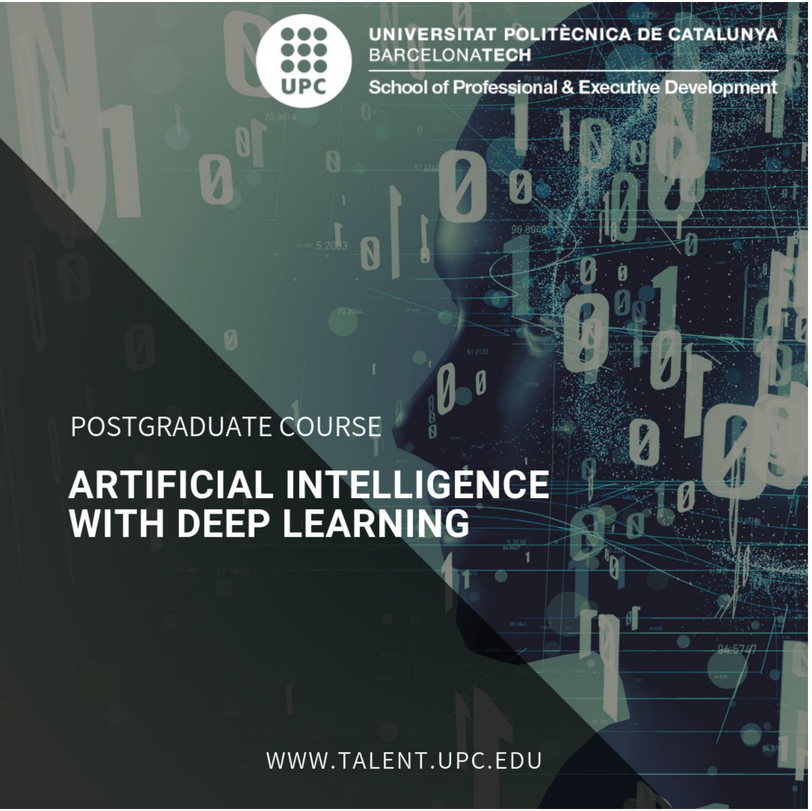


#DLUPC

Day 1 Lecture 4

Multilayer Perceptron MLP

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

Associate Professor
Universitat Politècnica de Catalunya
Technical University of Catalonia

Acknowledgements

Kevin McGuinness
kevin.mcguinness@dcu.ie

Research Fellow
Insight Centre for Data Analytics
Dublin City University

Eva Mohedano
eva.mohedano@insight-centre.org

Postdoctoral Researcher
Insight Centre for Data Analytics
Dublin City University

Elisa Sayrol
elisa.sayrol@upc.edu

Associate Professor
ETSETB TelecomBCN
Universitat Politècnica de Catalunya

Antonio Bonafonte
antonio.bonafonte@upc.edu

Associate Professor
ETSETB TelecomBCN
Universitat Politècnica de Catalunya

Video lecture

Overview

- Limitations the perceptron model
- Multilayer perceptron

Overview

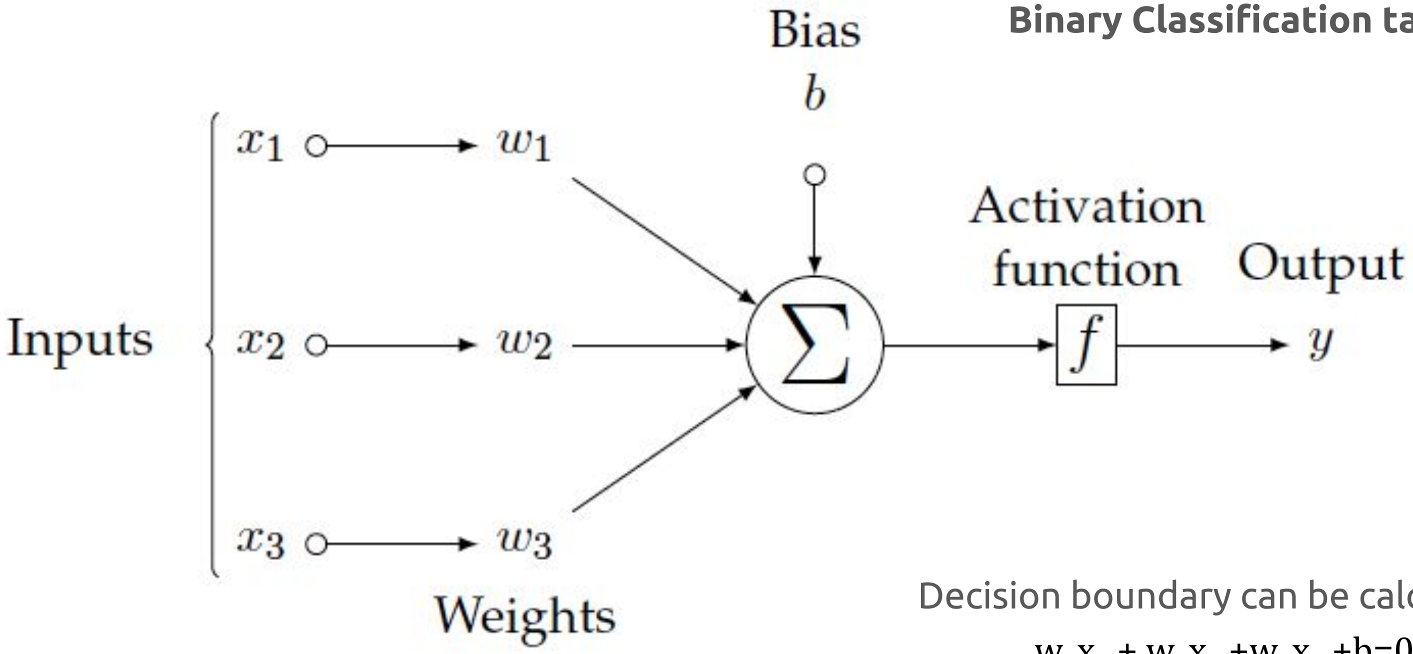
- **Limitations the perceptron model**

- Multilayer perceptron

Single Neuron

If the weighted sum of the input exceeds a threshold the neuron fires a signal.

Binary Classification task

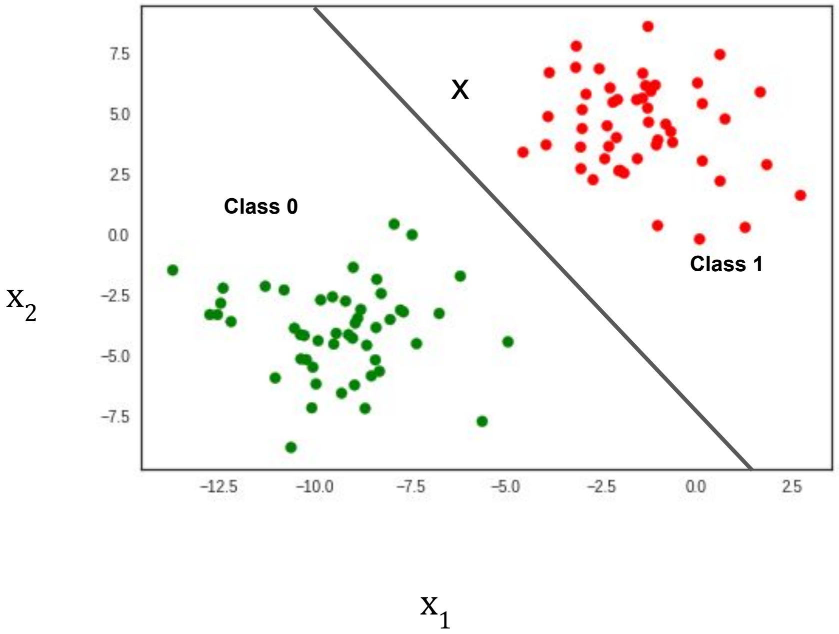


Decision boundary can be calculated by:

$$w_1x_1 + w_2x_2 + w_3x_3 + b = 0$$

Linear decision decision boundary

2D input space data



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

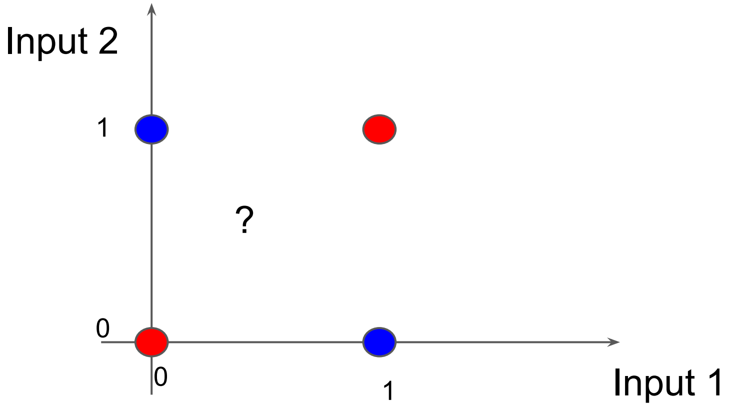
The XOR problem

Limitation: Data might be **non linearly separable**

→ One single neuron is not enough

XOR logic table

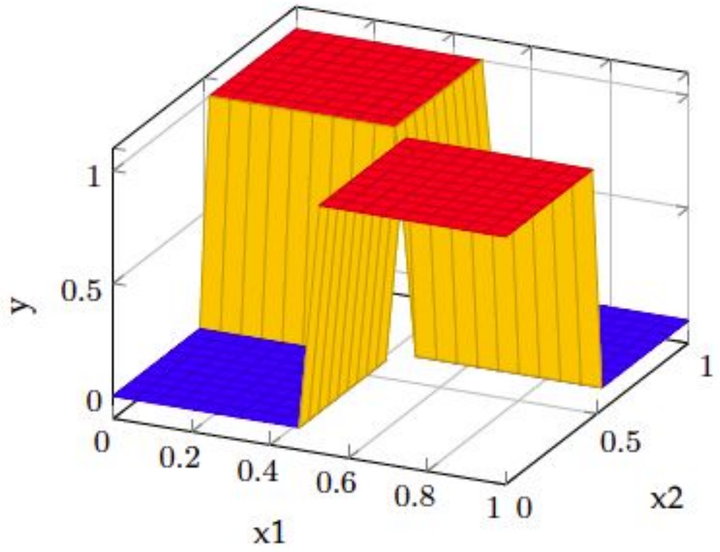
Input 1	Input 2	Desired Output
0	0	0
0	1	1
1	0	1
1	1	0

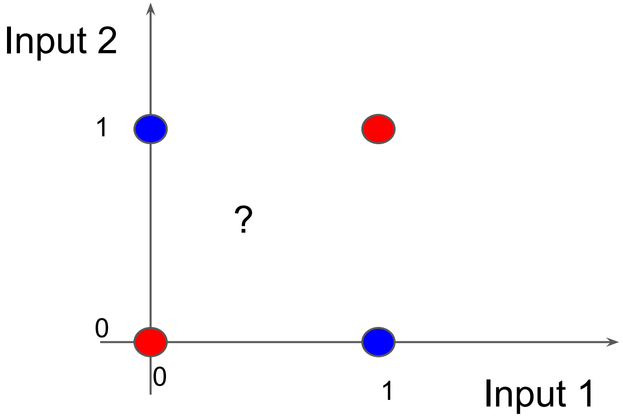


The XOR problem

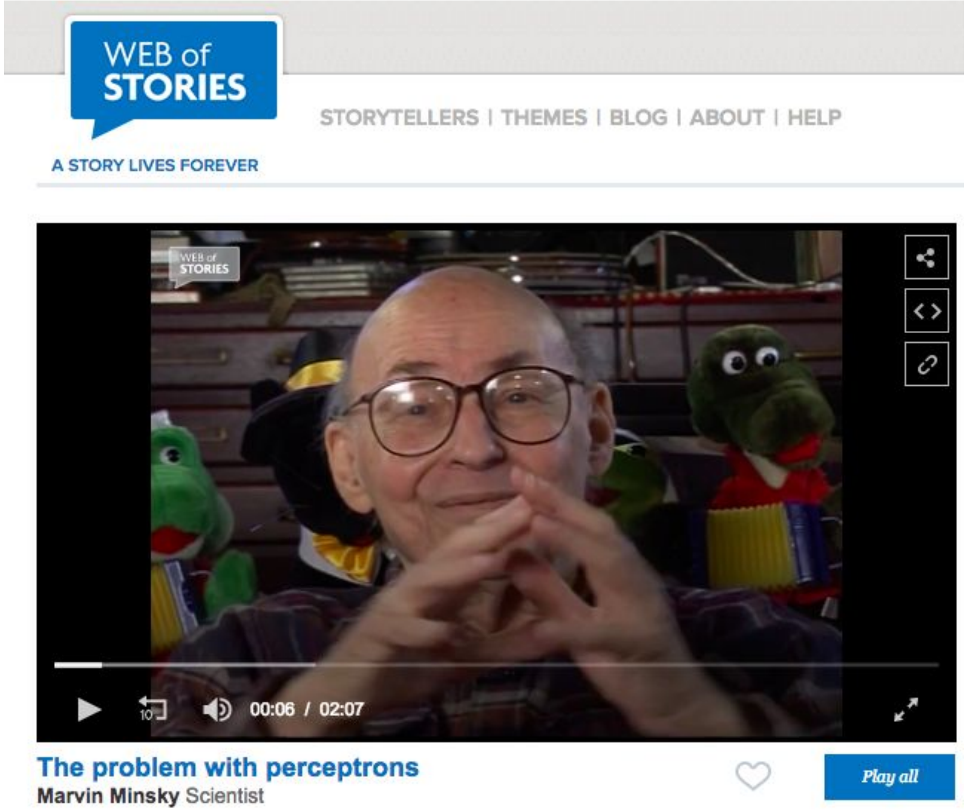
Limitation: Data might be **non linearly separable**

→ One single neuron is not enough





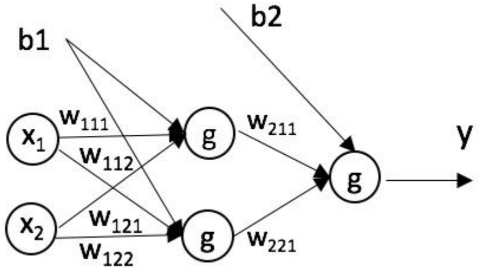
The limitations of the perceptron



Minsky, Marvin, and Seymour A. Papert. [Perceptrons: An introduction to computational geometry](). 1969

The XOR problem

Given the following network to obtain a XNOR operation, Indicate which set of parameters are correct (a, b, c or d):

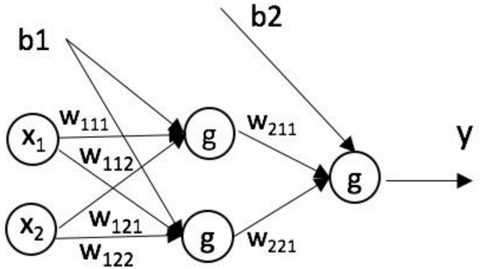


Input vector (x1,x2)	Class XNOR
(0,0)	1
(0,1)	0
(1,0)	0
(1,1)	1

	w_{111}	w_{112}	w_{121}	w_{122}	b_1	w_{211}	w_{221}	b_2
a	2	2	2	2	-1	2	2	-1
b	-2	2	2	-2	-1	2	2	1
c	-2	2	2	-2	-1	2	-2	1
d	-2	2	2	-2	-1	2	-2	-1

The XOR problem

Given the following network to obtain a XNOR operation, Indicate which set of parameters are correct (a, b, c or d):

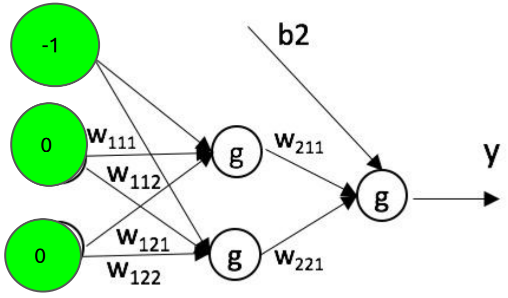


Input vector (x1,x2)	Class XNOR
(0,0)	1
(0,1)	0
(1,0)	0
(1,1)	1

	w_{111}	w_{112}	w_{121}	w_{122}	b_1	w_{211}	w_{221}	b_2
a	2	2	2	2	-1	2	2	-1
b	-2	2	2	-2	-1	2	2	1
c	-2	2	2	-2	-1	2	-2	1
d	-2	2	2	-2	-1	2	-2	-1

The XOR problem

Given the following network to obtain a XNOR operation, Indicate which set of parameters are correct (a, b, c or d):

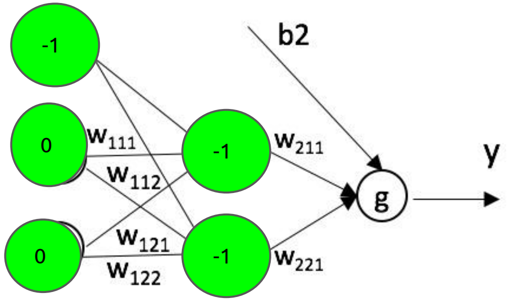


Input vector (x1,x2)	Class XNOR
(0,0)	1
(0,1)	0
(1,0)	0
(1,1)	1

	w_{111}	w_{112}	w_{121}	w_{122}	b_1	w_{211}	w_{221}	b_2
a	2	2	2	2	-1	2	2	-1
b	-2	2	2	-2	-1	2	2	1
c	-2	2	2	-2	-1	2	-2	1
d	-2	2	2	-2	-1	2	-2	-1

The XOR problem

Given the following network to obtain a XNOR operation, Indicate which set of parameters are correct (a, b, c or d):

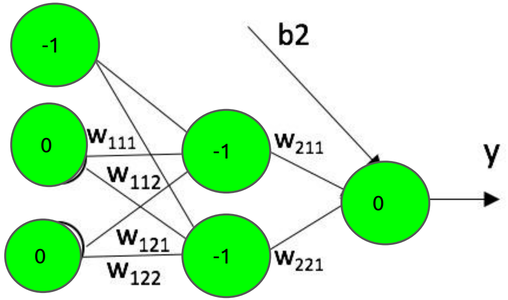


Input vector (x1,x2)	Class XNOR
(0,0)	1
(0,1)	0
(1,0)	0
(1,1)	1

	w_{111}	w_{112}	w_{121}	w_{122}	b_1	w_{211}	w_{221}	b_2
a	2	2	2	2	-1	2	2	-1
b	-2	2	2	-2	-1	2	2	1
c	-2	2	2	-2	-1	2	-2	1
d	-2	2	2	-2	-1	2	-2	-1

The XOR problem

Given the following network to obtain a XNOR operation, Indicate which set of parameters are correct (a, b, c or d):

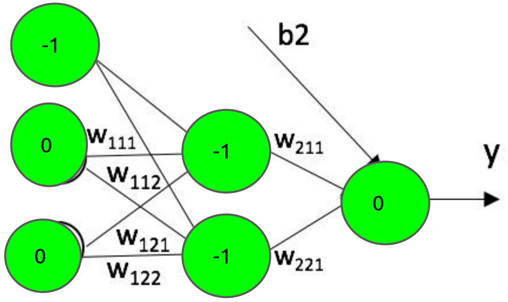


Input vector (x1,x2)	Class XNOR
(0,0)	1
(0,1)	0
(1,0)	0
(1,1)	1

	w_{111}	w_{112}	w_{121}	w_{122}	b_1	w_{211}	w_{221}	b_2
a	2	2	2	2	-1	2	2	-1
b	-2	2	2	-2	-1	2	2	1
c	-2	2	2	-2	-1	2	-2	1
d	-2	2	2	-2	-1	2	-2	-1

The XOR problem

Given the following network to obtain a XNOR operation, Indicate which set of parameters are correct (a, b, c or d):

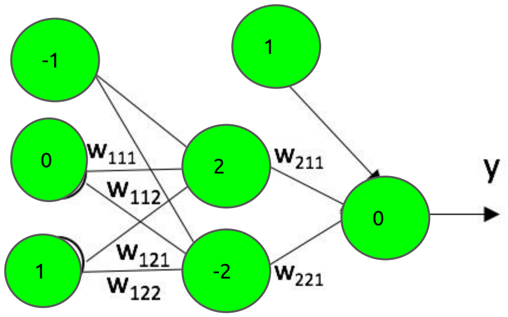


Input vector (x1,x2)	Class XNOR
(0,0)	1
(0,1)	0
(1,0)	0
(1,1)	1

	w_{111}	w_{112}	w_{121}	w_{122}	b_1	w_{211}	w_{221}	b_2
a	2	2	2	2	-1	2	2	-1
b	-2	2	2	-2	-1	2	2	1
c	-2	2	2	-2	-1	2	-2	1
d	-2	2	2	-2	-1	2	-2	-1

The XOR problem

Given the following network to obtain a XNOR operation, Indicate which set of parameters are correct (a, b, c or d):

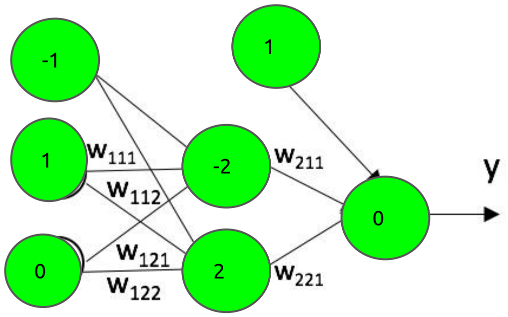


Input vector (x1,x2)	Class XNOR
(0,0)	1
(0,1)	0
(1,0)	0
(1,1)	1

	w_{111}	w_{112}	w_{121}	w_{122}	b_1	w_{211}	w_{221}	b_2
a	2	2	2	2	-1	2	2	-1
b	-2	2	2	-2	-1	2	2	1
c	-2	2	2	-2	-1	2	-2	1
d	-2	2	2	-2	-1	2	-2	-1

The XOR problem

Given the following network to obtain a XNOR operation, Indicate which set of parameters are correct (a, b, c or d):

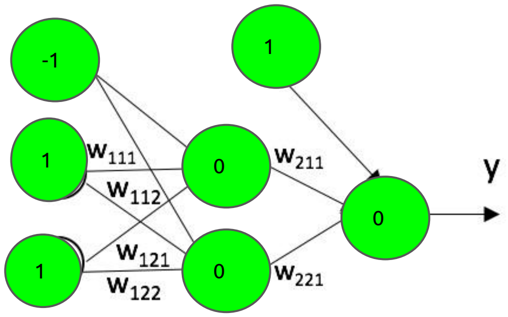


Input vector (x1,x2)	Class XNOR
(0,0)	1
(0,1)	0
(1,0)	0
(1,1)	1

	w_{111}	w_{112}	w_{121}	w_{122}	b_1	w_{211}	w_{221}	b_2
a	2	2	2	2	-1	2	2	-1
b	-2	2	2	-2	-1	2	2	1
c	-2	2	2	-2	-1	2	-2	1
d	-2	2	2	-2	-1	2	-2	-1

The XOR problem

Given the following network to obtain a XNOR operation, Indicate which set of parameters are correct (a, b, c or d):



Input vector (x1,x2)	Class XNOR
(0,0)	1
(0,1)	0
(1,0)	0
(1,1)	1

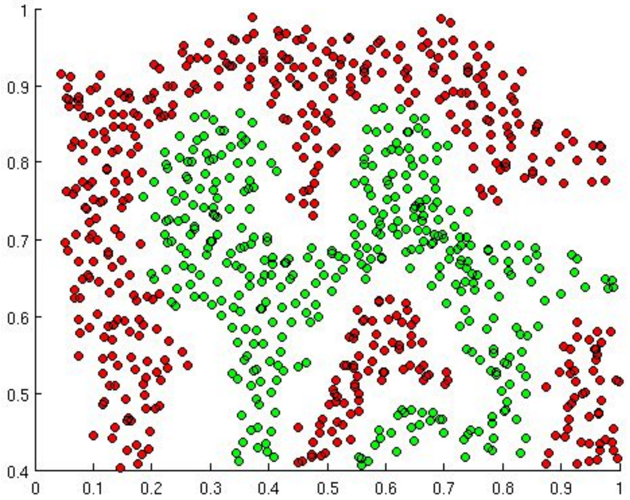
	w_{111}	w_{112}	w_{121}	w_{122}	b_1	w_{211}	w_{221}	b_2
a	2	2	2	2	-1	2	2	-1
b	-2	2	2	-2	-1	2	2	1
c	-2	2	2	-2	-1	2	-2	1
d	-2	2	2	-2	-1	2	-2	-1

Non-linear decision boundaries

Linear models can only produce linear decision boundaries

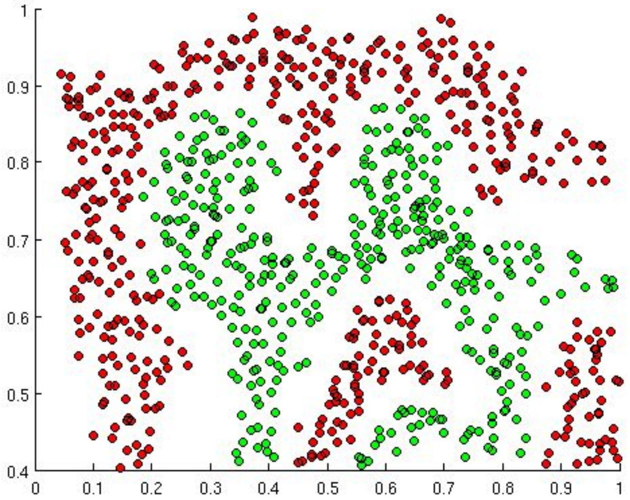
Real world data often needs a non-linear decision boundary

- Images
- Audio
- Text



Non-linear decision boundaries

What can we do?

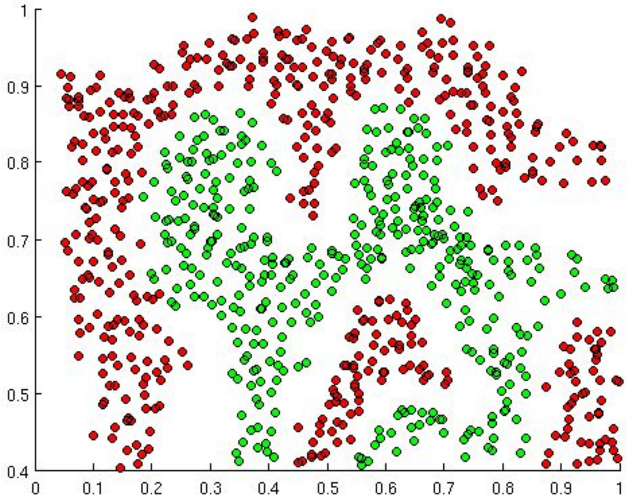


1. Use a non-linear classifier
   - Decision trees (and forests)
   - K nearest neighbours
2. Engineer a suitable representation
   - One in which features are more linearly separable
   - Then use a linear model
3. Engineer a kernel
   - Design a kernel **K(x$_1$, x$_2$)**
   - Use kernel methods (e.g. SVM)
4. Learn a suitable representation space from the data
   - Deep learning, deep neural networks
   - Boosted cascade classifiers like Viola Jones also take this approach

Non-linear decision boundaries

What can we do?

1. Use a non-linear classifier
   - Decision trees (and forests)
   - K nearest neighbours
2. Engineer a suitable representation
   - One in which features are more linearly separable
   - Then use a linear model
3. Engineer a kernel
   - Design a kernel $\mathbf{K}(\mathbf{x}_1, \mathbf{x}_2)$
   - Use kernel methods (e.g. SVM)
4. Learn a suitable representation space from the data
   - **Deep learning, deep neural networks**
   - Boosted cascade classifiers like Viola Jones also take this approach



Overview

- Limitations the perceptron model
- **Multilayer perceptron**

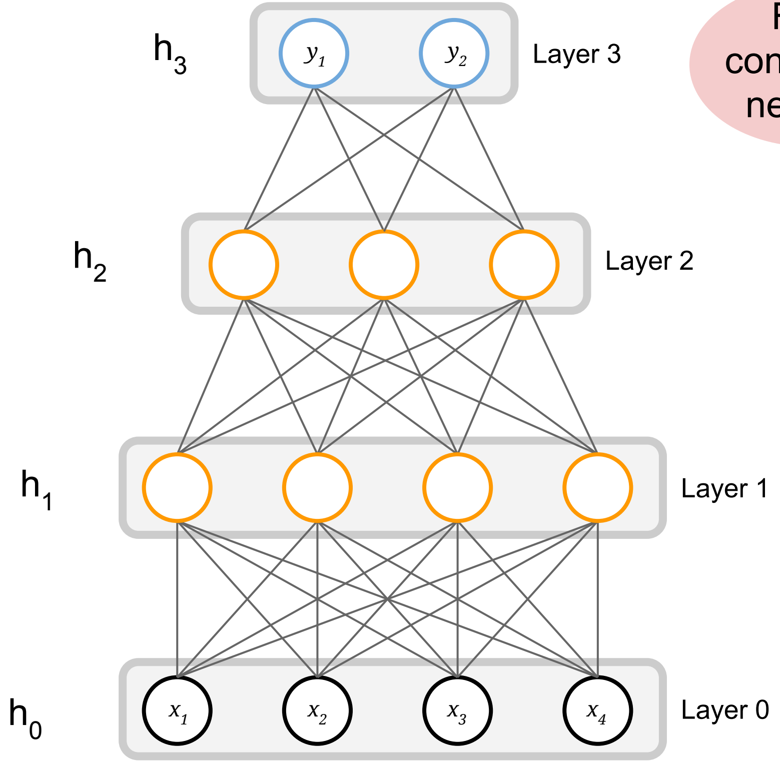
Multilayer perceptrons

When each node in each layer is a linear combination of **all inputs from the previous layer** then the network is called a multilayer perceptron (MLP)

Weights can be organized into matrices.

Forward pass computes

$$\mathbf{h}_0 = \mathbf{x}$$
$$\mathbf{h}^{(t)} = g(W^{(t)}\mathbf{h}^{(t-1)} + \mathbf{b}^{(t)})$$
$$f(\mathbf{x}) = \mathbf{h}^{(L)}$$



Fully connected network

Multilayer perceptrons

W_1

w_{11}	w_{12}	w_{13}	w_{14}
w_{21}	w_{22}	w_{23}	w_{24}
w_{31}	w_{32}	w_{33}	w_{34}
w_{41}	w_{42}	w_{43}	w_{44}

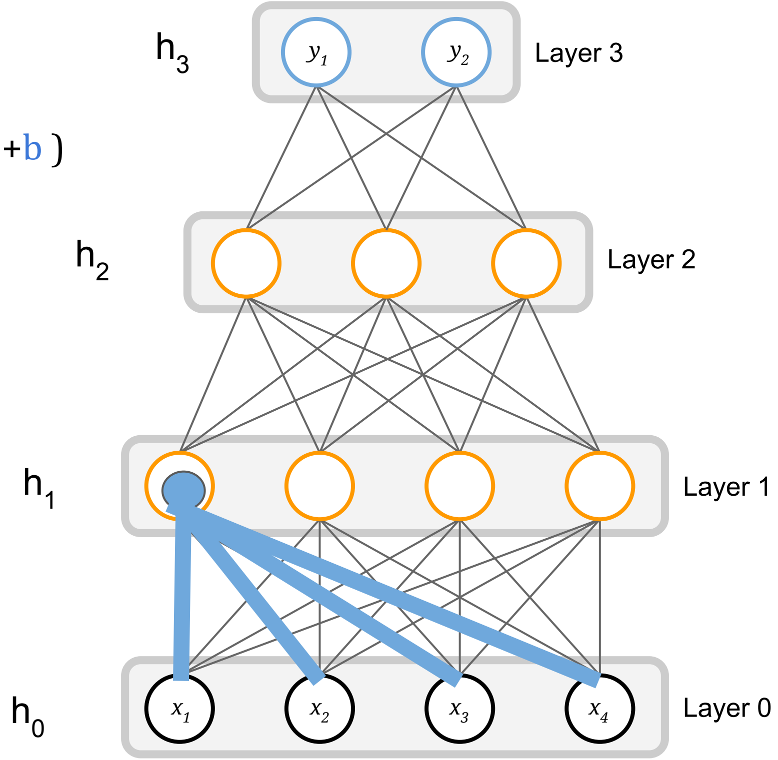
h_0

x_1
x_2
x_3
x_4

b_1

b_1
b_2
b_3
b_4

$h_{11} = g(\mathbf{wx} + b)$



Forward pass computes

$$\mathbf{h}_0 = \mathbf{x}$$
$$\mathbf{h}^{(t)} = g(W^{(t)}\mathbf{h}^{(t-1)} + \mathbf{b}^{(t)})$$
$$f(\mathbf{x}) = \mathbf{h}^{(L)}$$

Multilayer perceptrons

W_1

w_{11}	w_{12}	w_{13}	w_{14}
w_{21}	w_{22}	w_{23}	w_{24}
w_{31}	w_{32}	w_{33}	w_{34}
w_{41}	w_{42}	w_{43}	w_{44}

h_0

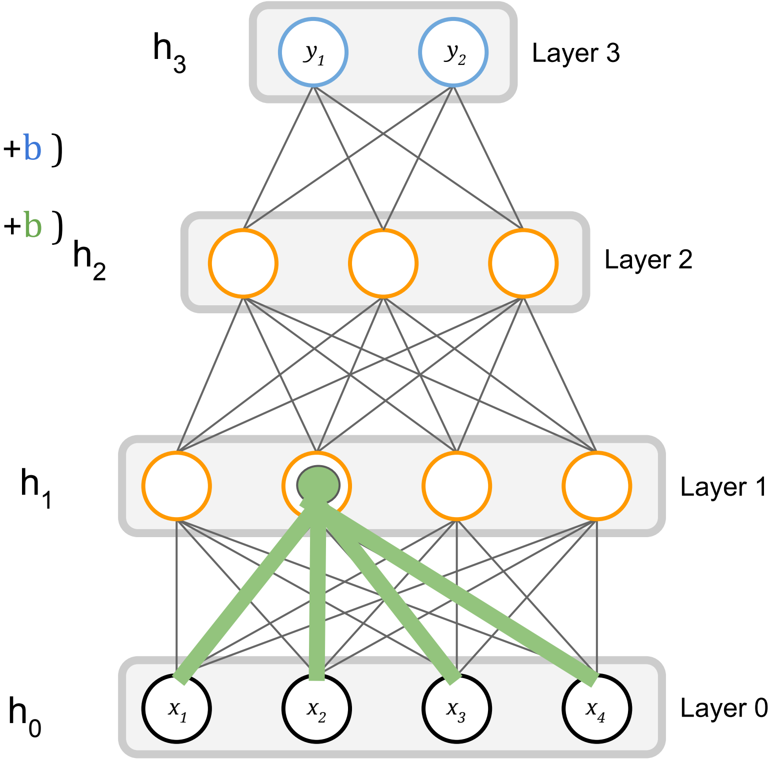
x_1
x_2
x_3
x_4

b_1

b_1
b_2
b_3
b_4

$$h_{11} = g(\mathbf{w}\mathbf{x} + b)$$

$$h_{12} = g(\mathbf{w}\mathbf{x} + b)$$



Forward pass computes

$$\mathbf{h}_0 = \mathbf{x}$$

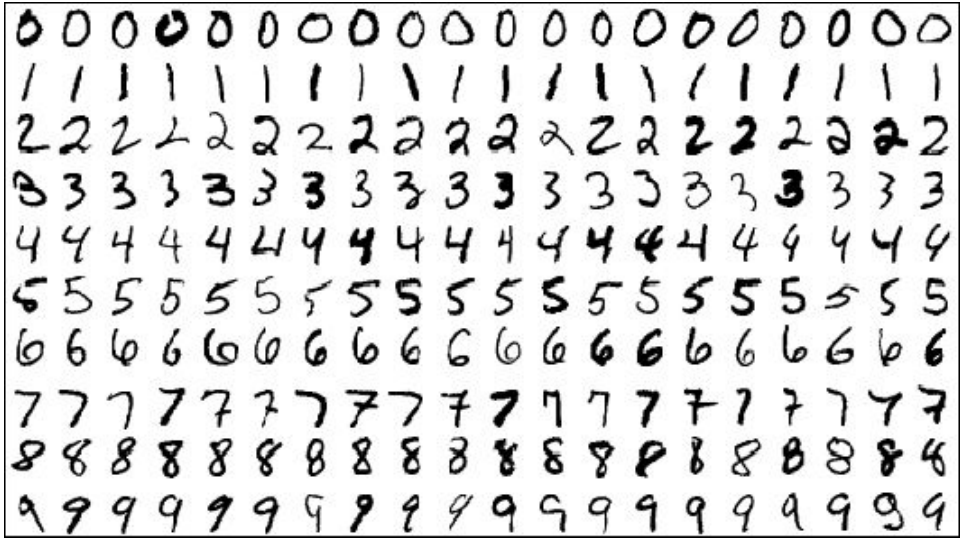
$$\mathbf{h}^{(t)} = g(W^{(t)}\mathbf{h}^{(t-1)} + \mathbf{b}^{(t)})$$

$$f(\mathbf{x}) = \mathbf{h}^{(L)}$$

Demo: MNIST digit classification

MNIST

- Popular dataset of handwritten digits
- 60,000 training examples
- 10,000 test examples
- 10 classes (digits 0-9)
- http://yann.lecun.com/exdb/mnist/
- 28x28 grayscale images (784D)



Objective

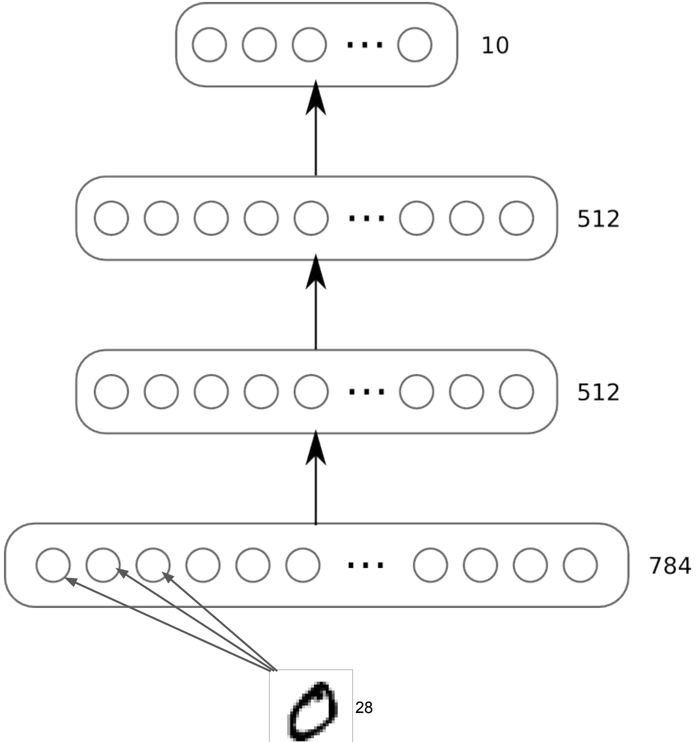
- Learn a function y = f(x) that predicts the digit from the image
- Measure accuracy on test set

Multilayer perceptrons - MLPs

How many parameters contains the following MLP ?

Model

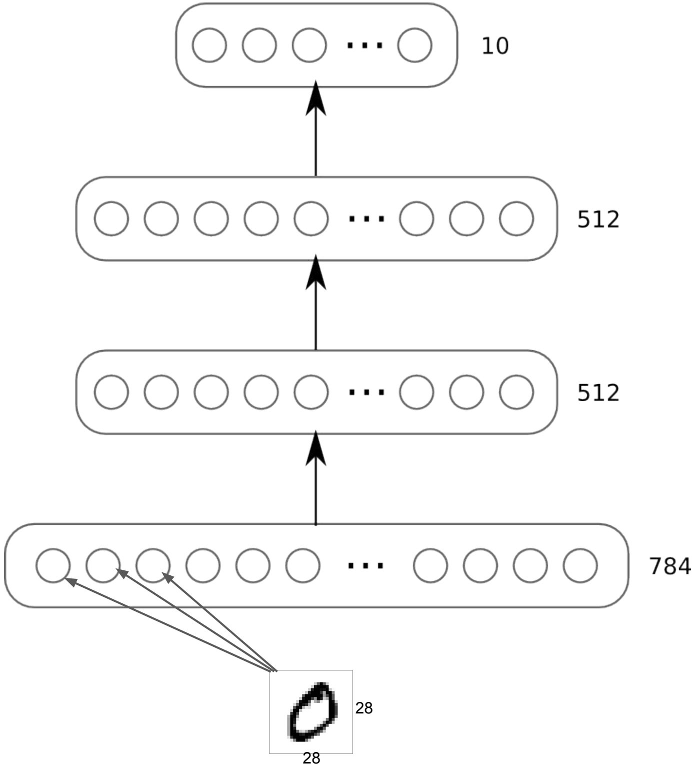
- 3 layer neural network (2 hidden layers)
- Tanh units (activation function)
- 512-512-10
- **Softmax** on top layer



Multilayer perceptrons - MLPs

How many parameters contains the following MLP ?

Layer	#Weights	#Biases	Total
1	784 x 512	512	401,920
2	512 x 512	512	262,656
3	512 x 10	10	5,130
			669,706



Multilayer perceptrons - MLPs

Which of the statements below is true?

A. Multi-layer perceptrons and CNNS are the same kind of networks
B. Each node in a given layer is connected to all inputs from the previous layer
C. In multi-layer perceptrons, the hidden layers are connected only to the input layer
D. There are no hidden layers in the Multi-layer perceptron only in deep networks

Multilayer perceptrons - MLPs

Which of the statements below is true?

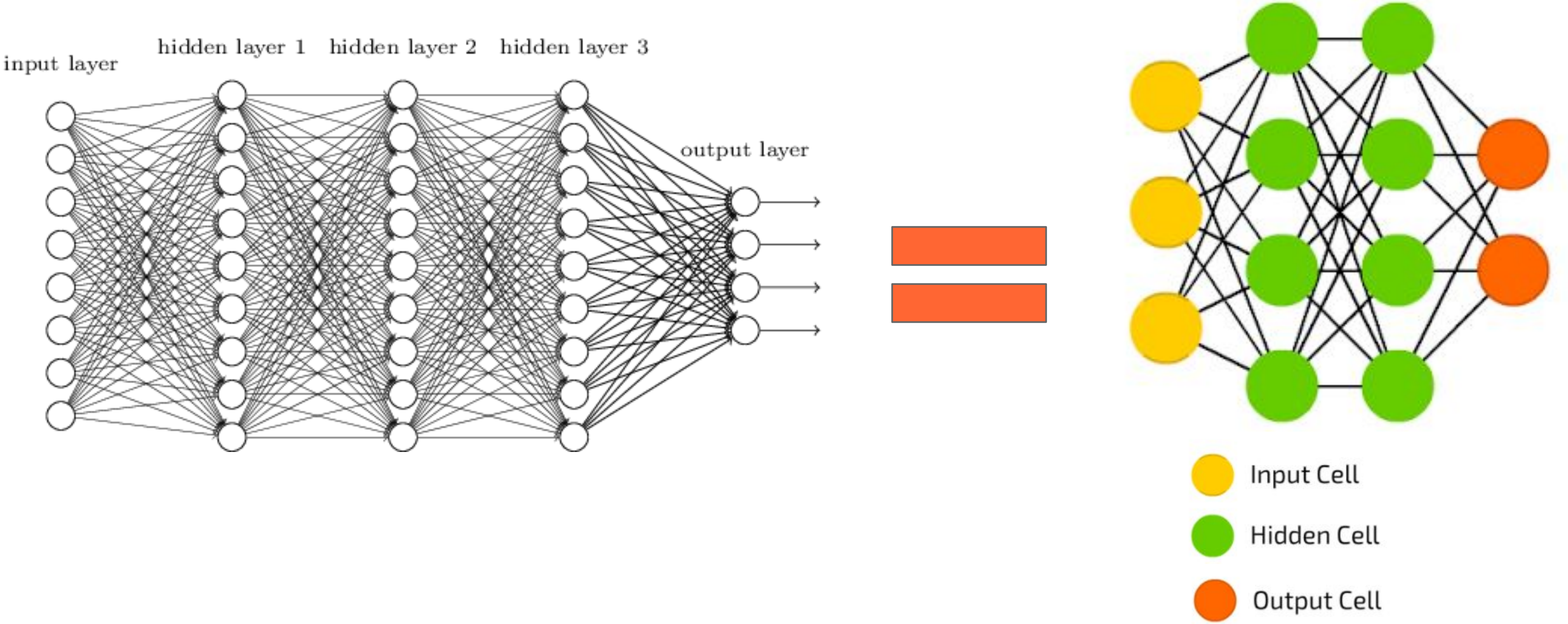
A. Multi-layer perceptrons and CNNS are the same kind of networks

B. Each node in a given layer is connected to all inputs from the previous layer

C. In multi-layer perceptrons, the hidden layers are connected only to the input layer

D. There are no hidden layers in the Multi-layer perceptron only in deep networks

Deep Neural Networks (DNN)



F. Van Veen, [“The Neural Network Zoo”]() (2016)

Universal approximation theorem

Universal approximation theorem states that "the standard multilayer feed-forward network with **a single hidden layer**, which contains **finite number of hidden neurons**, is a **universal approximator** among continuous functions on compact subsets of R^n, under mild assumptions on the activation function."

If a 2 layer NN is a universal approximator, then why do we need **deep nets** ??

The universal approximation theorem:

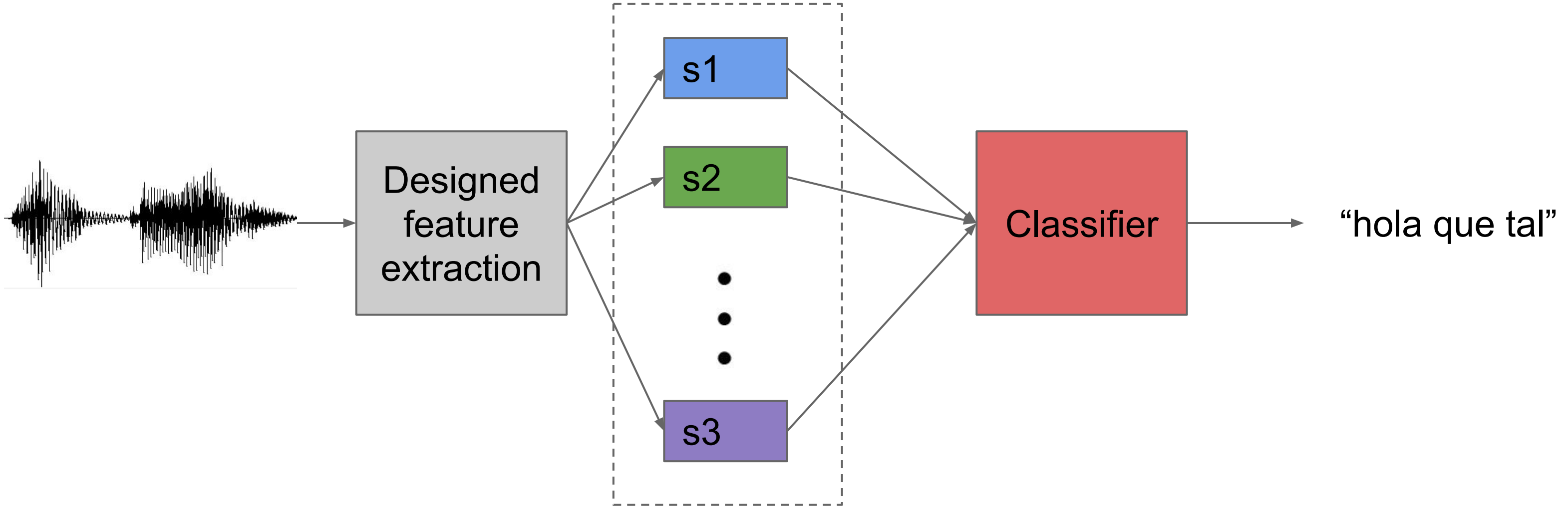
- Says nothing about the how easy/difficult it is to fit such approximators
- Needs a "finite number of hidden neurons": finite may be extremely large

In practice, deep nets can usually represent more complex functions with less total neurons (and therefore, less parameters)

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators." Neural networks 2, no. 5 (1989): 359-366.

Classic Machine Learning...

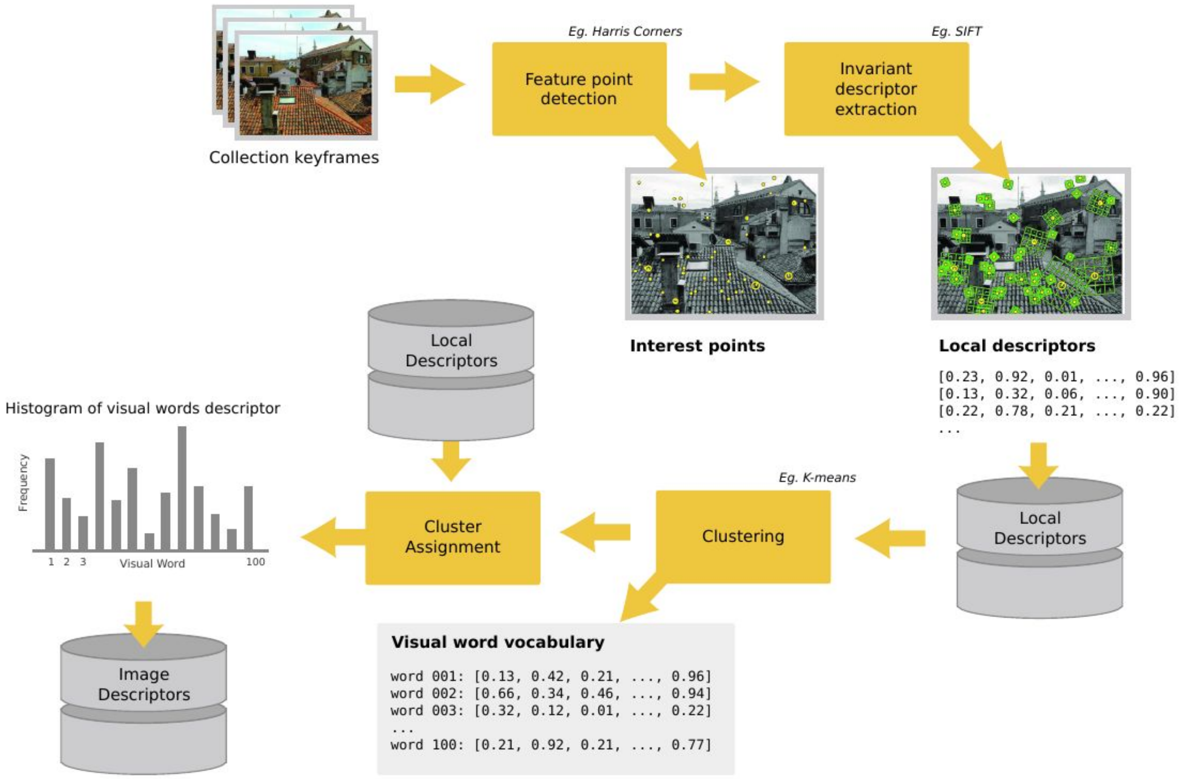
Feature engineering for Automatic Speech Recognition (ASR).



“Hand-crafted” features

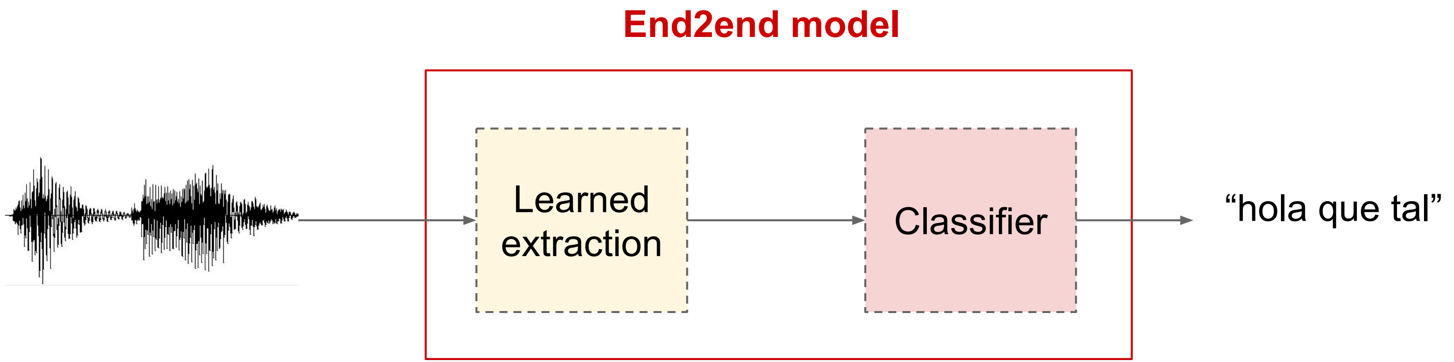
Classic Machine Learning...

Feature engineering for Computer Vision.



Classic Machine Learning vs Deep Learning

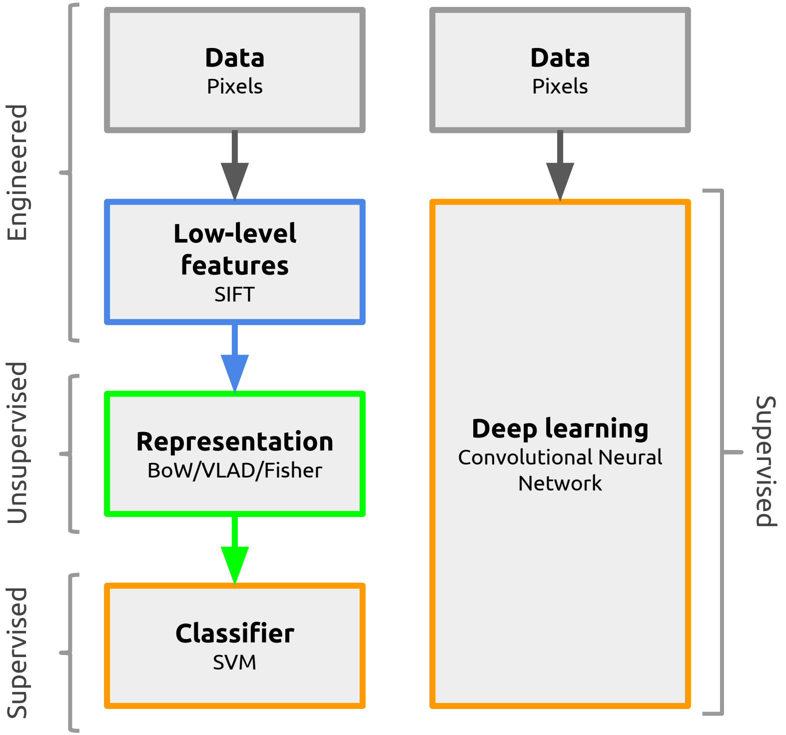
Learn the representations as well, not only the final mapping → **end2end**



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

Classic Machine Learning vs Deep Learning

- Old style machine learning:
  - Engineer features (by some unspecified method)
  - Create a representation (descriptor)
  - Train shallow classifier on representation
- Example:
  - SIFT features (engineered)
  - BoW representation (engineered + unsupervised learning)
  - SVM classifier (convex optimization)
- **Deep learning**
  - Learn layers of features, representation, and classifier in one go based on the data alone
  - Primary methodology: deep neural networks (non-convex)



Undergradese

What undergrads ask vs. what they're REALLY asking