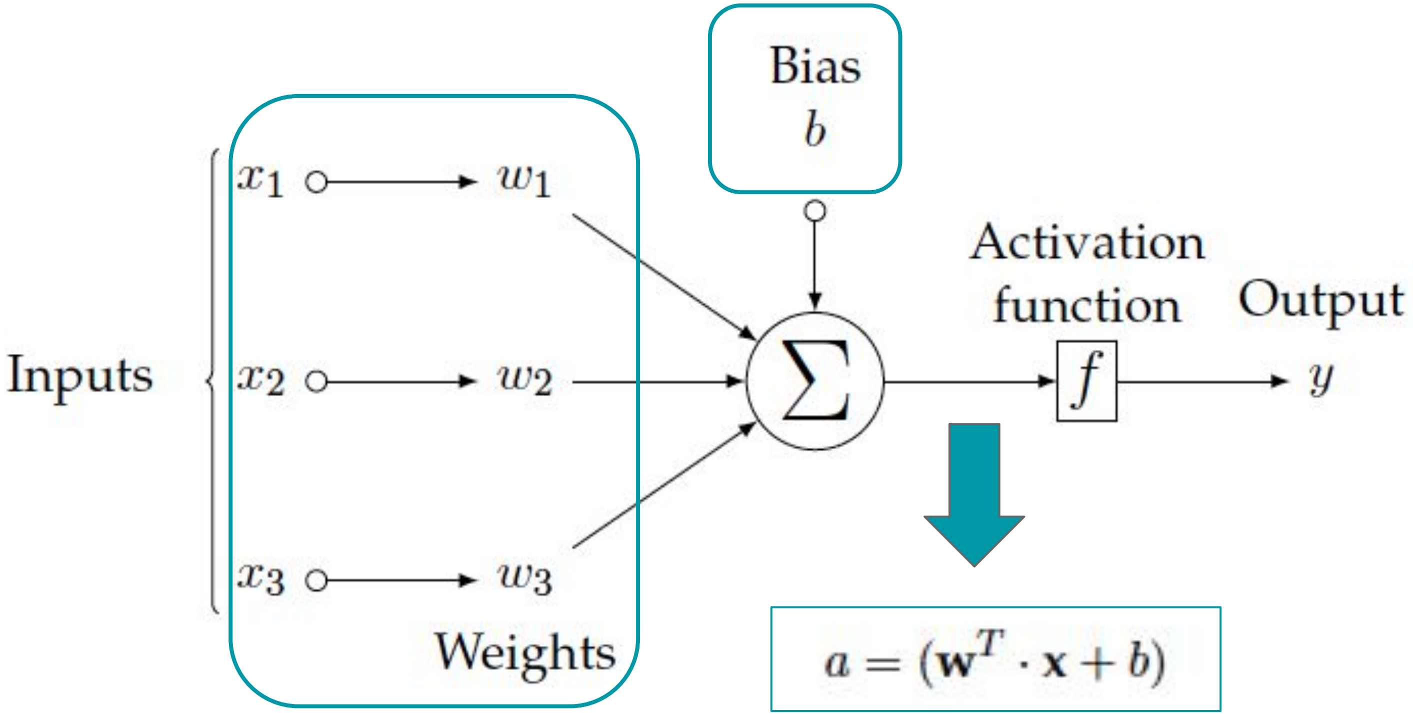
Single neuron model (perceptron)

Weights and bias are the parameters that define the behavior. They must be estimated during training.



Single neuron model (perceptron)

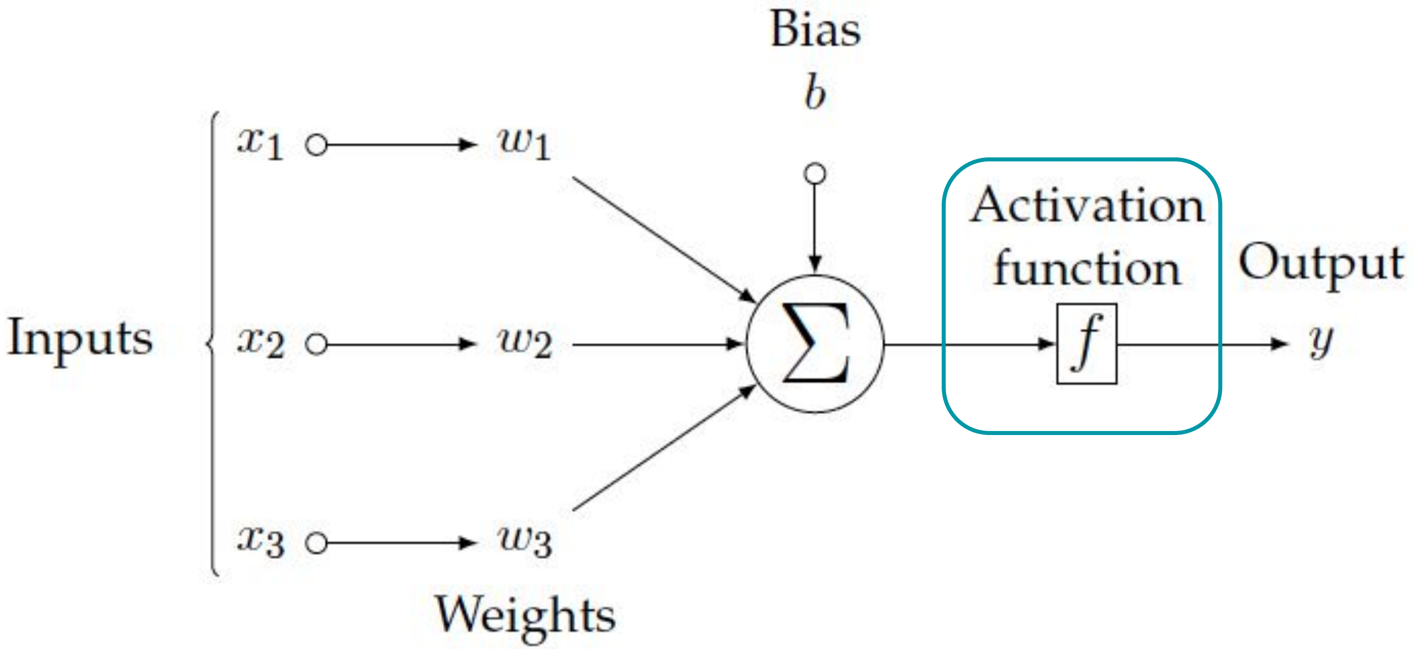
The output y is derived from a sum of the **weighted** inputs plus a **bias** term.



$$a = (\mathbf{w}^T \cdot \mathbf{x} + b)$$

Single neuron model (perceptron)

The **activation function** introduces non-linearities.



Single neuron model

Activation functions:

- They act as a **threshold**

Desirable properties

- Mostly smooth, continuous, differentiable
- Fairly linear

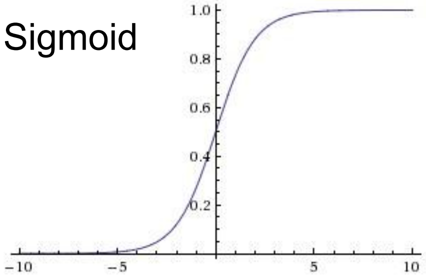
Common nonlinearities

- Sigmoid
- Tanh
- ReLU = max(0, x)

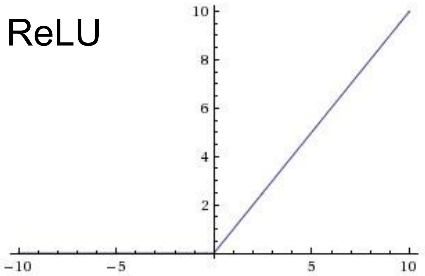
Why do we need them?

If we only use linear layers we are only able to learn linear transformations of our input.

Sigmoid

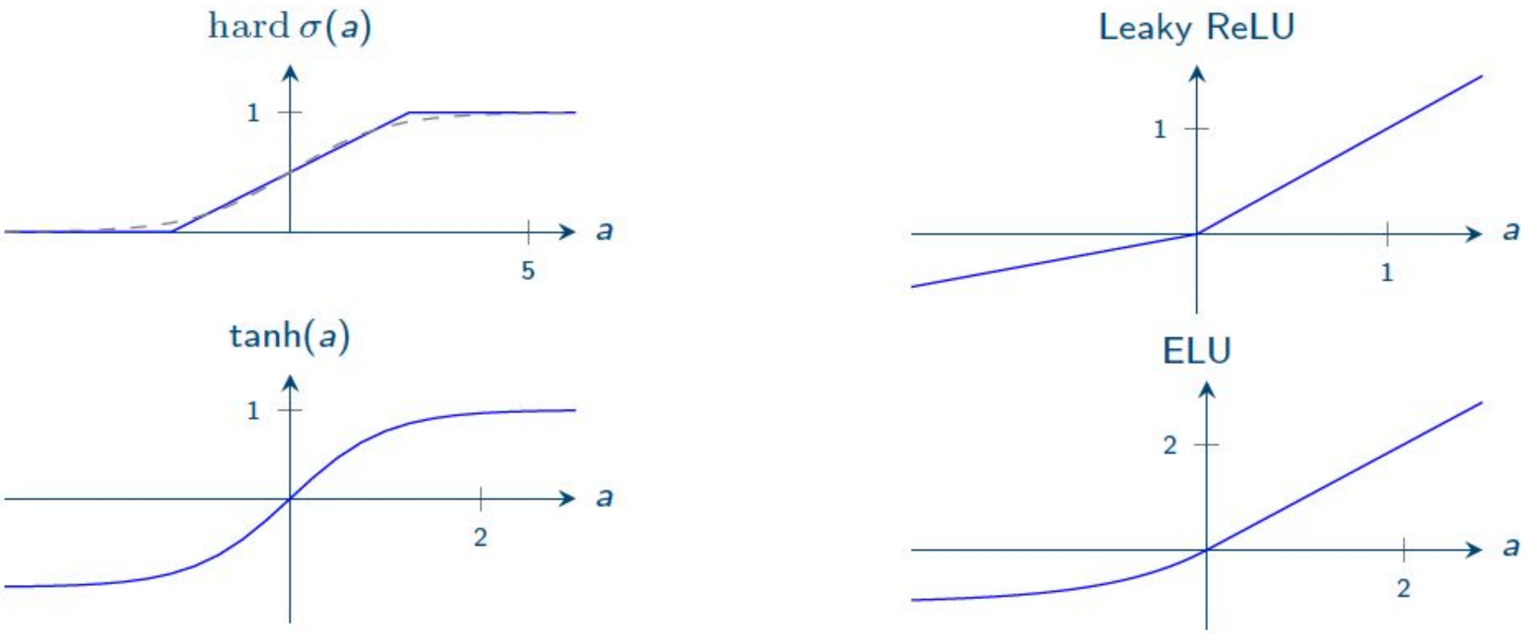


ReLU



Single neuron model: Regression

Other popular activation functions:



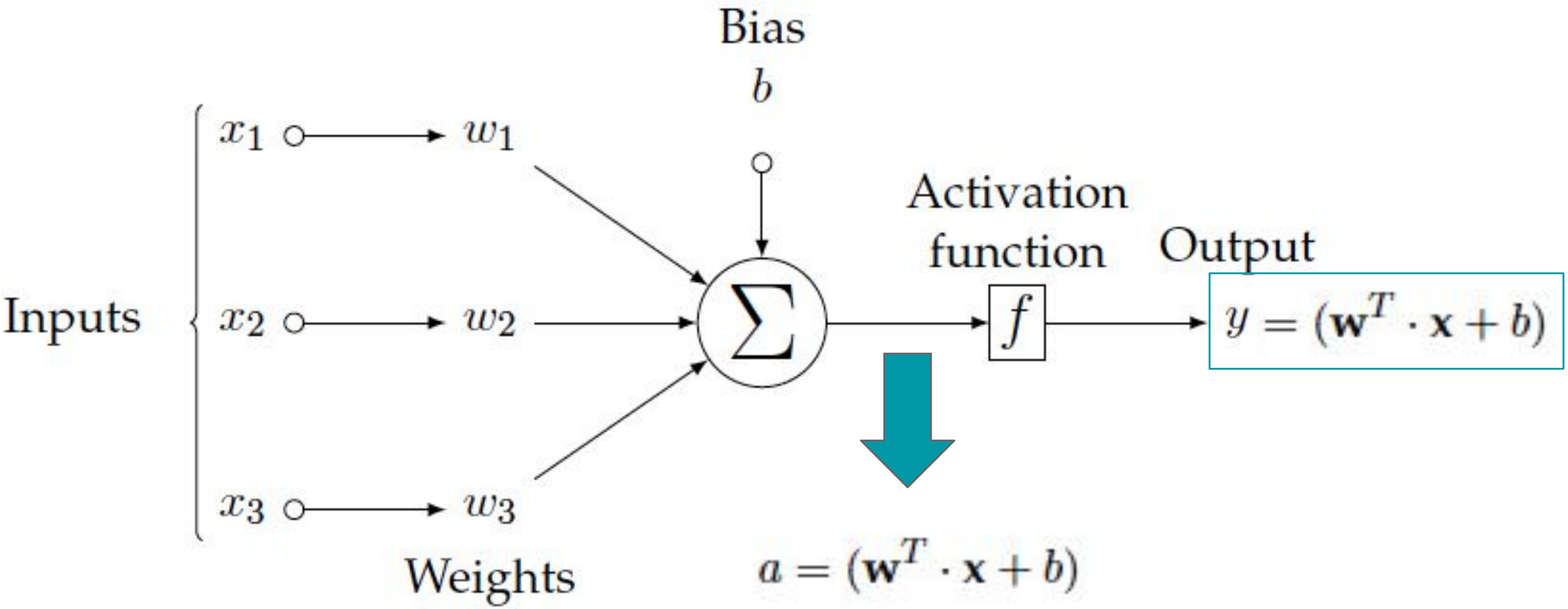
Outline

1. Single neuron models (perceptrons)
    a. **Linear regression**
    b. Logistic regression
    c. Multiple outputs and softmax regression
2. Limitations of the perceptron

Single neuron model: Linear Regression

A single neuron scheme can solve linear regression problems when f(a)=a.

[identity]

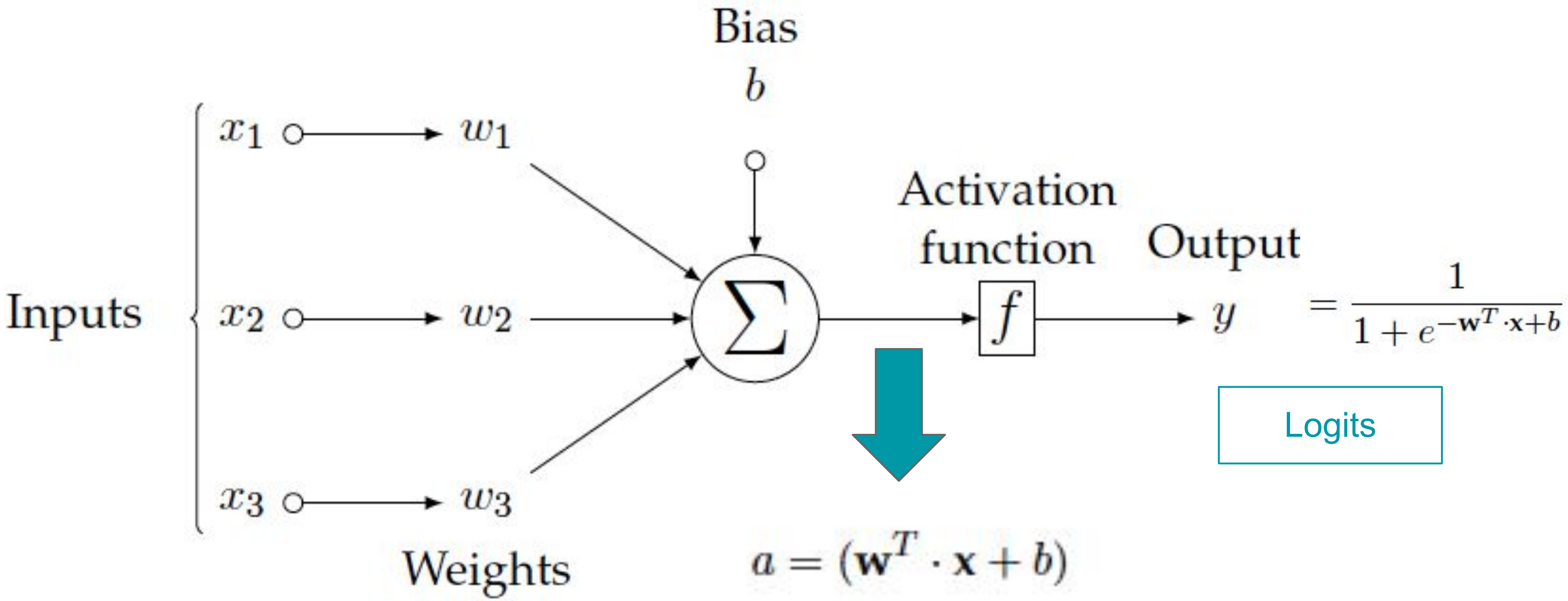


Outline

1. Supervised learning: regression/classification
2. Single neuron models (perceptrons)
    a. Linear regression
    b. Logistic regression
    c. Multiple outputs and softmax regression

Single neuron model: Logistic Regression

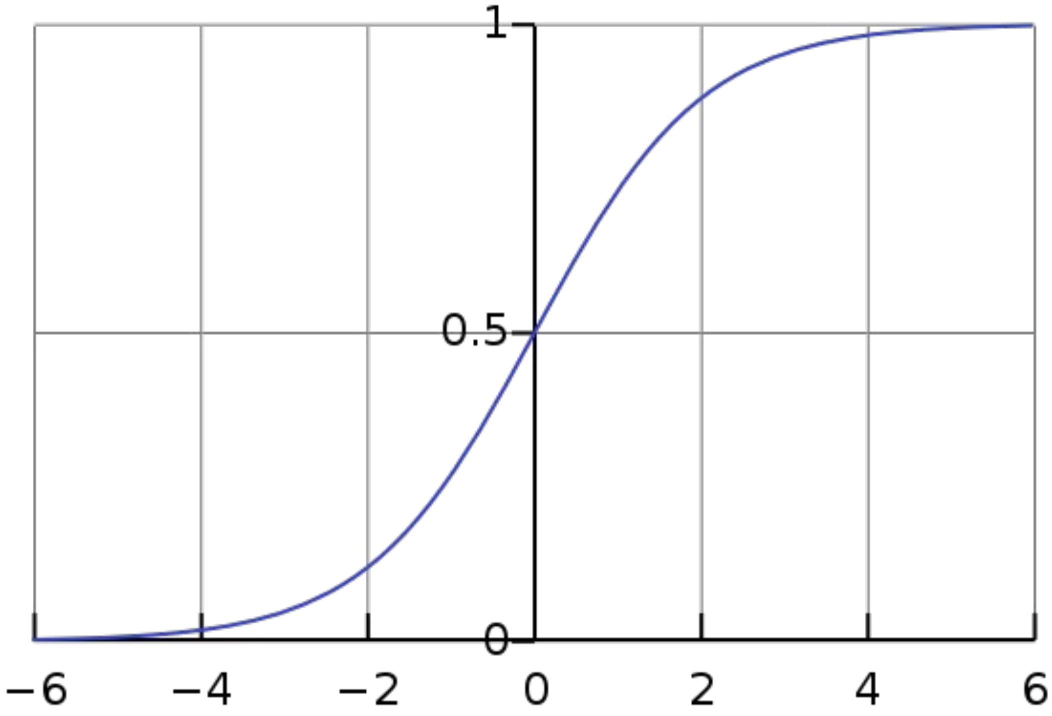
The perceptron is suitable for <u>classification</u> problems when f(a)=σ(a). [sigmoid]



Single neuron model: Logistic Regression

The **sigmoid function σ(x)** or **logistic curve** maps any input x between [0,1]:

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

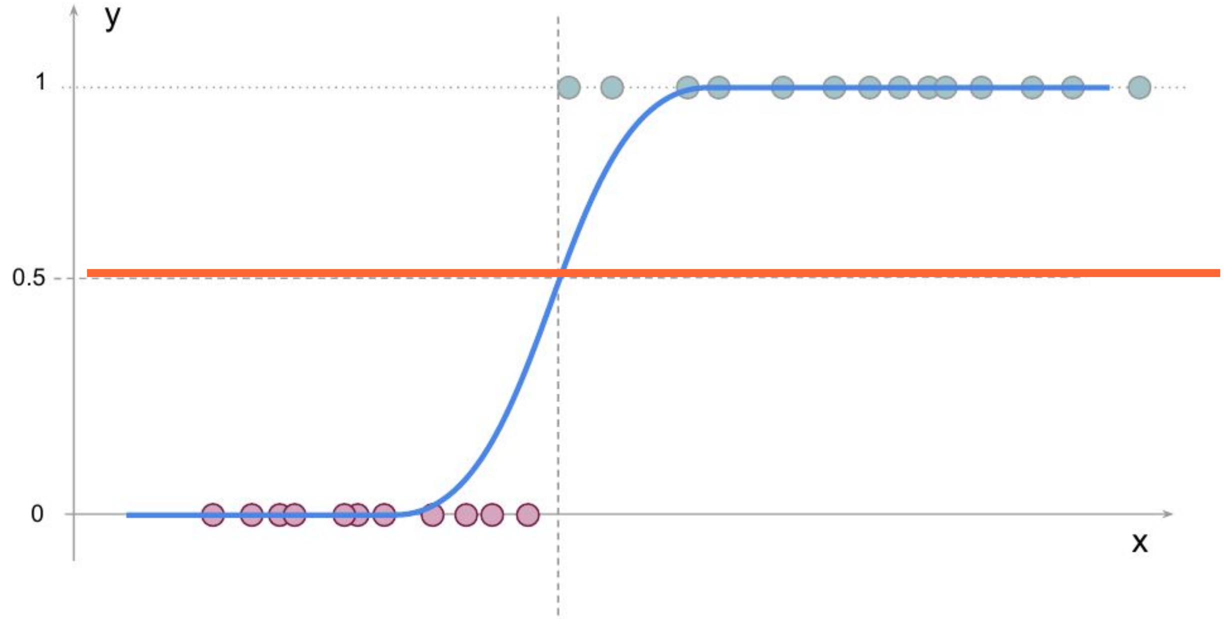


Single neuron model: Binary Classification

For classification, regressed values should be collapsed into 0 and 1 to quantize the confidence of the predictions ("probabilities").

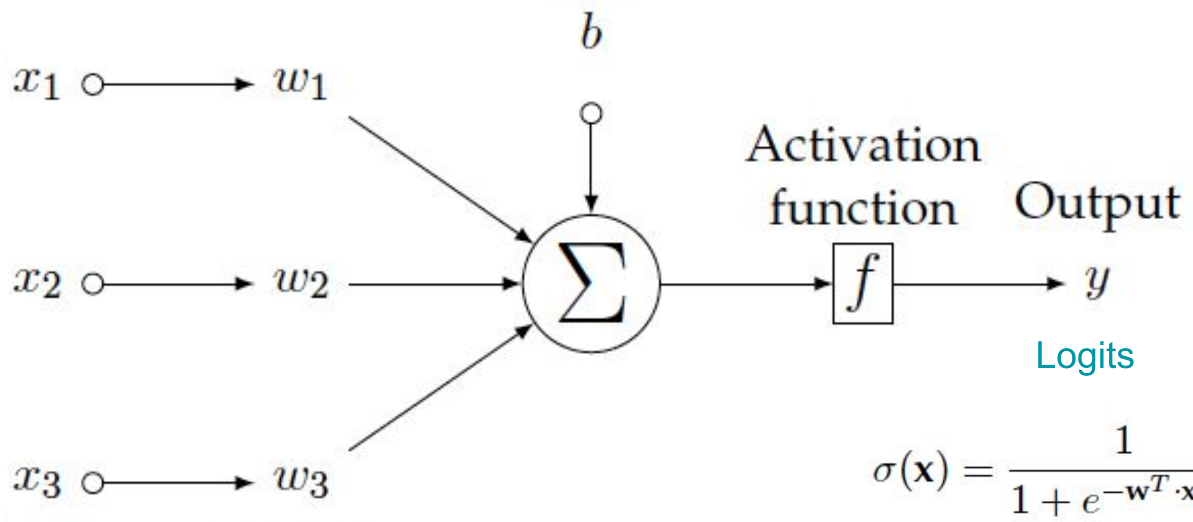
$$\sigma(\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T \cdot \mathbf{x} + b}}$$

Threshold (thr)



Single neuron model: Binary Classification

Setting a **threshold (thr)** at the output of the perceptron allows solving classification problems between two classes (binary):



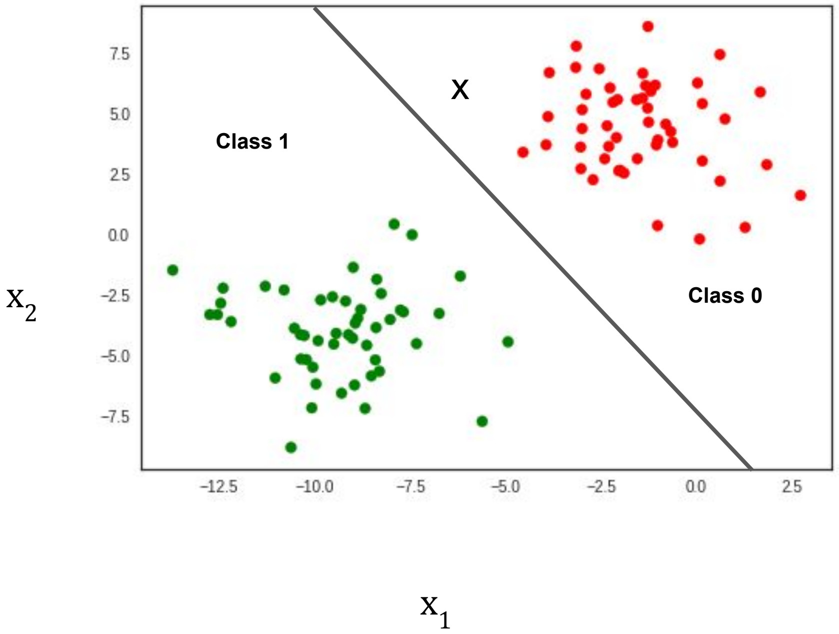
$$\sigma(\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T \cdot \mathbf{x} + b}}$$

y > thr → class 1 (eg. green)

y < thr → class 2 (eg. red)

Single neuron model: Binary Classification

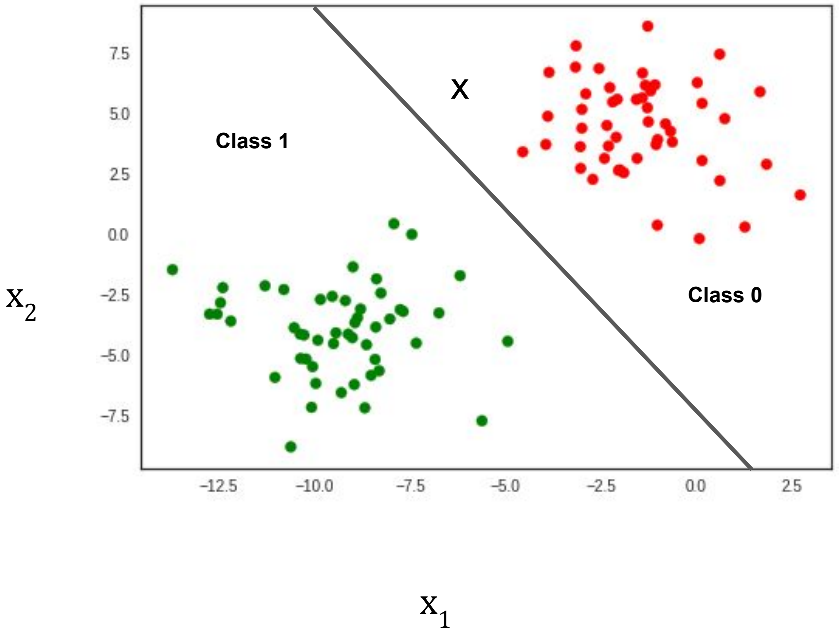
2D input space data



$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > \ 0.5 \\ 0 & \text{otherwise} \end{cases}$$

Single neuron model: Binary Classification

2D input space data

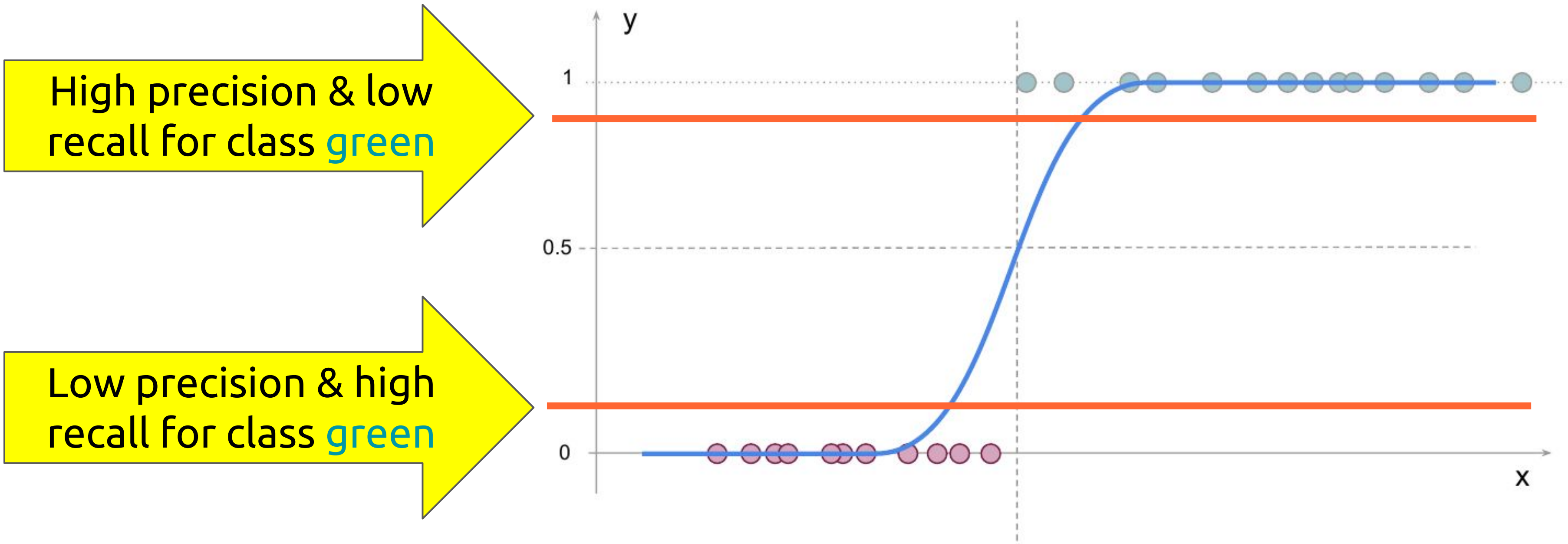


Parameters of the line.
They are find based on training data
- *Learning Stage*.

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

Single neuron model: Binary Classification

The classification threshold can be adjusted based on the desired precision - recall trade-off:



Single neuron model: Binary Classification

Consider a binary classifier implemented with a single neuron modelled by two weights $w_1=0.2$ and $w_2=0.8$ and a bias $b=-1$. Consider the activation function to be a sigmoid $f(x) = 1 / (1+e^{-x})$.

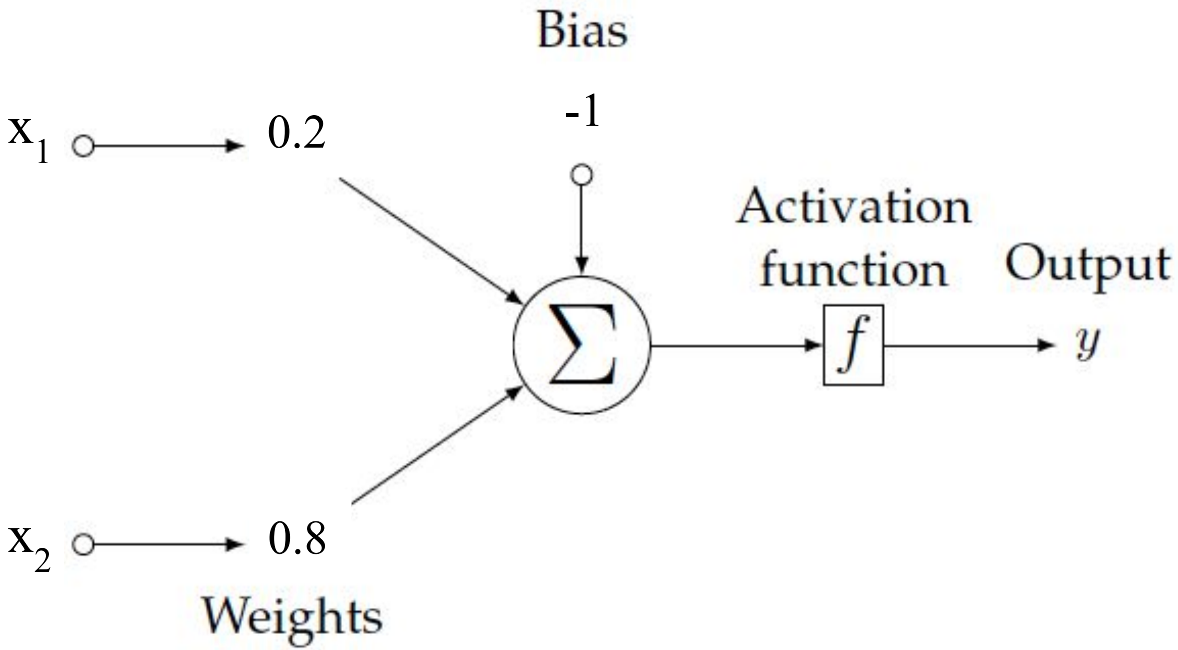
a) Draw a scheme of the model.

b) Compute the output of the logistic regressor for a given input $x=[1,1]$.

c) Considering a classification threshold of $y_{th}=0$ ($y_{th}>0.9$ for class A, and $y_{th}<0.9$ for class B), which class would be predicted for the considered input $x=[1,1]$?

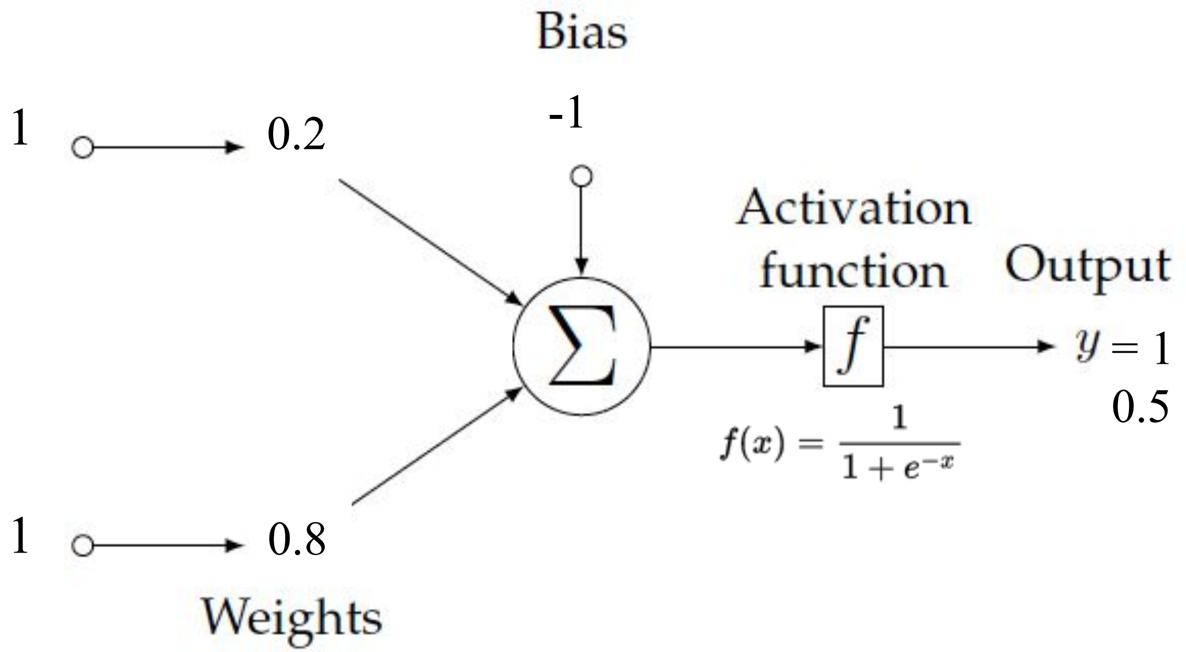
Single neuron model: Binary Classification

Solution: Draw a scheme of the model.



Single neuron model: Binary Classification

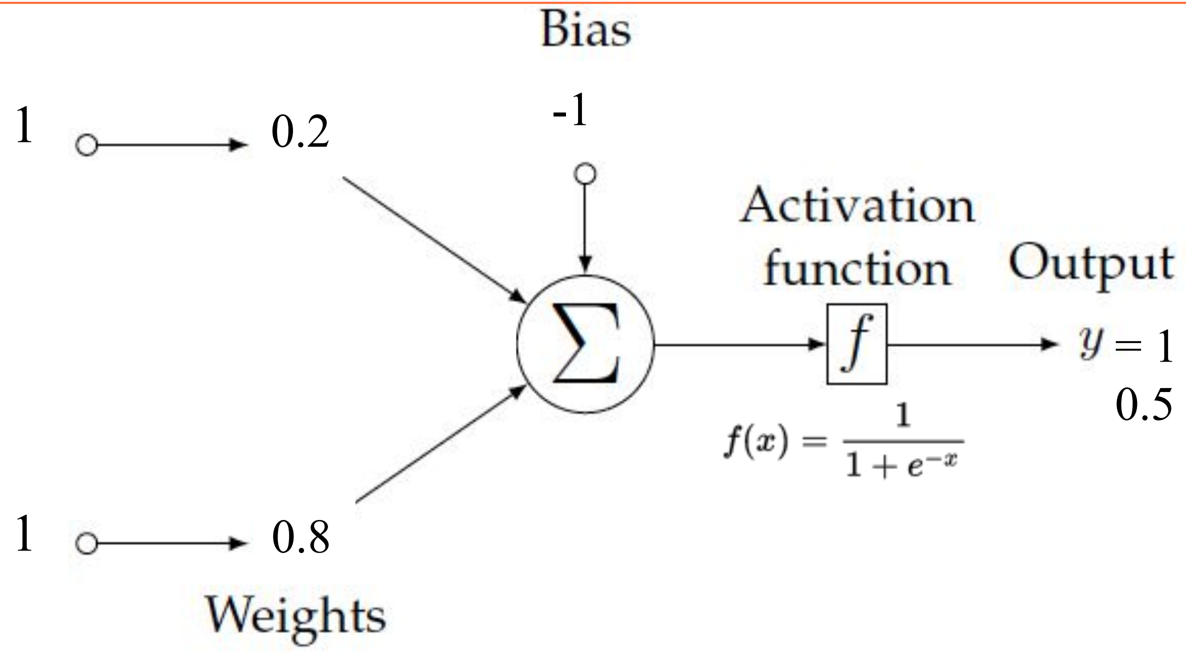
Solution: Compute the output of the logistic regressor for a given input x=[1,1].



$y = 1 \,/\, (1+e^{-(1\cdot 0.2+1\cdot 0.8-1)})=1/(1+1) = 0.5$

Single neuron model: Binary Classification

<u>Solution</u>: Considering a classification threshold of y_{th}=0 (y_{th}>0.9 for class A, and y_{th}<0.9 for class B), which class would be predicted for the considered input x=[1,1] ?



$y = 1 / (1+e^{-(1\cdot0.2+1\cdot0.8-1)})=1/(1+1) = 0.5$

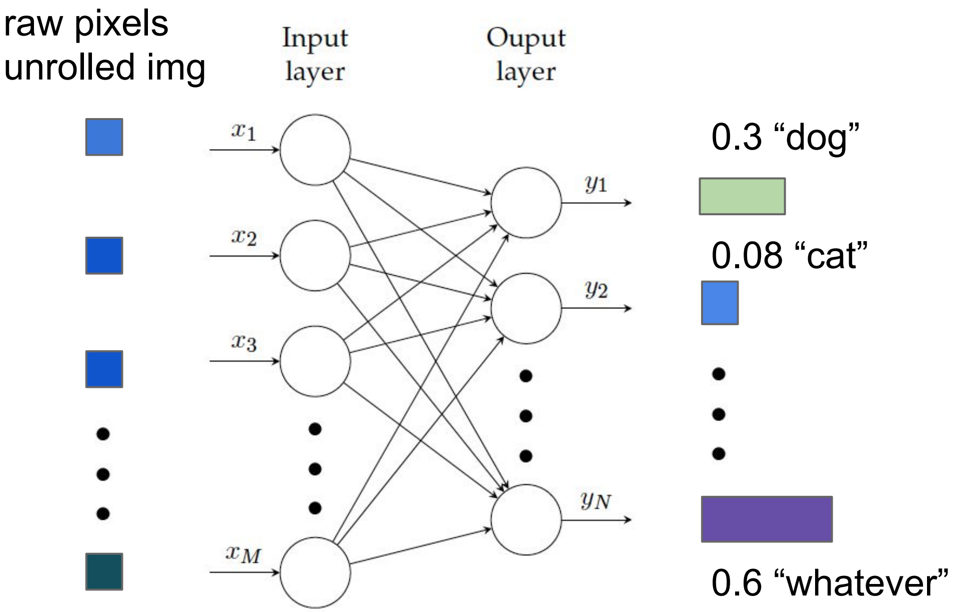
y=0.5 < 0.9 → class B

Outline

1. Supervised learning: regression/classification
2. Single neuron models (perceptrons)
   a. Linear regression
   b. Logistic regression
   c. **Softmax regression**
3. Limitations of the perceptron

Softmax regression: Multiclass (N classes)

Multiple classes can be predicted by putting many perceptrons in parallel, and normalizing their outputs with an exponential function:



Softmax regression

$$P(y = k|\mathbf{x}) = \frac{\exp \mathbf{x}^T \mathbf{w}_k}{\boxed{\sum_{n=1}^{N} \exp \mathbf{x}^T \mathbf{w}_n}}$$

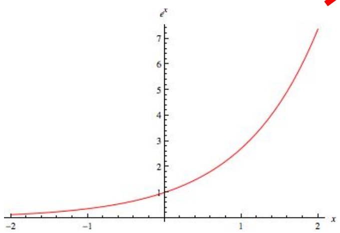
Normalization factor so that the sum of probabilities sum up to 1.

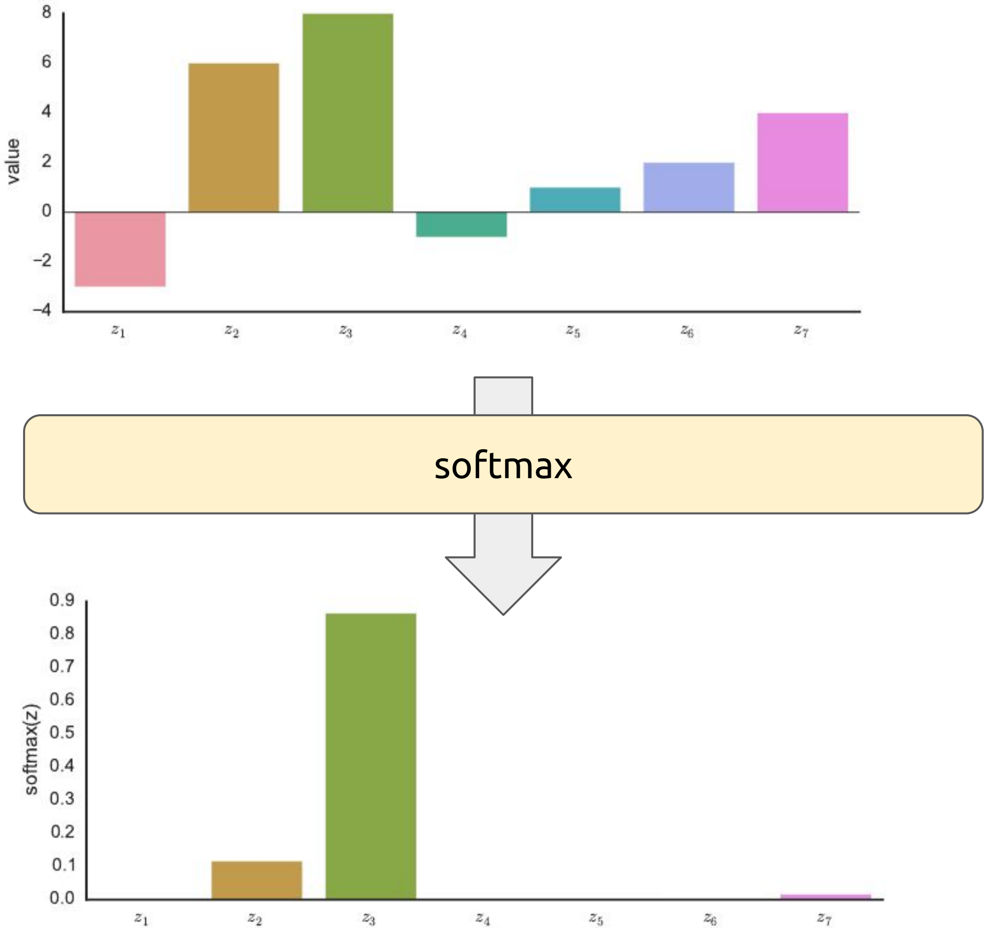
28

Softmax regression

Exponential *exp(xi)* boosts higher logits.

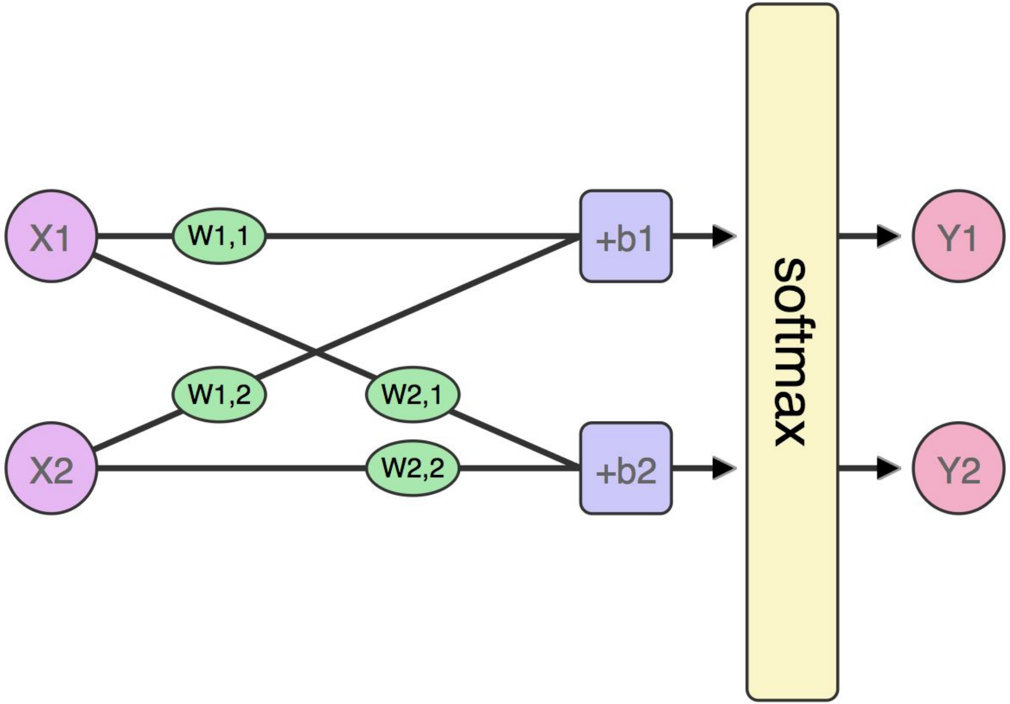
$$\mathrm{softmax}(\mathbf{x}) = \frac{1}{\sum_{j=1}^{K} \exp(x_j)} \begin{bmatrix} \exp(x_1) \\ \exp(x_2) \\ \vdots \\ \exp(x_K) \end{bmatrix}$$





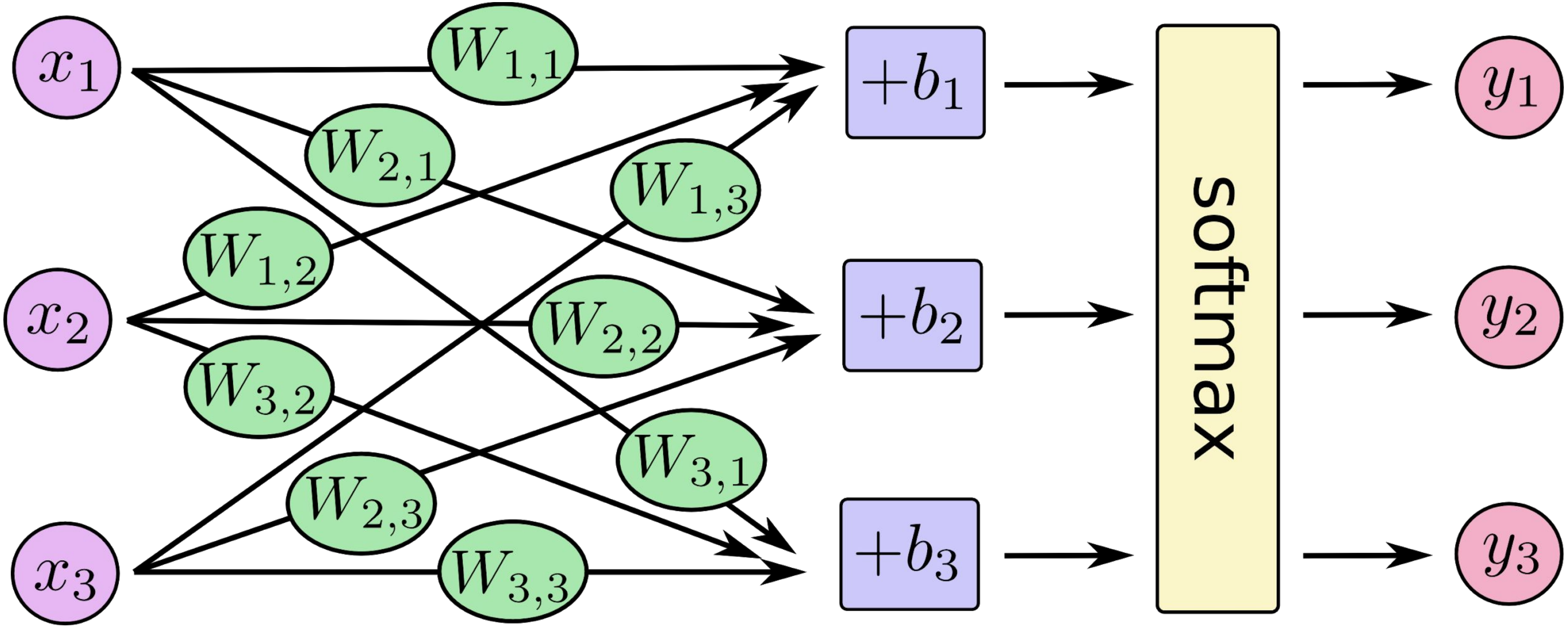
Softmax regression: Binary case

Example: Binary classification can also be solved with two perceptrons + softmax.



J. Alammar, [“A visual and interactive guide to the Basics of Neural Networks”]() (2016)

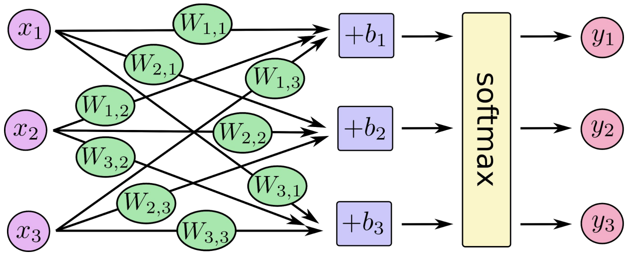
Softmax regression: Multiclass (3 classes)



TensorFlow, [“MNIST for ML beginners”]

Softmax regressor: Multiclass (3 classes)

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \text{softmax} \begin{pmatrix} W_{1,1}x_1 + W_{1,2}x_2 + W_{1,3}x_3 + b_1 \\ W_{2,1}x_1 + W_{2,2}x_2 + W_{2,3}x_3 + b_2 \\ W_{3,1}x_1 + W_{3,2}x_2 + W_{3,3}x_3 + b_3 \end{pmatrix}$$

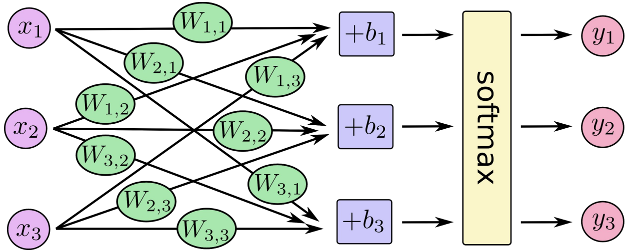


TensorFlow, [“MNIST for ML beginners”]

Softmax regressor: Multiclass (3 classes)

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \text{softmax}\left(\begin{bmatrix} W_{1,1} & W_{1,2} & W_{1,3} \\ W_{2,1} & W_{2,2} & W_{2,3} \\ W_{3,1} & W_{3,2} & W_{3,3} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \right)$$

$$\boxed{y = \text{softmax}(Wx + b)}$$



TensorFlow, [“MNIST for ML beginners”]

Outline

1. Supervised learning: regression/classification
2. Single neuron models (perceptrons)
    a. Linear regression
    b. Logistic regression
    c. Multiple outputs and softmax regression

Undergradese

What undergrads ask vs. what they're REALLY asking