Insight
Centre for Data Analytics

DEEP
LEARNING
WORKSHOP

Dublin City University
21-22 May 2018

DCU

#InsightDL2018

Day 1 Lecture 1

The Perceptron

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

Associate Professor
Intelligent Data Science and Artificial Intelligence Center
Universitat Politecnica de Catalunya (UPC)

Acknowledgements

Santiago Pascual

Kevin McGuinness
kevin.mcguinness@dcu.ie

Research Fellow
Insight Centre for Data Analytics
Dublin City University

Video lecture (DLSL 2017)

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:

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

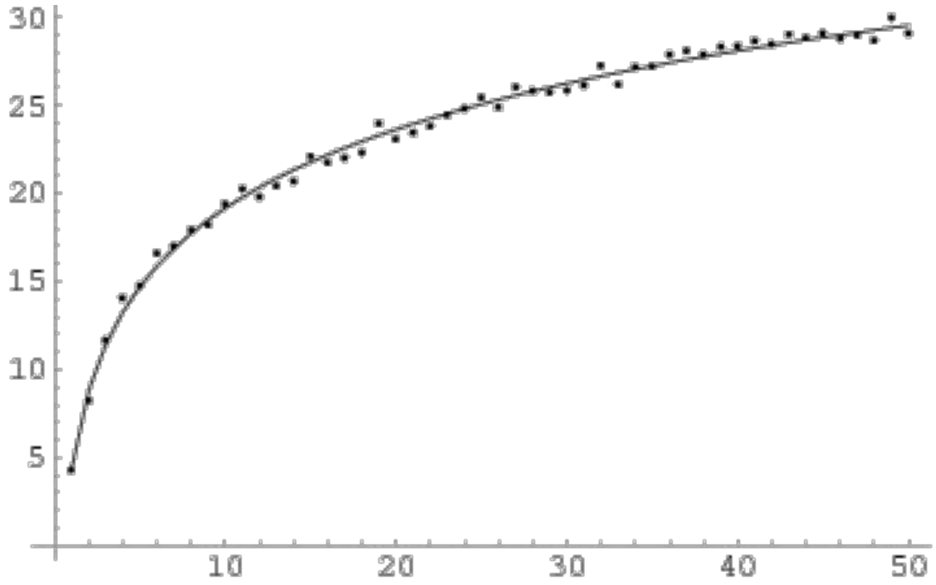
3. Reinforcement Learning:

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

$$\mathbf{z}$$

Supervised learning

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



Supervised learning

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

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

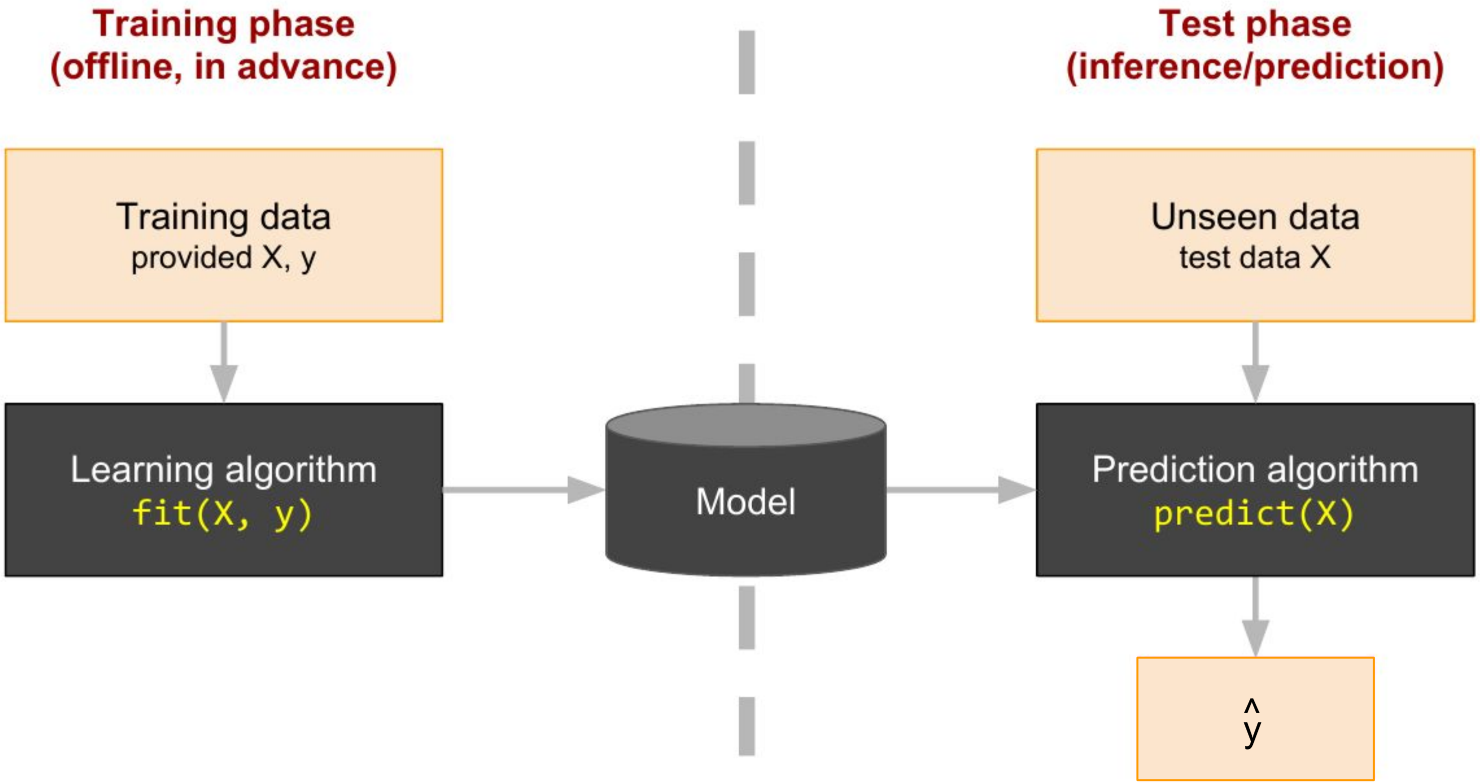
Supervised learning

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

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

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

Black box abstraction of supervised learning



Regression vs Classification

Depending on the type of target $\mathbf{y}$ we get:

- **Regression:** $\mathbf{y} \in \mathbb{R}^N$ is continuous (e.g. temperatures $\mathbf{y}$ = {19º, 23º, 22º})

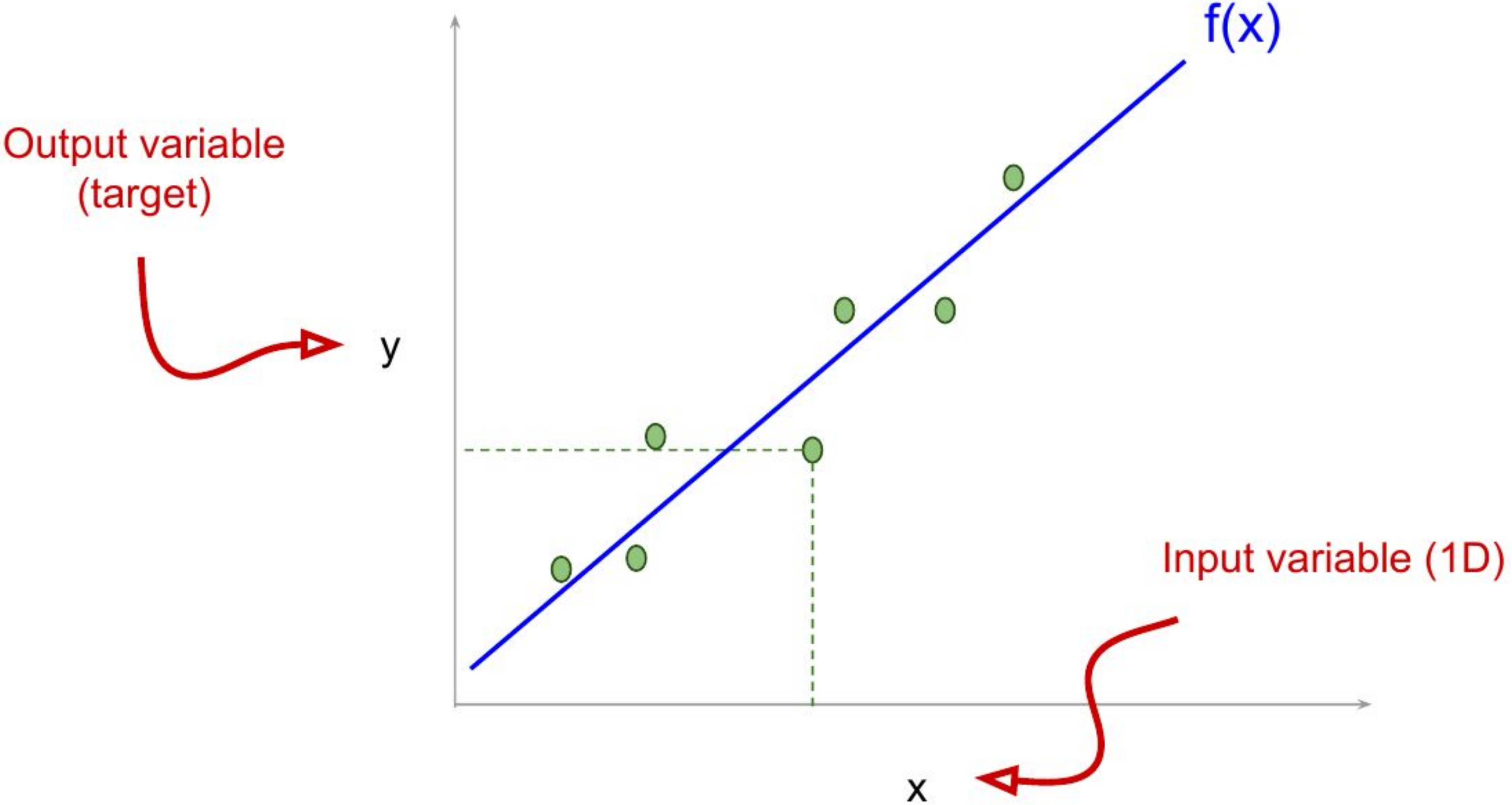
- **Classification:** $\mathbf{y}$ is discrete (e.g. $\mathbf{y}$ = {1, 2, 5, 2, 2}).

Regression vs Classification

Depending on the type of target $\mathbf{y}$ we get:

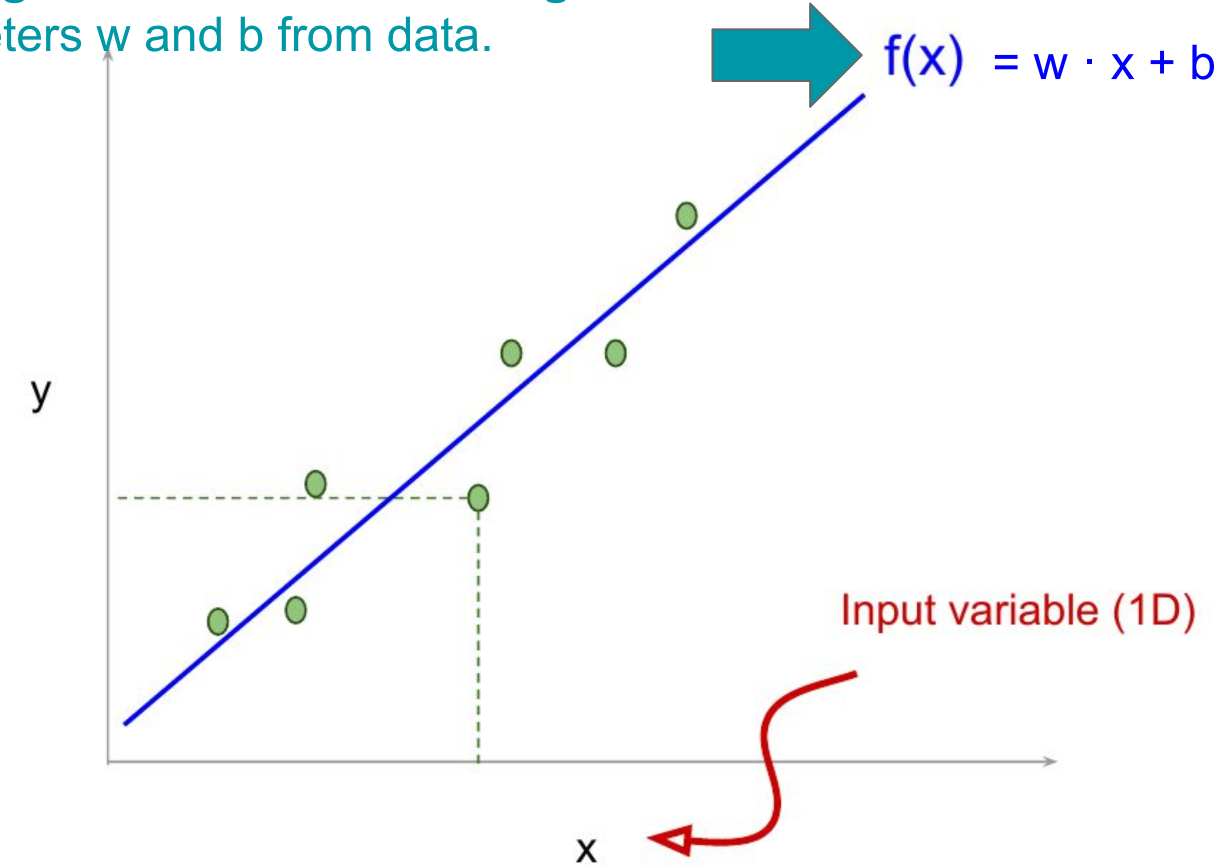
- **Regression:** $\mathbf{y} \in \mathbb{R}^N$ is continuous (e.g. temperatures $\mathbf{y} = \{19^\circ, 23^\circ, 22^\circ\}$)
- **Classification:** $\mathbf{y}$ is discrete (e.g. $\mathbf{y} = \{1, 2, 5, 2, 2\}$).

Linear Regression (eg. 1D input - 1D ouput)



Linear Regression (eg. 1D input - 1D ouput)

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 **x**:

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

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)

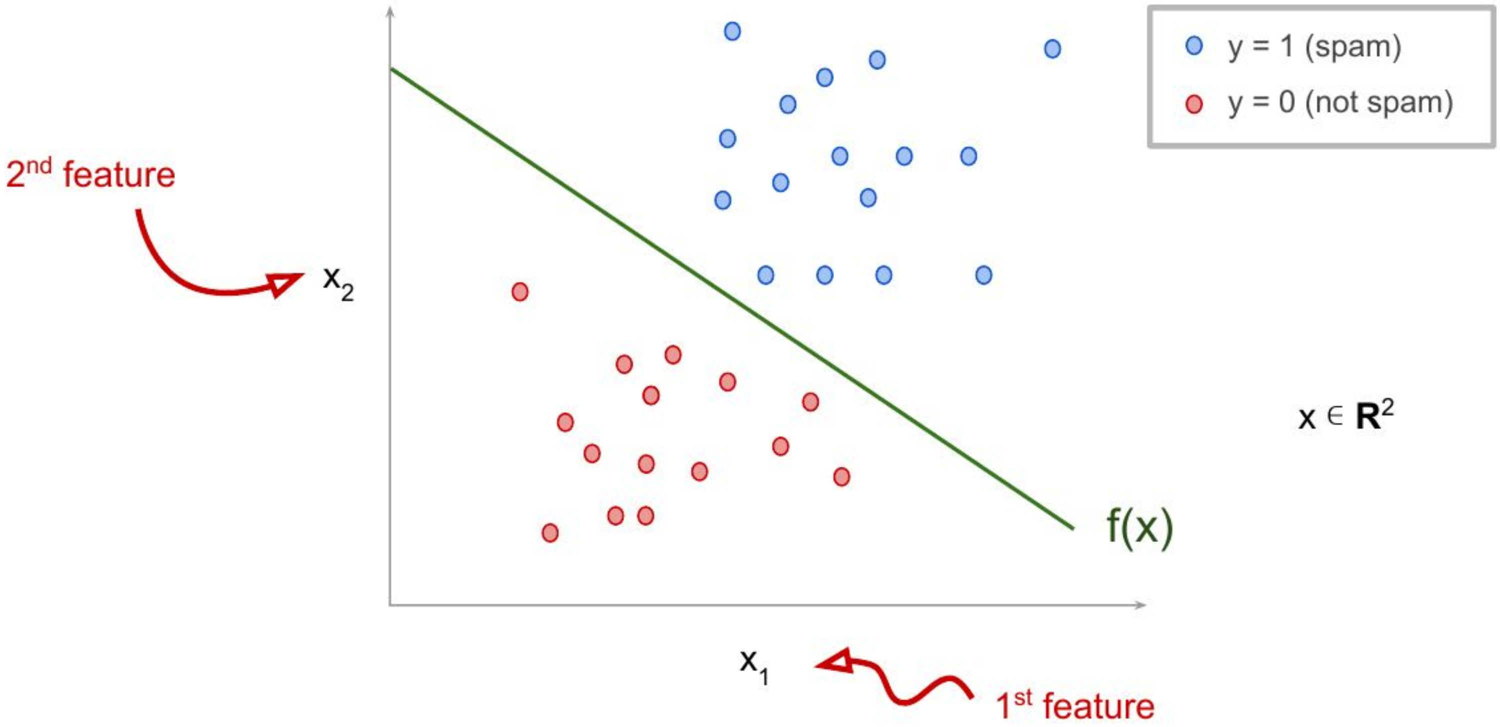
y = price = *w1*·(sqm) + *w2*·(lat) + *w3*·(lon) + *w4*·(yoc) + b

Regression vs Classification

Depending on the type of target $\mathbf{y}$ we get:

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

Binary Classification (eg. 2D input, 1D ouput)

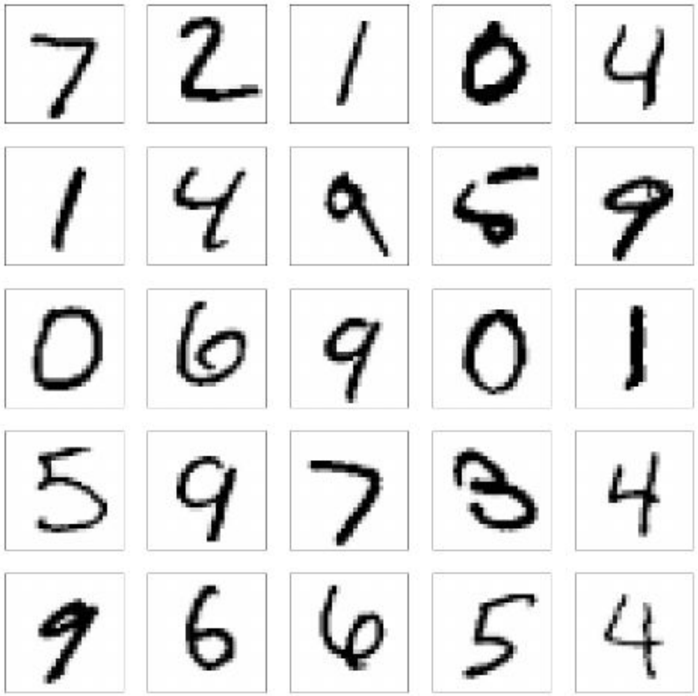


Multi-class Classification

Produce a classifier to map from pixels to the digit.

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

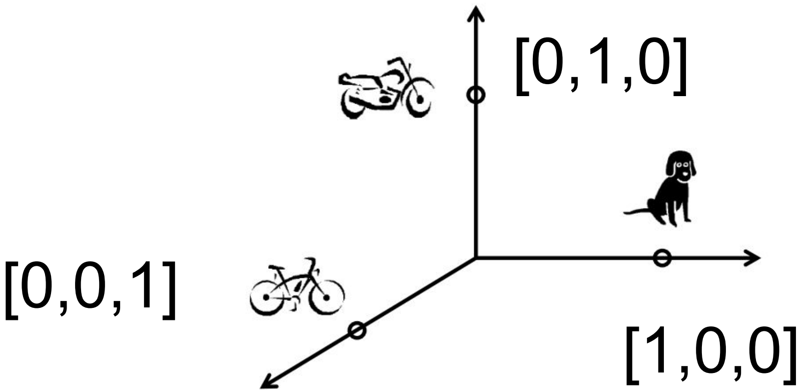
Example of a **multi-class classification** task.



Multi-class Classification

- **<u>Classification</u>:** $\mathbf{y}$ is discrete (e.g. $\mathbf{y} = \{1, 2, 5, 2, 2\}$).
  - 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)

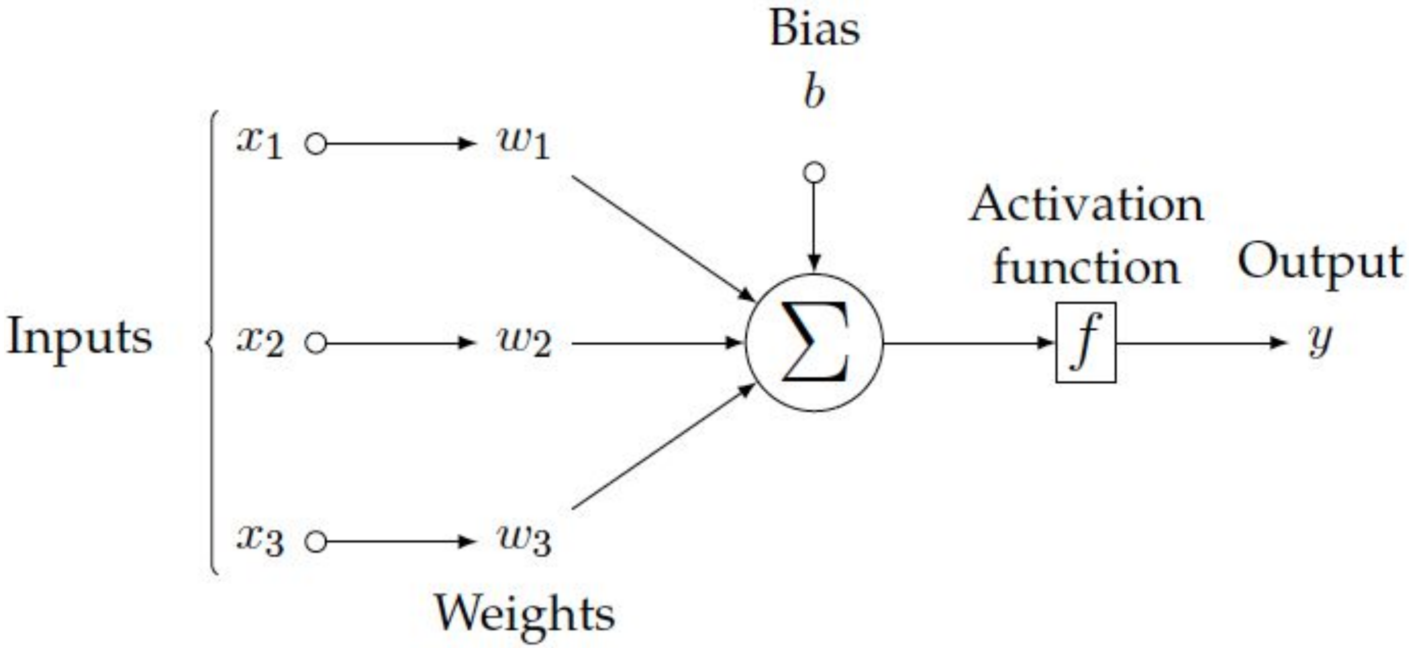
One-hot representation



Perronin, F., CVPR Tutorial on LSVR @ CVPR’14, Output embedding for LSVR

Single Neuron Model (Perceptron)

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

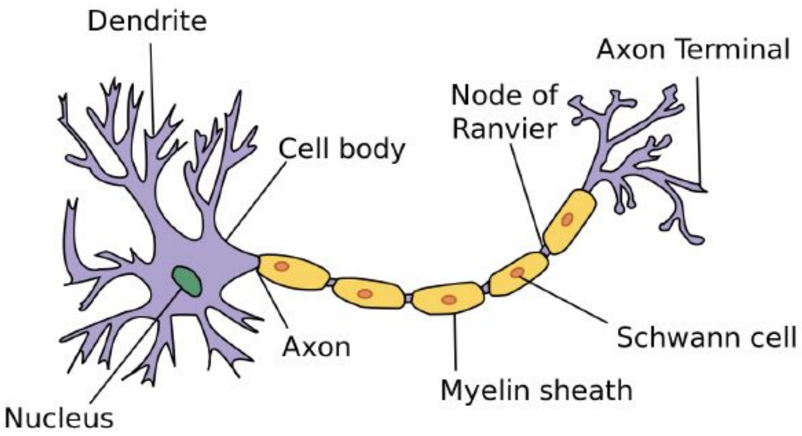


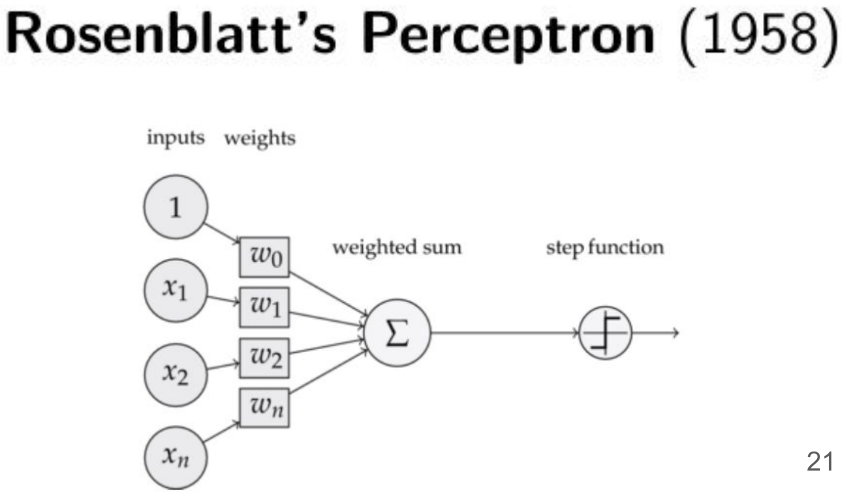
Single neuron model (perceptron)

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

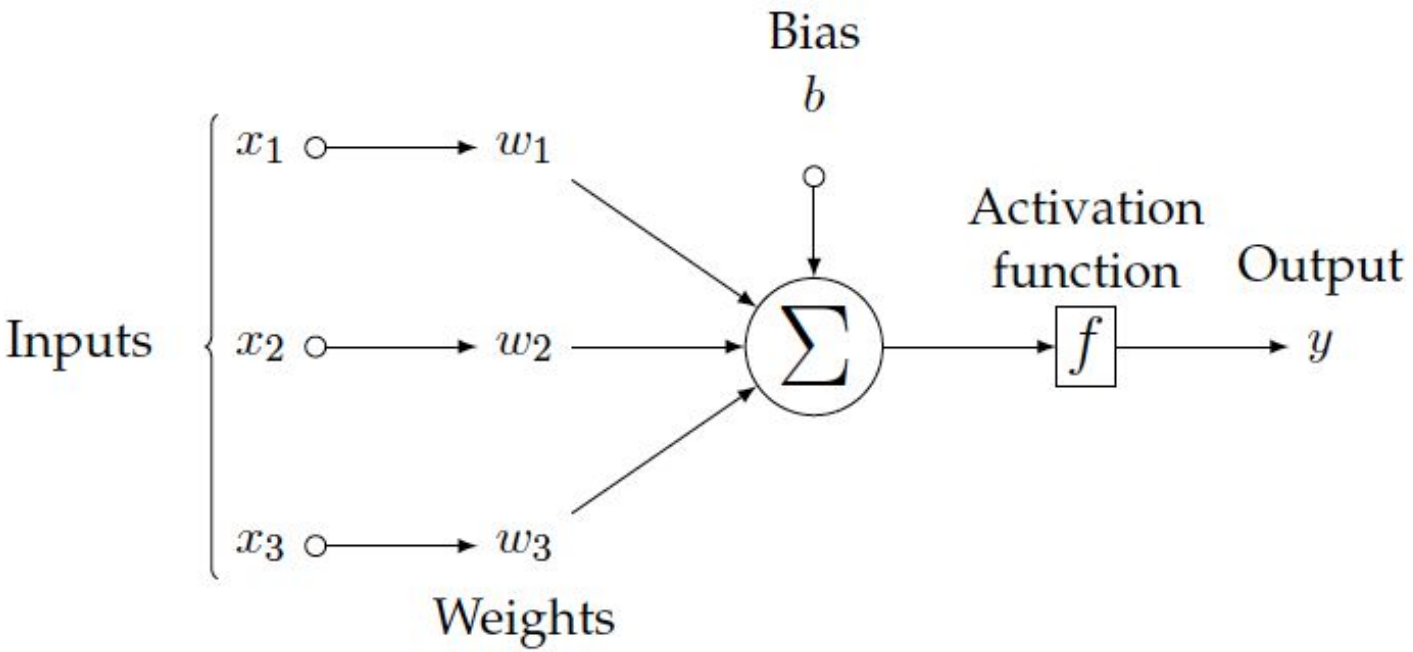
Biological neurons fire an impulse once the sum of all inputs is over a threshold.

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



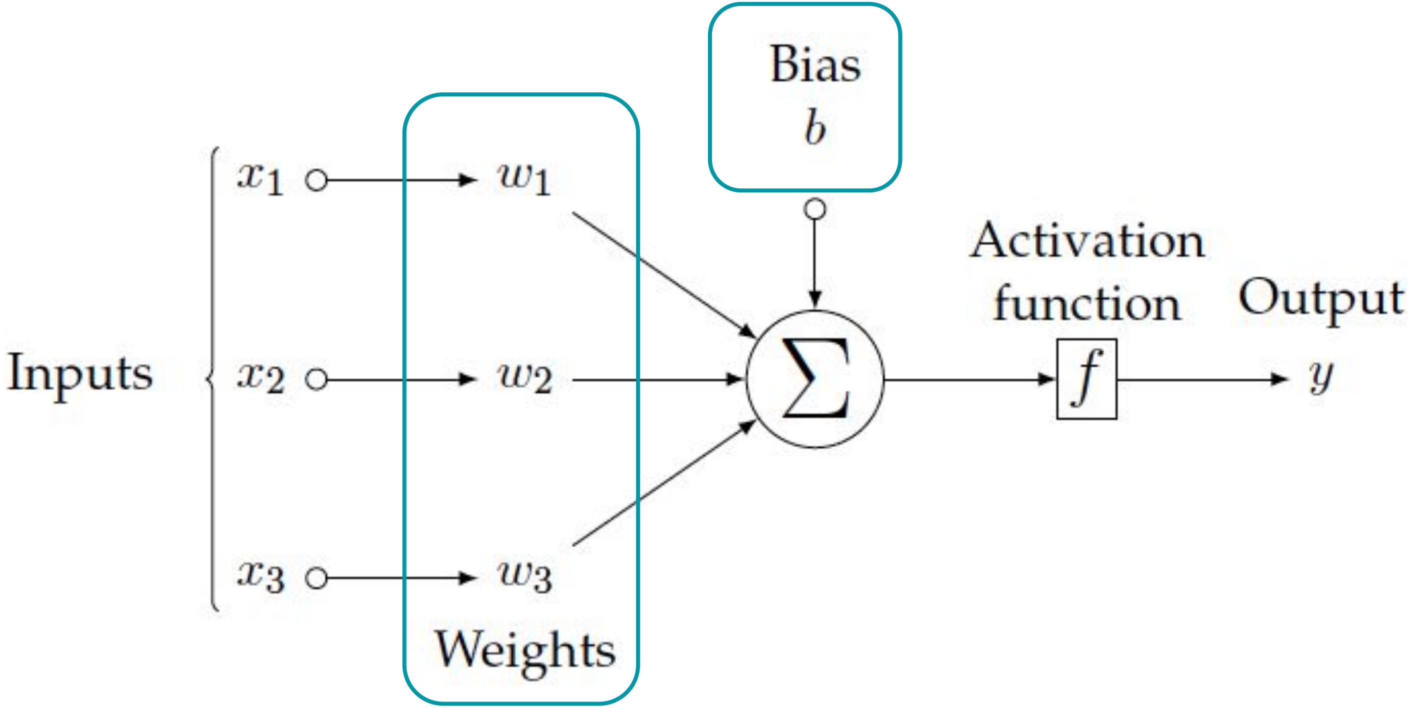


Single neuron model (perceptron)



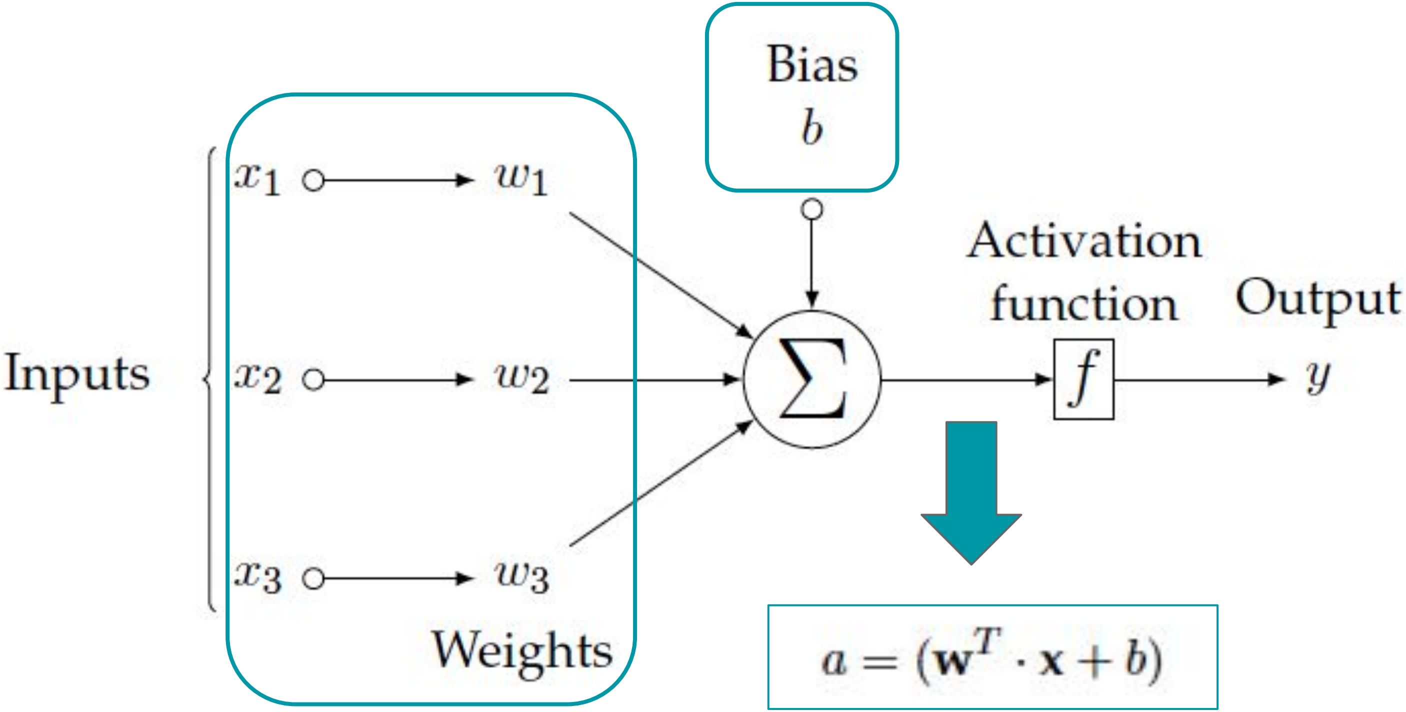
Single neuron model (perceptron)

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



Single neuron model (perceptron)

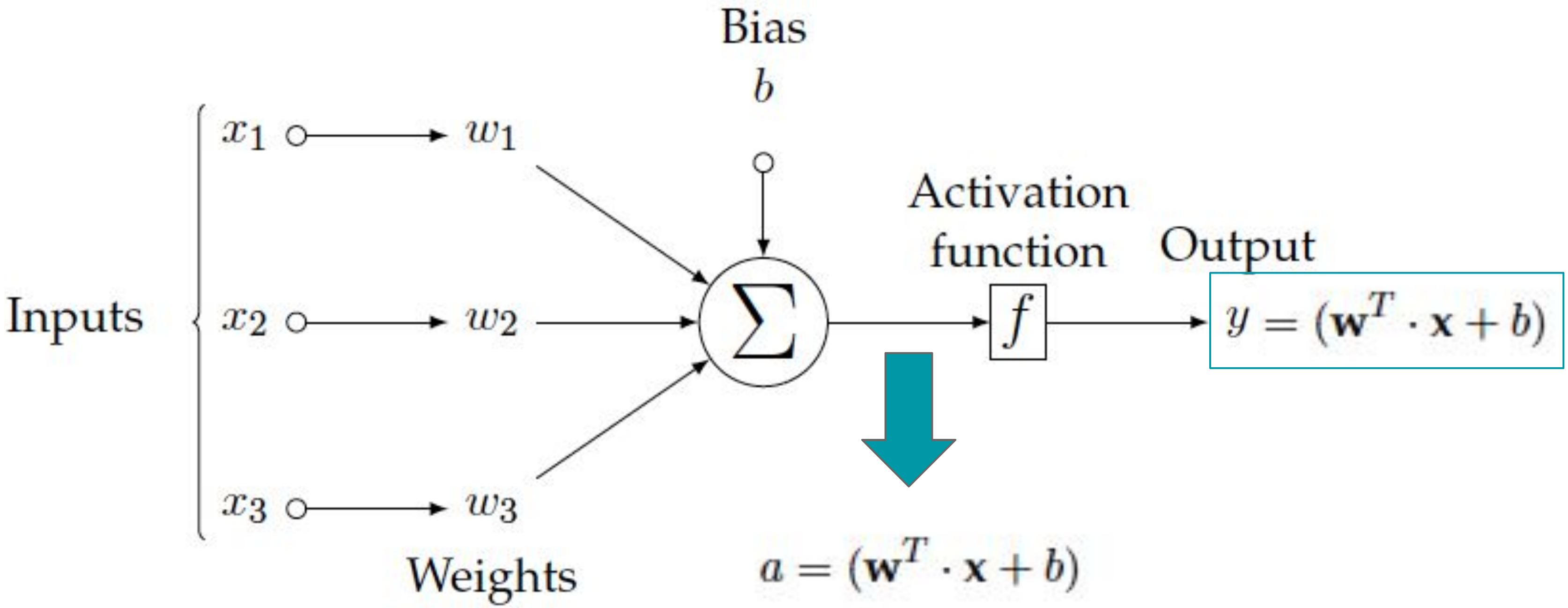
The output y is derived from a sum of the **weighted** inputs plus a **bias** term.



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

Single neuron model: Regression

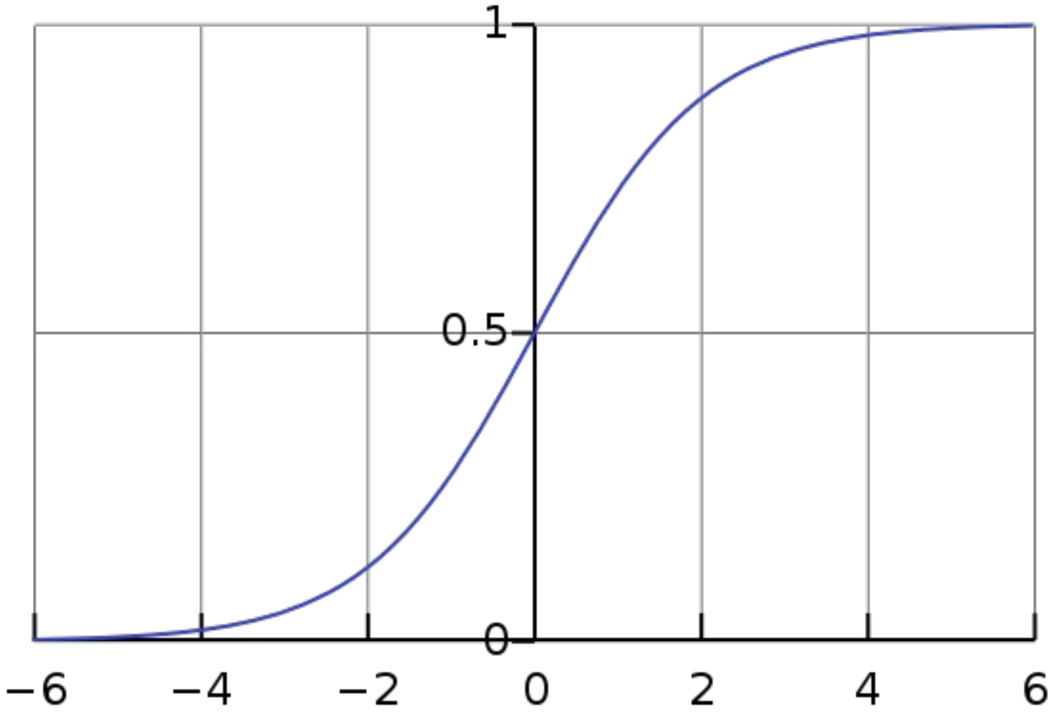
The perceptron can solve regression problems when f(a)=a. [identity]



Single neuron model: Binary Classification

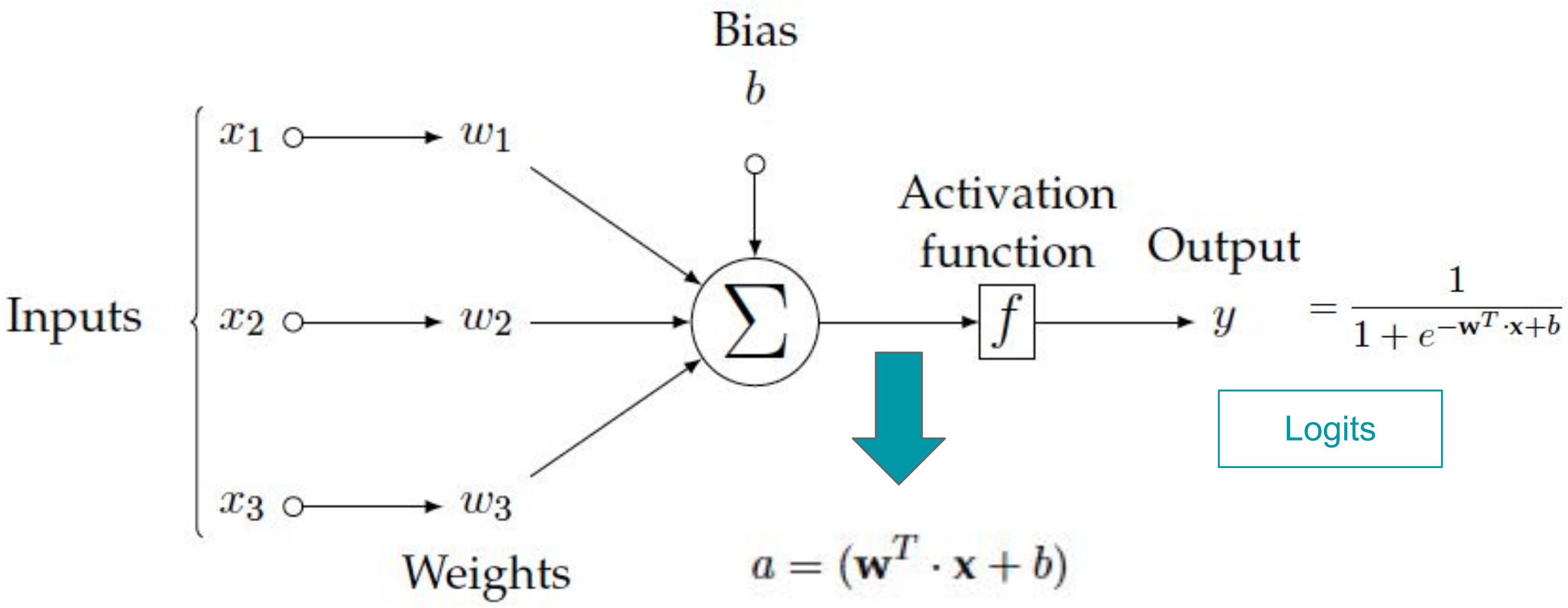
The **sigmoid function σ(x)** or **logistic curve** maps any input x between [0,1]:

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



Single neuron model: Binary Classification

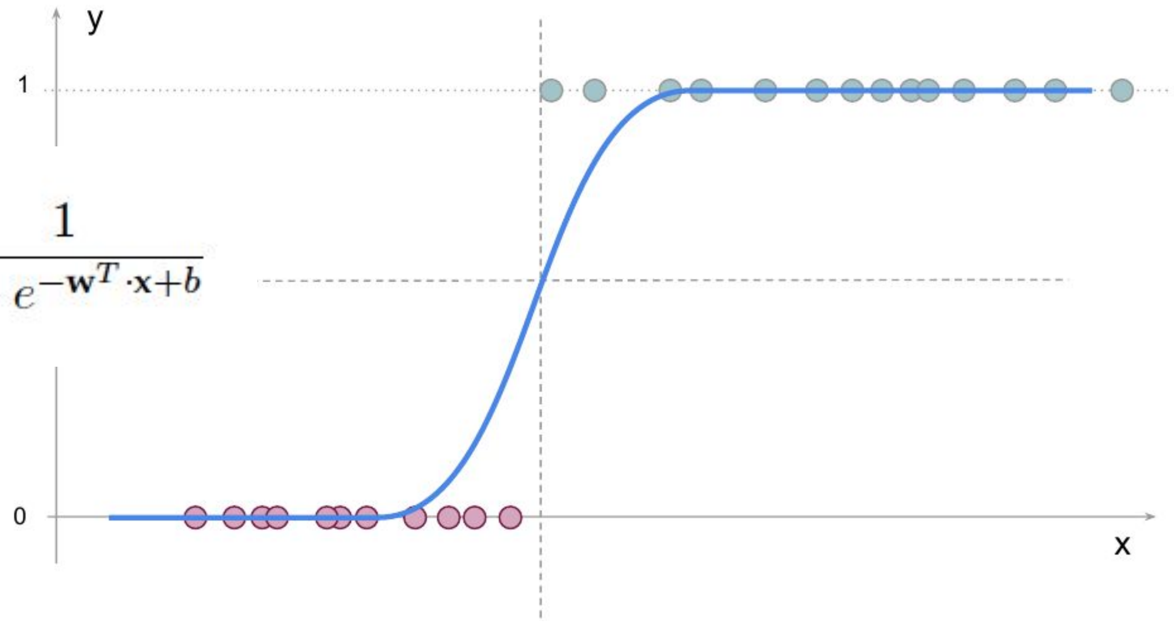
The perceptron can solve <u>classification</u> problems when f(a)=σ(a). [sigmoid]



Single neuron model: Binary Classification

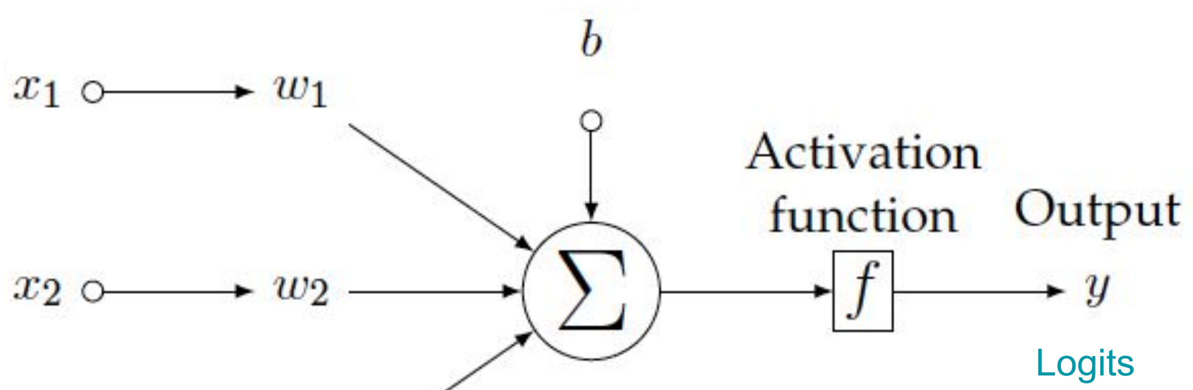
For classification, regressed values must be bounded between 0 and 1 to represent "probabilities".

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



Single neuron model: Binary Classification

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

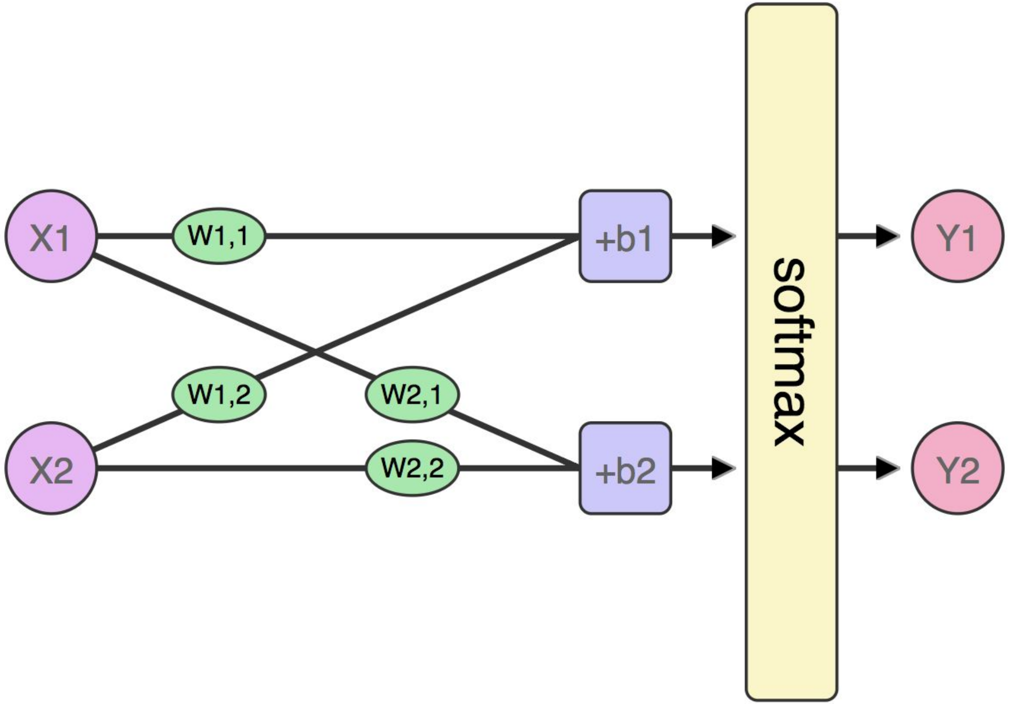


$$\sigma(\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T \cdot \mathbf{x} + b}}$$

y > thr → class 1 (eg. green)

y < thr → class 2 (eg. no green)

Softmax classifier: Binary classification



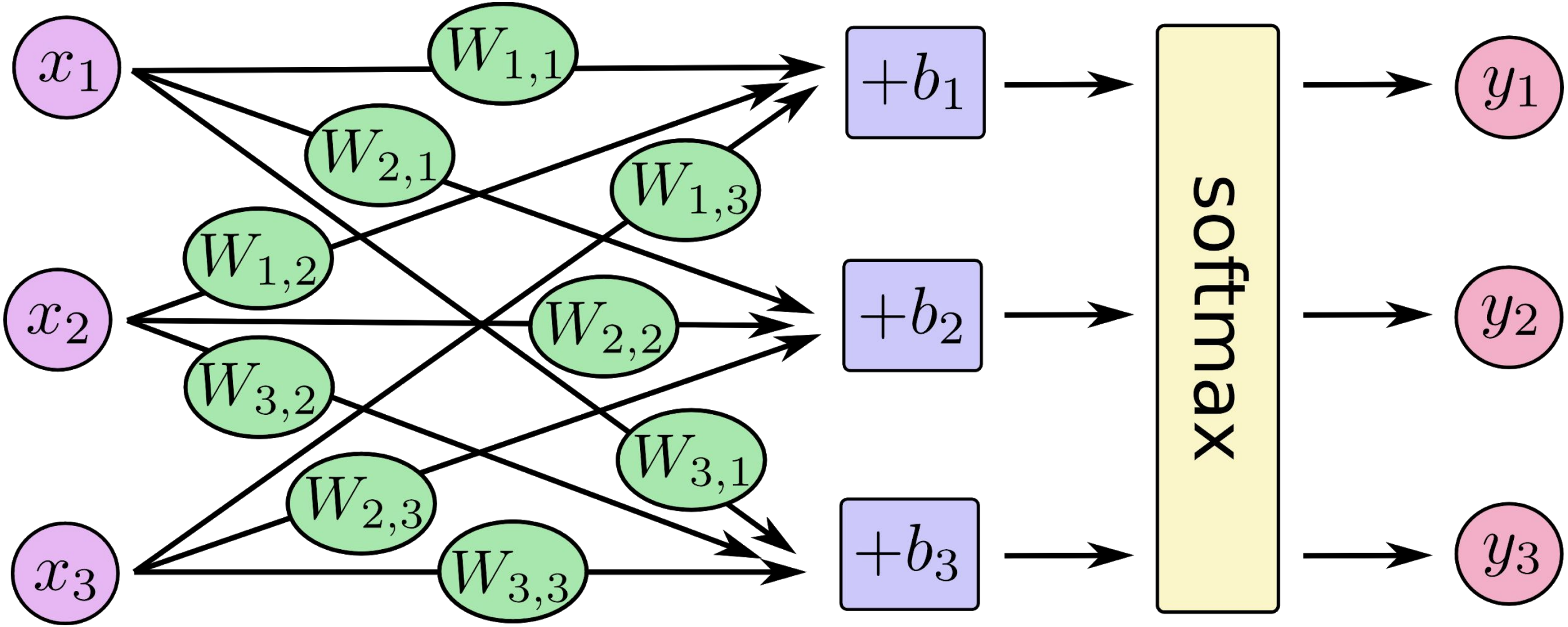
Softmax regression

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

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

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

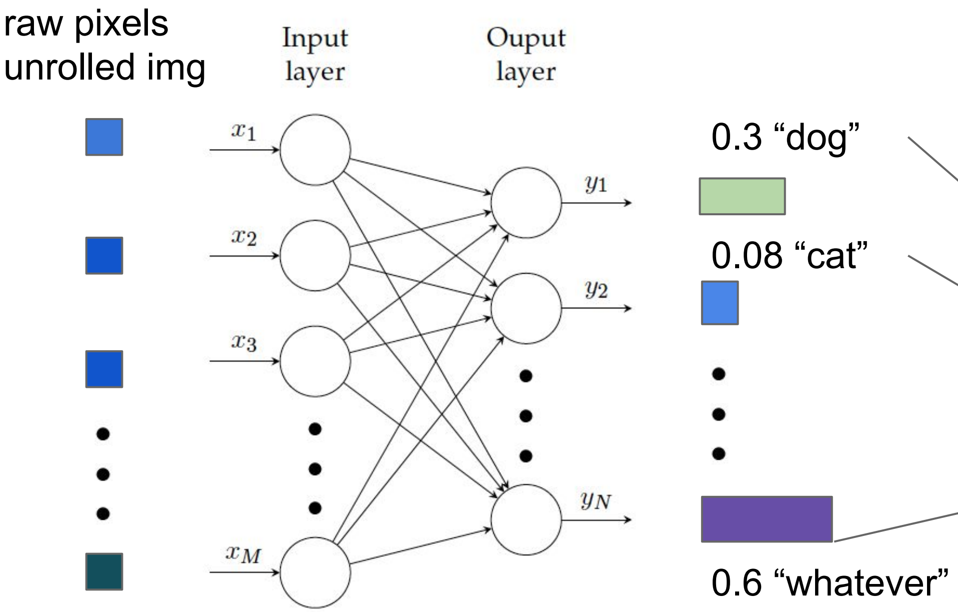
Softmax classifier: Multiclass (3 classes)



TensorFlow, [“MNIST for ML beginners”]

Softmax classifier: Multiclass (N classes)

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



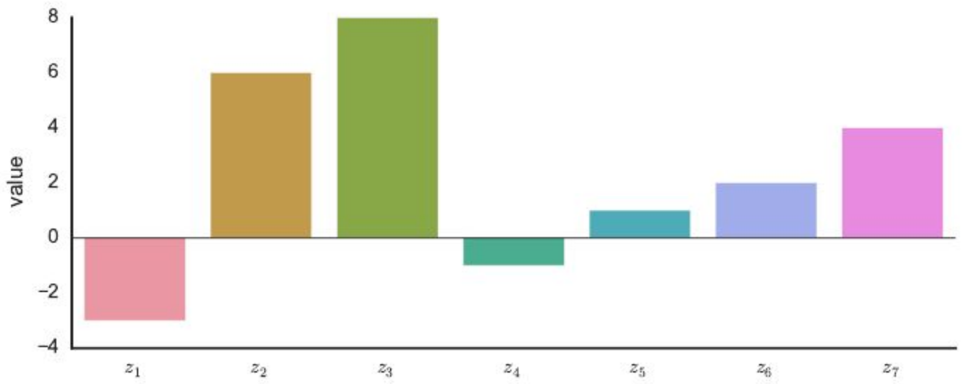
Softmax function

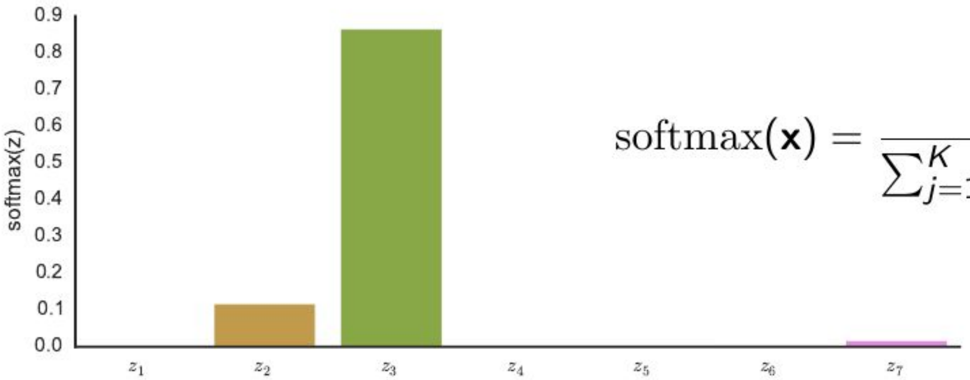
$$P(y=k|\mathbf{x}) = \frac{\exp \mathbf{x}^T \mathbf{w}_k}{\sum_{n=1}^{N} \exp \mathbf{x}^T \mathbf{w}_n}$$

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

32

Effect of the softmax





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

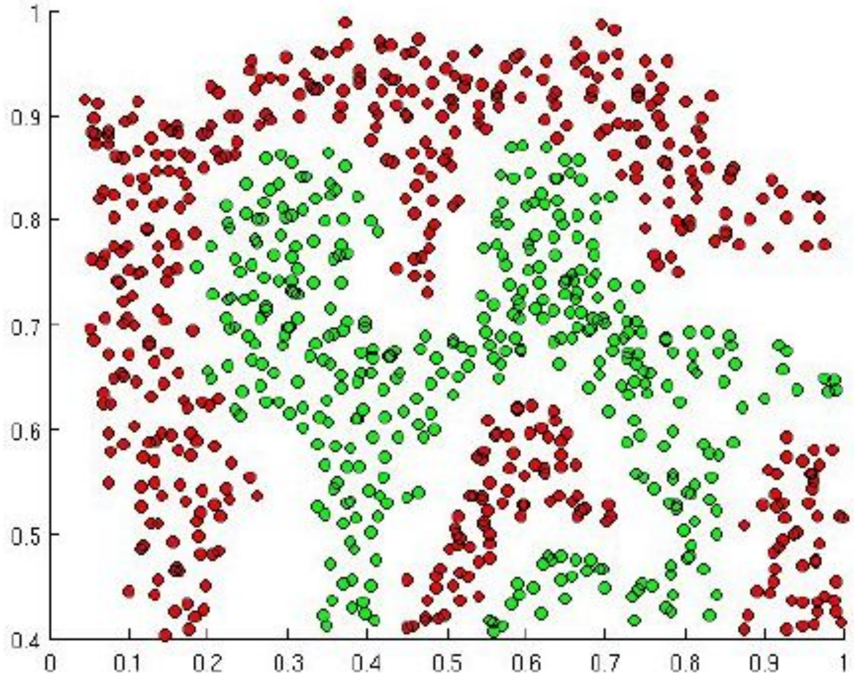
Next lecture...

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?