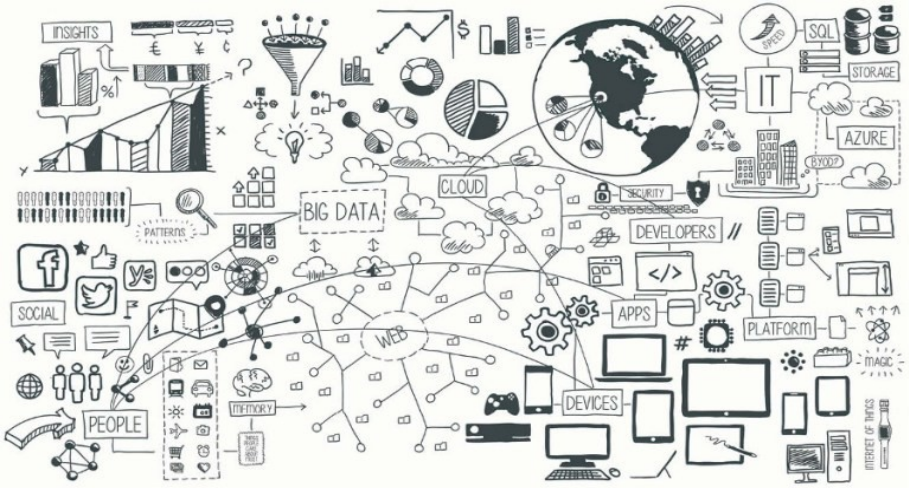
Machine Learning I
Neural Networks

INTRODUCTION AND HISTORICAL PERSPECTIVE



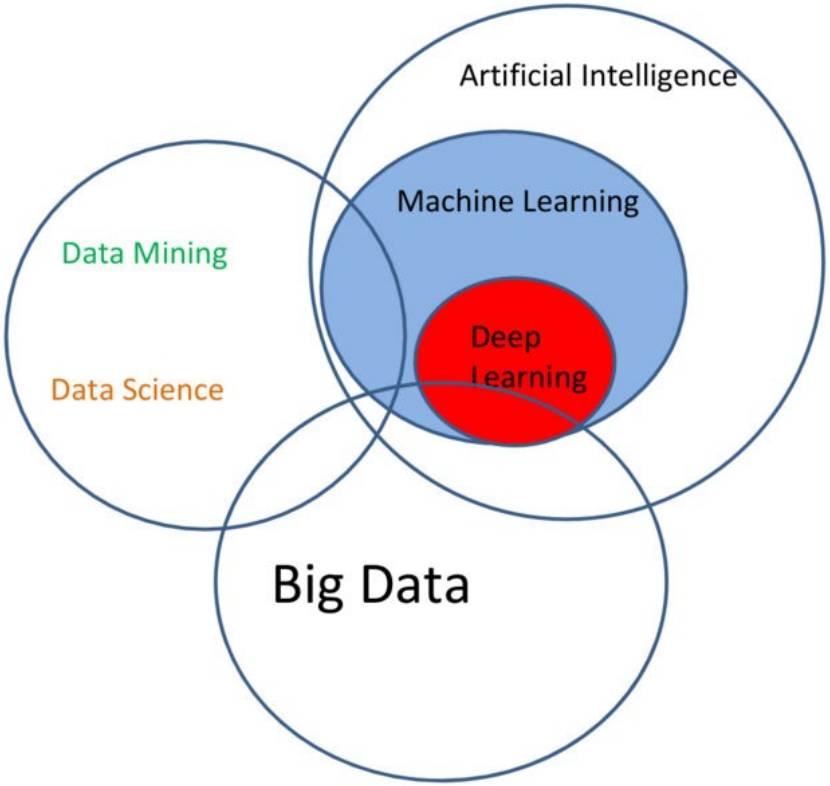
José Manuel Gutiérrez
Javier Díez Sierra
Jose González Abad
(IFCA)

Lara Lloret
Ignacio Heredia
(IFCA)

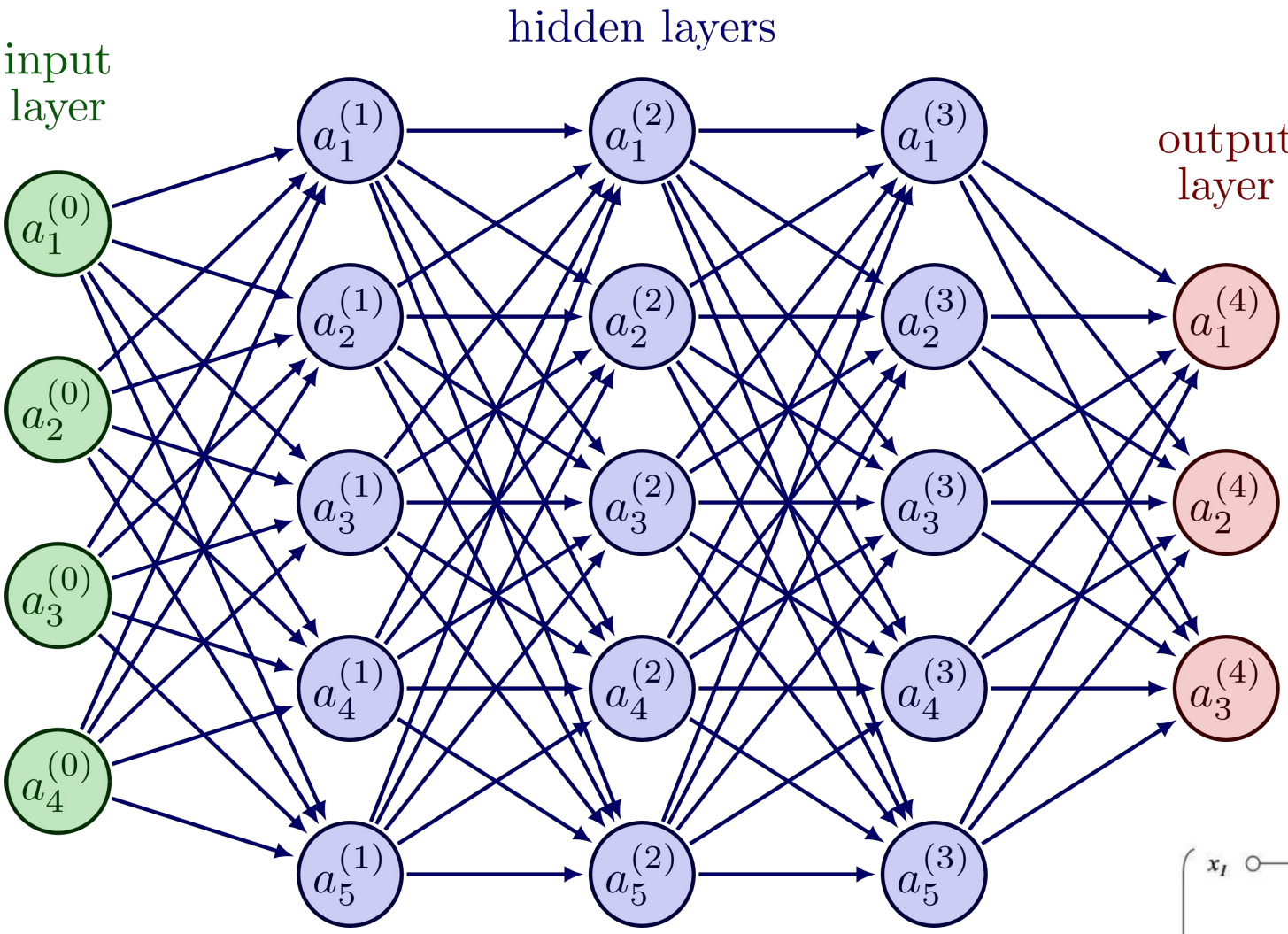
Dpto. Matemática Aplicada y Ciencias de la Computación
Instituto de Física de Cantabria

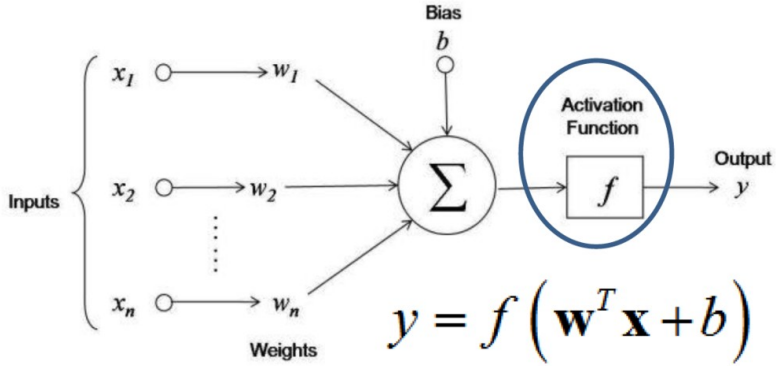
Martes y Jueves de 15:30 a 17:30

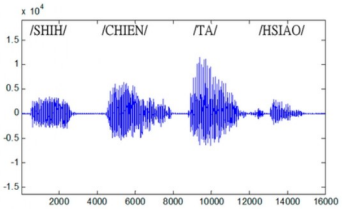
3 Trabajos: 60%. Examen (test): 40%



Título del tema	Día	Profesor	Tipo
Fundamentos de Redes Neuronales	04/02/2025	José Manuel Gutiérrez y Javier Díez Sierra	Teoría
Aprendizaje en redes neuronales I	06/02/2025	Javier Díez Sierra	Teoría
Aprendizaje en redes neuronales II	11/02/2025	Javier Díez Sierra	Teoría
Frameworks para redes neuronales I (Keras)	13/02/2025	Javier Díez Sierra	Práctica
Frameworks para redes neuronales II (Keras)	18/02/2025	Javier Díez Sierra	Práctica
Fundamentos de Deep Learning	20/02/2025	Lara Lloret Iglesias	Teoría
Redes de convolución	25/02/2025	Lara Lloret Iglesias	Teoría
Redes de convolución (Keras)	27/02/2025	Jose González Abad	Práctica
Autoencoders	04/03/2025	Ignacio Heredia	Teoría
Difusión (PyTorch)	06/03/2025	Jose González Abad	Teoría
Difusión / Redes Recurrentes (PyTorch)	11/03/2025	Jose González Abad y Lara Lloret Iglesias	Práctica
Transformers | RL | LLMs	13/03/2025	Ignacio Heredia	Teoría
Transformers | RL | LLMs	18/03/2025	Jose González Abad	Teoría
Examen	20/03/2025	Lara Lloret Iglesias	Teoría
New trends in medical images with AI	21/03/2025	Wilson Silva	Teoría
New trends in medical images with AI	24/03/2025	Wilson Silva	Teoría

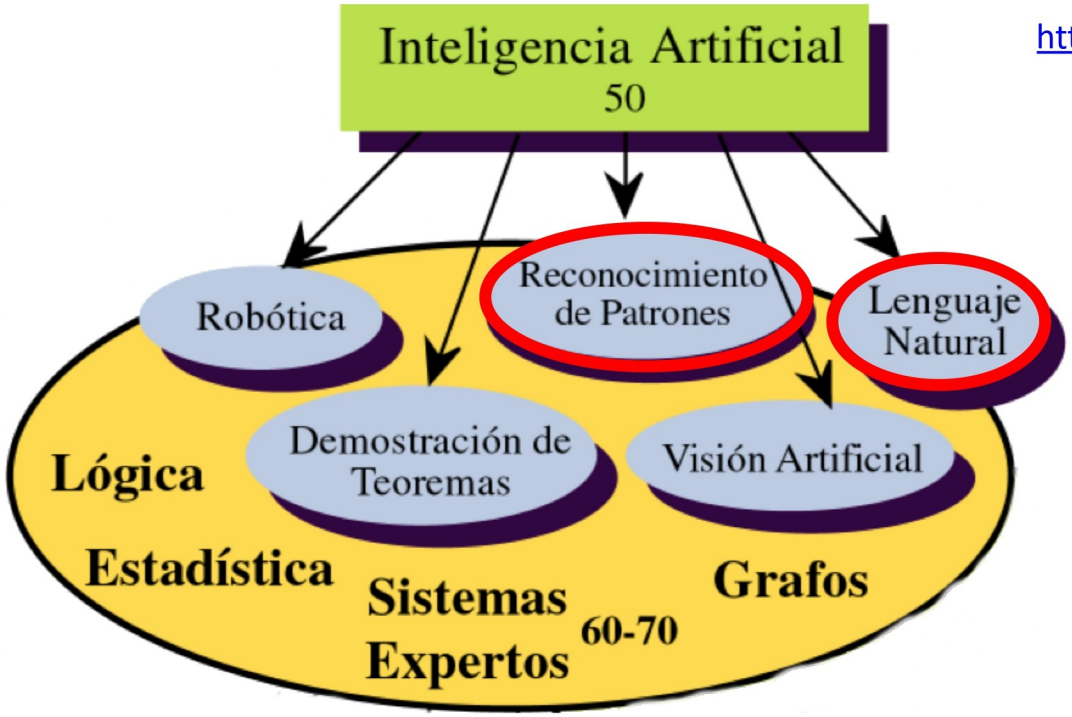

hidden layers
input layer
output layer
$a_1^{(0)}$
$a_2^{(0)}$
$a_3^{(0)}$
$a_4^{(0)}$
$a_1^{(1)}$
$a_2^{(1)}$
$a_3^{(1)}$
$a_4^{(1)}$
$a_5^{(1)}$
$a_1^{(2)}$
$a_2^{(2)}$
$a_3^{(2)}$
$a_4^{(2)}$
$a_5^{(2)}$
$a_1^{(3)}$
$a_2^{(3)}$
$a_3^{(3)}$
$a_4^{(3)}$
$a_5^{(3)}$
$a_1^{(4)}$
$a_2^{(4)}$
$a_3^{(4)}$


Bias
b
x_1
x_2
x_n
w_1
w_2
w_n
Inputs
Weights
$\sum$
Activation Function
f
Output
y
$y = f\left(\mathbf{w}^T\mathbf{x} + b\right)$



R + openNLP

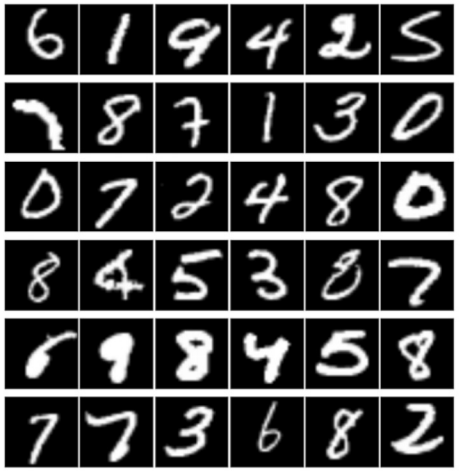
Overview of Natural Language Processing(NLP) with R and OpenNLP



http://yann.lecun.com/exdb/mnist/

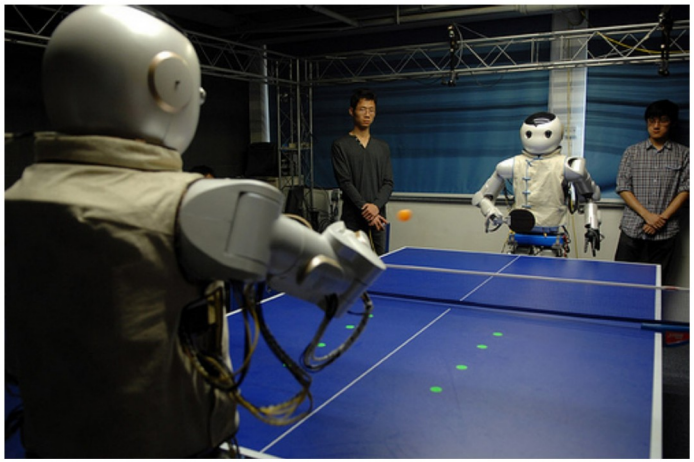
60000+10000 images 28x28

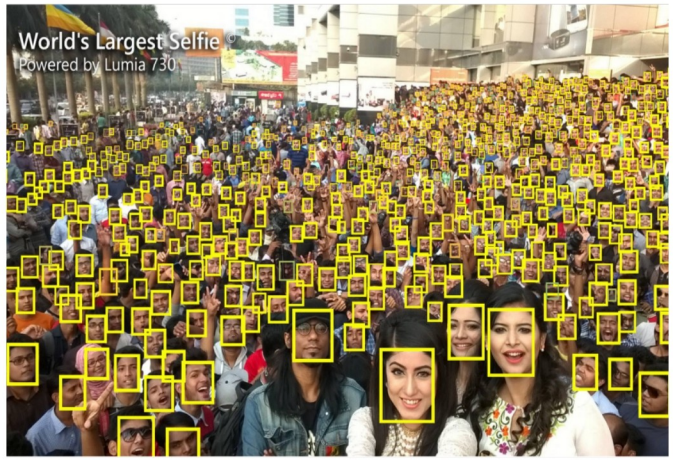
Labeled as {0,…,9}



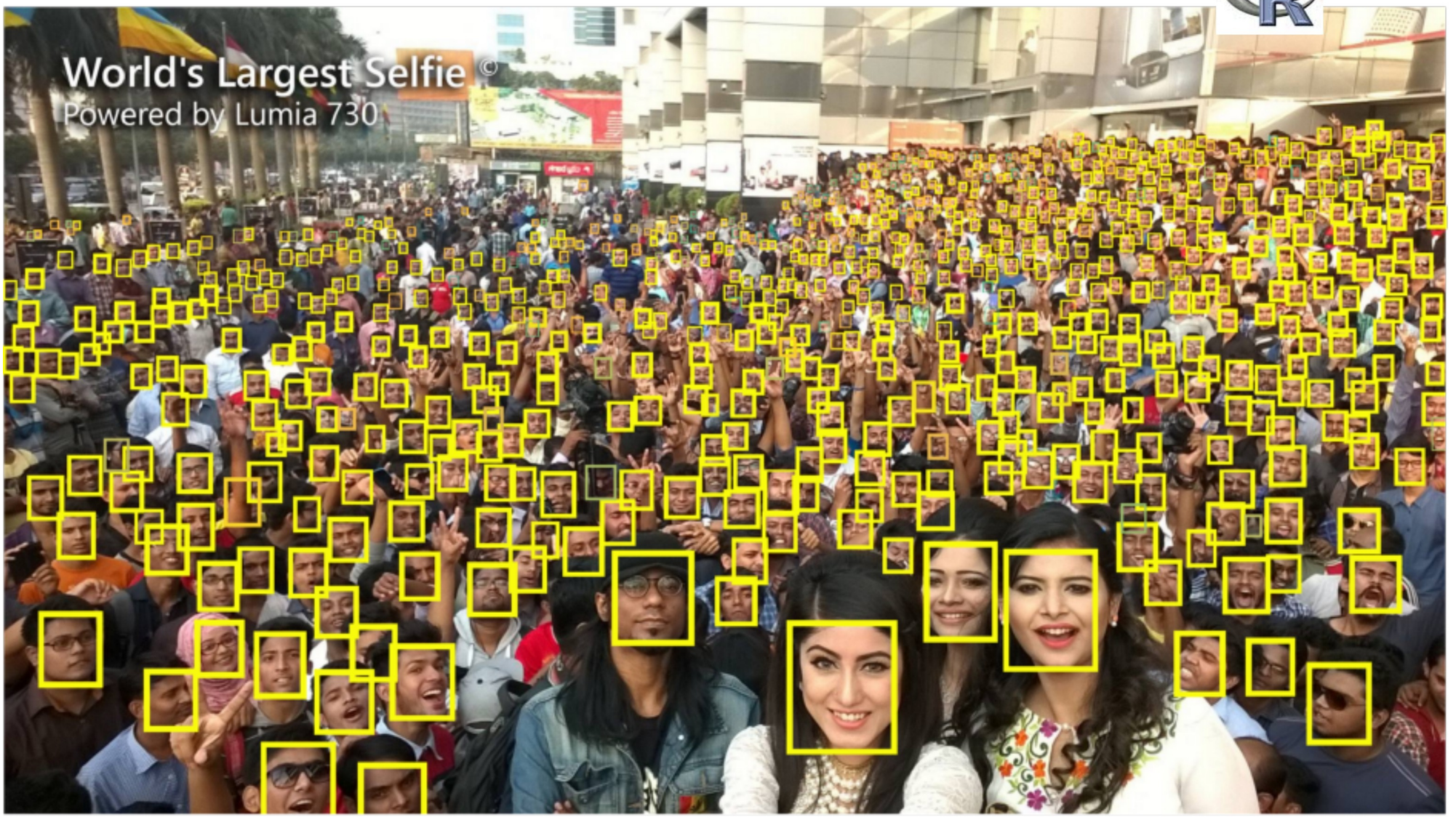
Lineal: 10%. k-NN: 3%. SVM: 1%.

Deep: 0.3%





https://github.com/peiyunh/tiny



We develop a face detector (Tiny Face Detector) that can find ~800 faces out of ~1000 reportedly present, by making use of novel characterization of scale, resolution, and context to find small objects.

EL PAÍS

Educación

INFANTIL Y PRIMARIA · SECUNDARIA, BACHILLERATO Y FP · UNIVERSIDADES · ÚLTIMAS NOTICIAS

TECNOLOGÍA >

La inteligencia artificial aprueba por los pelos la Selectividad

EL PAÍS somete a ChatGPT a un ejercicio real de Historia de las pruebas de acceso a la Universidad y pide a tres docentes con experiencia como correctores de la EVAU que lo revisen

http://chat.openai.com

nature

Search Log in

Explore content · About the journal · Publish with us

nature > news > article

NEWS | 12 January 2023

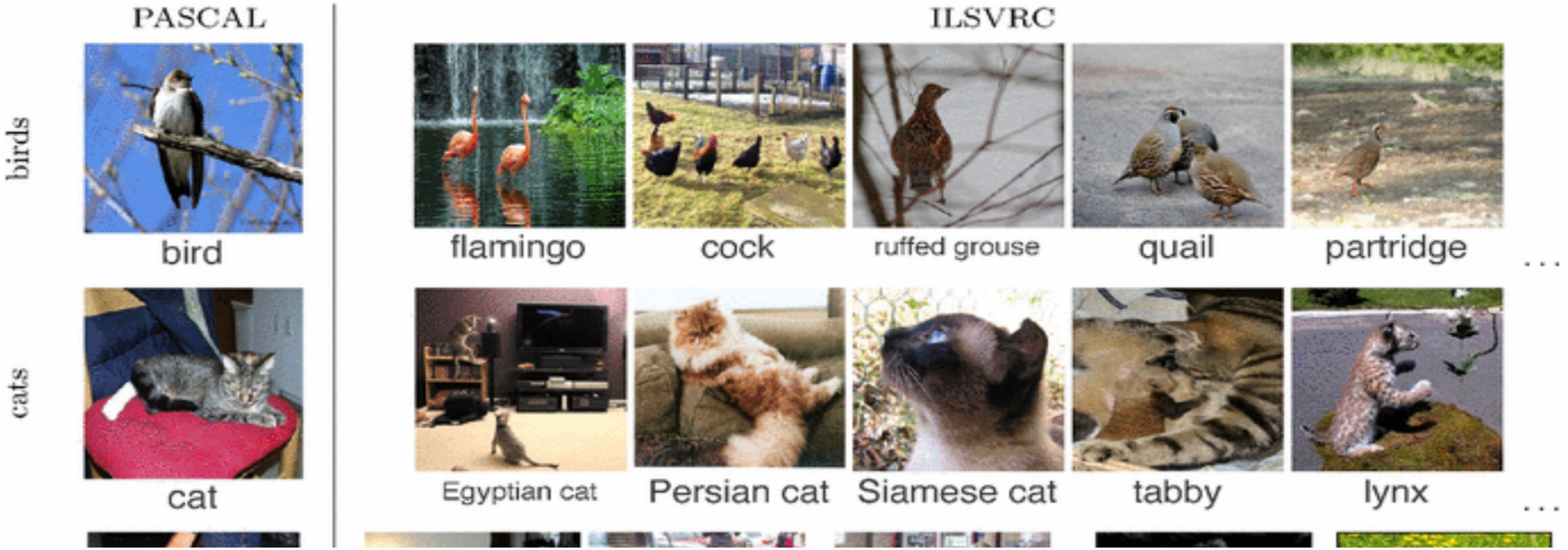
Abstracts written by ChatGPT fool scientists

Researchers cannot always differentiate between AI-generated and original abstracts.

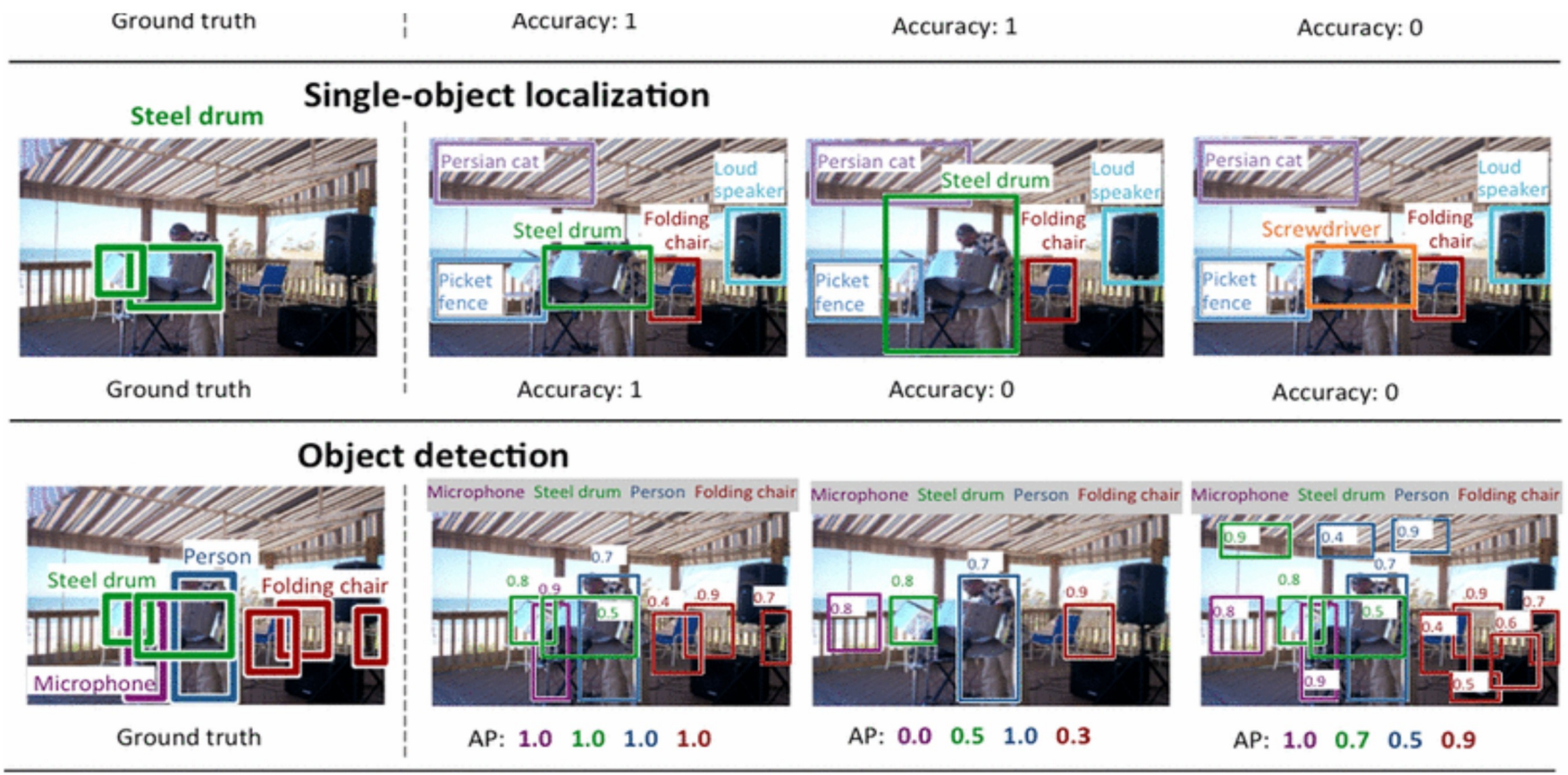
ImageNet is an image database organized according to the (nouns of the) WordNet hierarchy, in which each node of the hierarchy is depicted by an average of over five hundred images.

#synsets: 21841
#images: 14197122

150 GB [kaggle]



David G. Lowe, **Distinctive Image Features from Scale-Invariant Keypoints**. *International Journal of Computer Vision, 2004.*



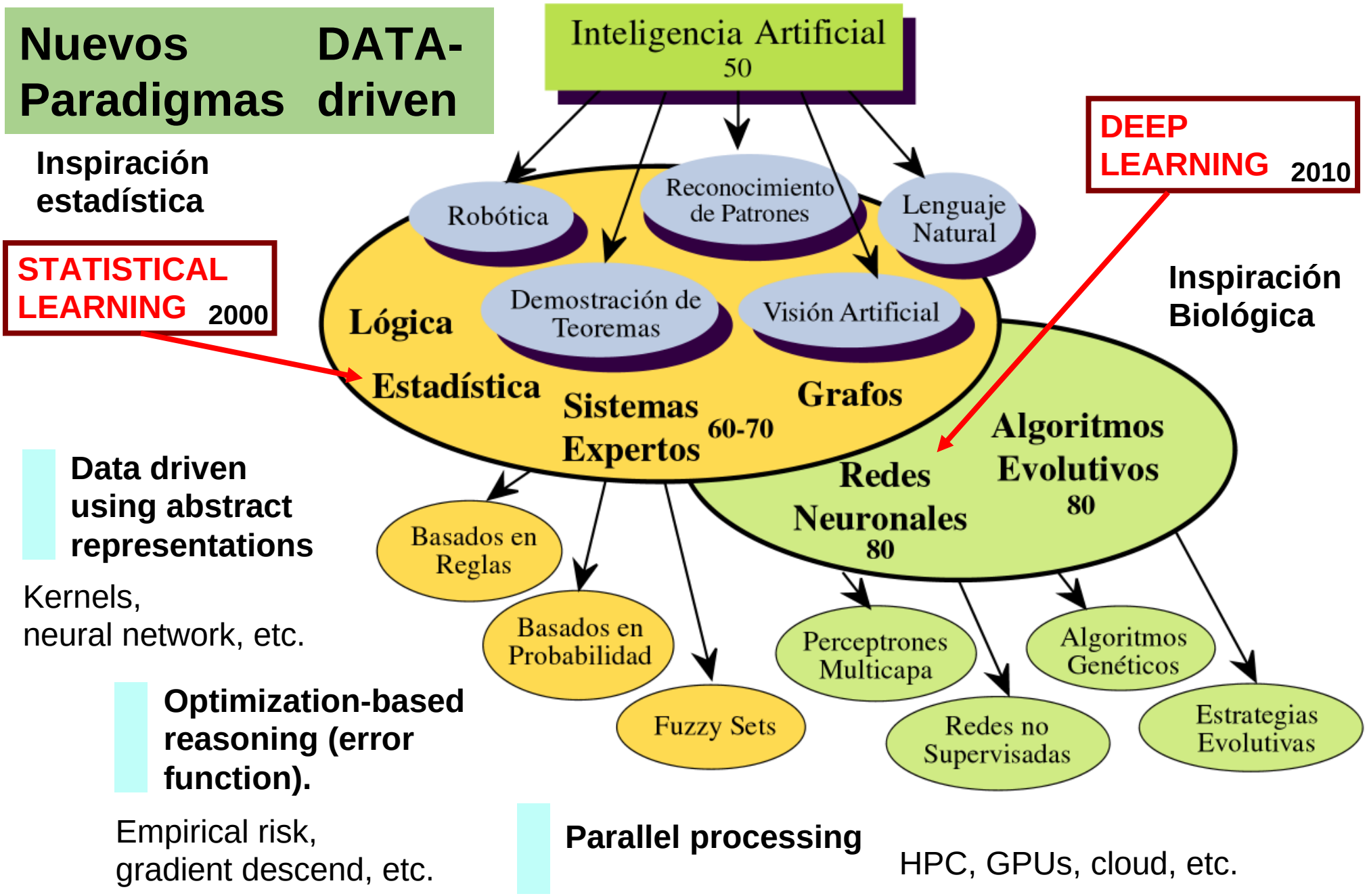
Validation: top-5 error rate

2017 video included

[Inception-v3](): 3.46% top-5 and 17.3% top-1 (25 million parameters). [Inception In [kaggle]()]

O. Russakovsk (2015) [ImageNet Large Scale Visual Recognition Challenge](), International Journal of Computer Vision, 115, 211–252

Nuevos Paradigmas DATA-driven



Inspiración estadística

STATISTICAL LEARNING 2000

DEEP LEARNING 2010

Inspiración Biológica

Data driven using abstract representations

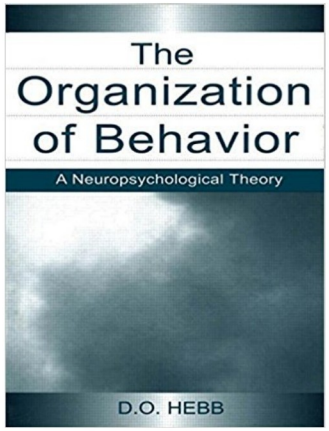
Kernels, neural network, etc.

Optimization-based reasoning (error function).

Empirical risk, gradient descend, etc.

Parallel processing

HPC, GPUs, cloud, etc.



Altmetric: 80 Citations: 6342 More detail »

Letter

Learning representations by back-propagating errors

David E. Rumelhart, Geoffrey E. Hinton & Ronald J. Williams

Nature **323**, 533–536 (09 October 1986)
doi:10.1038/323533a0
Download Citation

Received: 01 May 1986
Accepted: 31 July 1986
Published online: 09 October 1986

Altmetric: 795 Citations: 2124

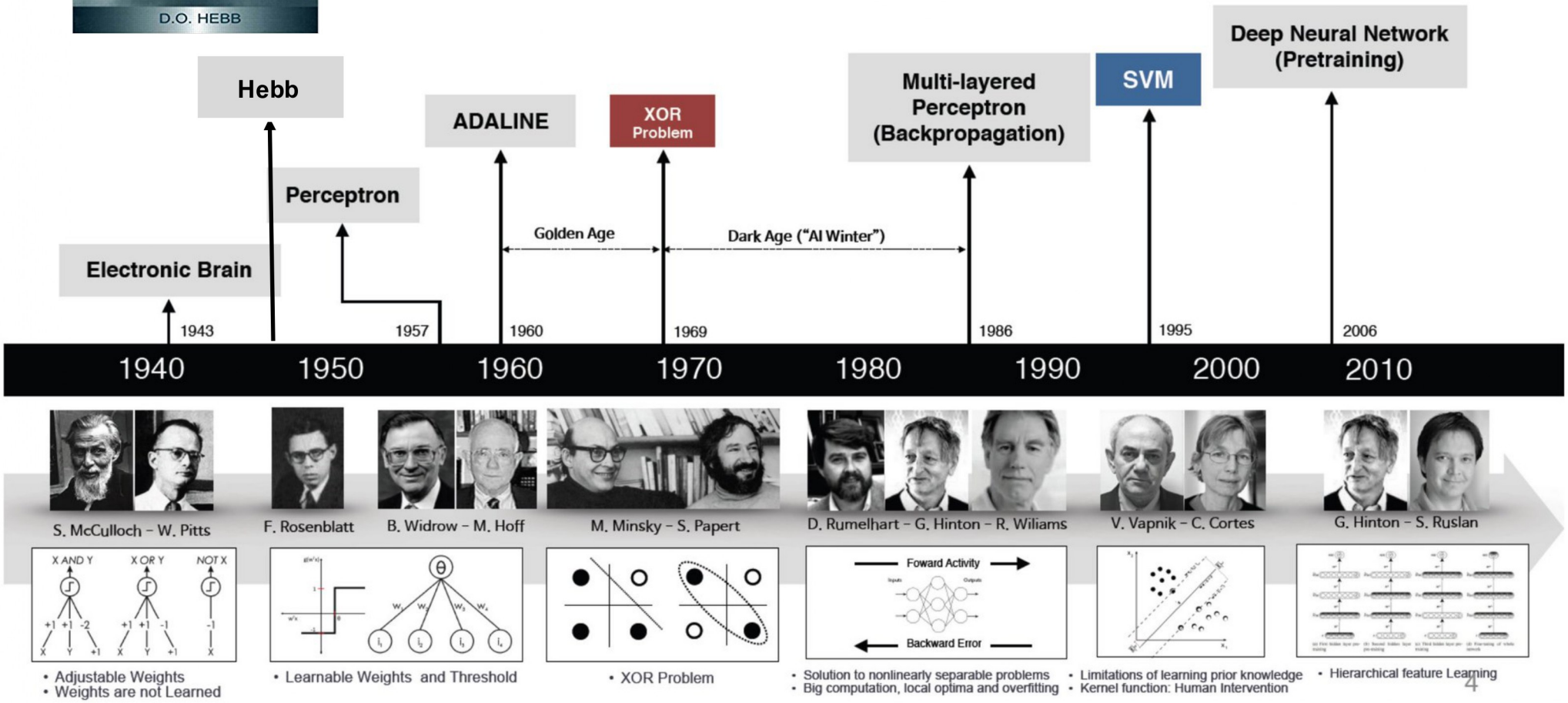
Review

Deep learning

Yann LeCun, Yoshua Bengio & Geoffrey Hinton

Nature **521**, 436–444 (28 May 2015)
doi:10.1038/nature14539
Download Citation

Received: 25 February 2015
Accepted: 01 May 2015
Published online: 27 May 2015



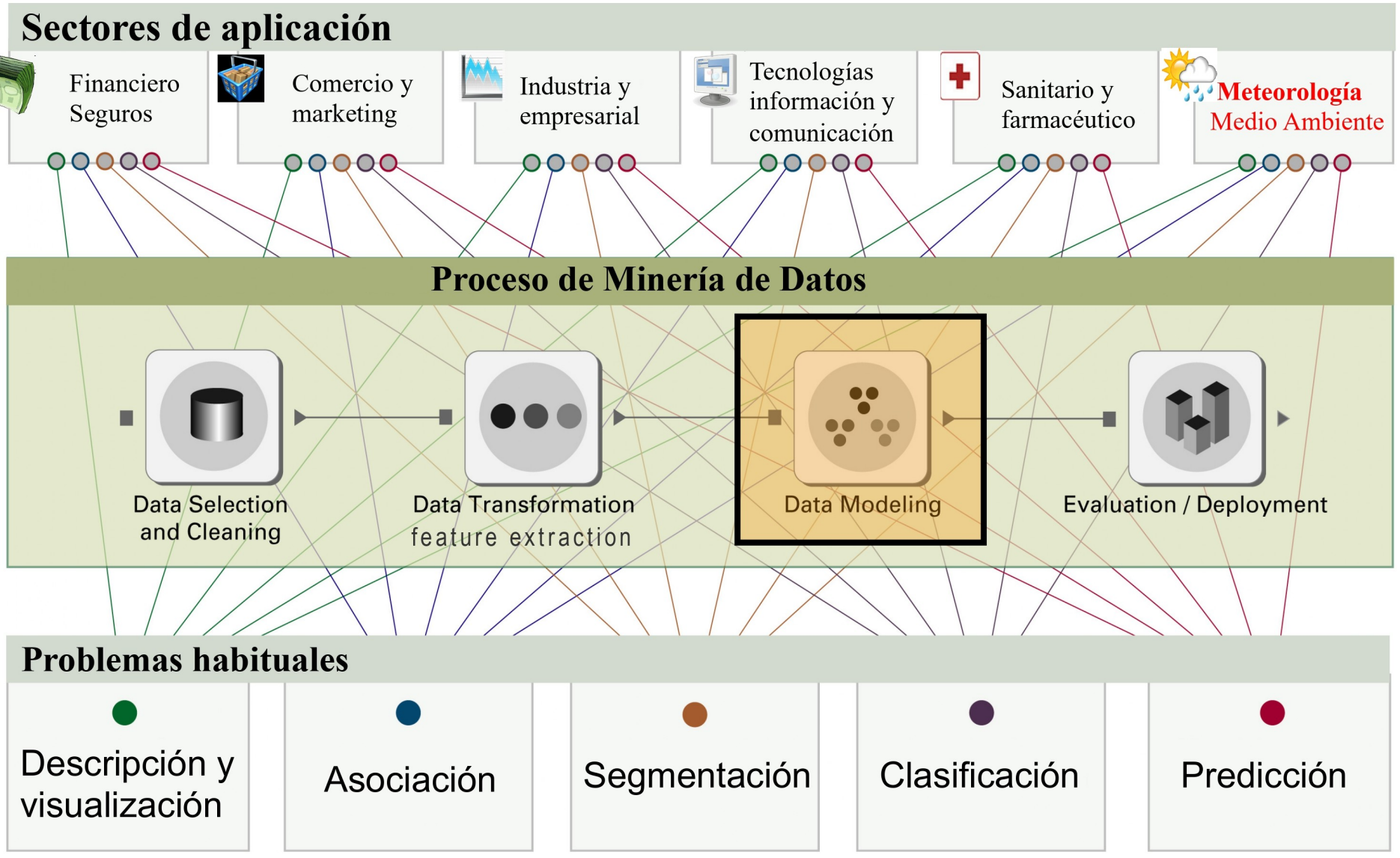
Neural Network Study (1988, AFCEA International Press, p. 60):

*... a neural network is a system composed of **many simple processing elements operating in parallel** whose function is determined by network structure, connection strengths, and the processing performed at computing elements or nodes.*

Haykin, S. (1994), Neural Networks: A Comprehen- sive Foundation, NY: Macmillan, p. 2:

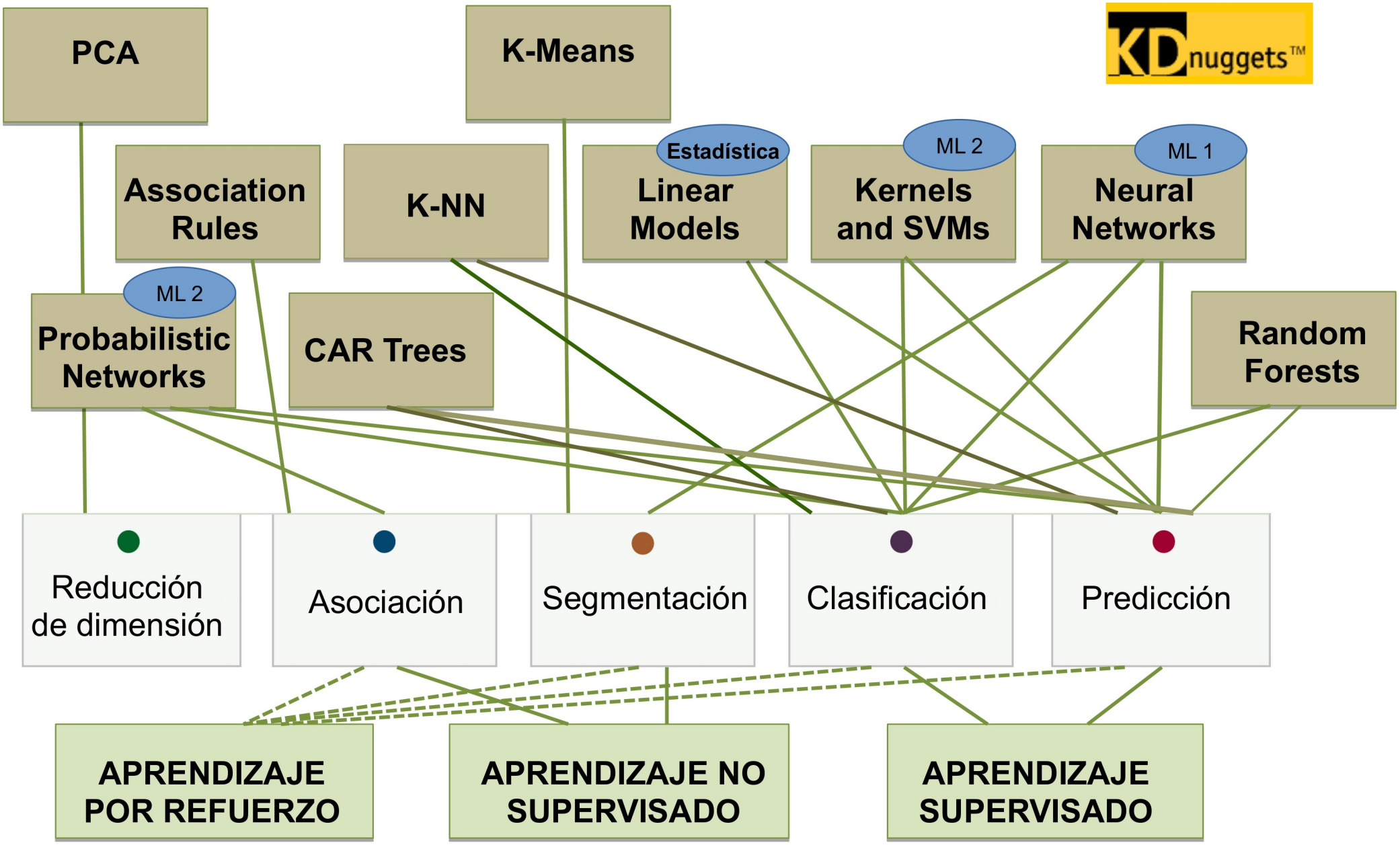
A neural network is a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:

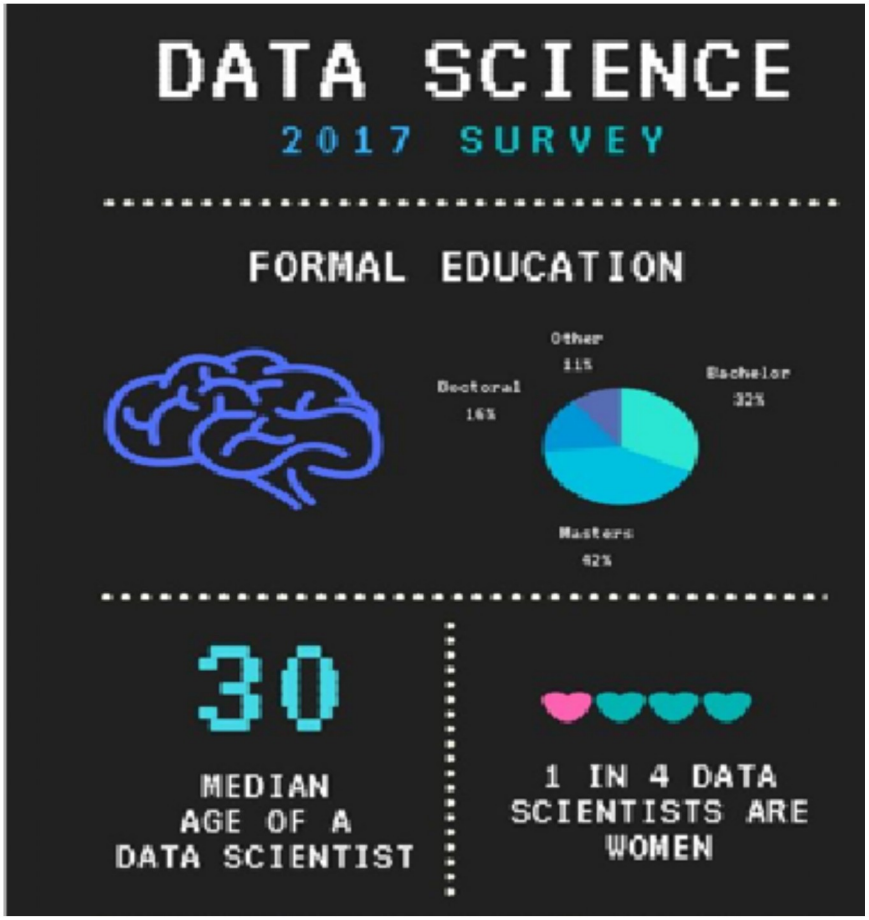
1. ***Knowledge is acquired through a learning process.***
2. ***Neuron weights are used to store the knowledge.***

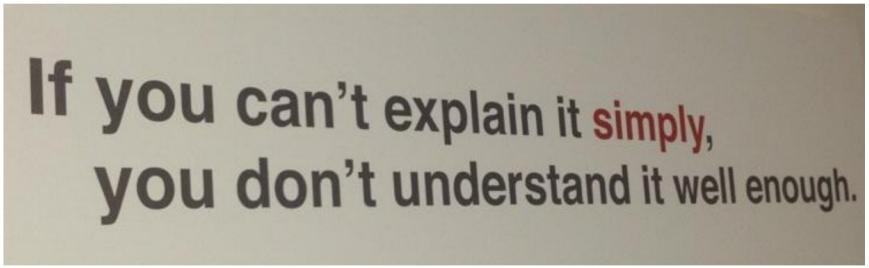


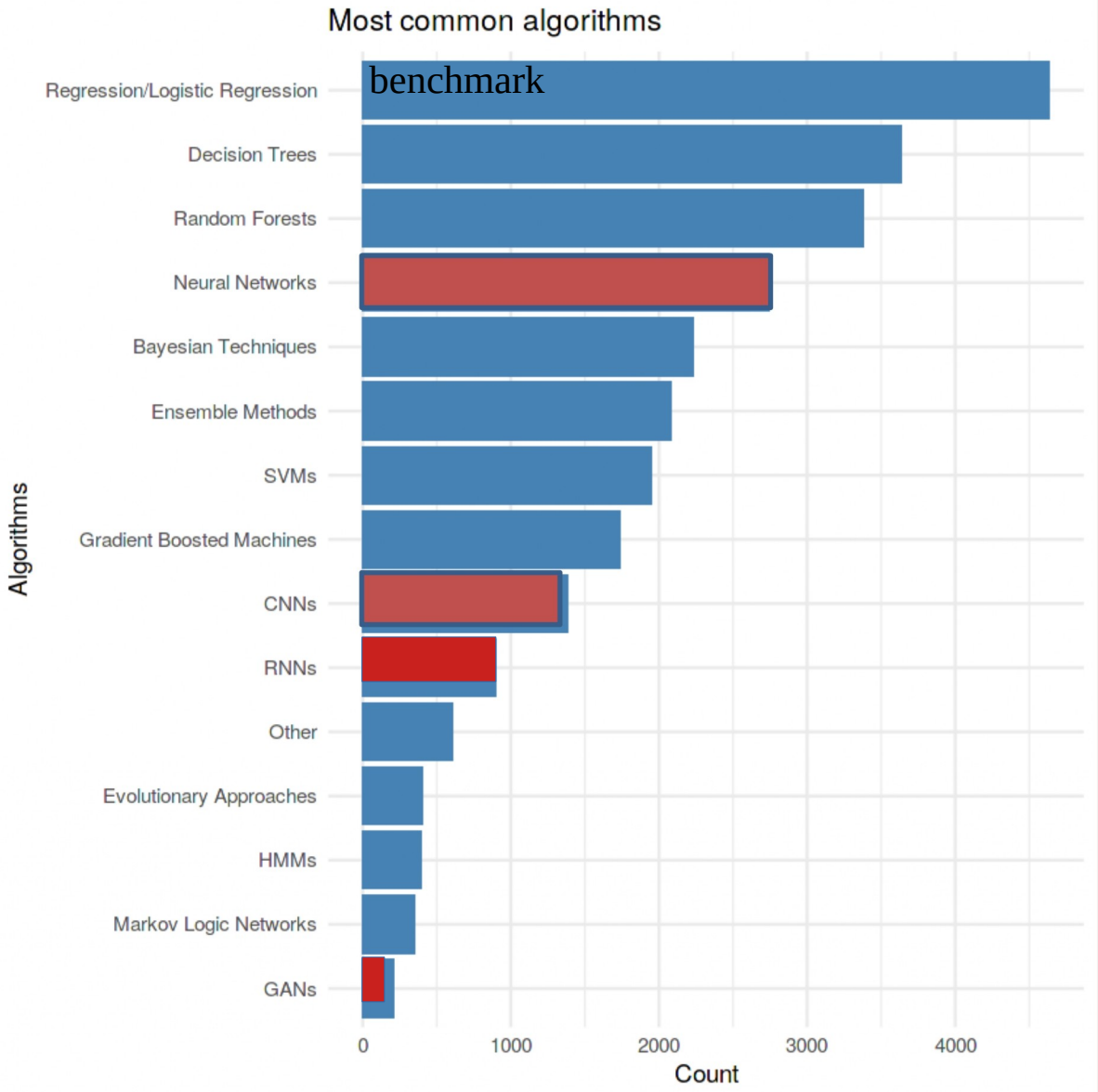
Machine learning develop methods for data modelling and prognosis.

https://www.kdnuggets.com/2017/10/top-10-machine-learning-algorithms-beginners.html

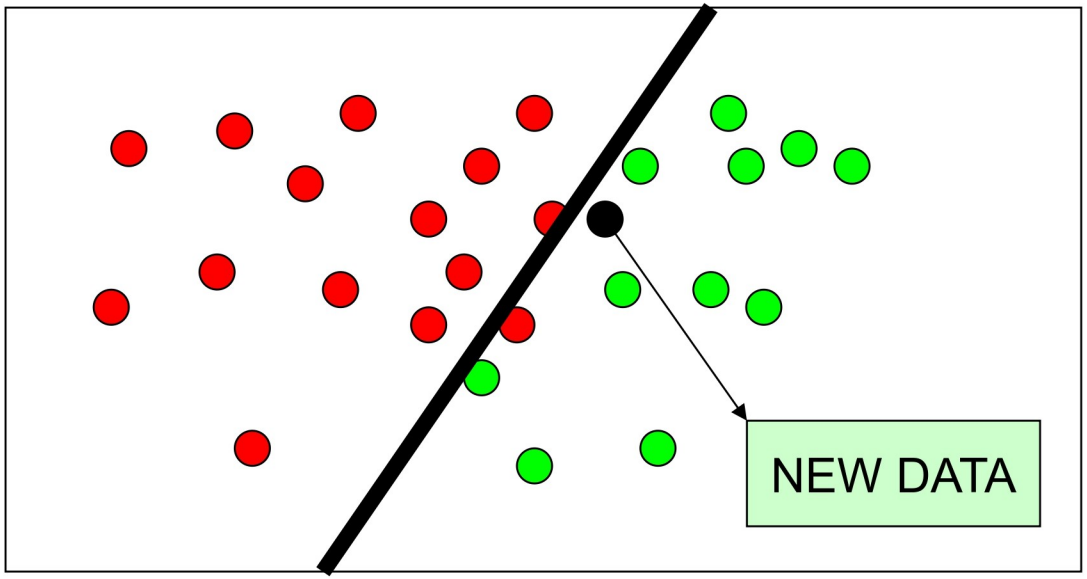




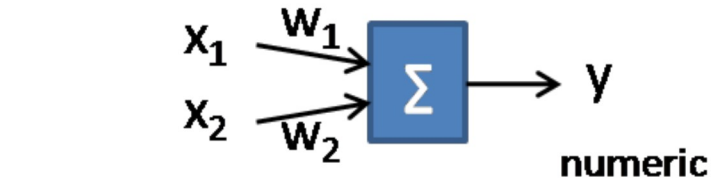




https://www.kaggle.com/kaggle/kaggle-survey-2017

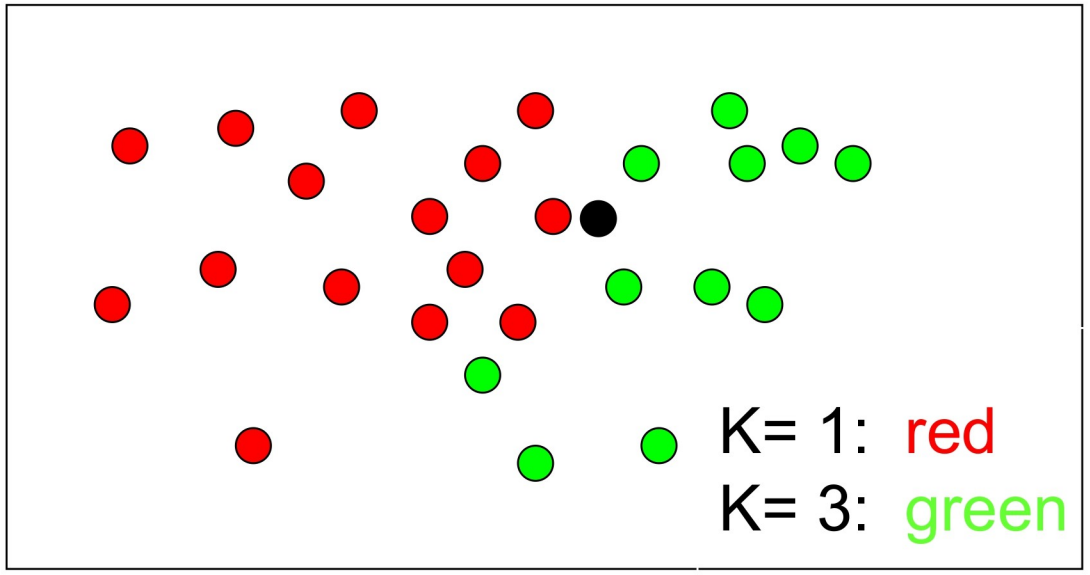


GENERATIVE METHODS:
Linear models are the simplest family for machine learning and have good generalization properties.



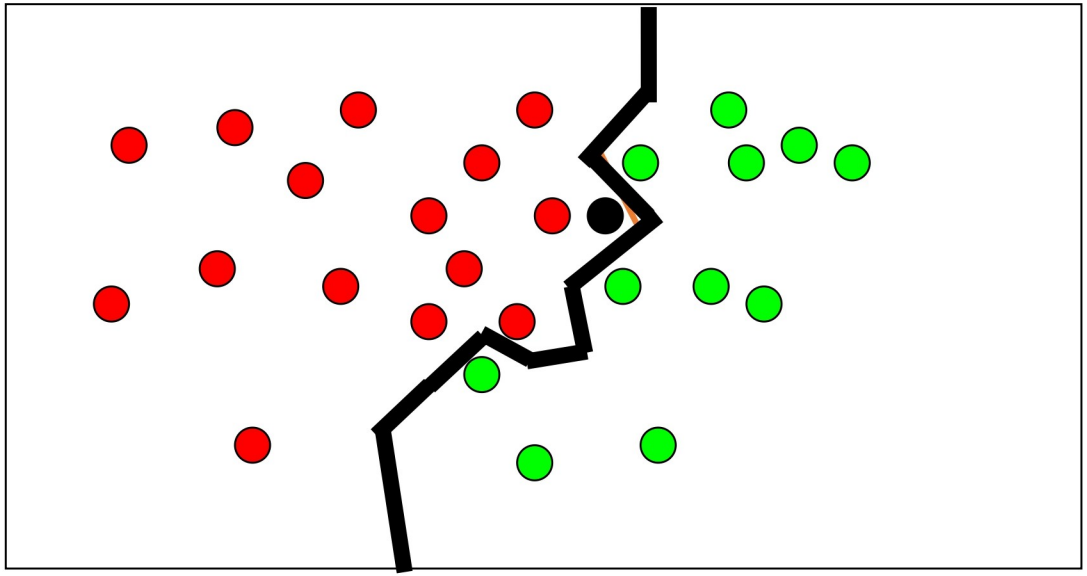
$$y = w_0 + w_1x_1 + w_2x_2$$

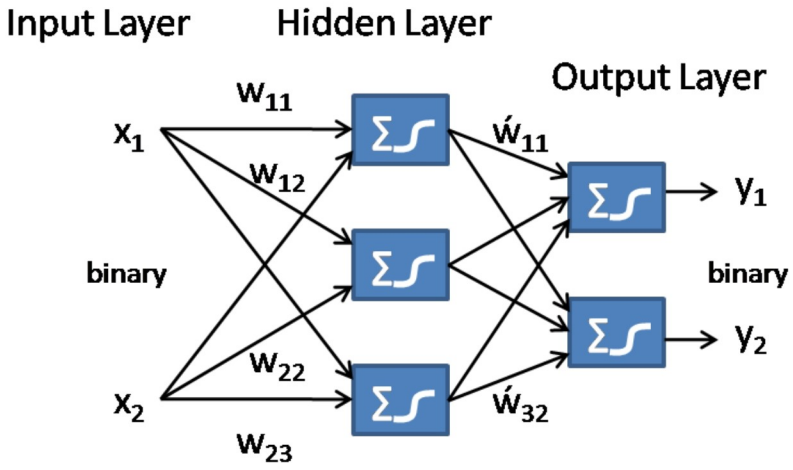
$$y = f(\mathbf{X},\mathbf{W}) = \mathbf{X}^T.\mathbf{W}$$



NON-GENERATIVE (OR ALGORITHMIC)
K Nearest Neighbours is the simples non-generative method. It depends on a single parameter (K) to be tunned (generalization depends on K).

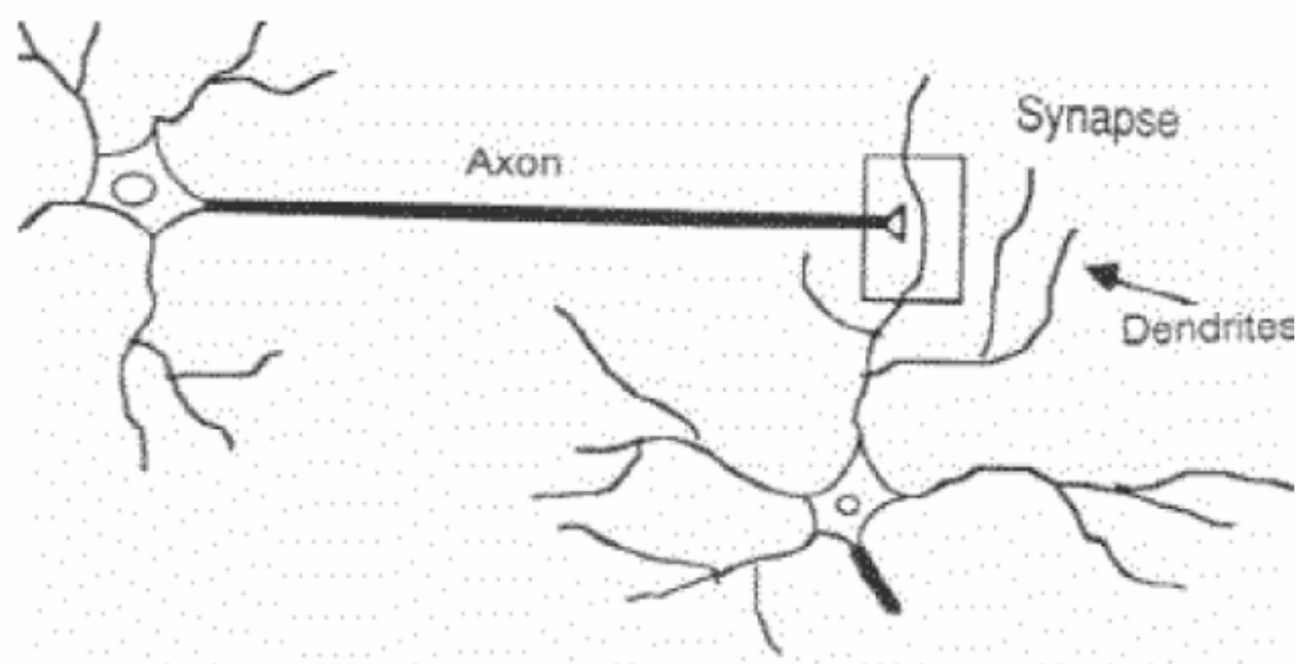
Increasing model complexity (e.g. number of parameters) can result in **overfitting** (lack of generalization).





Artificial Neural Networks are inspired in the structure and functioning of the **brain**, which is a collection of **interconnected neurons** (the simplest computing elements performing information processing):

- ✓Each neuron consists of a cell body, that contains a cell **nucleus**.
- ✓There are number of fibers, called **dendrites**, and a single long fiber called **axon** branching out from the cell body.
- ✓The axon connects one neuron to others (through the dendrites).
- ✓The connecting junction is called **synapse**.



There are over **10^{11} neurons** in a human brain, each **connected with 1000** on average.

- The synapses releases chemical transmitter substances, entering the dendrite, raising or lowering (**excitatory and inhibitory synapses**) the electrical potential of the cell body.
- When the potential **reaches a threshold**, an electric pulse or action potential is sent down to the axon affecting other neurons *(there is a **nonlinear activation**)*.

y = w x + b
Y = f(y)
E = 1/2 \sum (Y_os - Y)^2

y = w x + b, alla prima iterazione i pesi w e il bias b sono aleatori, dopo calcolo la funzione di perdita per valutare l'errore e con questa nelle iterazioni successive si aggiustano i parametri

per aggiustare i parametri si usa il gradient descent:
\Delta w = - \eta dE/dw
\Delta b = - \eta dE/db
dove \eta è un parametro da impostare

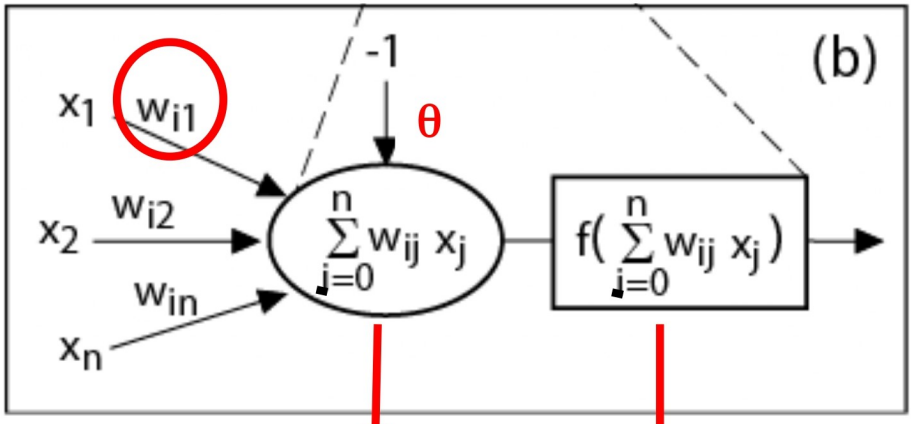
$$y = f(w^T x), \text{ with } x_0 = -1 \text{ to account for } \theta: f(w^T x - \theta).$$

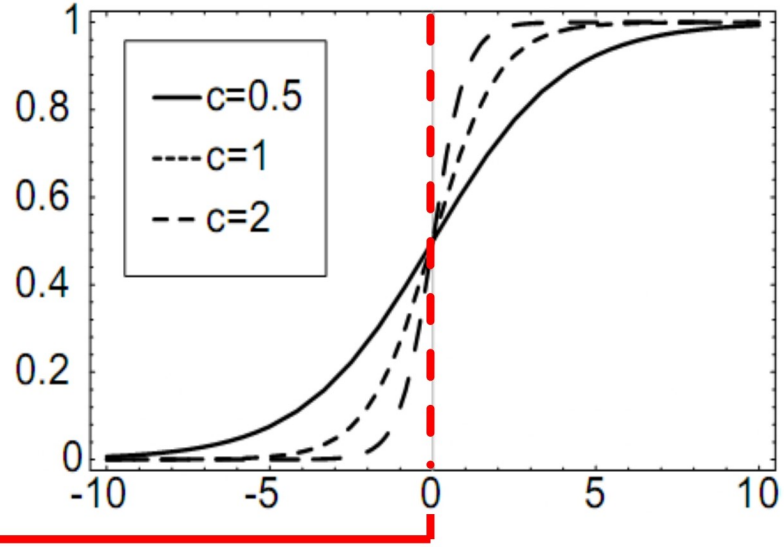
per calcolare le derivate uso il Back Propagation con la regola della catena:
dE/dw = dE/dY dY/dy dy/dw
dE/db = dE/dY dY/dy dy/db

dE/dY = Y - Y_os
dx/dy = f'(y)
dy/dw = x
dy/db =1

McCulloc & Pitts (1943)

weights (+ or -, excitatory or inhibitory)





neuron potential:
mixed input of neighboring neurons

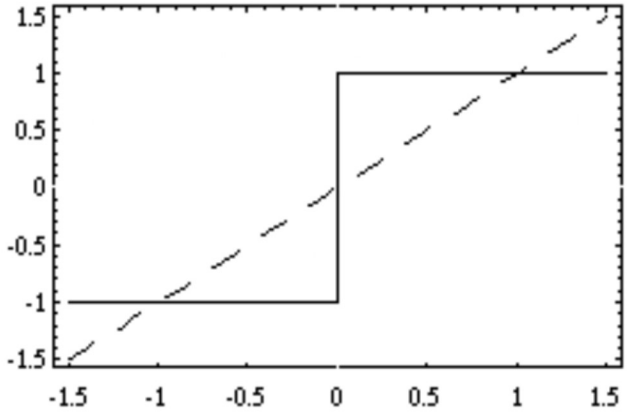
nonlinear activation function

(threshold = θ)

- **Funciones lineales:** $f(x) = x$.

- **Funciones paso:** Dan una salida binaria dependiente de si el valor de entrada está por encima o por debajo del valor umbral.

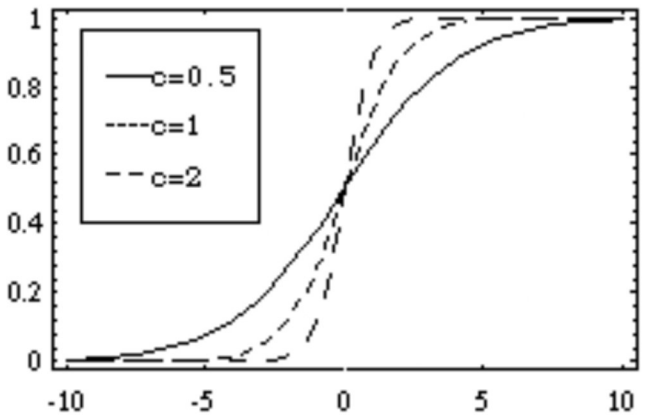
$$sgn(x) = \begin{cases} -1, \text{ si } x < 0, \\ 1, \text{ sino,} \end{cases} \quad , \quad \Theta(x) = \begin{cases} 0, \text{ si } x < 0, \\ 1, \text{ sino.} \end{cases}$$



- **Funciones sigmoidales:** Funciones monótonas acotadas que dan una salida gradual no lineal.

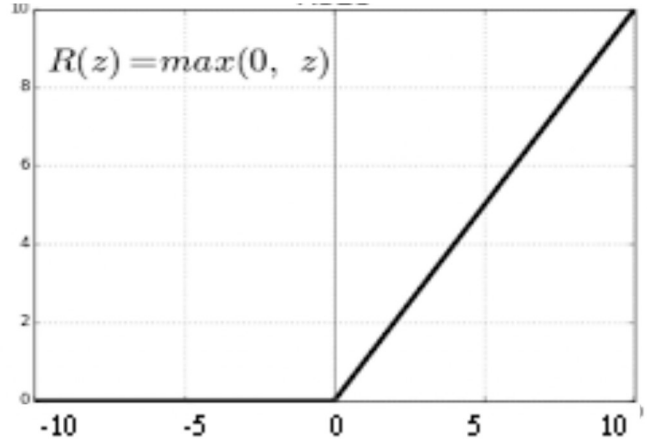
  1. La función logística de 0 a 1:

$$f_c(x) = \frac{1}{1 + e^{-c\,x}}.$$

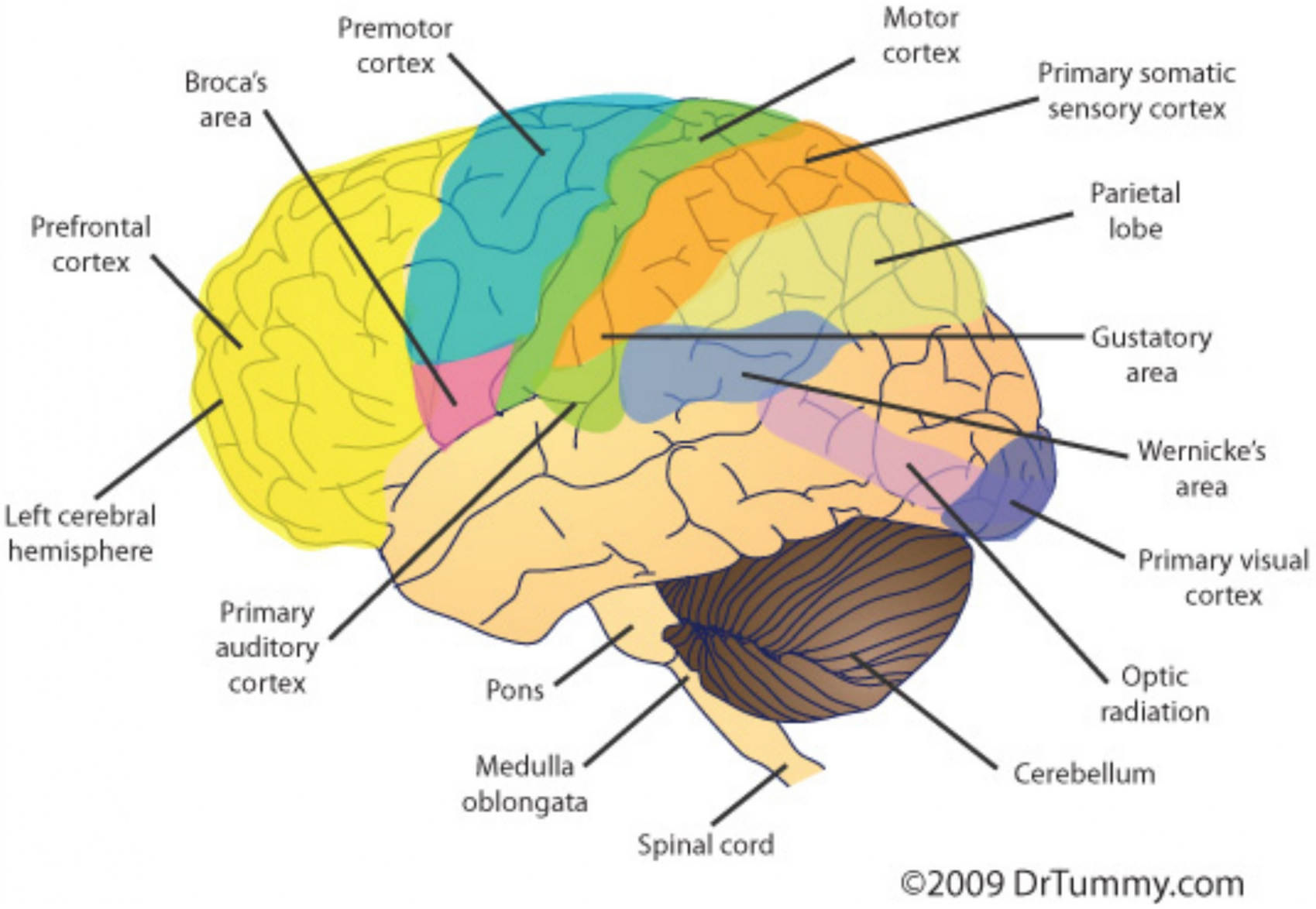


  2. La función tangente hiperbólica de -1 a 1 $f_c(x) = tanh(c\,x)$.

- **Rectified linear unit (ReLU):** Utilizadas para evitar el "desvanecimiento del gradiente".

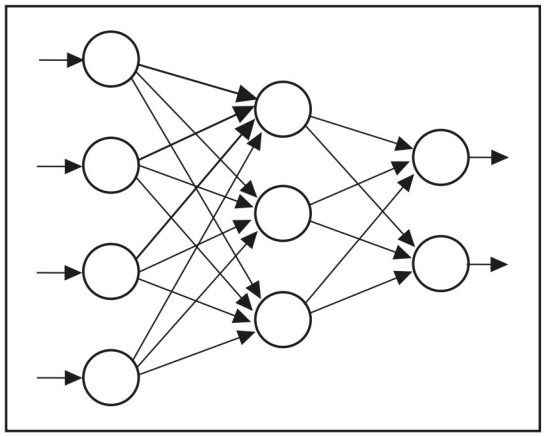


TanH	$f(x) = tanh(x) = \frac{2}{1+e^2x} - 1$	$f'(x) = 1 - f(x)^2$	$(-1, 1)$	C^∞
SoftSign	$f(x) = \frac{x}{1+\|x\|}$	$f'(x) = 1 - f(x)^2$	$(-1, 1)$	C^1
SoftPlus	$f(x) = \ln(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$	$(0, \infty)$	C^∞
SoftExponential	$f(\alpha, x) = \begin{cases} -\frac{\ln(1-\alpha(x+\alpha))}{\alpha} \text{ for } \alpha < 0 \\ x \text{ for } \alpha = 0 \\ \frac{e^{\alpha x}-1}{\alpha} + \alpha \text{ for } \alpha > 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} \frac{1}{1-\alpha(\alpha+x)} \text{ for } \alpha < 0 \\ e^{\alpha x} \text{ for } \alpha \geq 0 \end{cases}$	$(-\infty, \infty)$	C^∞
Sinusoid	$f(x) = \sin(x)$	$f'(x) = \cos(x)$	$[-1, 1]$	C^∞
Sinc	$f(x) = \begin{cases} 1 \text{ for } x = 0 \\ \frac{\sin(x)}{x} \text{ for } x \neq 0 \end{cases}$	$f'(x) = \begin{cases} 0 \text{ for } x = 0 \\ \frac{\cos(x)}{x} - \frac{\sin(x)}{x^2} \text{ for } x \neq 0 \end{cases}$	$[\approx -.217234, 1]$	C^∞
Scaled exponential linear unit (SELU)	$f(\alpha, x) = \lambda \begin{cases} \alpha(e^x - 1) \text{ for } x < 0 \\ x \text{ for } x \geq 0 \end{cases}$ $\lambda = 1.0507$ y $\alpha = 1.67326$	$f'(\alpha, x) = \lambda \begin{cases} f(\alpha, x) + \alpha \text{ for } x < 0 \\ 1 \text{ for } x \geq 0 \end{cases}$	$(-\lambda\alpha, \infty)$	C^0
Rectified linear unit (ReLU)	$f(x) = \begin{cases} 0 \text{ for } x < 0 \\ x \text{ for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 \text{ for } x < 0 \\ 1 \text{ for } x \geq 0 \end{cases}$	$[0, \infty)$	C^0
Randomized leaky rectified linear unit (RReLU)	$f(\alpha, x) = \begin{cases} \alpha x \text{ for } x < 0 \\ x \text{ for } x \geq 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} \alpha \text{ for } x < 0 \\ 1 \text{ for } x \geq 0 \end{cases}$	$(-\infty, \infty)$	C^0
Parametric rectified linear unit (PReLU)	$f(\alpha, x) = \begin{cases} \alpha x \text{ for } x < 0 \\ x \text{ for } x \geq 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} \alpha \text{ for } x < 0 \\ 1 \text{ for } x \geq 0 \end{cases}$	$(-\infty, \infty)$	C^0
Logistic (a.k.a soft step)	$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1 - f(x))$	(0,1)	C^∞


Premotor cortex
Motor cortex
Broca's area
Primary somatic sensory cortex
Parietal lobe
Prefrontal cortex
Gustatory area
Wernicke's area
Left cerebral hemisphere
Primary visual cortex
Primary auditory cortex
Pons
Optic radiation
Medulla oblongata
Cerebellum
Spinal cord
©2009 DrTummy.com

Supervised Problems. **Input-Output pairs are provided:**
(x1,y1), (x2,y2), ..., (xn,yn) and the network learns y = f(x+ε).

***Multilayer* Networks or Feedforward Nets.**
Several layers connected (input+hidden+output)



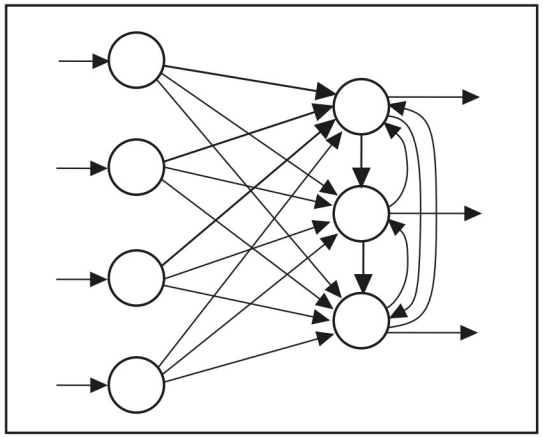
Pattern Recognition
OCR, images
Interpolation and fitting

Prediction: Input => Output

Learning: Backpropagation

Unsupervised Problems. **Only input data is provided:**
x1, x2, ..., xn and the network self-organizes it to provide a clustering.

***Competitive* Networks**
Multilayer networks with lateral connections (competitive) in the last layer.



Segmentation

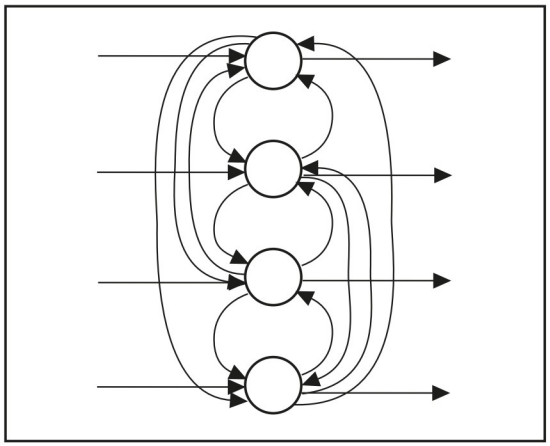
Feature extraction.

Prediction: Input => Clusters

Learning: Ad hoc
Winner-takes-all

Supervised Problems. Input-Input pairs are provided:
(x1,x1), (x2,x2), ..., (xn,xn) and the network learns x = f(x+ε).

Autoassociative memories (Hopfield).
Single layer with lateral delayed connections.



Pattern Recognition
OCR, images
Memories (robust to noise)

Prediction: Input => Input

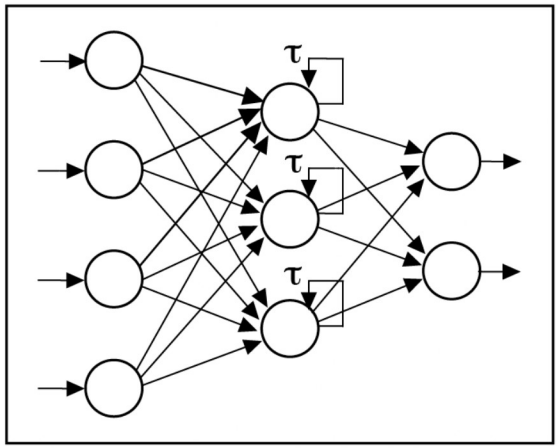
Learning: Hegg

Autoencoders (later)

Feature extraction, compression.

Supervised Problems (with memory). Input-Output pairs are provided:
(x1,y1), (x2,y2), ..., (xn,yn) and the network learns $y_t = f(x_{t-1,t-2,\ldots}+\varepsilon)$.

Recurrent Networks or Elman/Jordan nets.
Multilayer network with hidden/output delayed lines.



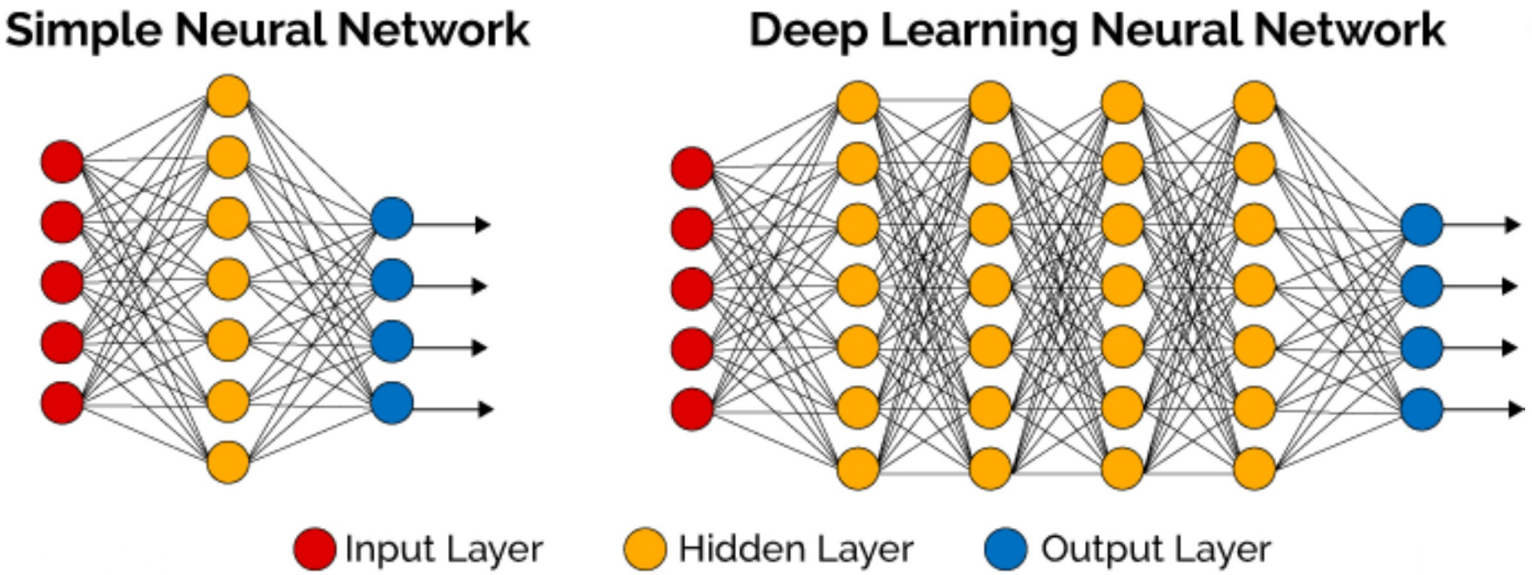
Time series analysis
Video, natural language
Interpolation and fitting

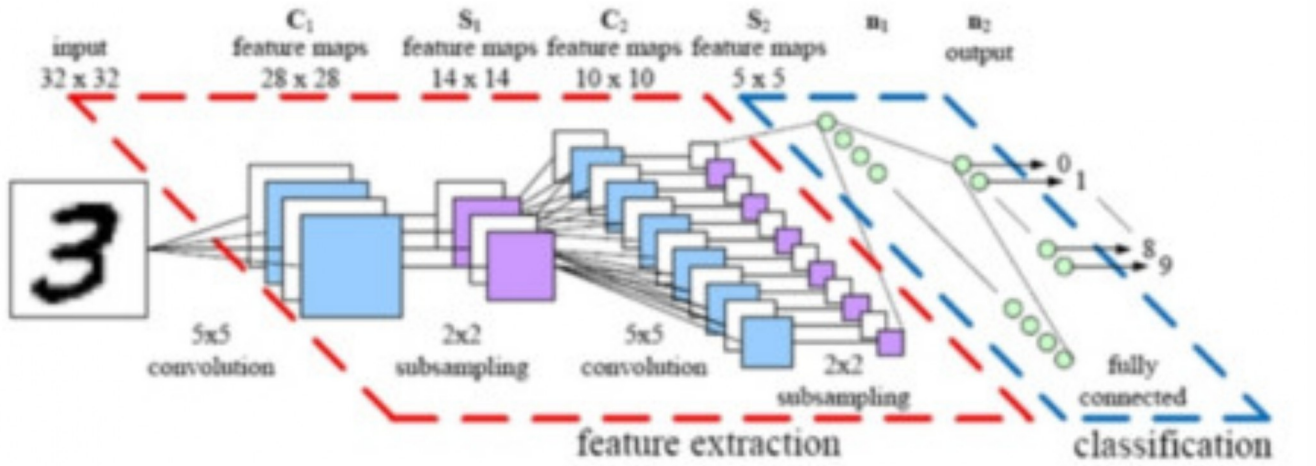
Prediction: Input => Output

Learning: Backpropagation in time

Deep Learning: Supervised and Reinforced Problems

(x1,y1), (x2,?), ..., (xn,yn) and the network self-organizes and learn y = f(x).

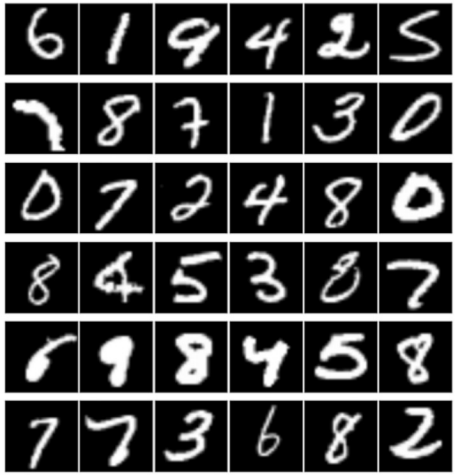




http://yann.lecun.com/exdb/mnist/

60000+10000 images 32x32

Labeled as {0,...,9}



Lineal: 10%. k-NN: 3%. SVM: 1%.

Deep: 0.3%

Include preprocessing layers for feature extraction:

- Convolutions
- Autoencoders

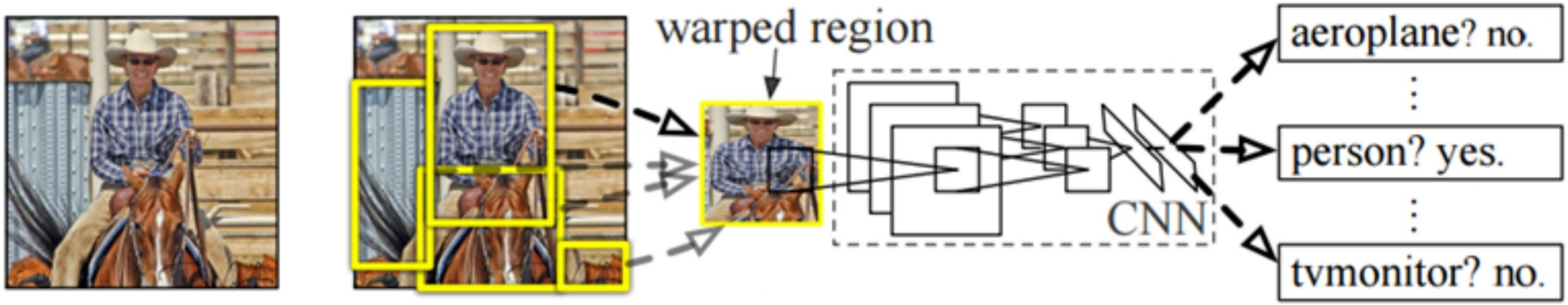
New optimization/learning.

http://www.kdnuggets.com/2017/08/convolutional-neural-networks-image-recognition.html

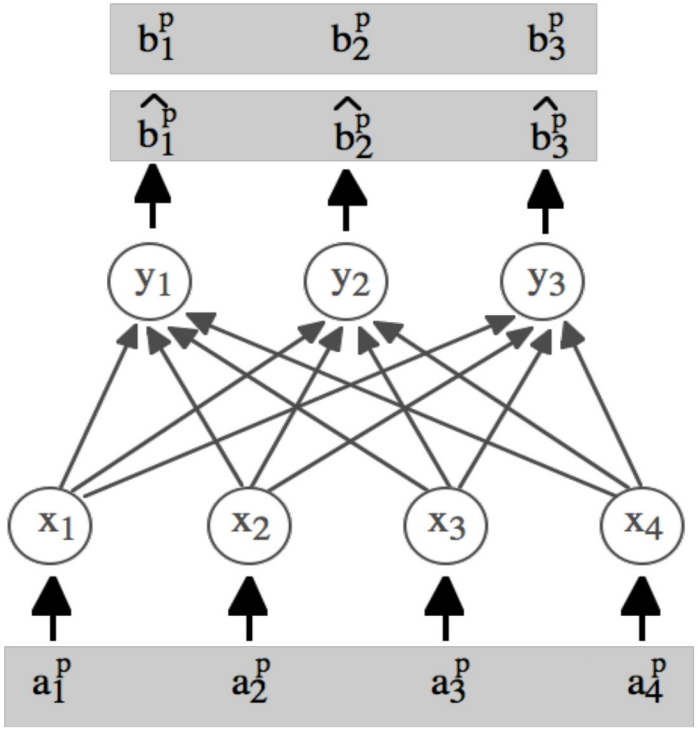
Traditional Machine Learning Flow

Deep Learning Flow

R-CNN: *Regions with CNN features*



R-CNN workflow



$$E(w) = \frac{1}{2}\sum_{i,p}(b_i^p - \hat{b_i^p})^2.$$

Inicialmente se eligen valores aleatorios para los pesos.

Aprendizaje Hebbiano (1949): Se modifican los pesos acorde a la correlación entre las unidades. Se eligen los patrones (a^p, b^p) de uno en uno y se modifican los pesos de los nodos con salidas incorrectas:

$$\Delta w_{ij} = \eta(b_i^p - \hat{b_i^p})a_j^p$$

Descenso de gradiente: Se modifican los pesos acorde la dirección del gradiente del error.

$$\Delta w_{ij} = -\eta\frac{\partial E}{\partial w_{ij}} = \eta\sum_p(b_i^p - \hat{b_i^p})f'(B_i^p)a_j^p$$

η : Tasa de aprendizaje

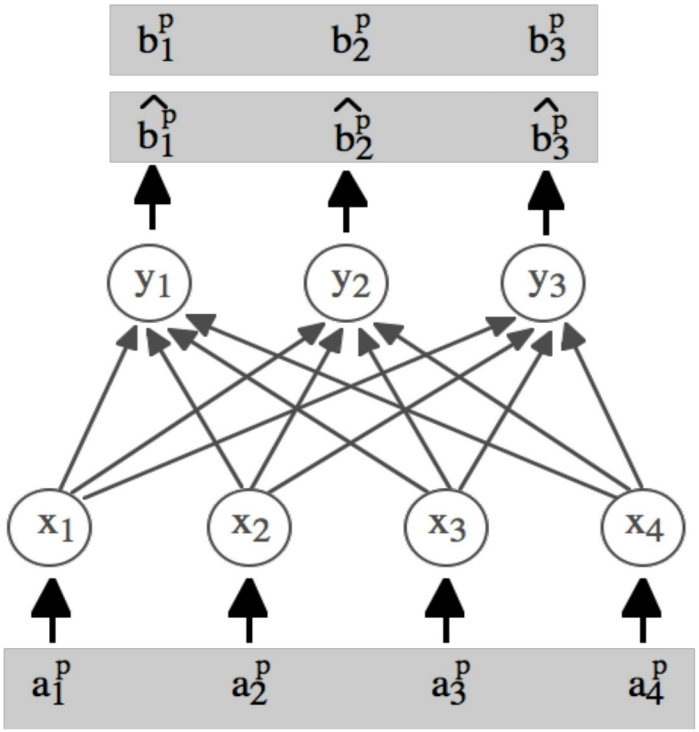
Inercia

$$\Delta w_{ij}(t+1) = -\eta\frac{\partial E}{\partial w_{ij}} + \alpha\Delta w_{ij}(t-1)$$

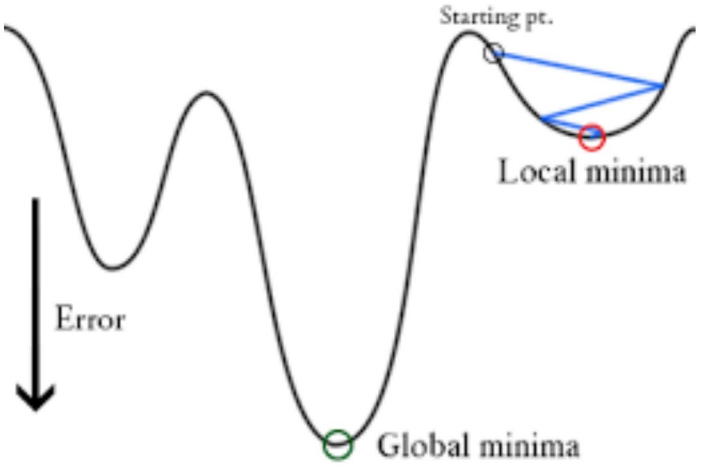
Regularización

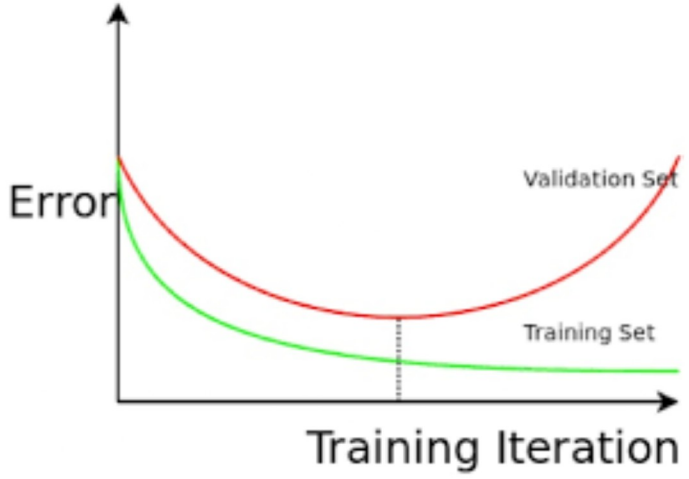
$$E(w) = \sum_{p=1}^{r}(y_p - \hat{y}_p)^2 + \lambda\sum_{i,j} w_{ij}^2$$

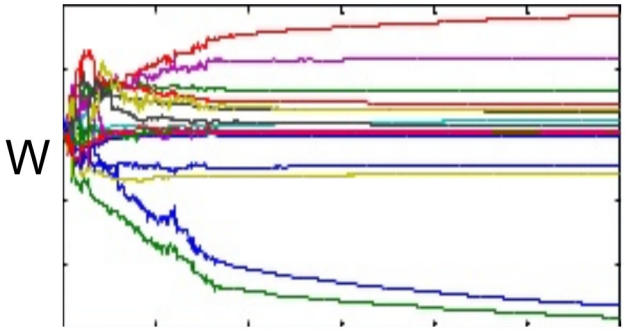
RSNNS



$$E(w) = \frac{1}{2}\sum_{i,p}(b_i^p - \hat{b}_i^p)^2.$$







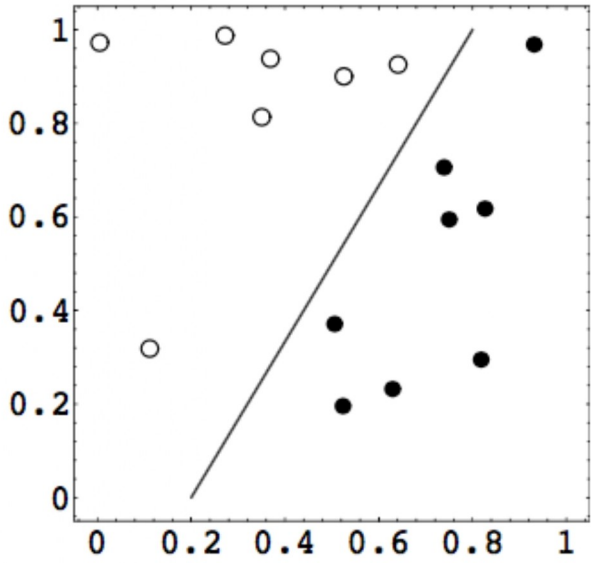
Overfiting is a critical problem in neural networks. The network should be carefully designed and/or **early stopping** learning should be adopted.

Error functions can be **highly nonlinear** and optimization can get trapped in local minima.

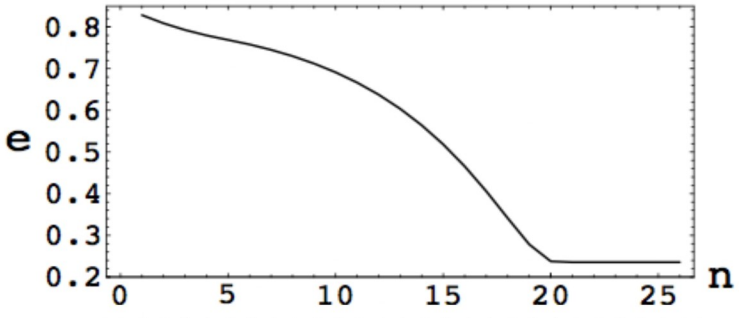
Several **replications** of the learning process are necessary (from different random initial weights).

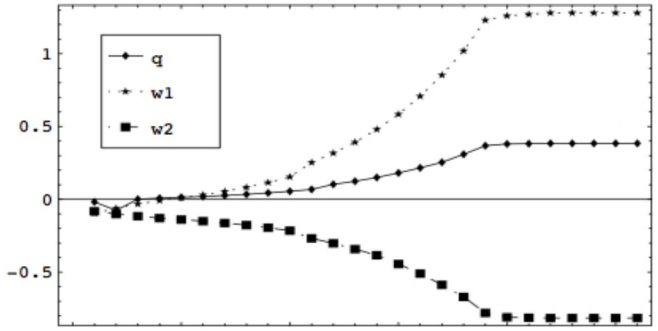
This process can be very **time consuming**.

Recent **advances** mitigate these problems.

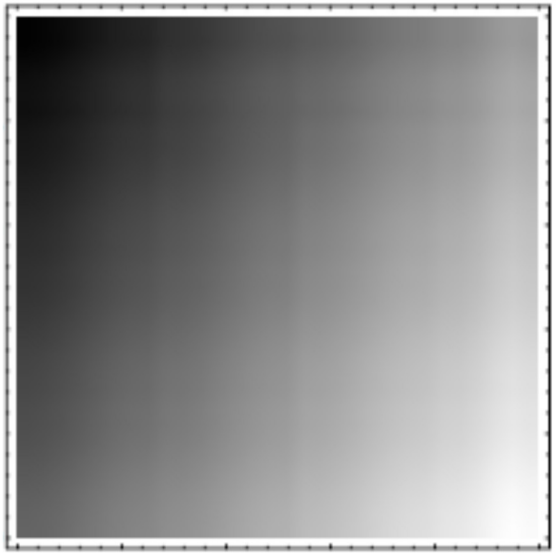


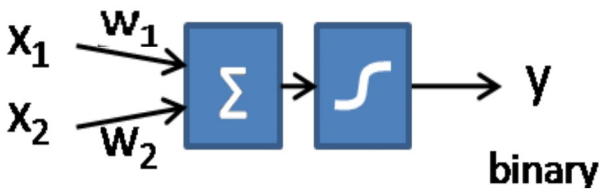
$$c_i = w_1 x_i + w_2 y_i + q,$$





$$c_i = 1.28x_i - 0.815y_i + 0.384.$$

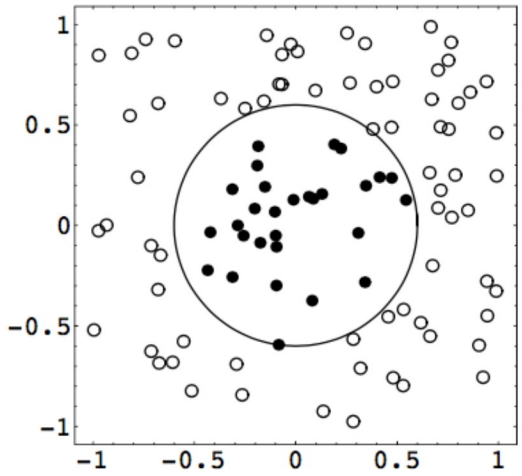




y = f(**X**,**W**) = sigmod(**X**^{T}.**W**)

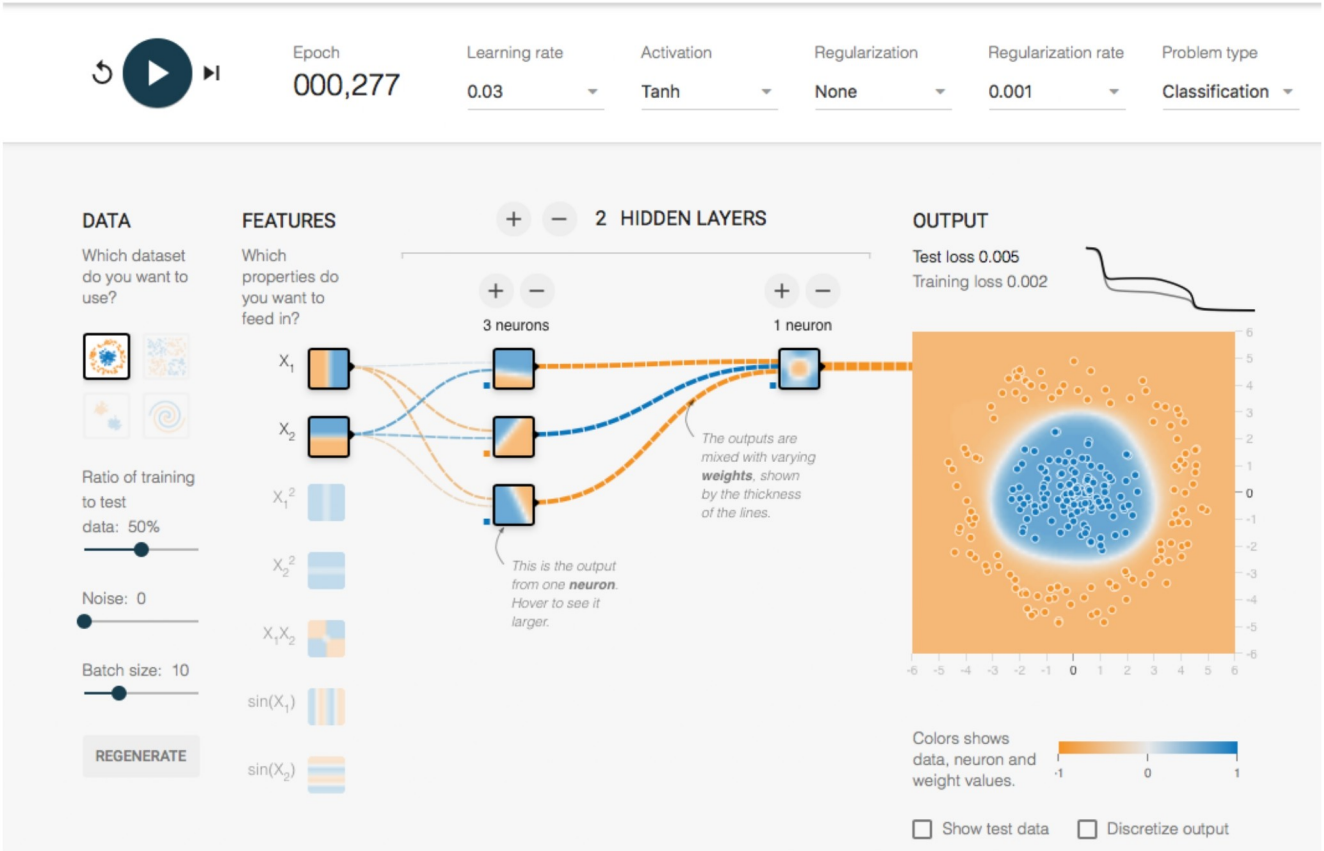
LOGISTIC REGRESSION

Single-layer networks cannot approximate nonlinear problems.



1. Watch an introductory video (19') on multi-layer neural networks.
2. Play around with the tensorflow illustrative tool.

Introductory video: https://www.youtube.com/watch?v=aircAruvnKk



http://playground.tensorflow.org/