

# DEEP LEARNING FOR ARTIFICIAL INTELLIGENCE

Master Course UPC ETSETB TelecomBCN Barcelona. Autumn 2017.



## Instructors



Xavier  
Giró-i-Nieto



Marta R.  
Costa-jussà



Jordi  
Torres



Elisa  
Sayrol



Santiago  
Pascual



Verónica  
Vilaplana



Ramon  
Morros



Javier  
Ruiz

## Organizers



UNIVERSITAT POLITÈCNICA  
DE CATALUNYA  
BARCELONATECH



Barcelona  
Supercomputing  
Center  
Centre Nacional de Supercomputació

## Supporters

aws  educate

GitHub Education

+ info: <http://dlai.deeplearning.barcelona>

[\[course site\]](http://dlai.deeplearning.barcelona)



#DLUPC

## Day 1 Lecture 2

# The Perceptron



## Xavier Giro-i-Nieto

[xavier.giro@upc.edu](mailto:xavier.giro@upc.edu)

Associate Professor

Universitat Politècnica de Catalunya  
Technical University of Catalonia



# Acknowledgements



Santiago Pascual



Kevin McGuinness

[kevin.mcguinness@dcu.ie](mailto:kevin.mcguinness@dcu.ie)

Research Fellow

Insight Centre for Data Analytics  
Dublin City University



# Video lecture (DLSL 2017)

Winter Seminar UPC TelecomBCN, 24 - 25 January 2017



Instructors



Organizers



+ info: [TelecomBCN.DeepLearning.Barcelona](#)

[\[course site\]](#)

Day 1 Lecture 2

## The Perceptron



Santiago Pascual







UNIVERSITAT POLITÈCNICA DE CATALUNYA  
BARCELONATECH

# Outline

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

# Types of machine learning

Yann Lecun's Black Forest cake



## ■ "Pure" Reinforcement Learning (cherry)

- ▶ The machine predicts a scalar reward given once in a while.
- ▶ **A few bits for some samples**

## ■ Supervised Learning (icing)

- ▶ The machine predicts a category or a few numbers for each input
- ▶ Predicting human-supplied data
- ▶ **10→10,000 bits per sample**

## ■ Unsupervised/Predictive Learning (cake)

- ▶ The machine predicts any part of its input for any observed part.
- ▶ Predicts future frames in videos
- ▶ **Millions of bits per sample**



■ (Yes, I know, this picture is slightly offensive to RL folks. But I'll make it up)

# Types of machine learning

We can categorize three types of learning procedures:

1. Supervised Learning:

$$\mathbf{y} = f(\mathbf{x})$$

2. Unsupervised Learning:

$$f(\mathbf{x})$$

3. Reinforcement Learning:

$$\mathbf{y} = f(\mathbf{x})$$

$\mathbf{z}$





# Types of machine learning

We can categorize three types of learning procedures:

1. Supervised Learning:

$$\mathbf{y} = f(\mathbf{x})$$

2. Unsupervised Learning:

$$f(\mathbf{x})$$

3. Reinforcement Learning:

$$\mathbf{y} = f(\mathbf{x})$$

$\mathbf{z}$

		LECTURES (D5-010)		LAB
Week	Tuesday	15:00	16:00	17:00
1	19/9/2017	Welcome (XG)	Perceptron (XG)	Getting Started (JT)
2	26/9/2017	Multilayer Perceptron (ES)	Visit BSC (JT)	Visit BSC (JT)
3	3/10/2017	Layers (ES)	Frameworks (JT)	Keras & Tensorboard (JT)
4	10/10/2017	Backpropagation (VV)	Software stack (JT)	PyTorch (JT)
5	17/10/2017	Optimizers (VV)	Computational requirements (JT)	Project (XG)
6	24/10/2017	Loss function (JR)	Transfer learning (RM)	TensorFlow (JT)
7	31/10/2017	Methodology (JR)	Incremental (RM)	GPU (JT)
8	7/11/2017	Midterm (ES)	Midterm (ES)	Project (XG)
9	14/11/2017	RNNs (MRC)	Gated RNNs (MRC)	Project (XG)
10	21/11/2017	Attention (MRC)	Reinforcement (XG)	Project (XG)
11	28/11/2017	Unsupervised (XG)	Generative (SP)	GAN (SP)
12	5/12/2017	HOLIDAYS	HOLIDAYS	HOLIDAYS
13	12/12/2017	Project presentations (XG)	Project presentations (MRC)	Project presentations (ES)
14	19/12/2017	Jose M. Álvarez (XG)	Cristian Canton (XG)	Sponsors & Logistics (ES)

# Types of machine learning

We can categorize three types of learning procedures:

1. Supervised Learning:

$$\mathbf{y} = f(\mathbf{x})$$

2. Unsupervised Learning:

$$f(\mathbf{x})$$

3. Reinforcement Learning:

$$\mathbf{y} = f(\mathbf{x})$$

$$\mathbf{z}$$



# Types of machine learning

We can categorize three types of learning procedures:

## 1. Supervised Learning:

$$y = f(\mathbf{x})$$



We have a labeled dataset with pairs  $(\mathbf{x}, \mathbf{y})$ , e.g. classify a signal window as containing speech or not:

$\mathbf{x}_1 = [x(1), x(2), \dots, x(T)]$      $\mathbf{y}_1 = \text{"no"}$   
 $\mathbf{x}_2 = [x(T+1), \dots, x(2T)]$      $\mathbf{y}_2 = \text{"yes"}$   
 $\mathbf{x}_3 = [x(2T+1), \dots, x(3T)]$      $\mathbf{y}_3 = \text{"yes"}$   
...



# Supervised learning

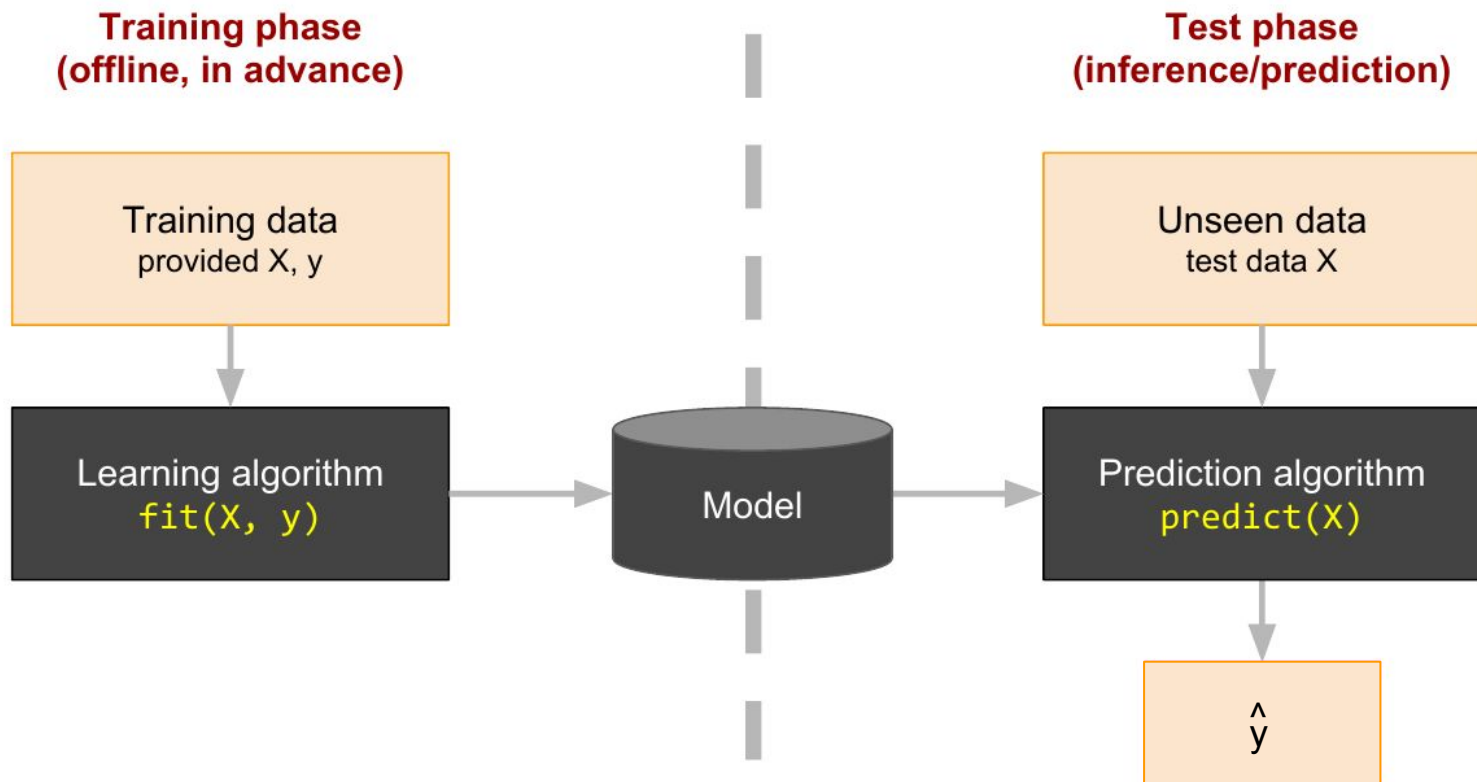
Fit a function:  $y = f(\mathbf{x})$ ,  $\mathbf{x} \in \mathbb{R}^m$

Given paired training examples  $\{(\mathbf{x}_i, y_i)\}$

Key point: **generalize well to unseen examples**



# Black box abstraction of supervised learning



# Regression vs Classification

Depending on the type of target  $y$  we get:

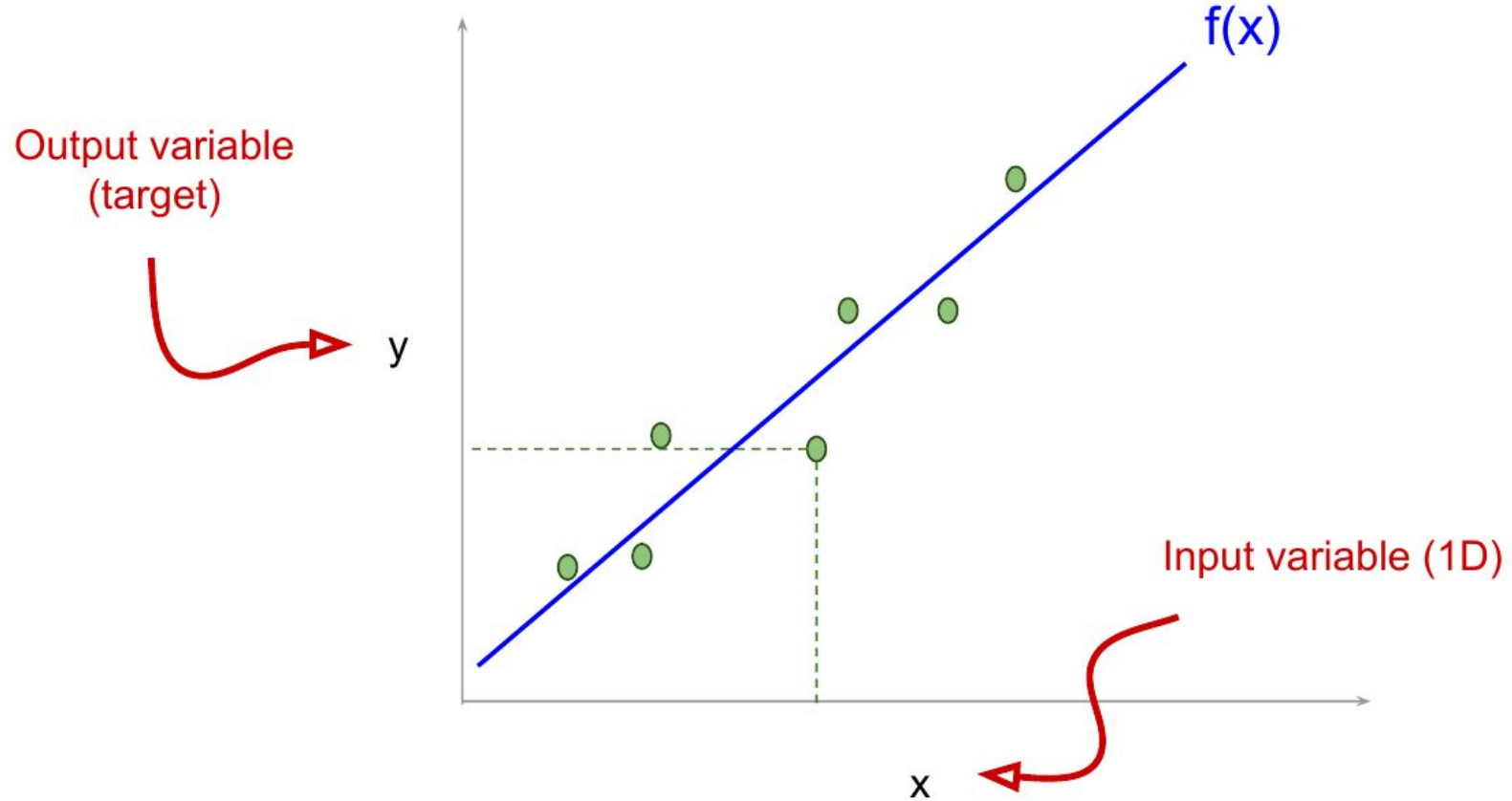
- **Regression**:  $y \in \mathbb{R}^N$  is continuous (e.g. temperatures  $y = \{19^\circ, 23^\circ, 22^\circ\}$ )
- **Classification**:  $y$  is discrete (e.g.  $y = \{1, 2, 5, 2, 2\}$ ).

# Regression vs Classification

Depending on the type of target  $y$  we get:

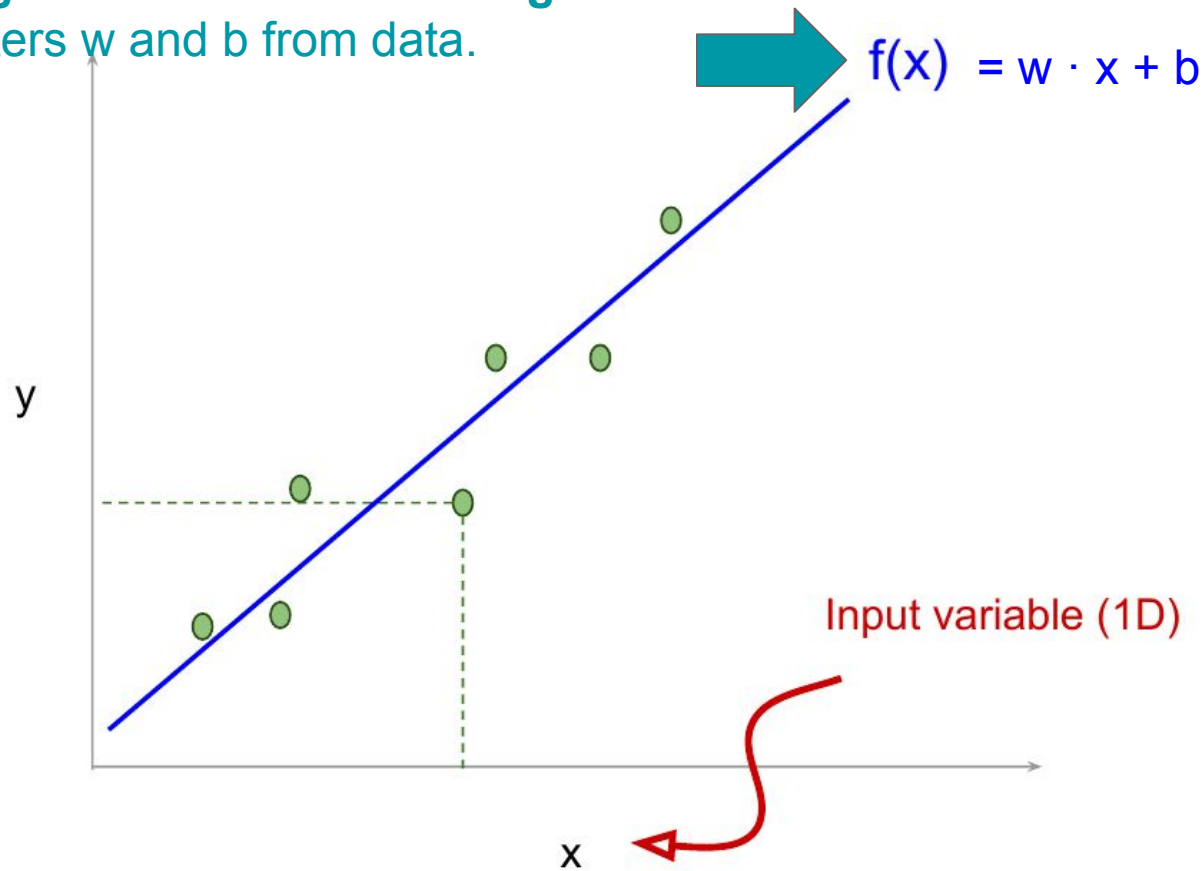
- **Regression**:  $y \in \mathbb{R}^N$  is **continuous** (e.g. temperatures  $y = \{19^\circ, 23^\circ, 22^\circ\}$ )
- **Classification**:  $y$  is **discrete** (e.g.  $y = \{1, 2, 5, 2, 2\}$ ).

# Linear Regression (eg. 1D input - 1D output)



# Linear Regression (eg. 1D input - 1D output)

Training a model means learning parameters  $w$  and  $b$  from data.





# Linear Regression (M-D input)

Input data can also be M-dimensional with vector  $\mathbf{x}$ :

$$y = \mathbf{w}^T \cdot \mathbf{x} + b = w1 \cdot x1 + w2 \cdot x2 + w3 \cdot x3 + \dots + wM \cdot xM + b$$

e.g. we want to predict the **price of a house (y)** based on:

$x1$  = square-meters (sqm)

$x2,3$  = location (lat, lon)

$x4$  = year of construction (yoc)

$y$  = price =  $w1 \cdot (\text{sqm}) + w2 \cdot (\text{lat}) + w3 \cdot (\text{lon}) + w4 \cdot (\text{yoc}) + b$

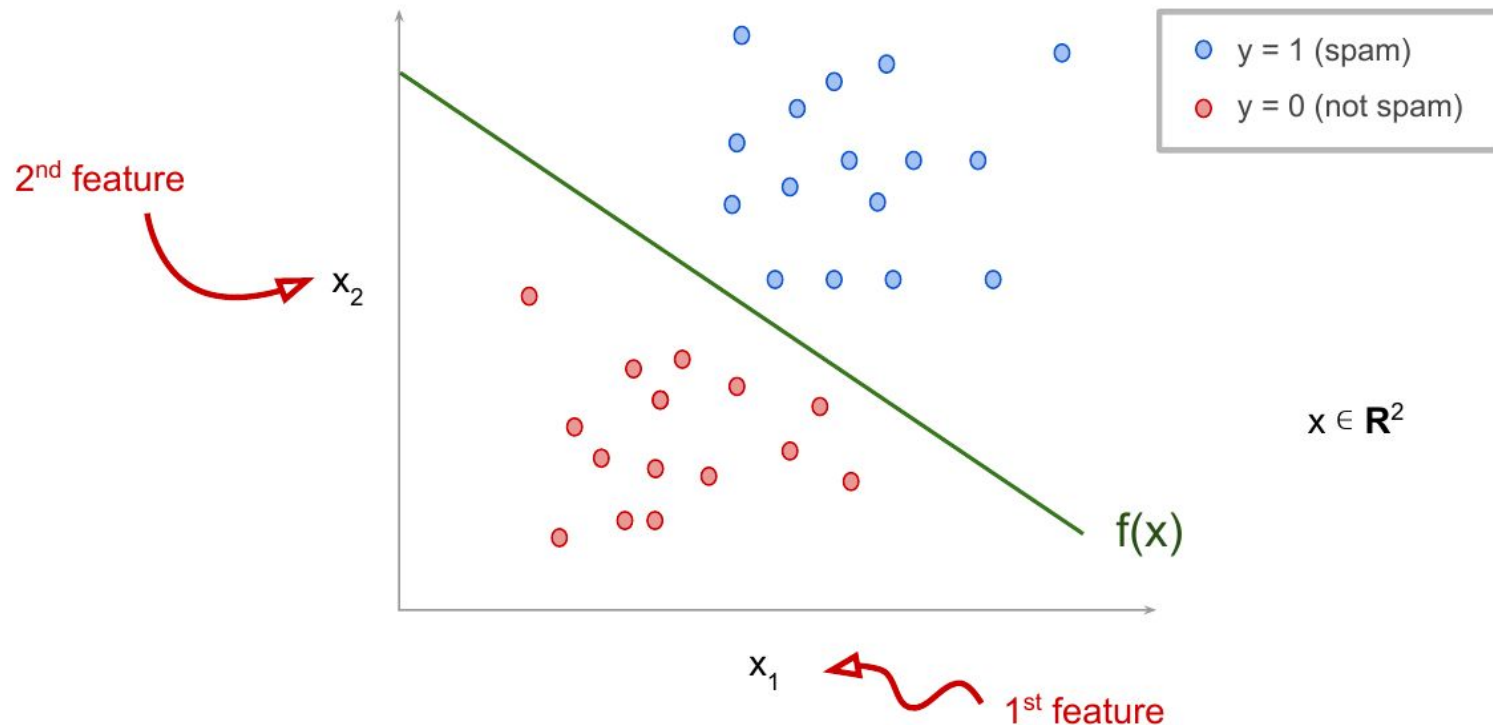


# Regression vs Classification

Depending on the type of target  $y$  we get:

- **Regression**:  $y \in \mathbb{R}^N$  is **continuous** (e.g. temperatures  $y = \{19^\circ, 23^\circ, 22^\circ\}$ )
- **Classification**:  $y$  is **discrete** (e.g.  $y = \{1, 2, 5, 2, 2\}$ ).

# Binary Classification (eg. 2D input, 1D output)

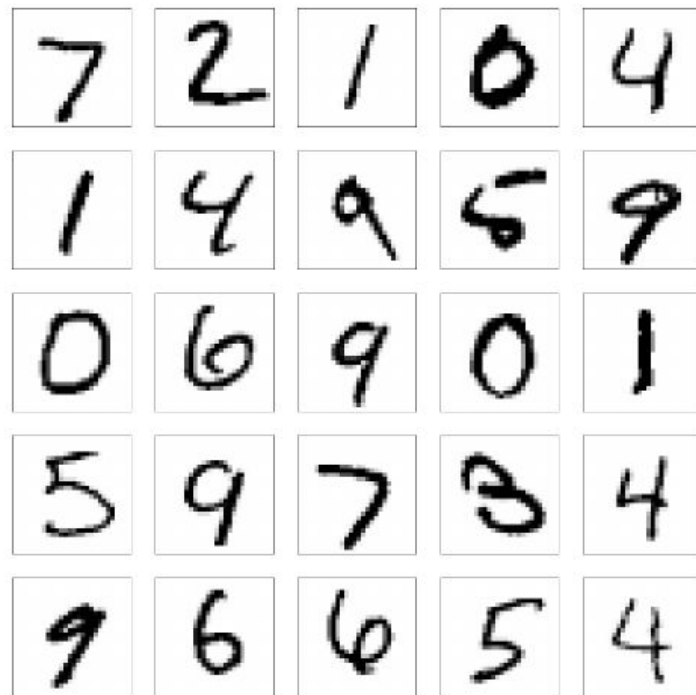


# Multi-class Classification

Produce a classifier to map from pixels to the digit.

- ▶ If images are grayscale and  $28 \times 28$  pixels in size, then  $\mathbf{x}_i \in \mathbb{R}^{784}$
- ▶  $y_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

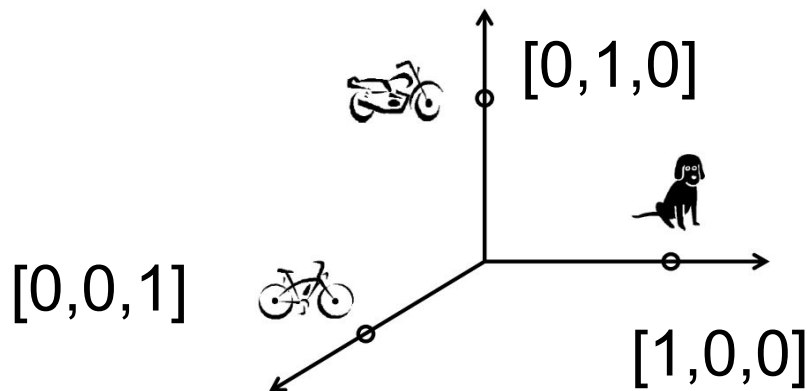
Example of a **multi-class classification** task.



# Multi-class Classification

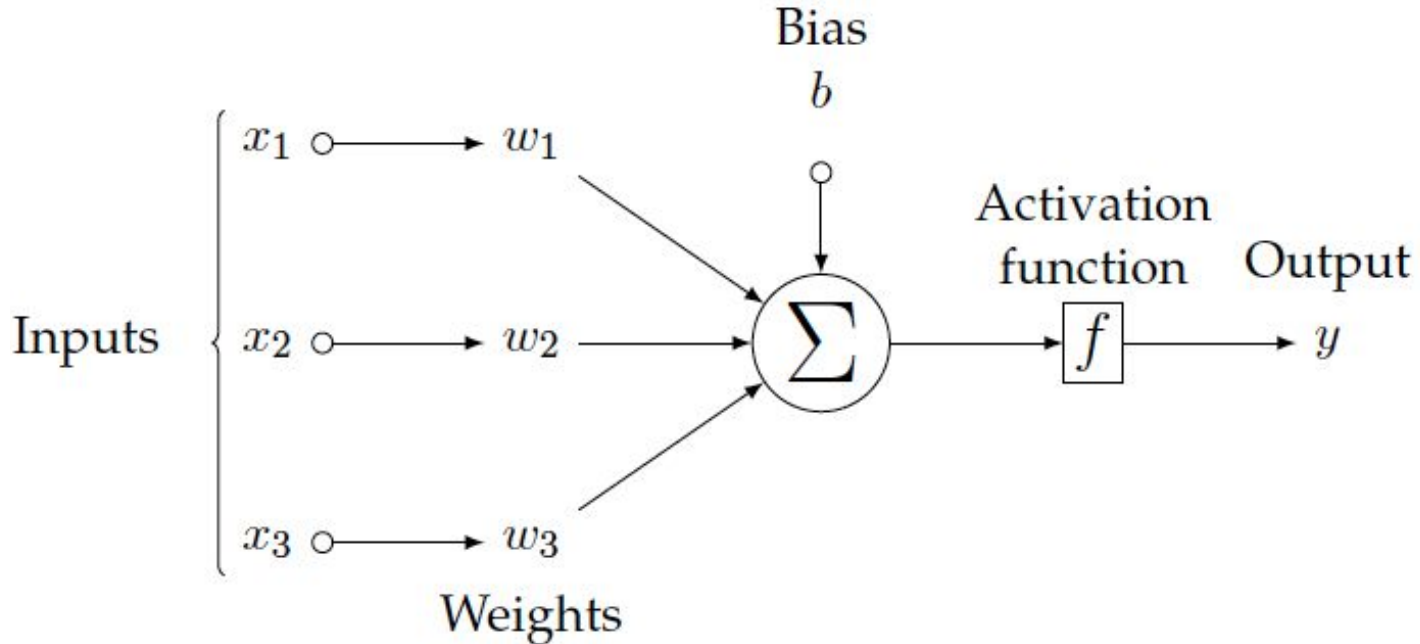
- **Classification:**  $y$  is **discrete** (e.g.  $y = \{1, 2, 5, 2, 2\}$ ).
  - Beware! These are unordered categories, not numerically meaningful outputs: e.g. `code[1] = "dog"`, `code[2] = "cat"`, `code[5] = "ostrich"`, ...
  - Classes are often coded as **one-hot vector** (each class corresponds to a different dimension of the output space)

One-hot  
representation



# Single Neuron Model (Perceptron)

Both regression and classification problems can be addressed with the perceptron:

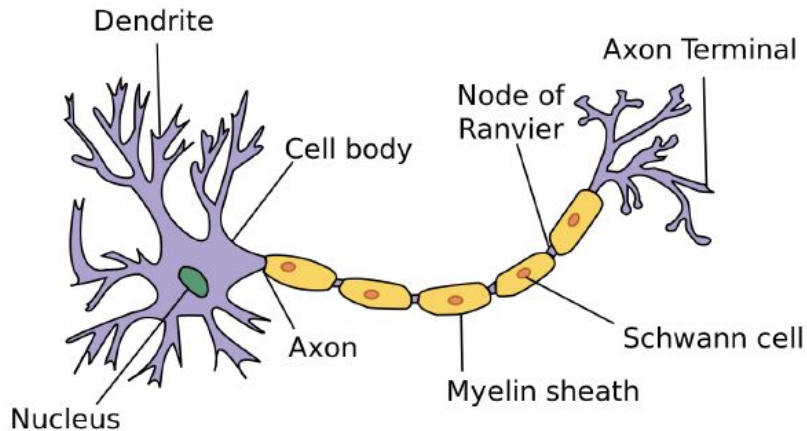


# Single neuron model (perceptron)

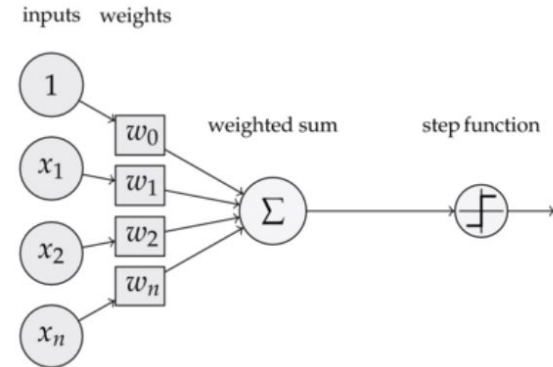
The Perceptron is seen as an **analogy** to a biological neuron.

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...).

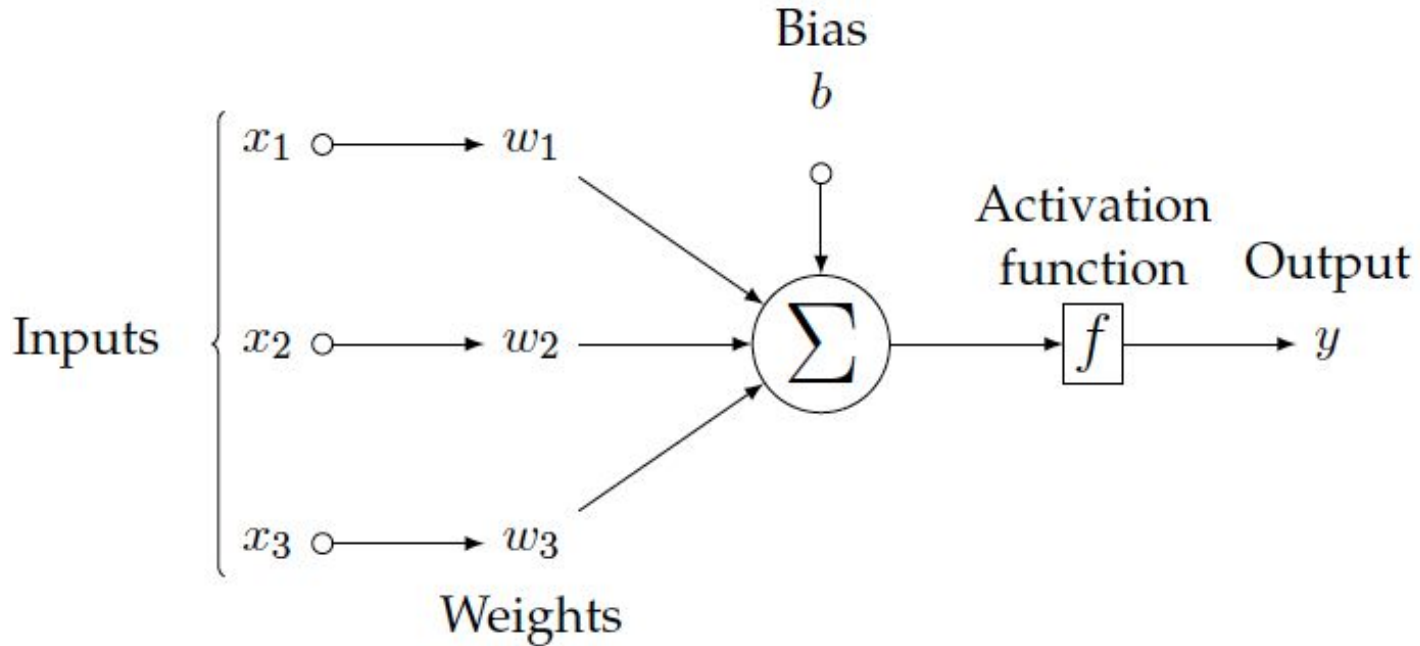


## Rosenblatt's Perceptron (1958)



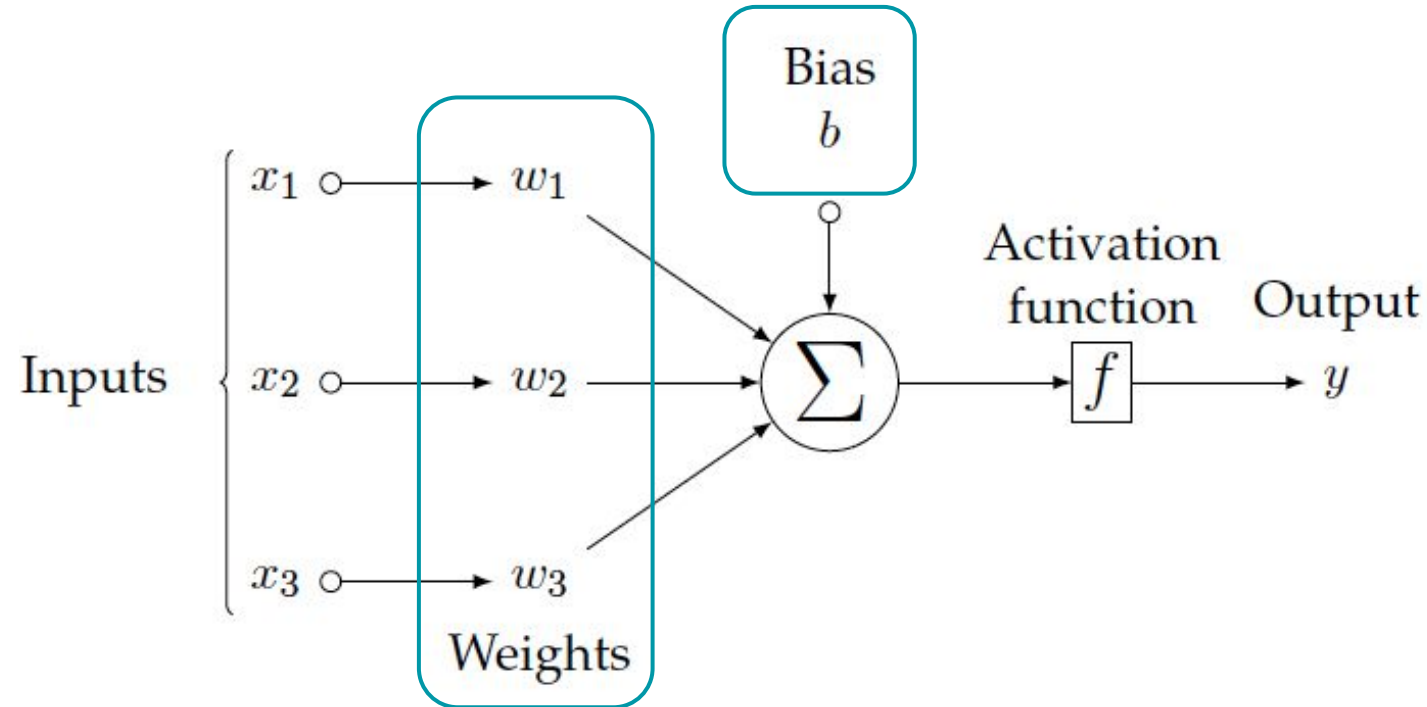


# Single neuron model (perceptron)



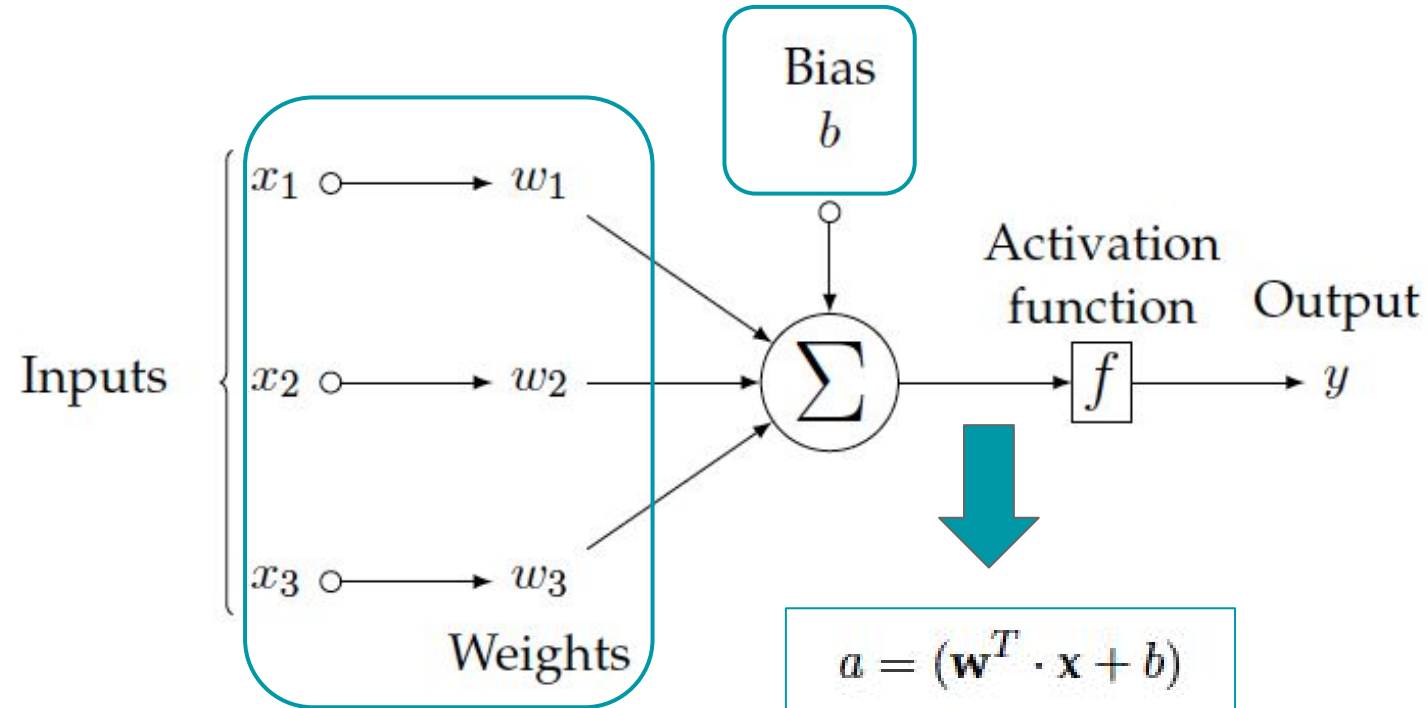
# Single neuron model (perceptron)

**Weights and bias** are the parameters that define the behavior (must be learned).



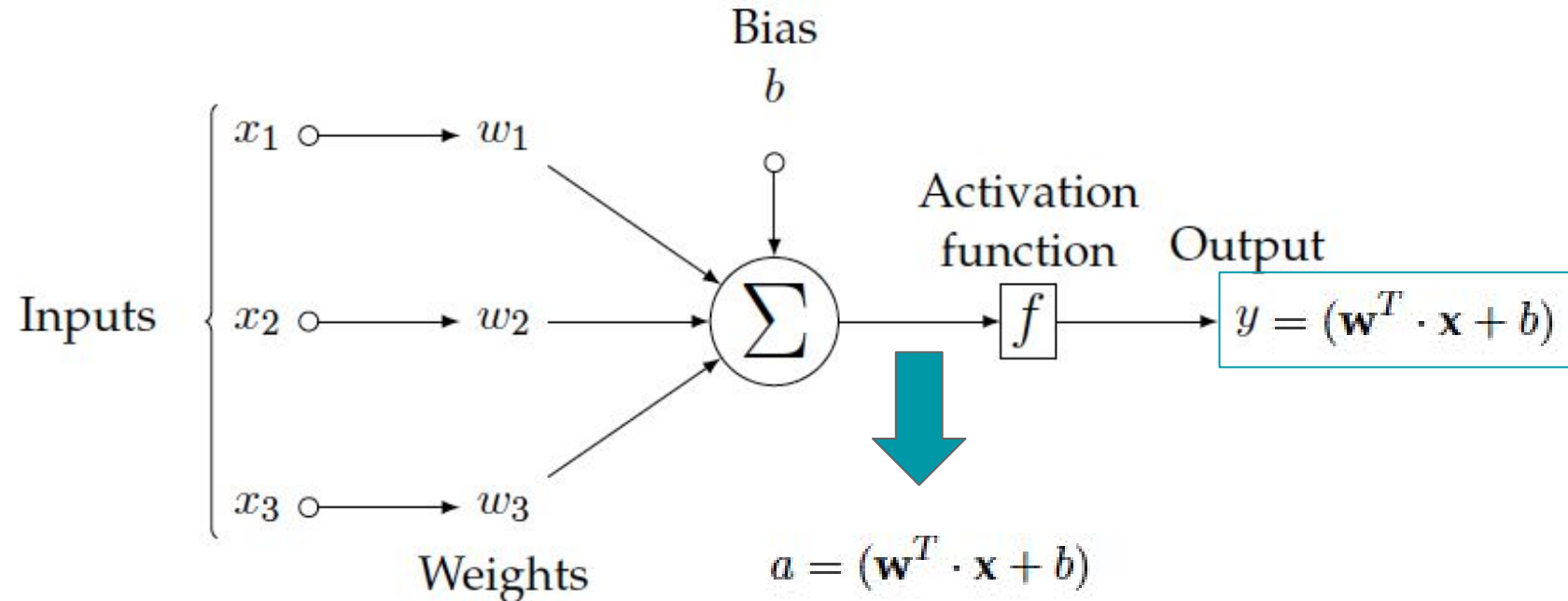
# Single neuron model (perceptron)

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



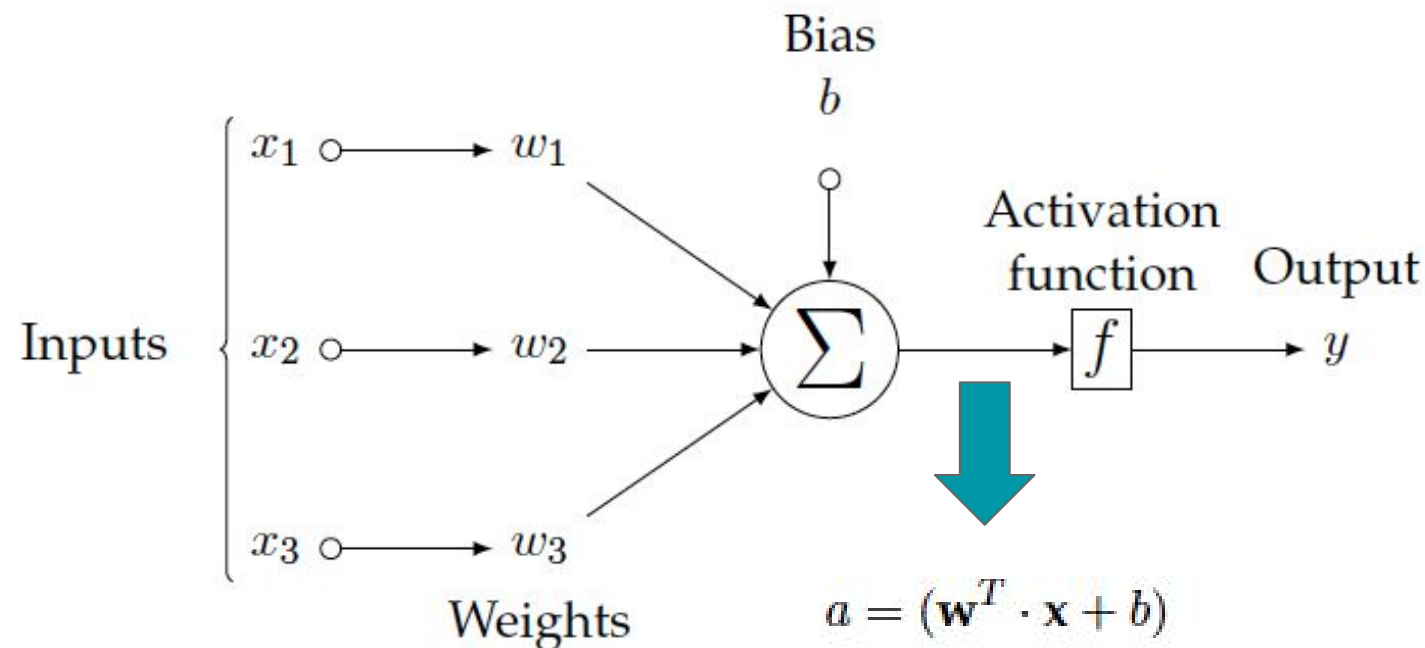
# Single neuron model: Regression

The perceptron can solve regression problems when  $f(a)=a$ . [identity]



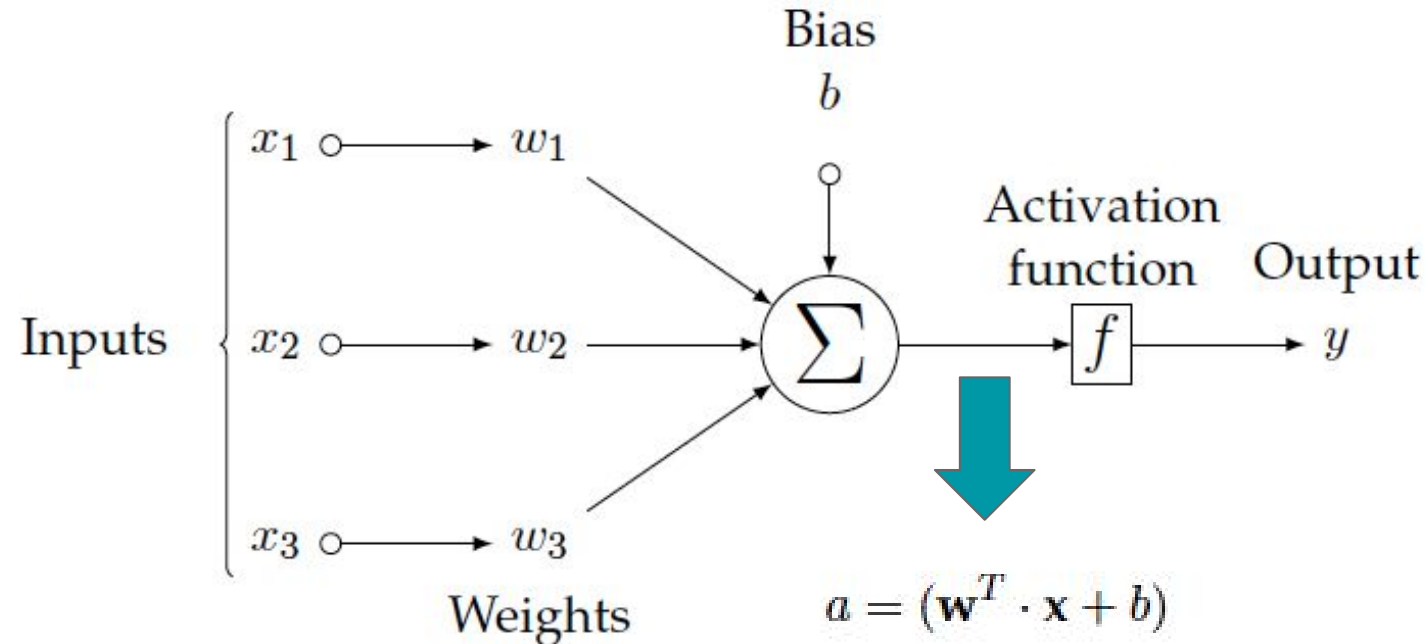
# Single neuron model: Binary Classification

The perceptron can solve classification problems when  $f(a)=\sigma(a)$ . [sigmoid]



# Single neuron model: Binary Classification

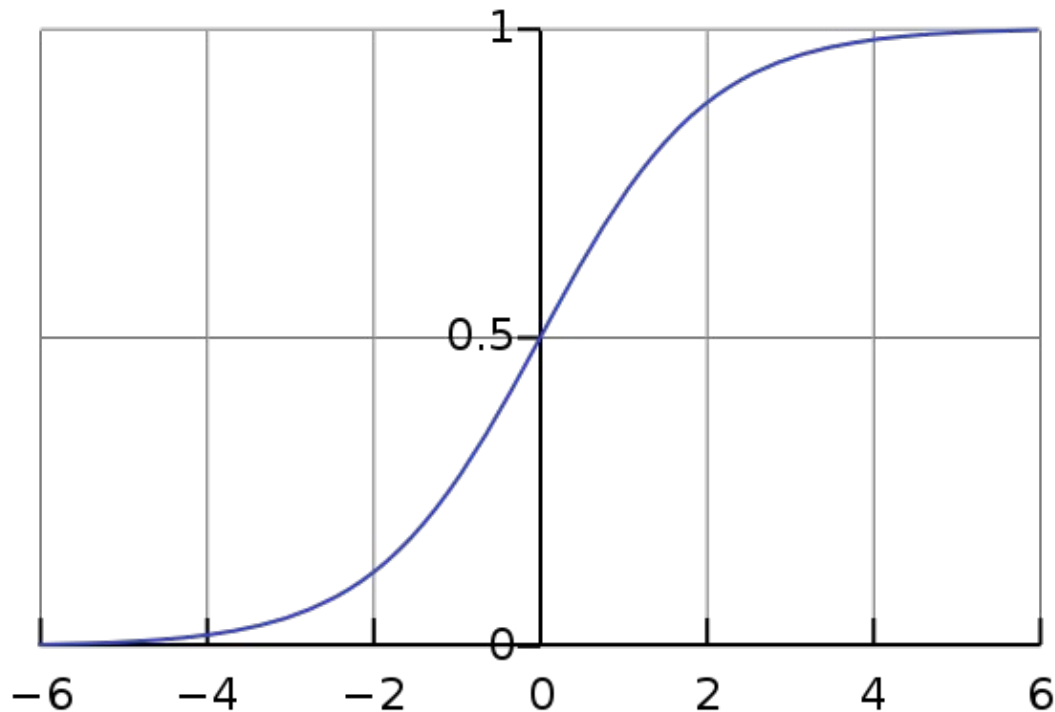
The perceptron can solve classification problems when  $f(a)=\sigma(a)$ . [sigmoid]



# Single neuron model: Binary Classification

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

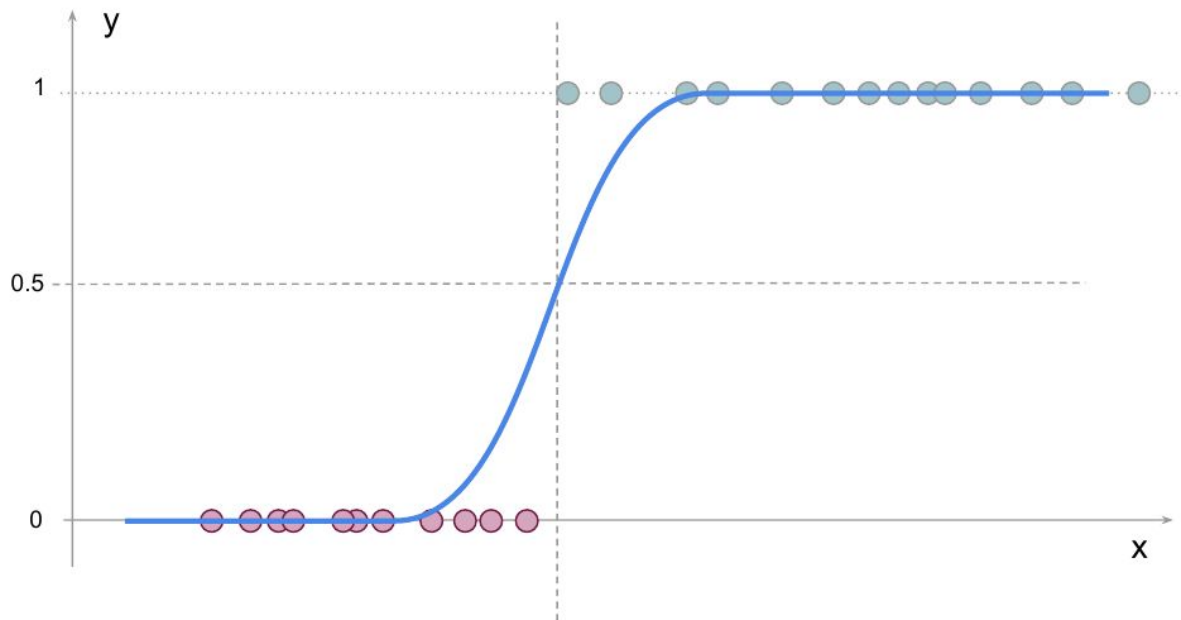
$$f(x) = \frac{1}{1 + e^{-x}}$$





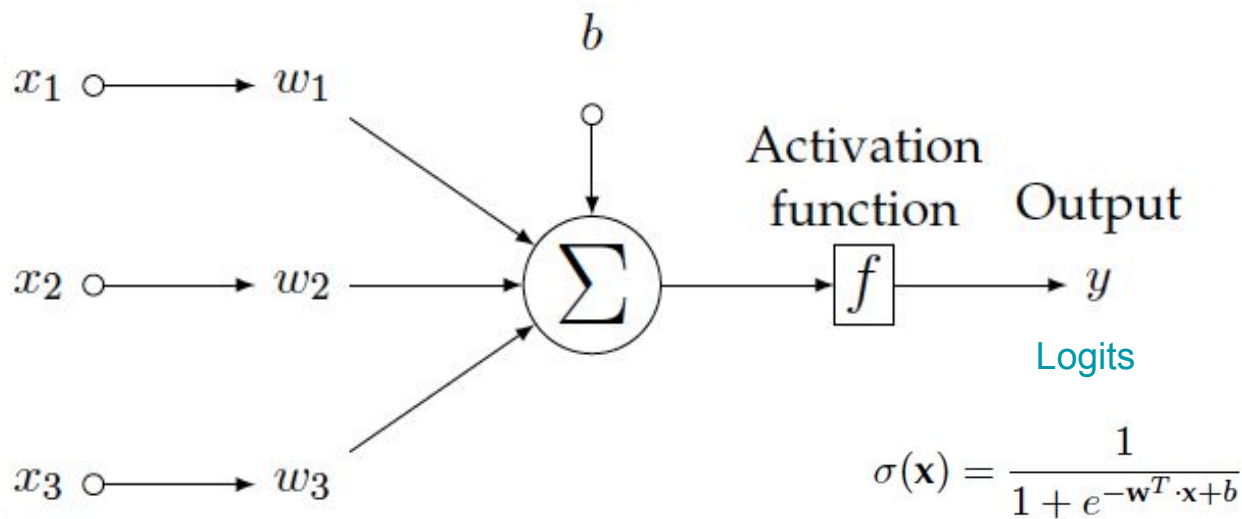
# Single neuron model: Binary Classification

For classification, regressed values must be bounded between 0 and 1 to represent probabilities.



# Single neuron model: Binary Classification

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

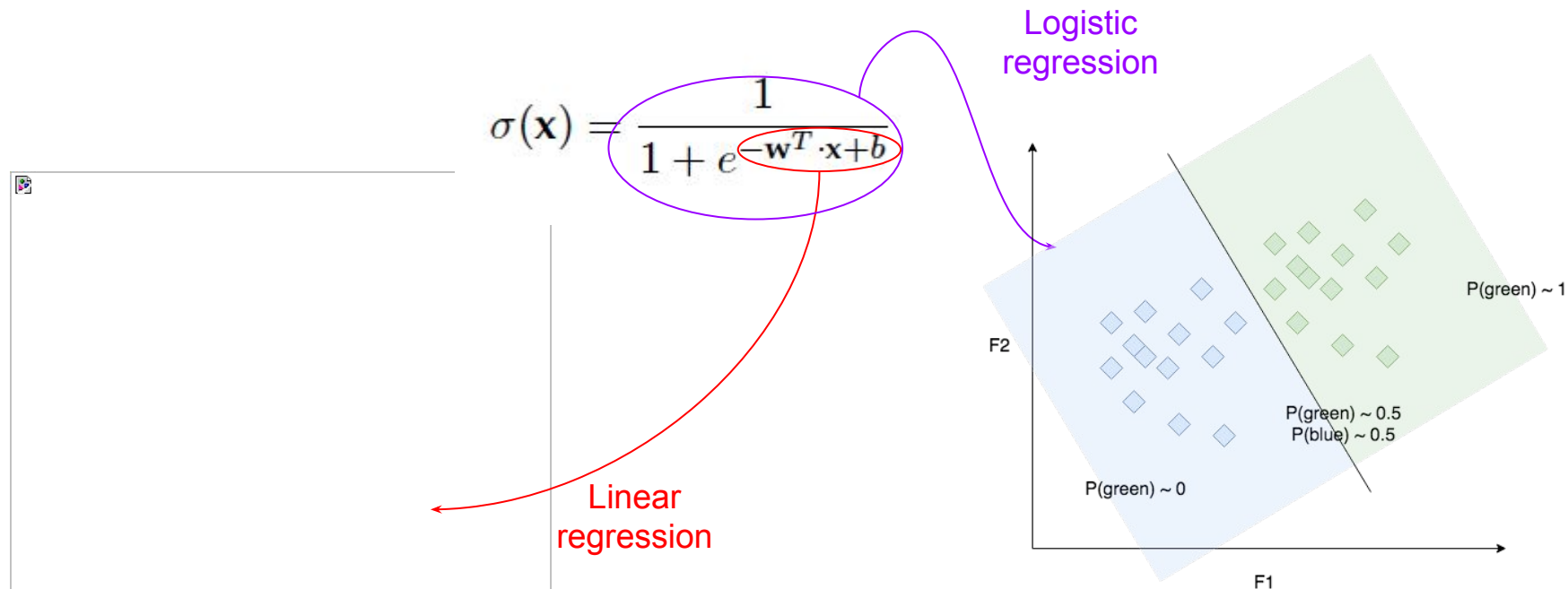


$y > \text{thr} \rightarrow \text{class 1}$   
(eg. green)

$y < \text{thr} \rightarrow \text{class 2}$   
(eg. no green)

# Single neuron model: Binary Classification

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



# Softmax classifier: Multiclass

Ideally we would like to predict the *probability* that  $y$  takes a particular value given  $\mathbf{x}$ ,

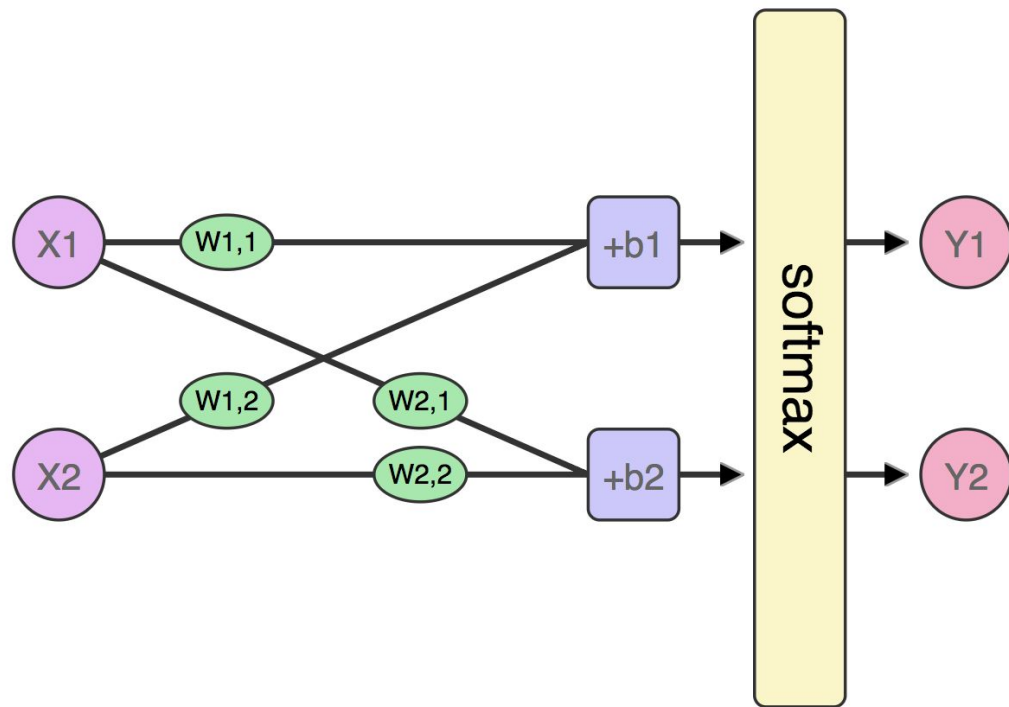
$$P(y = j|\mathbf{x}), \quad j \in \{1, \dots, K\}$$

with:

$$\sum_{j=1}^K P(y = j|x) = 1$$

The logistic regression classifier does this for the binary case. The **softmax classifier** extends it to the multiclass case.

# Softmax classifier: Multiclass

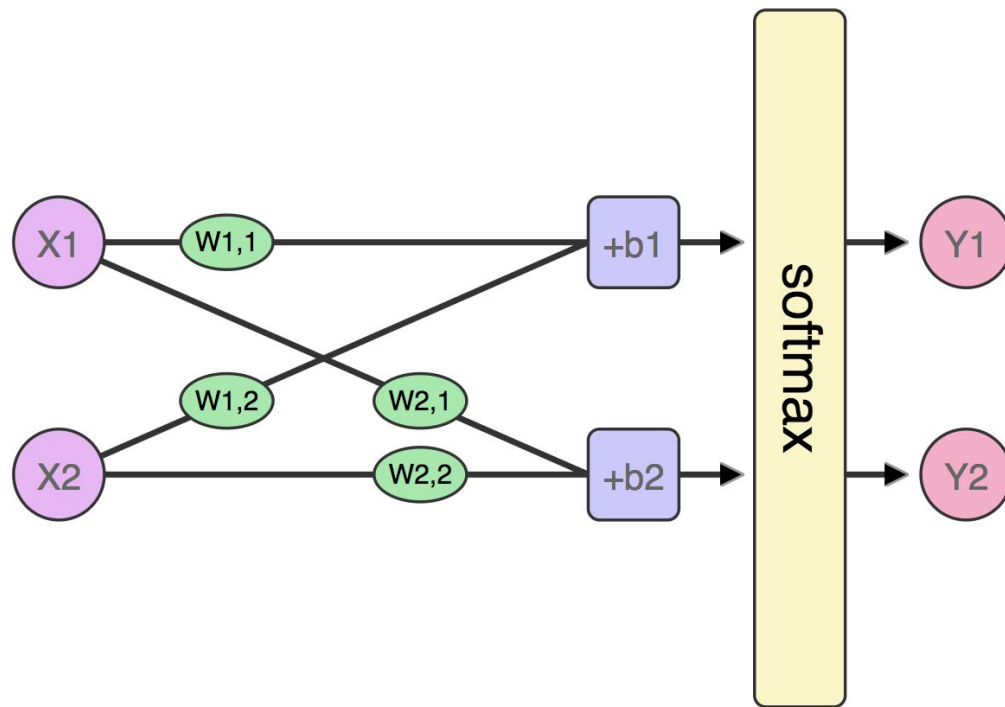


Probability estimations for each class can also be obtained by **softmax normalization** on the output of two neurons, one specialised for each class.

Softmax  
regression

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

# Softmax classifier: Multiclass

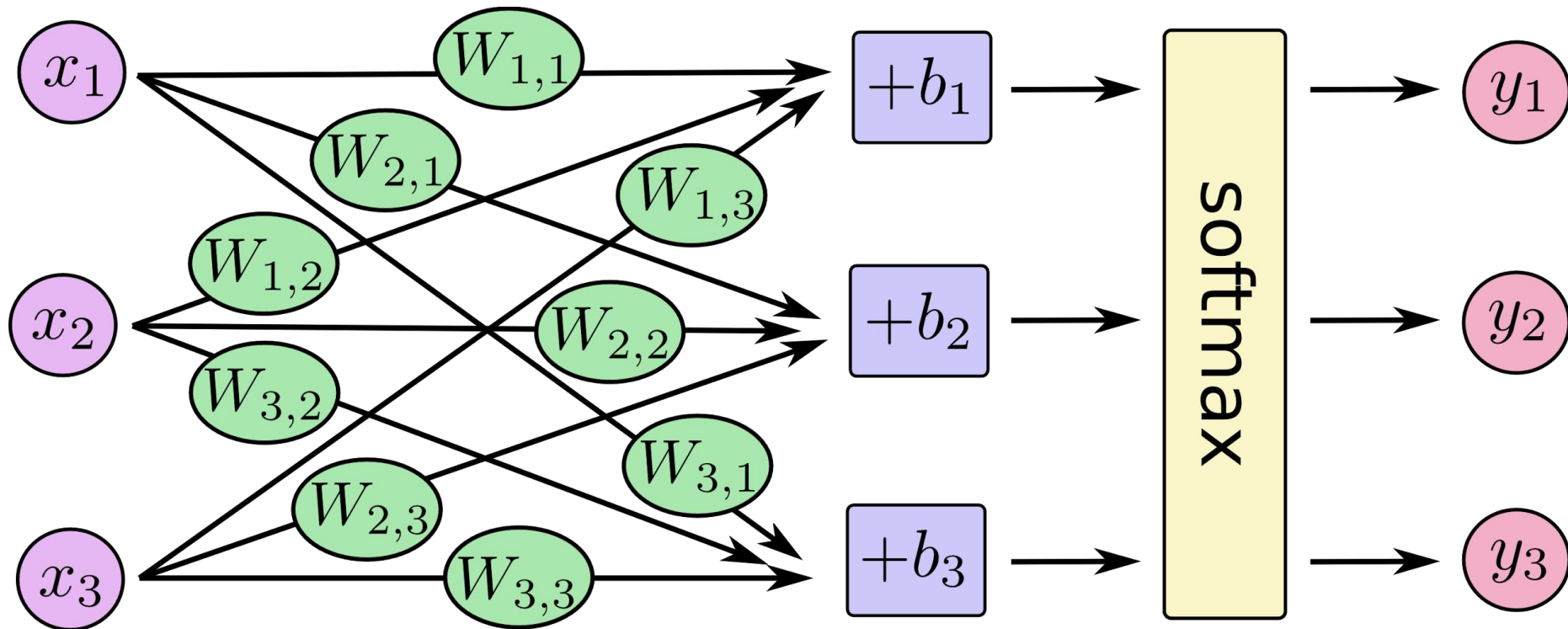


Softmax  
regression

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

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

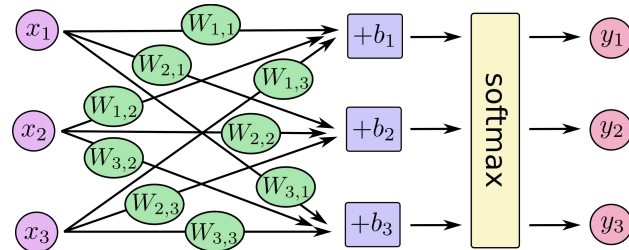
# Softmax classifier: Multiclass (3 classes)





# Softmax classifier: Multiclass (3 classes)

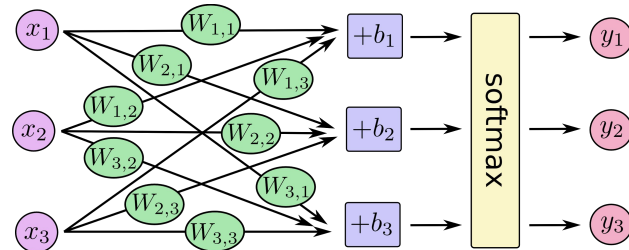
$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \text{softmax} \left( \begin{bmatrix} W_{1,1}x_1 + W_{1,2}x_2 + W_{1,3}x_3 + b_1 \\ W_{2,1}x_1 + W_{2,2}x_2 + W_{2,3}x_3 + b_2 \\ W_{3,1}x_1 + W_{3,2}x_2 + W_{3,3}x_3 + b_3 \end{bmatrix} \right)$$



# Softmax classifier: Multiclass (3 classes)

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

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



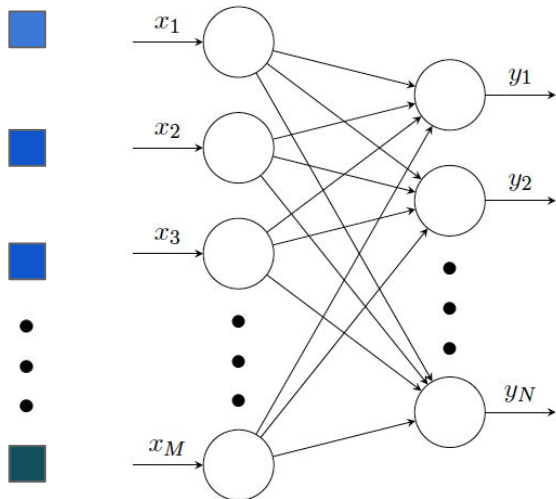
# Softmax classifier: Multiclass (3 classes)

Multiple classes can be predicted by putting many neurons in parallel, each processing its binary output out of N possible classes.

raw pixels  
unrolled img

Input  
layer

Ouput  
layer



0.3 “dog”



0.08 “cat”



•

•

•

0.6 “whatever”

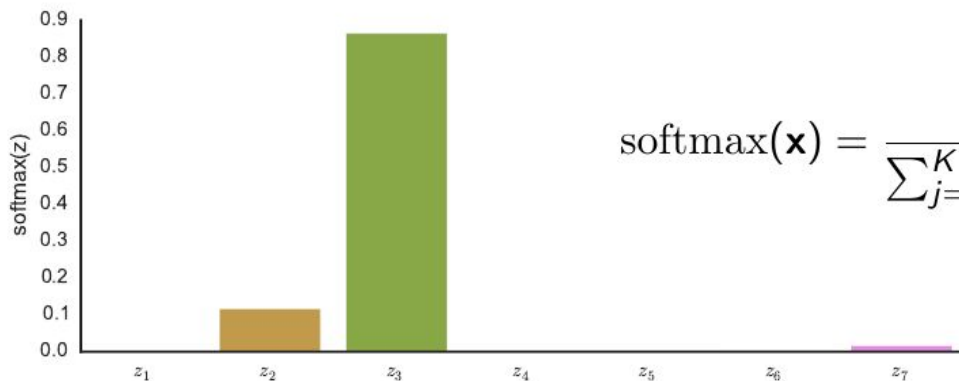
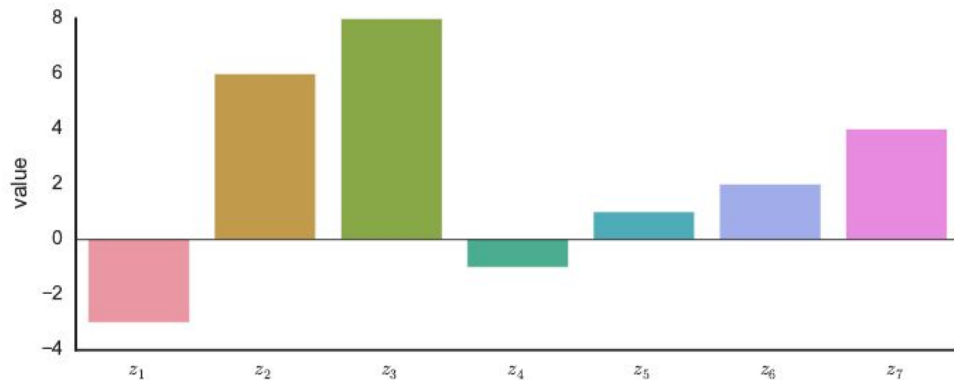


## Softmax function

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

Normalization factor,  
remember: we want a pdf at  
the output! → all output P's  
sum up to 1.

# Effect of the softmax



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

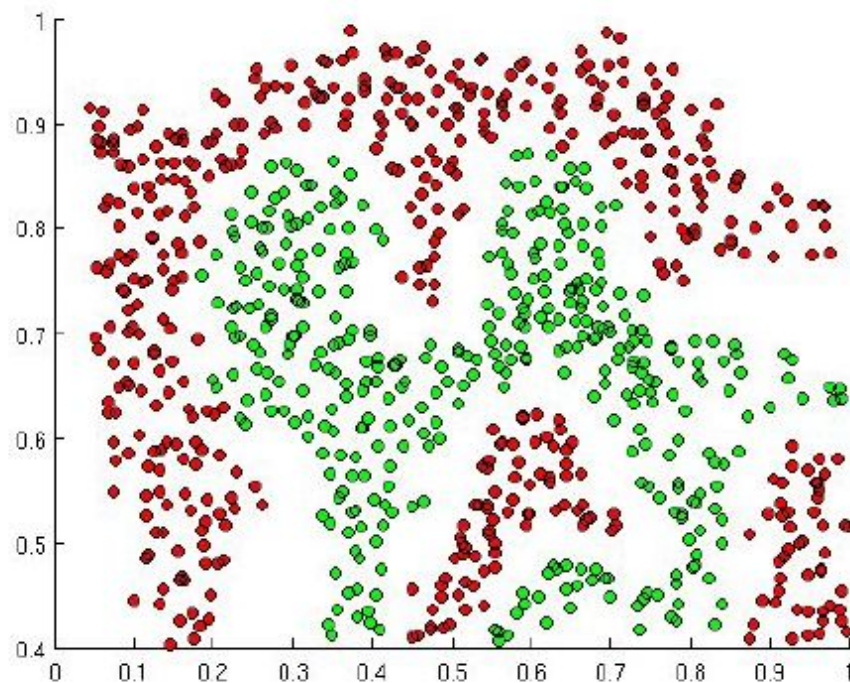
# Next lecture...

Perceptrons can only produce linear decision boundaries.

**Many interesting problems are not linearly separable.**

Real world problems often need non-linear boundaries

- Images
- Audio
- Text



# Questions?