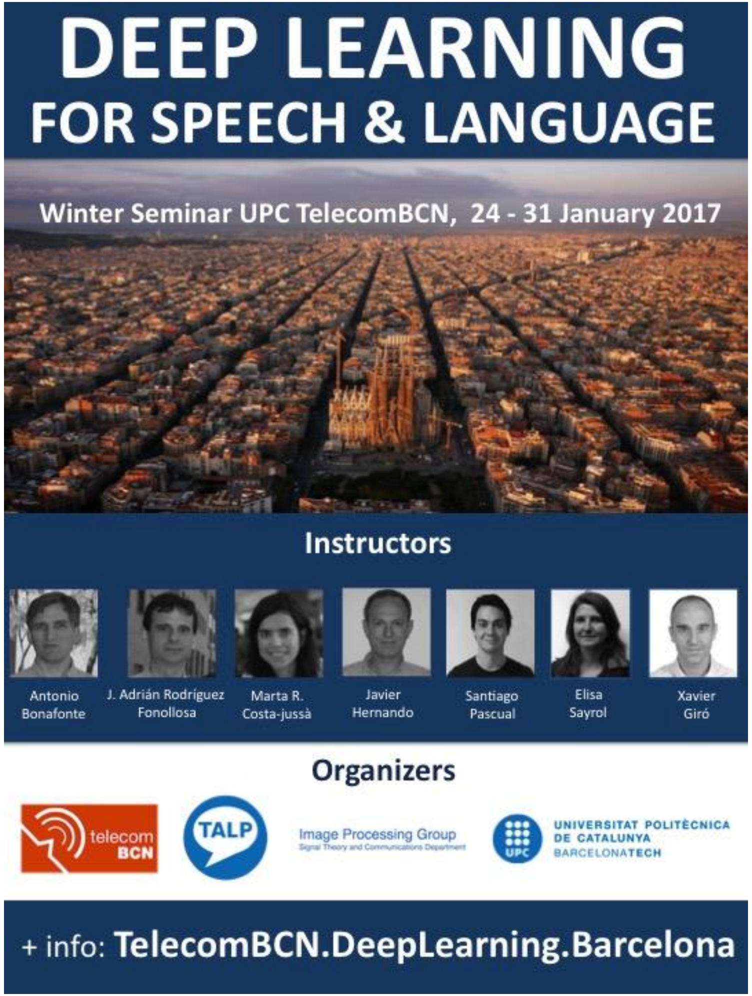


[course site]

Day 1 Lecture 2

The Perceptron

Santiago Pascual

Outline

1. Supervised learning: Regression/Classification
2. Linear regression
3. Logistic regression
4. The Perceptron
5. Multi-class classification
6. The Neural Network
7. Metrics

Machine Learning techniques

We can categorize three types of learning procedures:

1. **Supervised Learning:**

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

   We have a labeled dataset with pairs ($\mathbf{x}$, $\mathbf{y}$), e.g. classify a signal window as containing speech or not:
   x1 = [x(1), x(2), …, x(T)] **y1** = "no"
   x2 = [x(T+1), …, x(2T)] **y2** = "yes"
   x3 = [x(2T+1), …, x(3T)] **y3** = "yes"
   ...

2. Unsupervised Learning:

   $f(\mathbf{x})$

3. Reinforcement Learning:

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

   $\mathbf{z}$

Supervised Learning

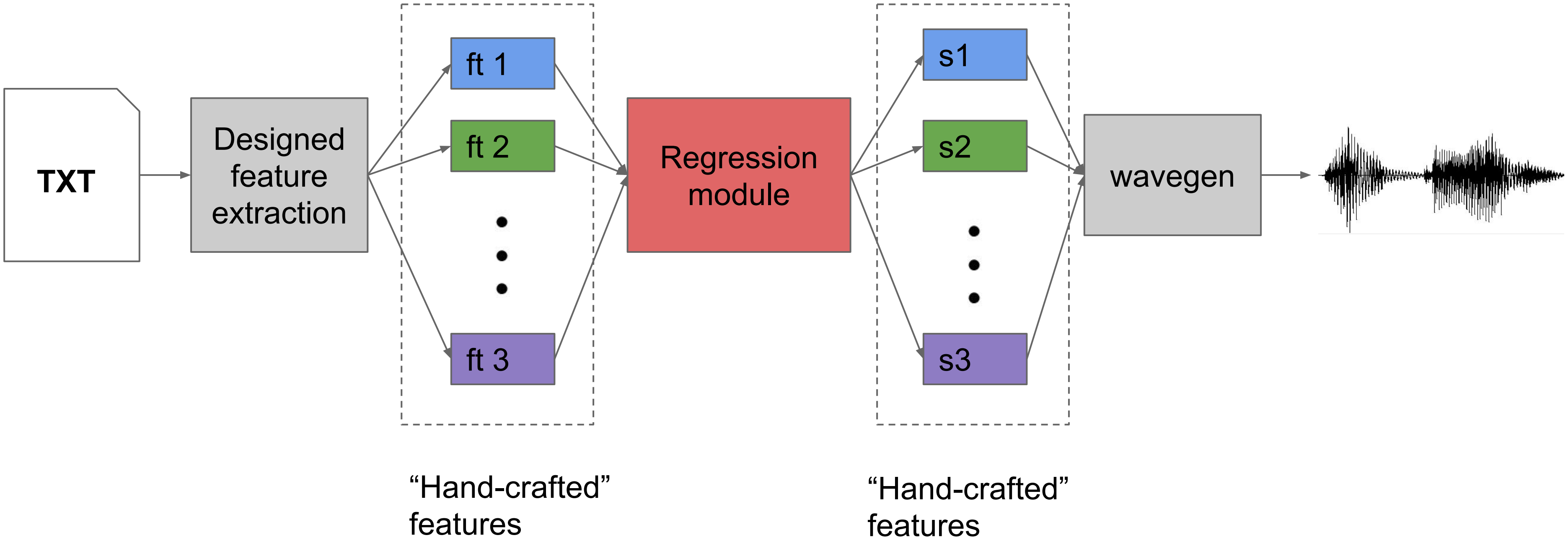
Build a function: $\mathbf{y} = f(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^m$, $\mathbf{y} \in \mathbb{R}^n$

Depending on the type of outcome we get…

- Regression: $\mathbf{y}$ is continous (e.g. temperature samples $\mathbf{y}$ = {19º, 23º, 22º})
- Classification: $\mathbf{y}$ 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”, ...

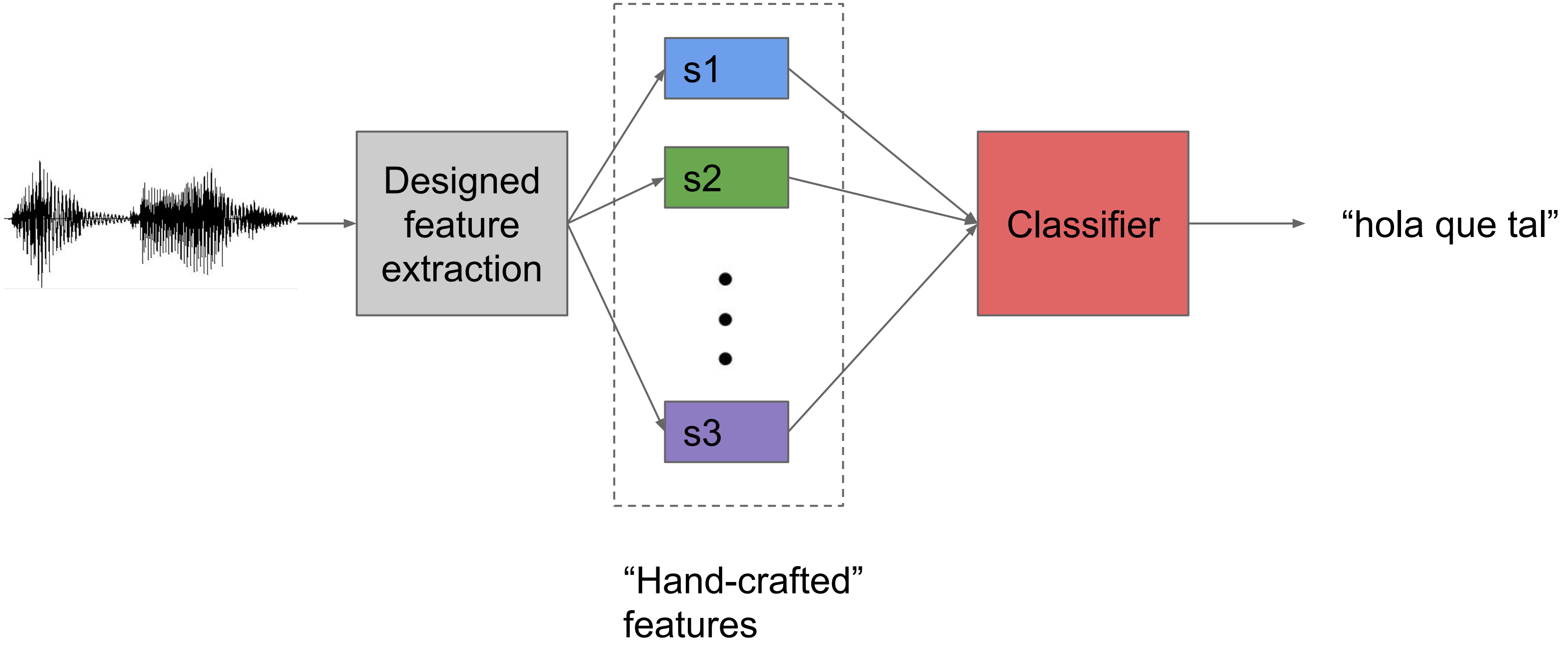
Regression motivation

Text to Speech: Textual features → Spectrum of speech (many coefficients)



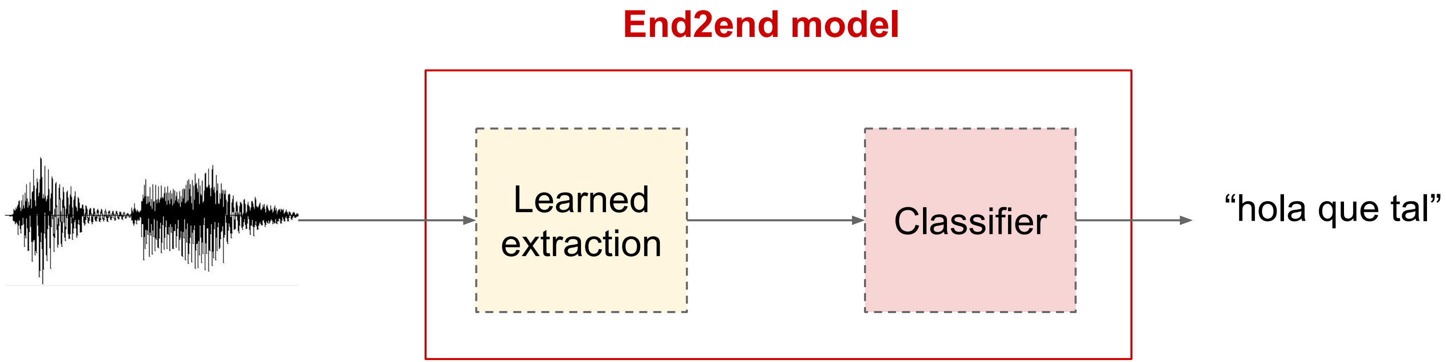
Classification motivation

Automatic Speech Recognition: Acoustic features → Textual transcription (words)



What “deep-models” means nowadays

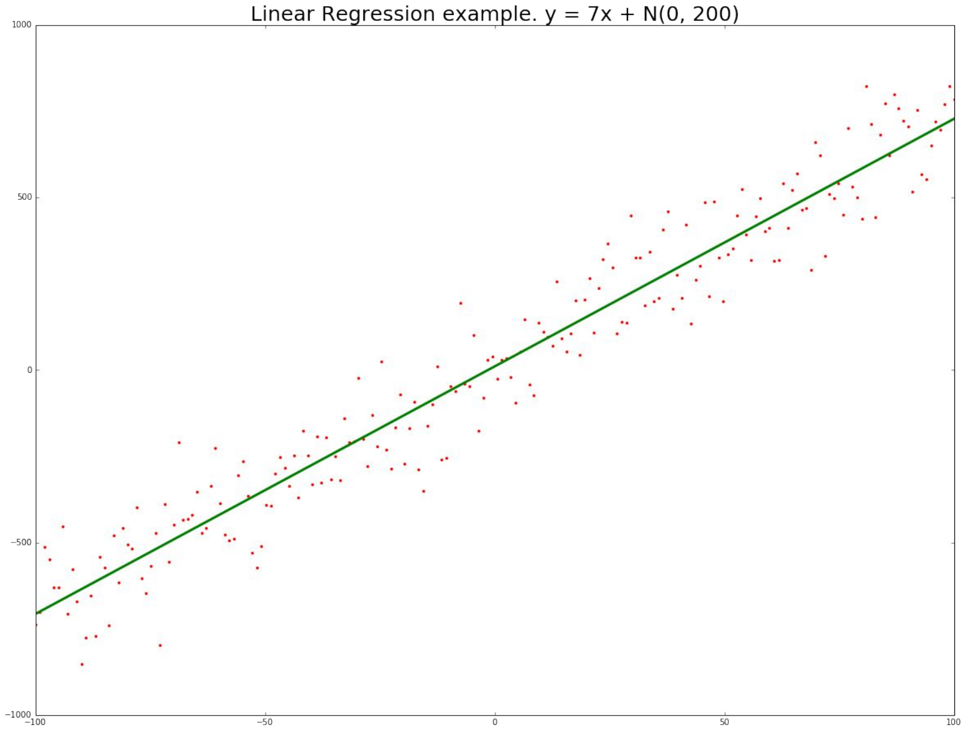
Learn the representations as well, not only the final mapping → **end2end**



Model maps raw inputs to raw outputs, no intermediate blocks.

Linear Regression

Function approximation $\mathbf{y} = \omega \cdot x + \beta$, with learnable parameters $\theta = \{\omega, \beta\}$



Linear Regression

We can also make the function more complex for **x** being an M-dimensional set of features: $\mathbf{y} = \omega1 \cdot x1 + \omega2 \cdot x2 + \omega3 \cdot x3 + \ldots + \omega M \cdot xM + \beta$

e.g. we want to predict the price of a house based on:

x1 = square-meters (sqm)

x2,3 = location (lat, lon)

x4 = year of construction (yoc)

price = $\omega1 \cdot$(sqm) + $\omega2 \cdot$(lat) + $\omega3 \cdot$(lon) + $\omega4 \cdot$(yoc) + β

- Fitting $f(\mathbf{x})$ means adjusting (**learning**) the values $\theta = \{\omega1, \omega2, \ldots, \omega M, \beta\}$
    - How? Will see in training chapter, stay tunned!

Logistic Regression

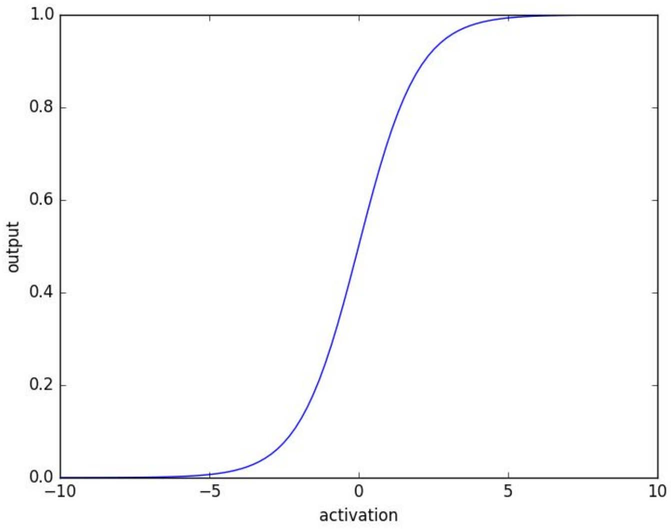
In the classification world we talk about **Probabilities**, and more concretely:

Given **x** input data features → Probability of y being:

- a dog $P(y=dog|\mathbf{x})$
- a cat $P(y=cat|\mathbf{x})$
- a horse $P(y=horse|\mathbf{x})$
- whatever $P(y=whatever|\mathbf{x})$.

We achieve so with the **sigmoid function**!

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



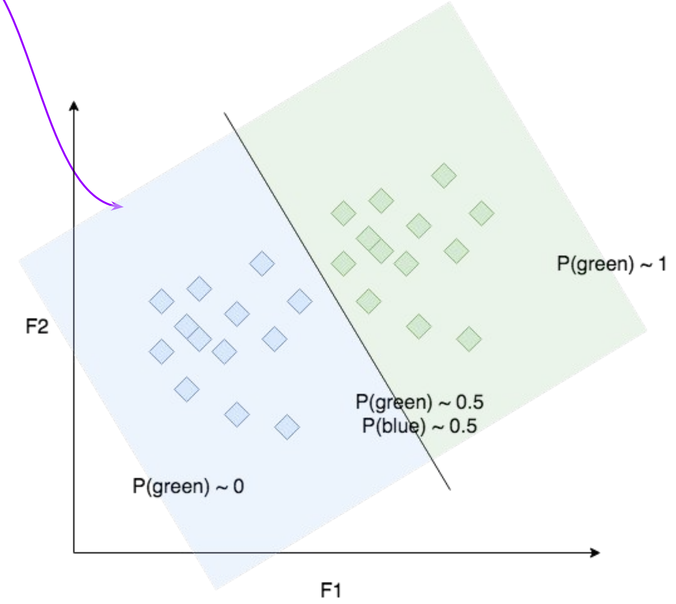
Note: This is a binary classification approach

Bounded $\sigma(\mathbf{x}) \in (0, 1)$

Logistic Regression

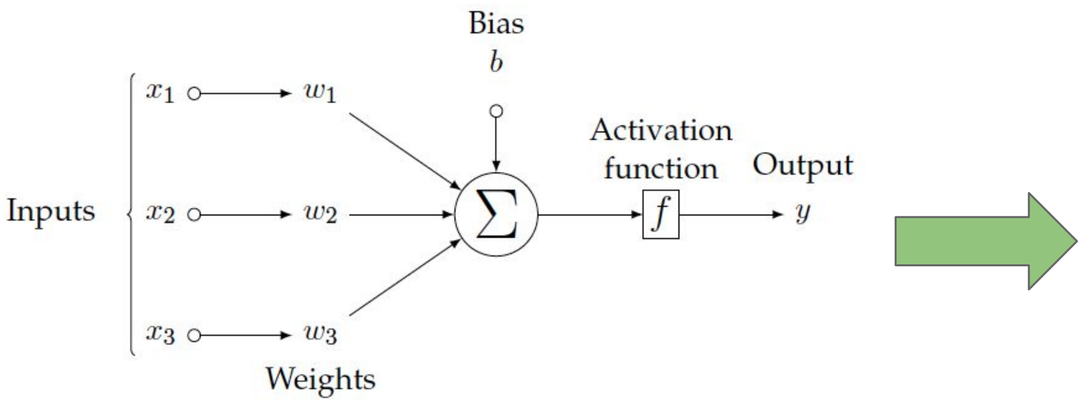
Interpretation: build a delimiting boundary between our data classes + apply the sigmoid function to estimate a probability in every point in the space.

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



The Perceptron

Both operations, linear regression and logistic regression, follow the scheme in the Figure:



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

$$y = f(a)$$

Depending on the Activation function f we have a linear/non-linear behavior:

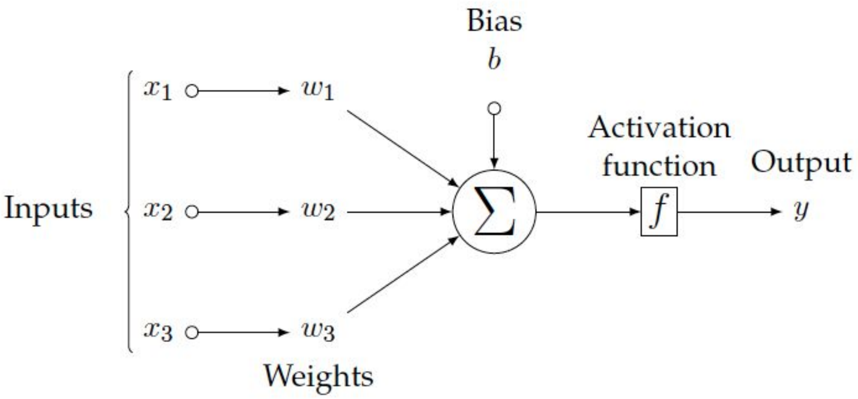
if f == identity → linear regression

if f == sigmoid → logistic regression

The Perceptron

The output is then derived by a **weighted sum** of the inputs **plus a bias term.**

Weights and bias are the parameters we keep (once learned) to define a neuron.



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

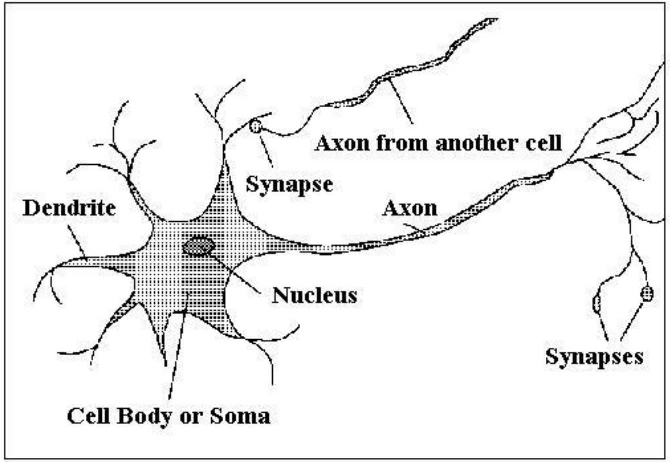
$$y = f(a)$$

The Perceptron

Actually the artificial neuron is seen as an analogy to a biological one.

Real neuron fires an impulse once the sum of all inputs is over a threshold.

The sigmoid emulates the thresholding behavior → act like a switch.



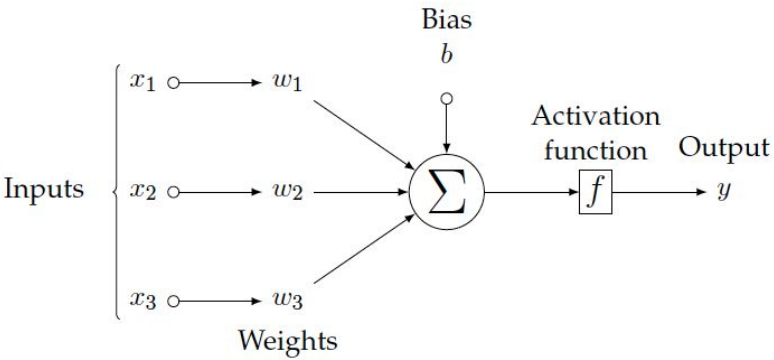
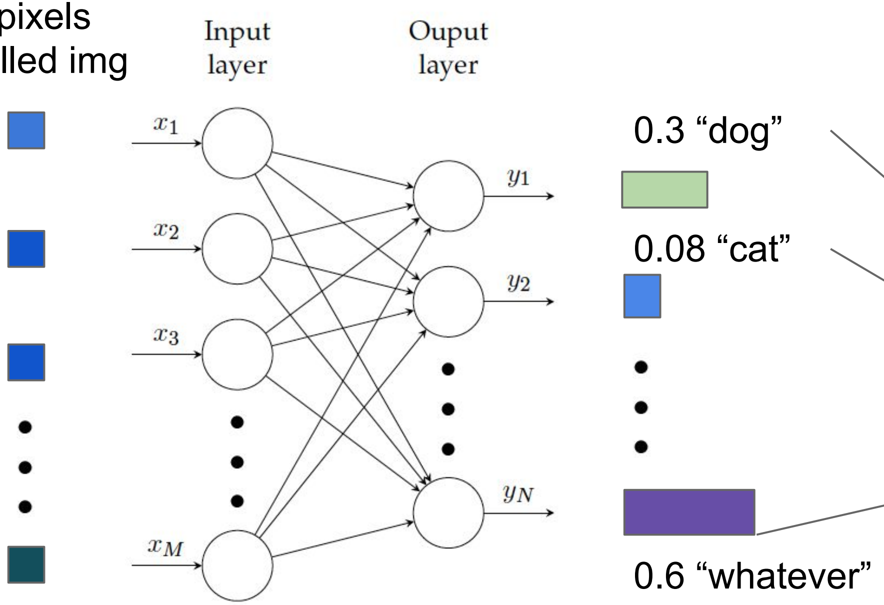


Figure credit:[Introduction to AI]

Multi-class classification

Natural extension: put many neurons in parallel, each processing its binary output out of N possible classes.

raw pixels
unrolled img



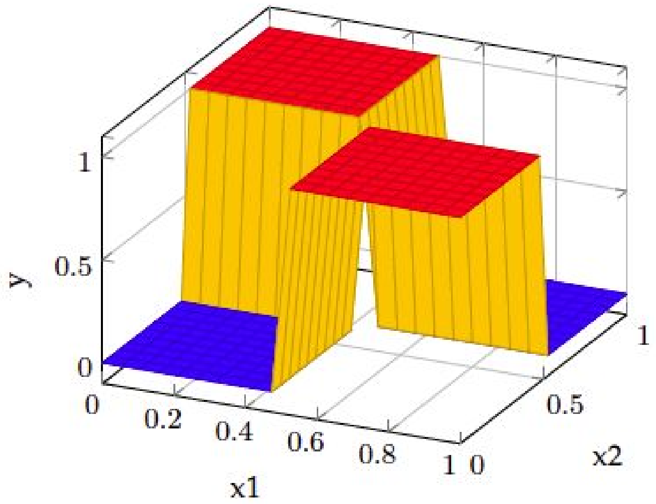
Softmax function

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

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

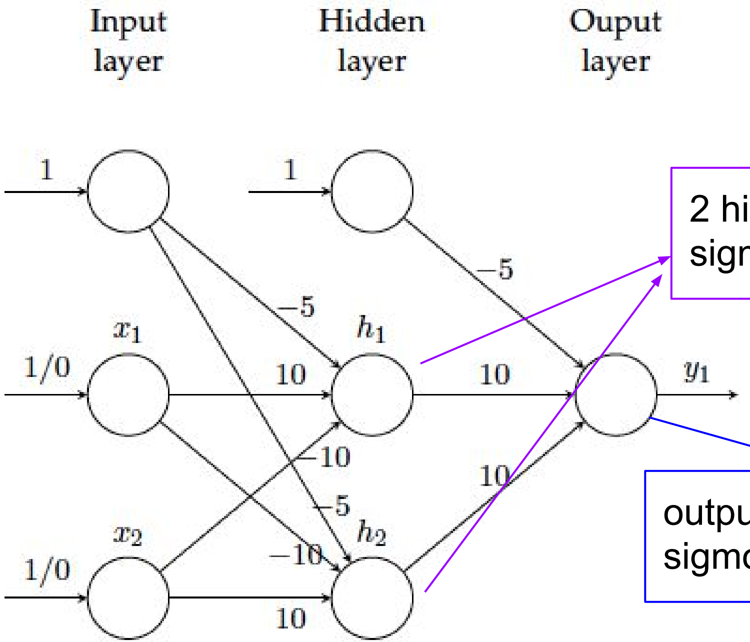
The Neural Network

The XOR problem: sometimes a single neuron is not enough → Just a single decision split doesn't work



The Neural Network

Solution: arrange many neurons in a first intermediate **non-linear** mapping (Hidden Layer), **connecting everything** from layer to layer in a **feed-forward** fashion.



Warning! Inputs are not neurons, but they are usually depicted like neurons.

2 hidden units with sigmoid activations

Example XOR NN

output unit with sigmoid activation

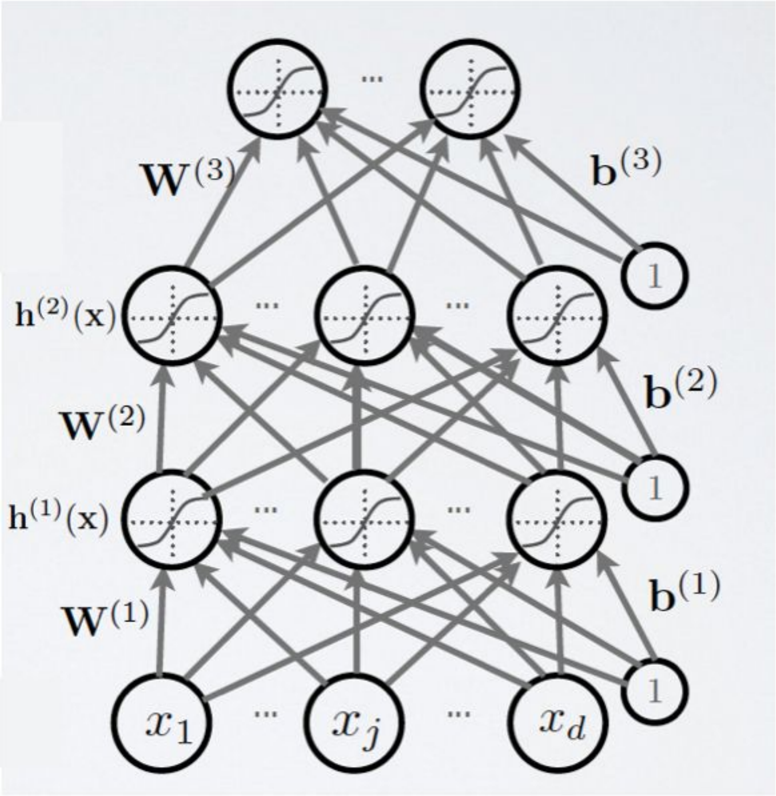
The Neural Network

The i-th layer is defined by a matrix **Wi** and a vector **bi,** and the activation is simply a dot product plus **bi**:

$$h_i = f(W_i \cdot h_{i-1} + b_i)$$

Num parameters to learn at i-th layer:

$$N^i_{params} = N^i_{inputs} \times N^i_{units} + N^i_{units}$$



Slide Credit: Hugo Laroche NN course

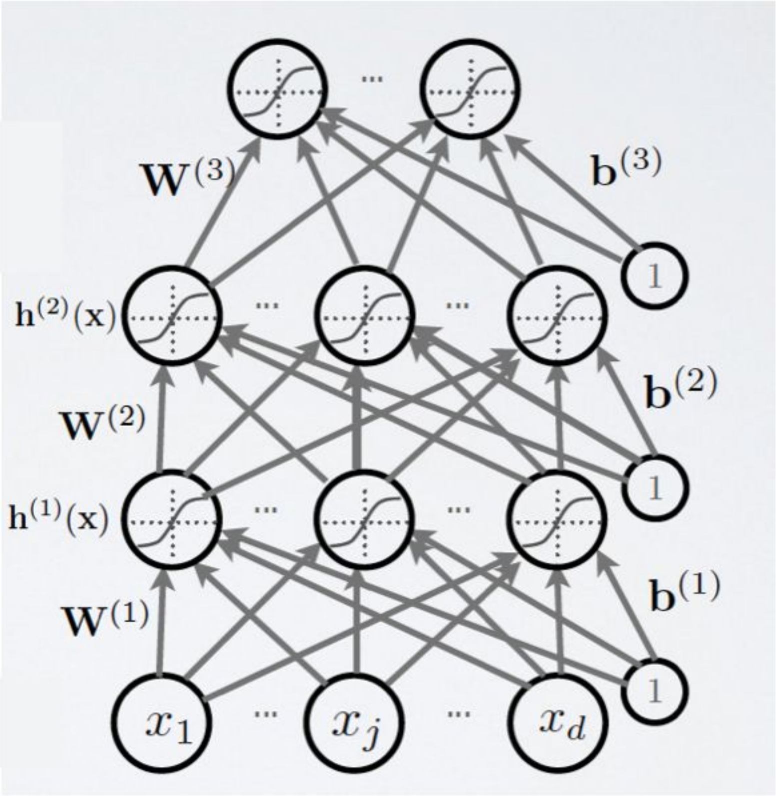
The Neural Network

The i-th layer is defined by a matrix **Wi** and a vector **bi,** and the activation is simply a dot product plus **bi**:

$$h_i = f(W_i \cdot h_{i-1} + b_i)$$

Num parameters to learn at i-th layer:

$$N^i_{params} = N^i_{inputs} \times N^i_{units} + N^i_{units}$$



Slide Credit: Hugo Laroche NN course

The Neural Network

Important remarks:

- We can put as many hidden layers as we want whenever training can be effectively done and we have enough data (next chapters)
  - **The amount of parameters to estimate grows very quickly** with the num of layers and units! → There is **no formula to know the amount of units per layer nor the amount of layers**, pitty...
- **The power of NNets comes from non-linear mappings**: hidden units must be followed by a non-linear activation!
  - *sigmoid, tanh, relu, leaky-relu, prelu, exp, softplus, …*

Regression metrics

In regression the metric is chosen based on the task:

- For example in TTS there are different metrics for the different predicted parameters:
  - Mel-Cepstral Distortion, Root Mean Squared Error F0, duration, ...

Classification metrics

Confusion matrices provide a by-class comparison between the results of the automatic classifications with ground truth annotations.

		Automatic		
		class1	class2	class3
Manual	class1	12	1	0
	class2	3	13	0
	class3	0	0	20

		Automatic		
		class1	class2	class3
Manual	class1	100%	0%	0%
	class2	0%	100%	0%
	class3	0%	0%	100%

Slide credit: Xavi Giró

Classification metrics

Correct classifications appear in the diagonal, while the rest of cells correspond to errors.

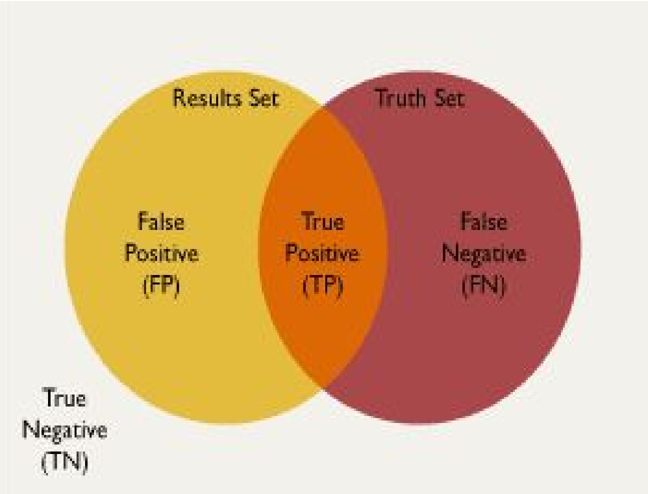
		Prediction		
		Class 1	Class 2	Class 3
Ground Truth	Class 1	x(1,1)	x(1,2)	x(1,3)
	Class 2	x(2,1)	x(2,2)	x(2,3)
	Class 3	x(3,1)	x(3,2)	x(3,3)

Slide credit: Xavi Giró

Classification metrics

Special case: Binary classifiers in terms of “Positive” vs “Negative”.

		Prediction	
		Positives	negative
Ground Truth	Positives	True positive (TP)	False negative (FN)
	negative	False positives (FP)	True negative (TN)



Slide credit: Xavi Giró

Classification metrics

The “accuracy” measures the proportion of correct classifications, not distinguishing between classes.

$$Accuracy = \frac{\boxed{\sum_{i=1}^{3} x(i,i)}}{\boxed{\sum_{i=1}^{3}\sum_{j=1}^{3} x(i,j)}}$$

Binary

$$Accuracy = \frac{\boxed{TP + TN}}{\boxed{TP + TN + FP + FN}}$$

		Prediction		
		Class 1	Class 2	Class 3
Ground Truth	Class 1	x(1,1)	x(1,2)	x(1,3)
	Class 2	x(2,1)	x(2,2)	x(2,3)
	Class 3	x(3,1)	x(3,2)	x(3,3)

		Prediction	
		Positives	negative
Ground Truth	Positives	True positive (TP)	False negative (FN)
	Negative	False positives (FP)	True negative (TN)

Slide credit: Xavi Giró

Classification metrics

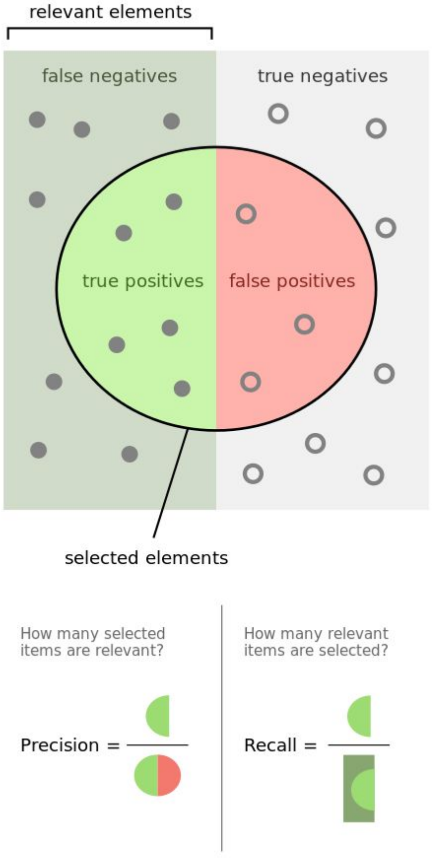
Given a reference class, its Precision (P) and Recall (R) are complementary measures of relevance.

Example: Relevant class is “Positive” in a binary classifier.

		Prediction	
		Positives	Negatives
Ground Truth	Positives	True positive (TP)	False negative (FN)
	Negatives	False positives (FP)	

$$Recall = \frac{TP}{TP+FN}$$

$$Precision = \frac{TP}{TP+FP}$$



Slide credit: Xavi Giró

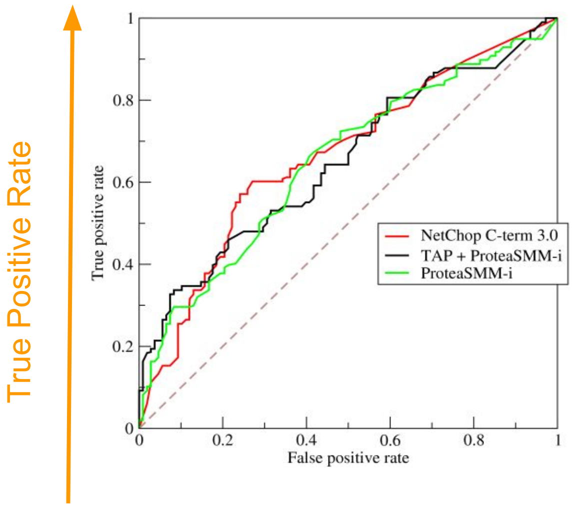
Classification metrics

Binary classification results often [depend from a parameter] (eg. decision threshold) whose value directly impacts precision and recall.

For this reason, in many cases a Receiver Operating Curve (ROC curve) is provided as a result.

$$True\ Positive\ Rate = \frac{TP}{TP+FN} = Recall = Sensitivity$$

$$False\ Positive\ Rate = \frac{FP}{TP+FN} = 1 - specificity$$



Thanks ! Q&A ?

@Santty128