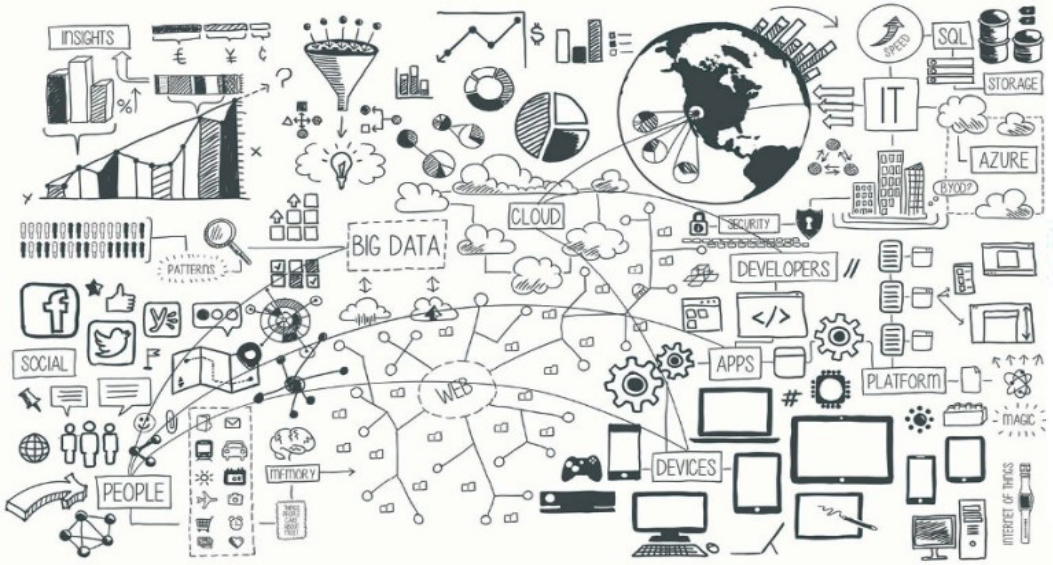
Machine Learning I (Neural Networks)

INTRODUCTION AND FEED-FORWARD MODELS

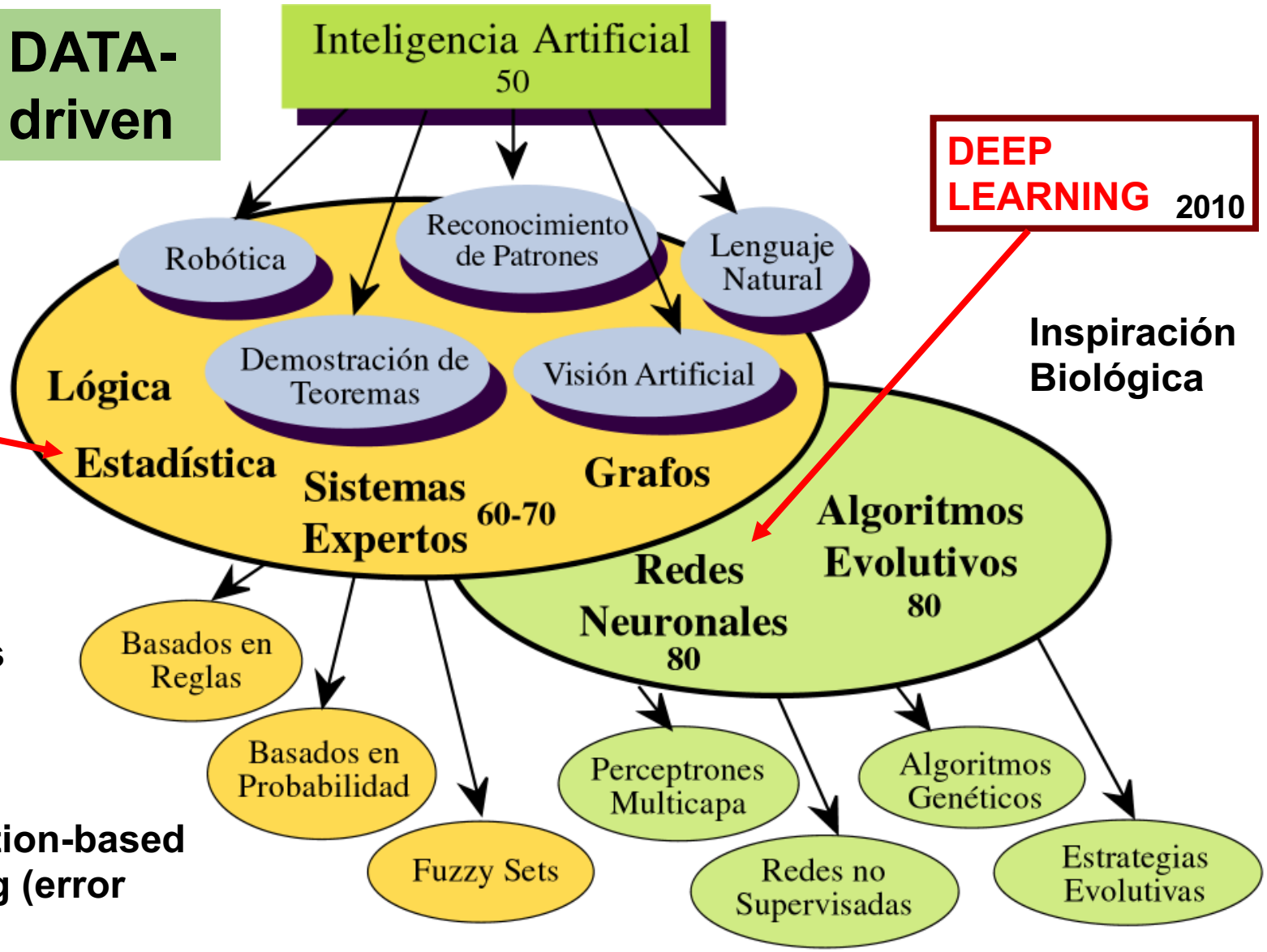


José Manuel Gutiérrez

Grupo de Meteorología

Univ. de Cantabria – CSIC
MACC / IFCA

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.

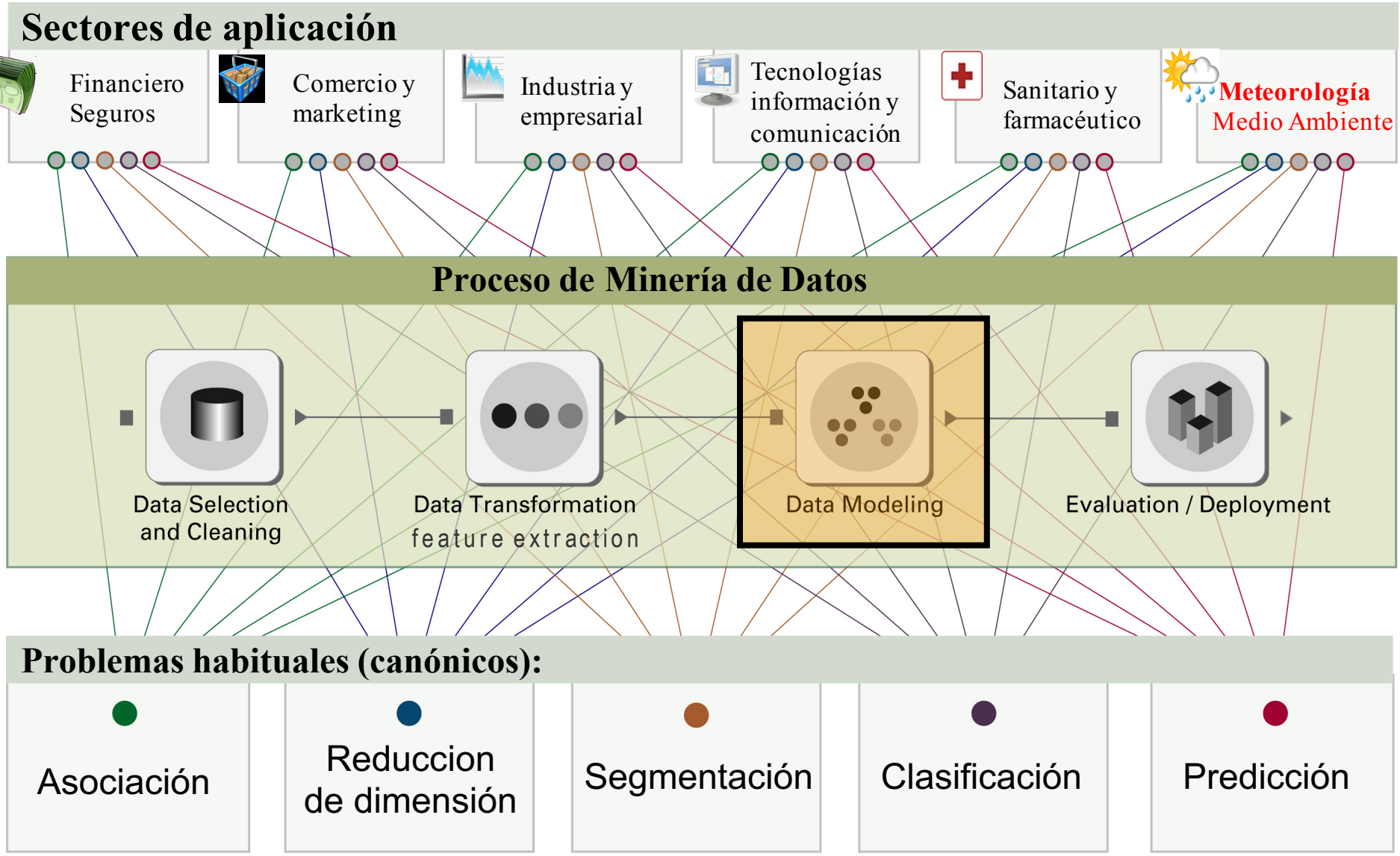
An Introduction to Functional Networks
E. Castillo, A. Cobo, J.M. Gutiérrez and E. Pruneda
Kluwer Academic Publishers (1999).

Paraninfo/International Thomson Publishing

LIBRO

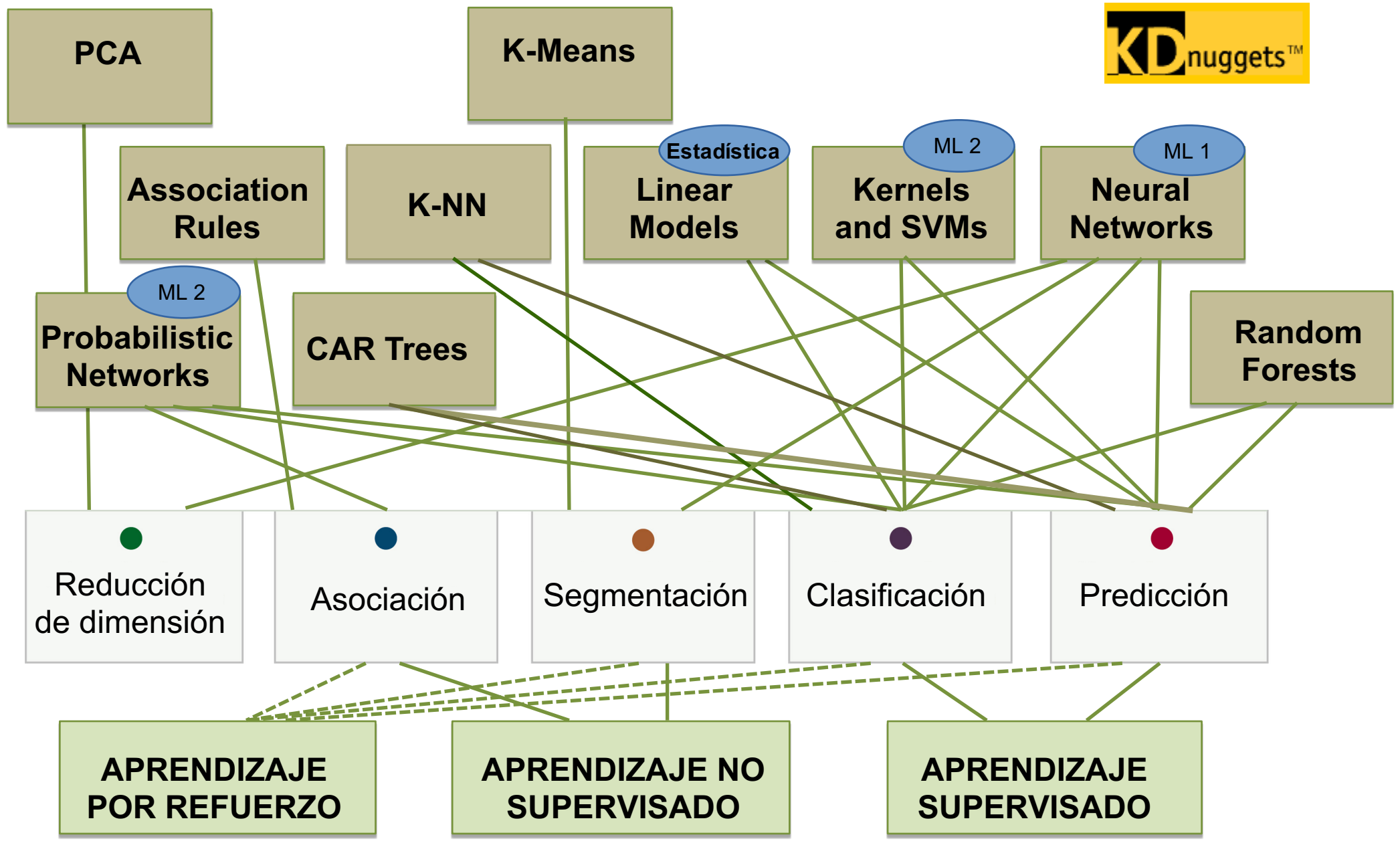
J.M. Gutiérrez, R. Cano, A.S. Cofiño, and C. Sordo
Redes Probabilísticas y Neuronales en las Ciencias Atmosféricas
Ministerio de Medio Ambiente (*Monografías del Instituto Nacional de Meteorología*), Madrid. 350 páginas, 2004

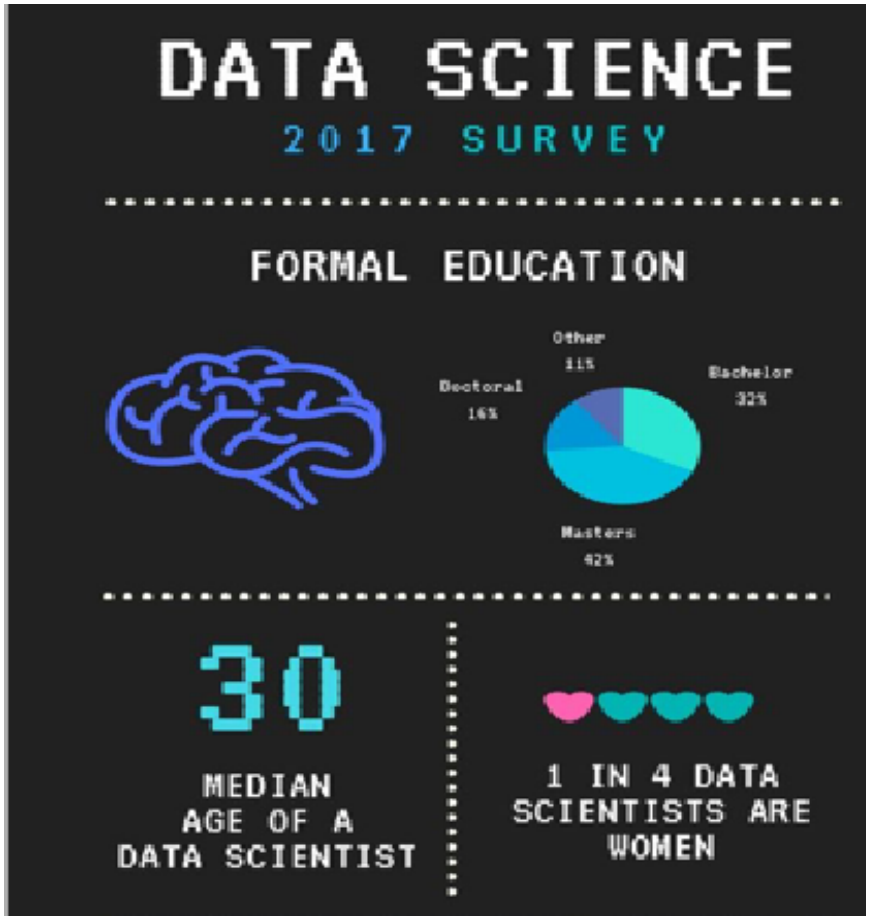
http://www.meteo.unican.es/files/pdfs/LibroINM.pdf

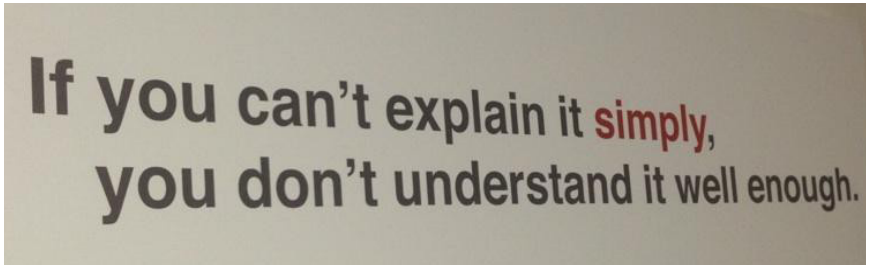


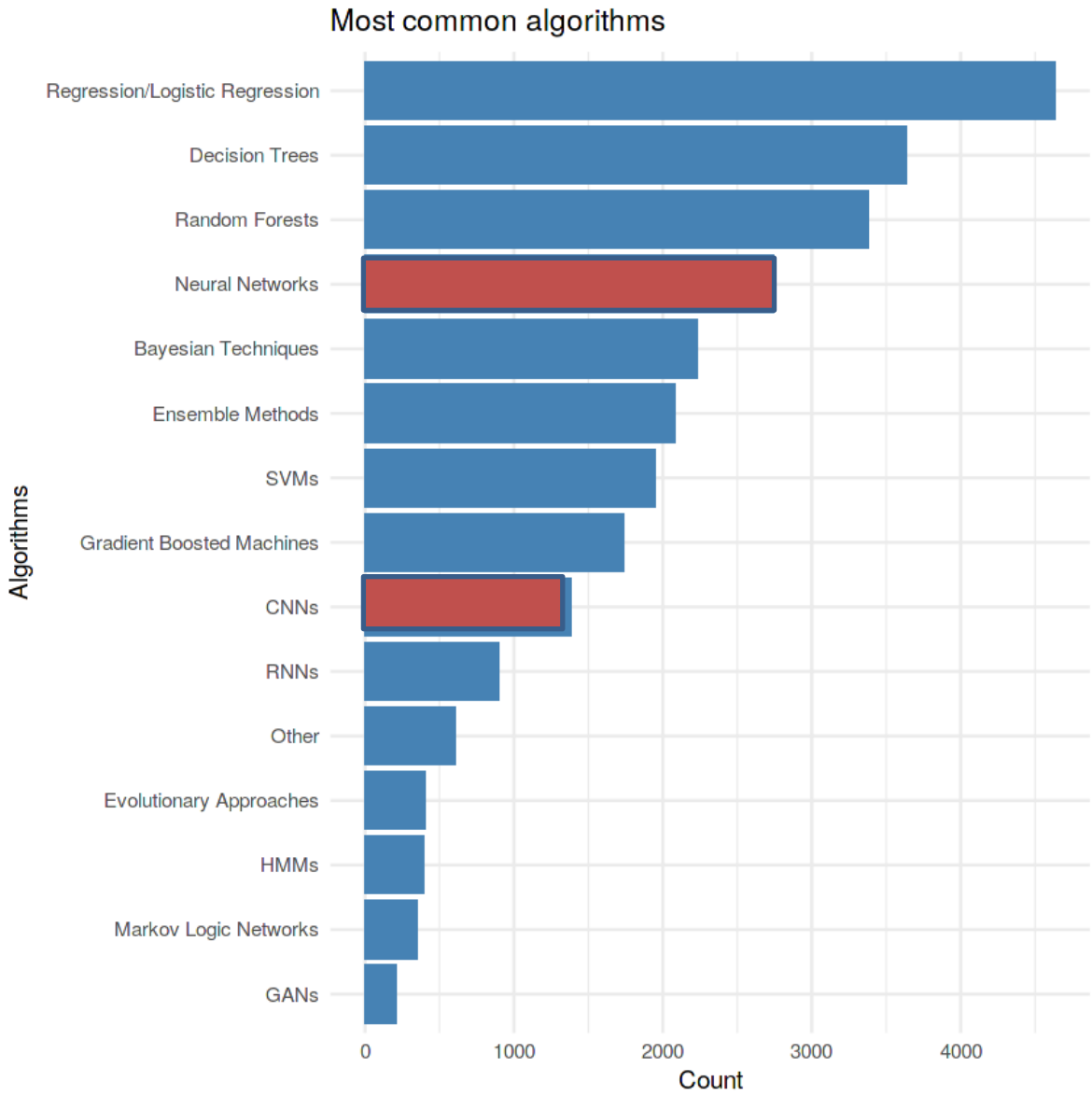
Machine learning develop methods for data modelling and prognosis.

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





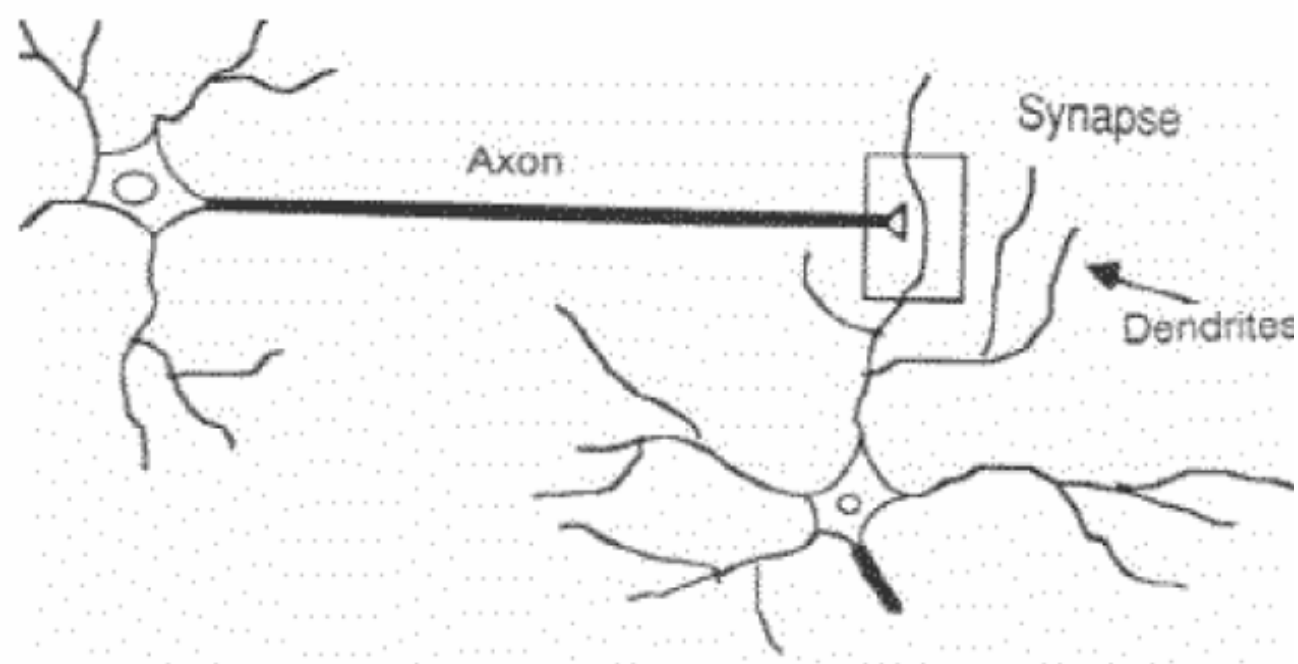




https://www.kaggle.com/kaggle/kaggle-survey-2017

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**.

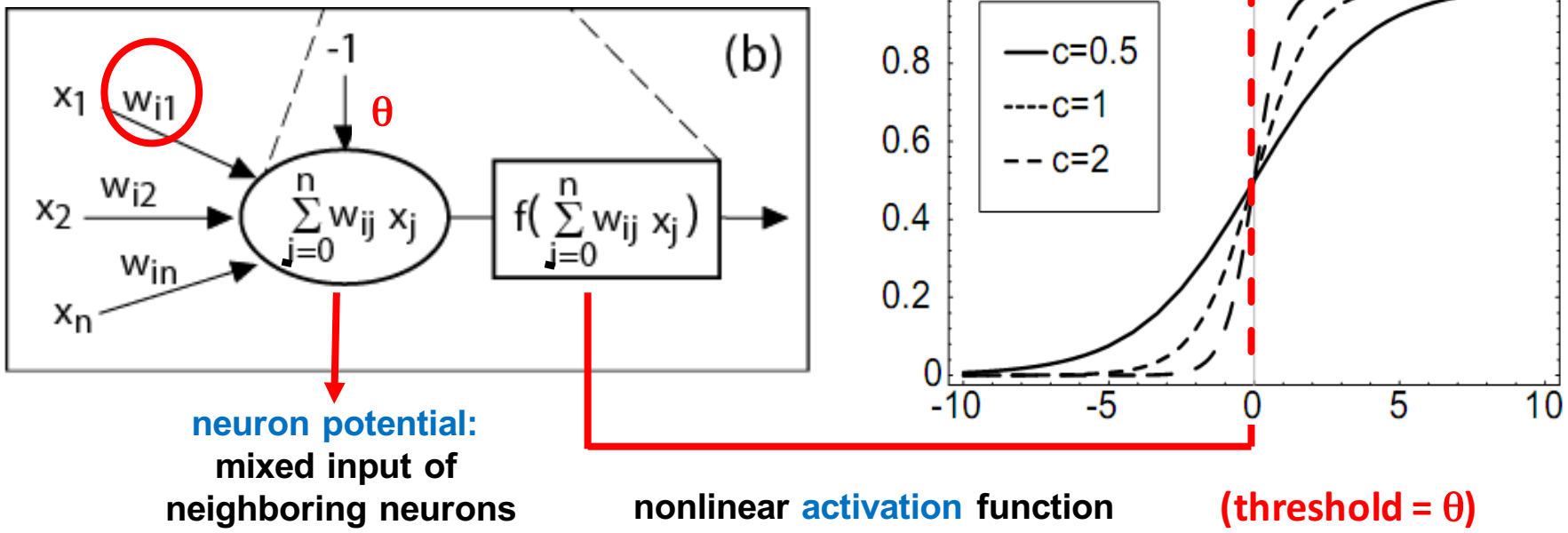


There are over **10^{11} neurons** in a human brain, each **connected with 1000** on average.

- 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**)*.

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

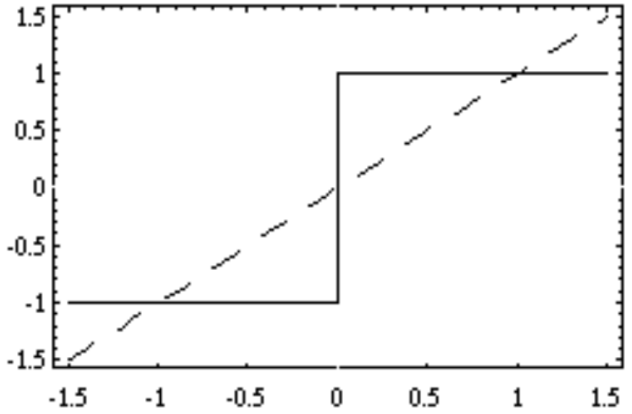
McCulloc & Pitts (1943)



- **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.

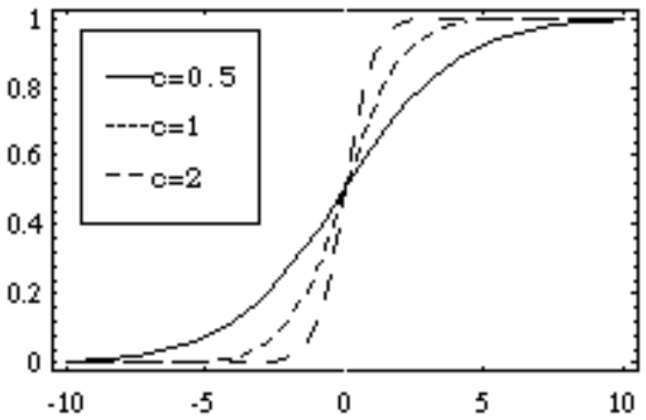
$$sgn(x) = \begin{cases} -1, \text{ si } x < 0, \\ 1, \text{ sino,} \end{cases} \quad , \quad \Theta(x) = \begin{cases} 0, \text{ si } x < 0, \\ 1, \text{ sino.} \end{cases}$$



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

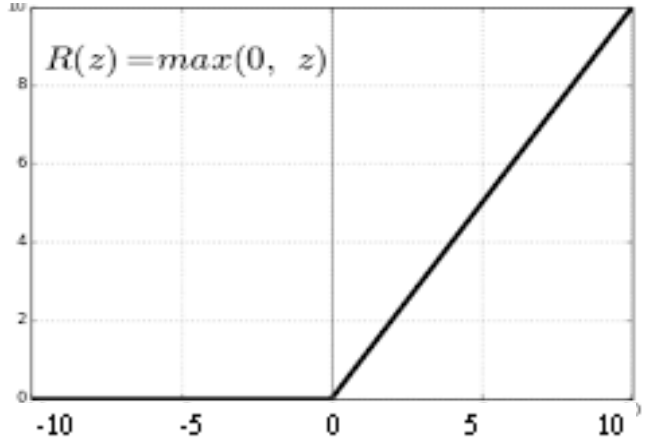
  1. La función logística de 0 a 1:

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

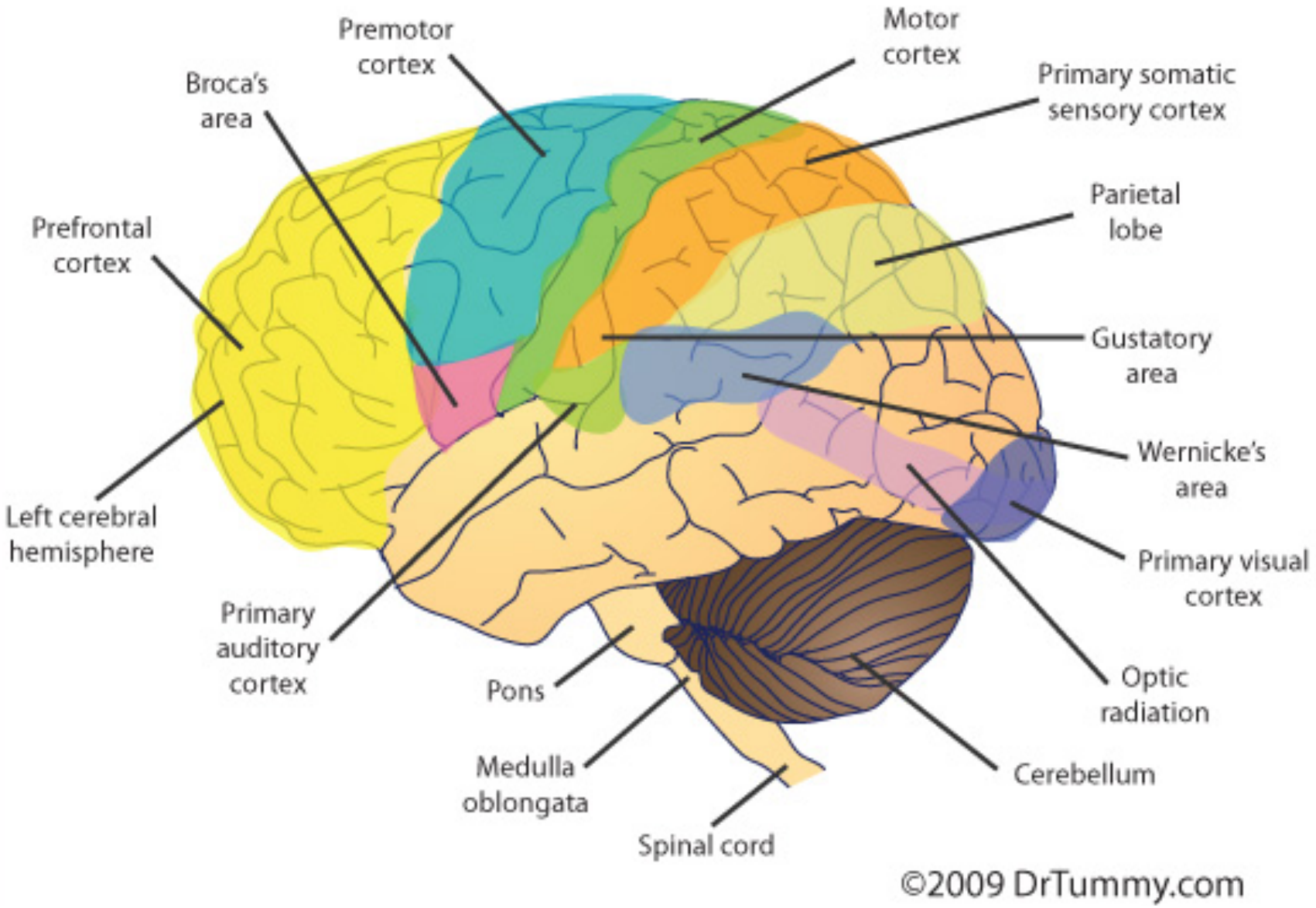


  2. La función tangente hiperbólica de -1 a 1 $f_c(x) = tanh(c\,x)$.

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

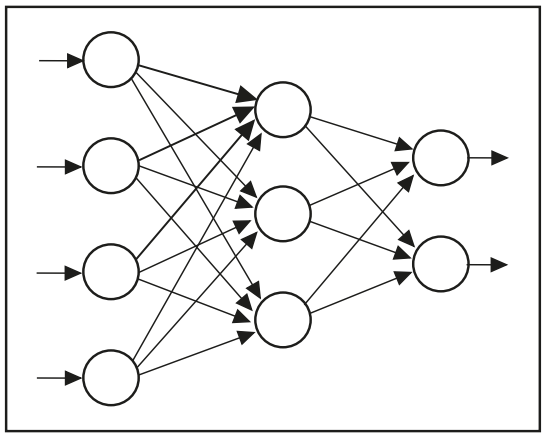


TanH	$f(x) = tanh(x) = \frac{2}{1+e^2x} - 1$	$f'(x) = 1 - f(x)^2$	$(-1, 1)$	C^∞
SoftSign	$f(x) = \frac{x}{1+\|x\|}$	$f'(x) = 1 - f(x)^2$	$(-1, 1)$	C^1
SoftPlus	$f(x) = \ln(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$	$(0, \infty)$	C^∞
SoftExponential	$f(\alpha, x) = \begin{cases} -\frac{\ln(1-\alpha(x+\alpha))}{\alpha} \text{ for } \alpha < 0 \\ x \text{ for } \alpha = 0 \\ \frac{e^{\alpha x}-1}{\alpha} + \alpha \text{ for } \alpha > 0 \end{cases}$	$f'(\alpha, x) = \begin{cases} \frac{1}{1-\alpha(\alpha+x)} \text{ for } \alpha < 0 \\ e^{\alpha x} \text{ for } \alpha \geq 0 \end{cases}$	$(-\infty, \infty)$	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)



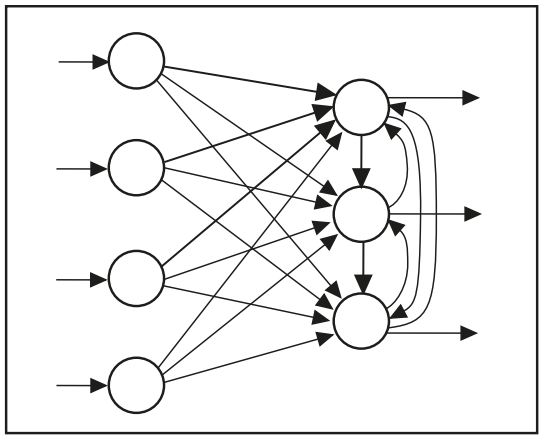
Pattern Recognition
OCR, images
Interpolation and fitting

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

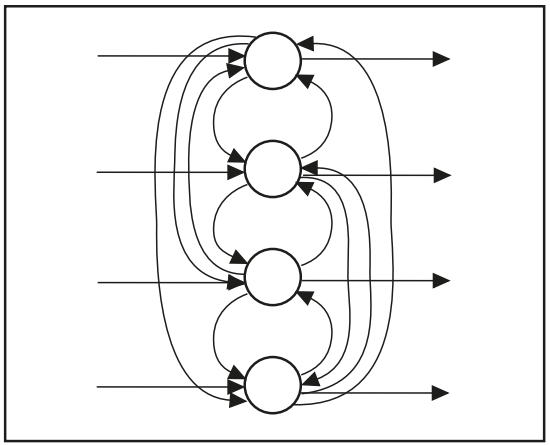
Feature extraction.

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

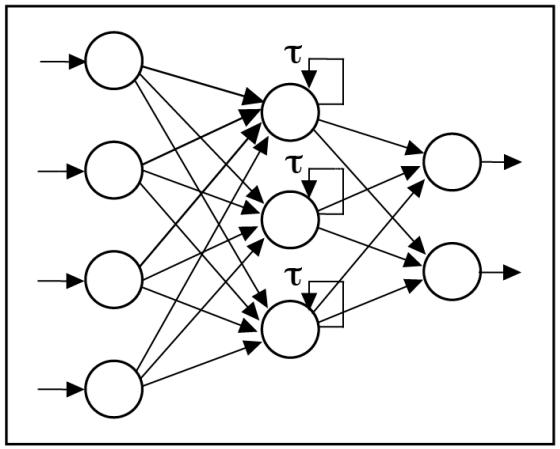
Learning: Hegg

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.



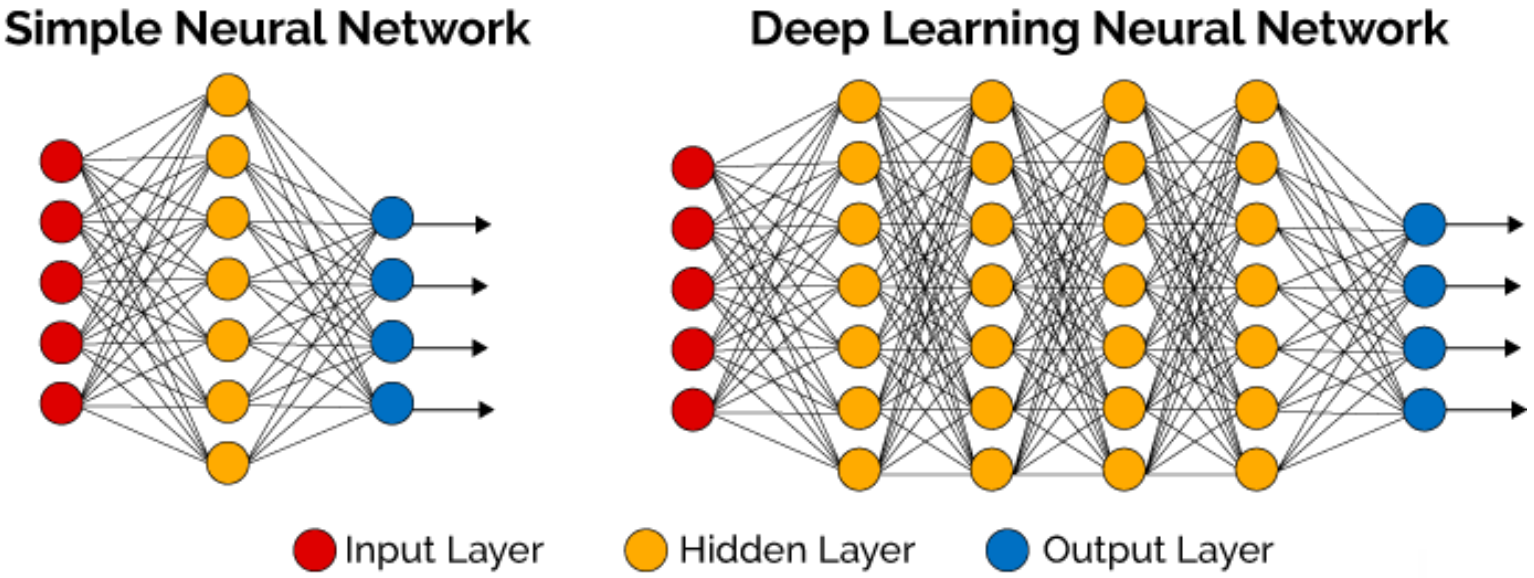
Time series analysis
Video, natural language
Interpolation and fitting

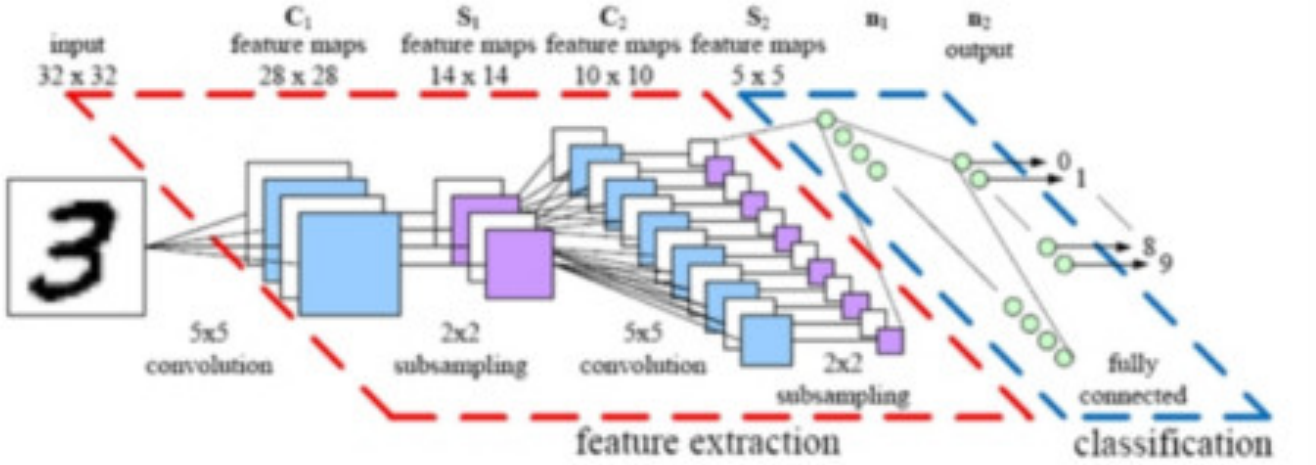
Prediction: Input => Output

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).

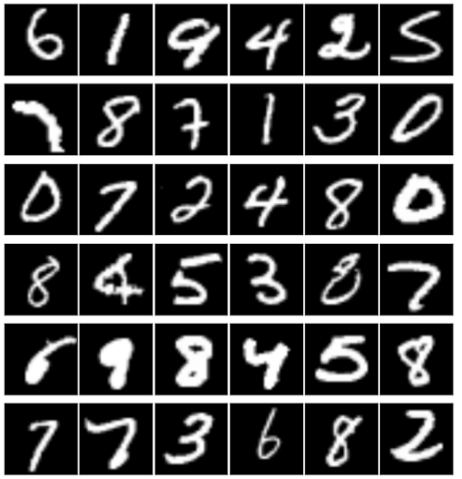




http://yann.lecun.com/exdb/mnist/

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

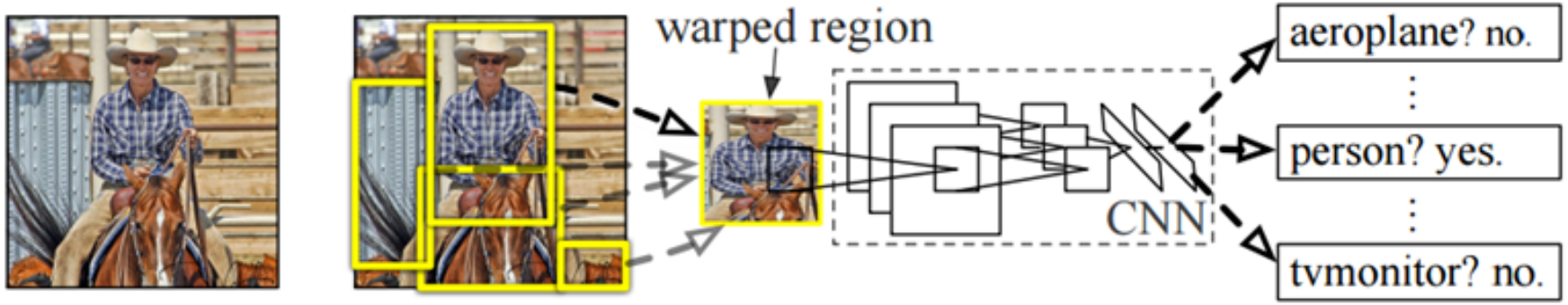
New optimization/learning.

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

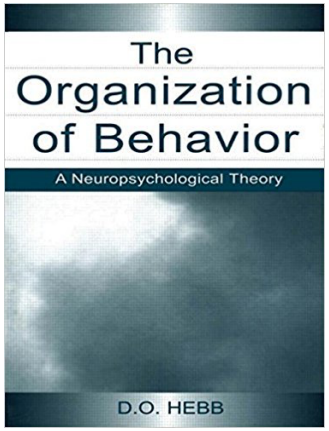
Neural Network Study (1988, AFCEA International Press, p. 60):

*... 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.***



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

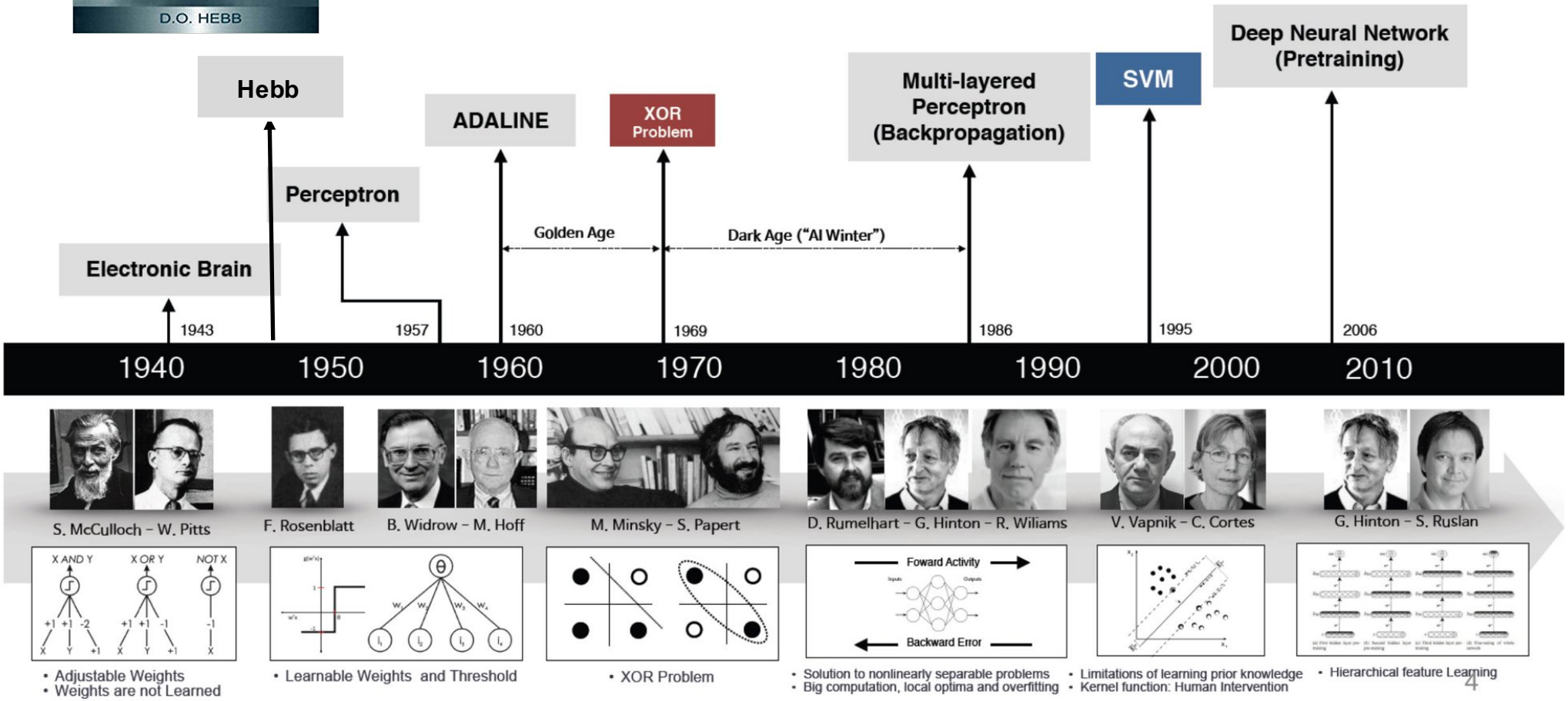
Review

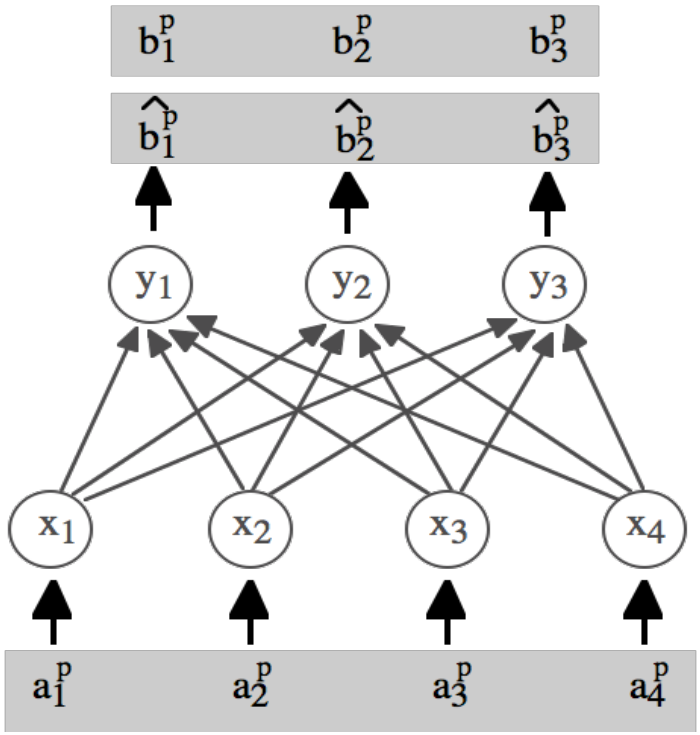
Deep learning

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





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

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

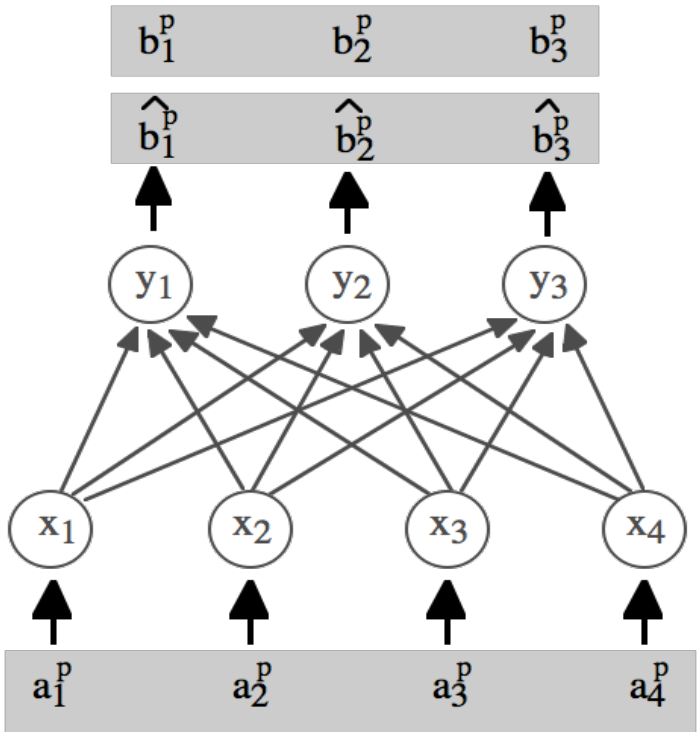
Inercia

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

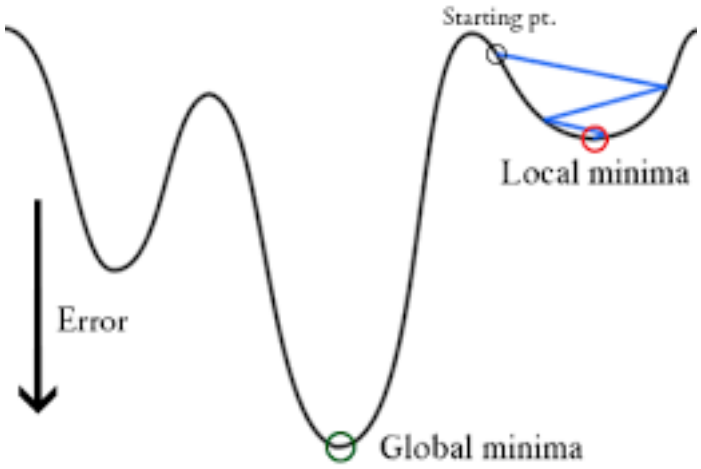
Regularización

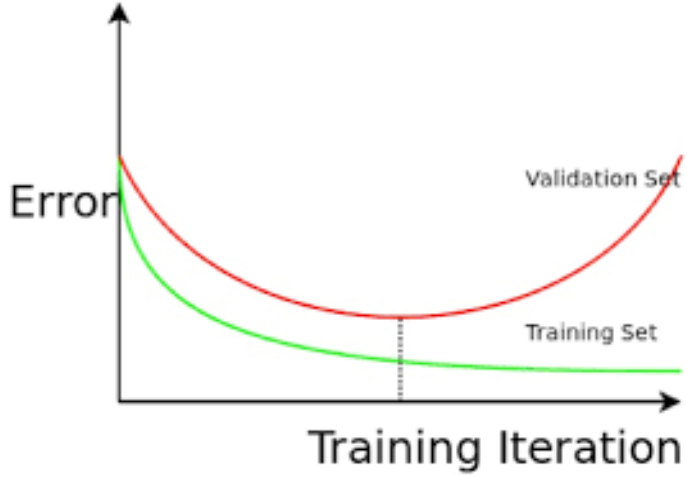
$$E(w) = \sum_{p=1}^{r}(y_p - \hat{y}_p)^2 + \lambda\sum_{i,j} w_{ij}^2$$

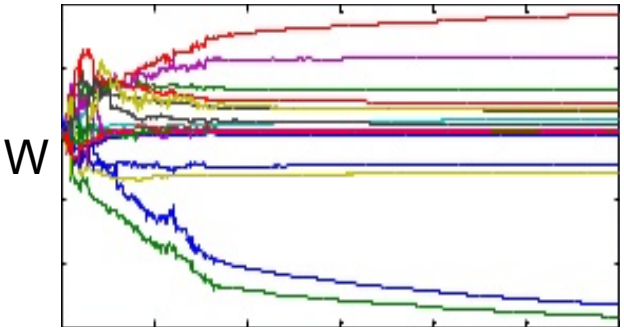
RSNNS



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







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**.

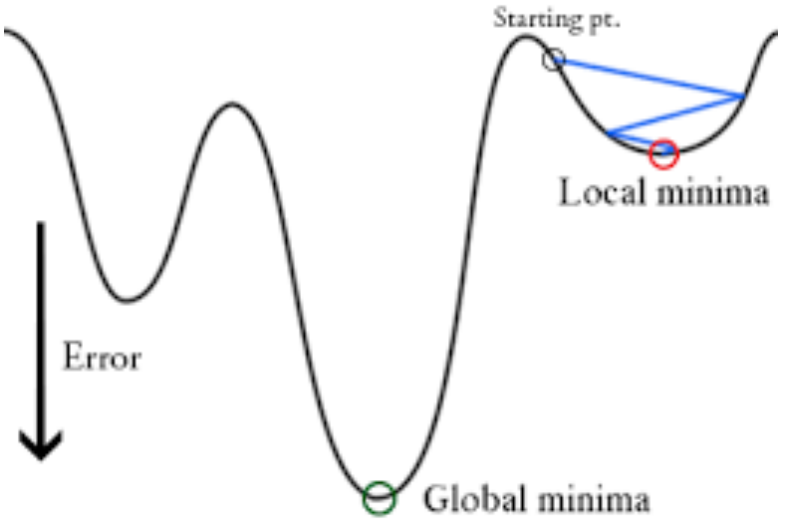
Recent **advances** mitigate these problems.

There are many extensions of the **backprop** method, such as conjugate gradient, Levenberg–Marquardt, etc.

nnet **RSNNS**

AMORE TAO-robust backpropagation

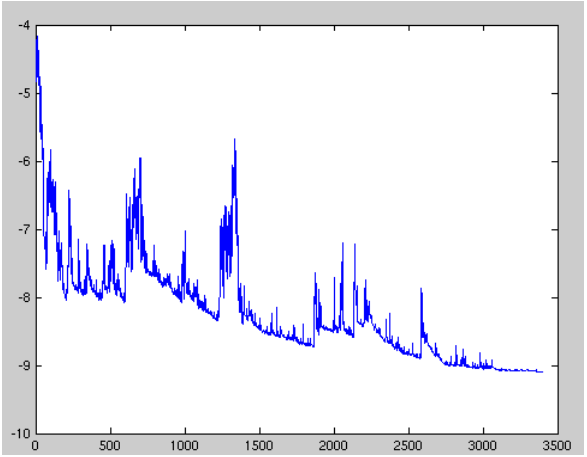
neuralnet Resilient backpropagation

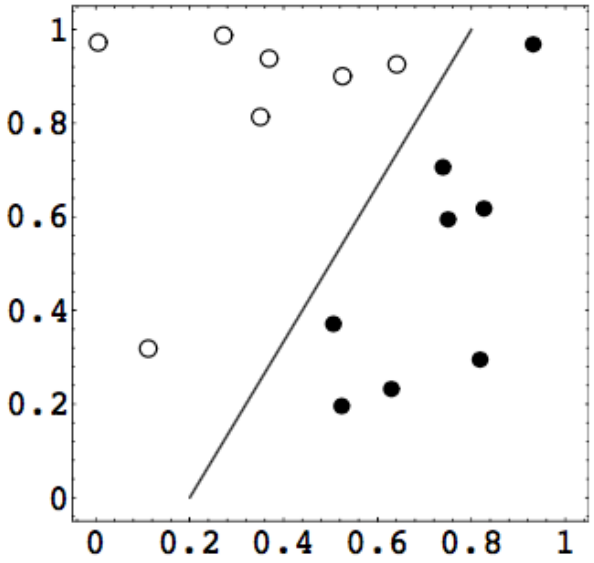


Recent advances in neural networks and deep learning provide new alternatives for the backprop (gradient descent) method including regularization in different ways.

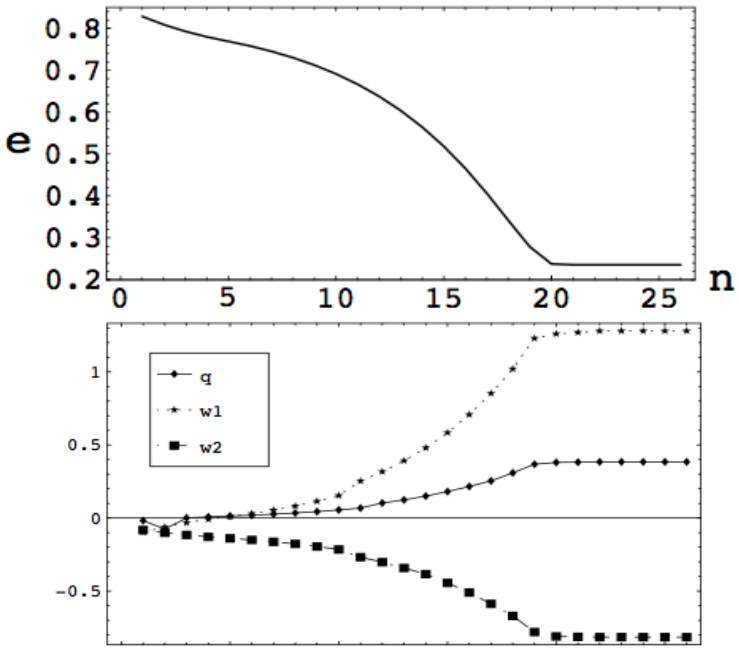
deepnet Stochastic/batch, dropout, etc.

Stochastic Gradient Descent (SGD) performs a parameter update for **each training example** (**or batch**). It is usually much faster and has higher variance, allowing to scape from local minima.

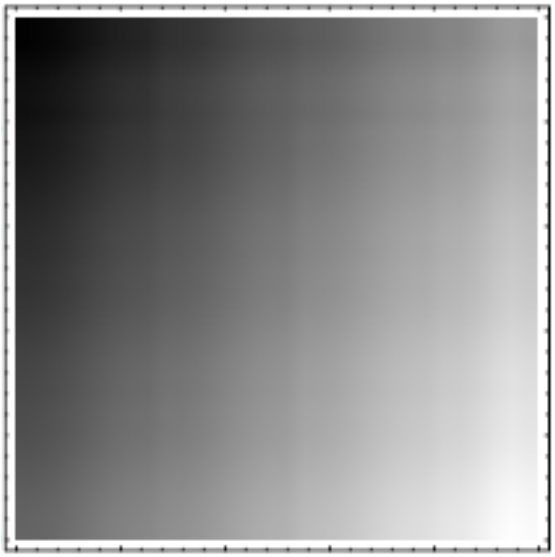


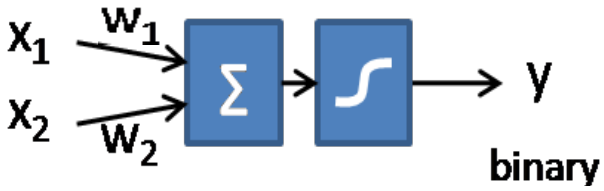


$$c_i = w_1 x_i + w_2 y_i + q,$$



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

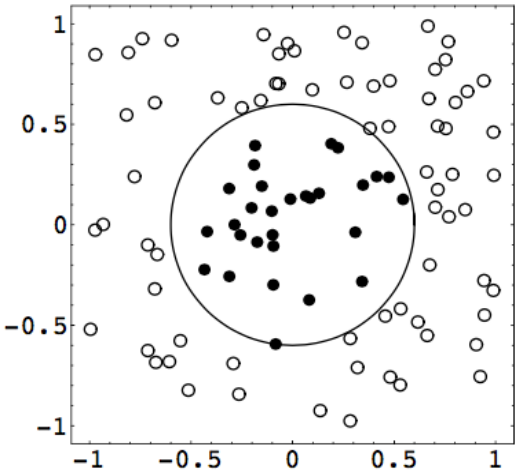




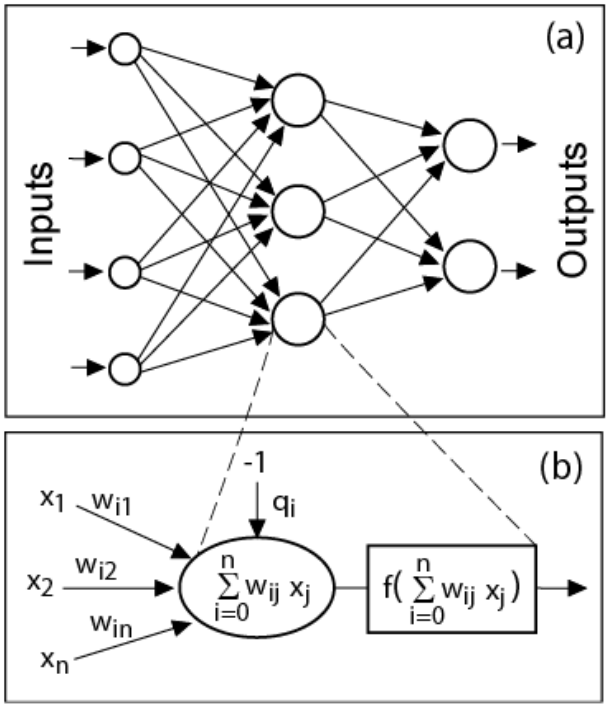
y = f(**X**,**W**) = sigmod(**X**^{T}.**W**)

LOGISTIC REGRESSION

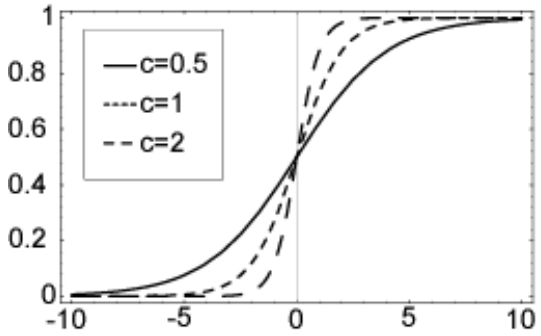
Single-layer networks cannot approximate nonlinear problems.



x : h : y



The neural activity (output) is given by a nonlinear function.



$$y_i = f(\sum_k W_{ik} f(\sum_j w_{kj} a_{pj}))$$ **h_i**

$$E(w) = \frac{1}{2}\sum_{p,i}(b_{pi} - f(\sum_k W_{ik}\, f(\sum_j w_{kj} a_{pj})))^2$$

Gradient descent

$$\Delta W_{ik} = -\eta \frac{\partial E}{\partial W_{ik}};\ \Delta w_{kj} = -\eta \frac{\partial E}{\partial w_{kj}},$$

1. Init the neural weight with random values
2. Select the input and propagate it (estimate hidden and output)
3. Compute the error associate with the output

$$\delta_{pi} = (b_{pi} - \hat{b}_{pi}) f'(\hat{B}_{pi})$$
$$= (b_{pi} - \hat{b}_{pi})\hat{b}_{pi}(1 - \hat{b}_{pi})$$

4. Compute the error associate with the hidden neurons

$$\psi_{pk} = \sum_i \delta_{pi}\, W_{ki}\, f'(\hat{H}_{pk})$$

5. Compute

$$\Delta W_{ik} = \eta\ \delta_{\mathrm{pi}}\, \hat{h}_{pk},\ \ \Delta w_{kj} = \eta\, \psi_{pk}\, a_{pj}$$

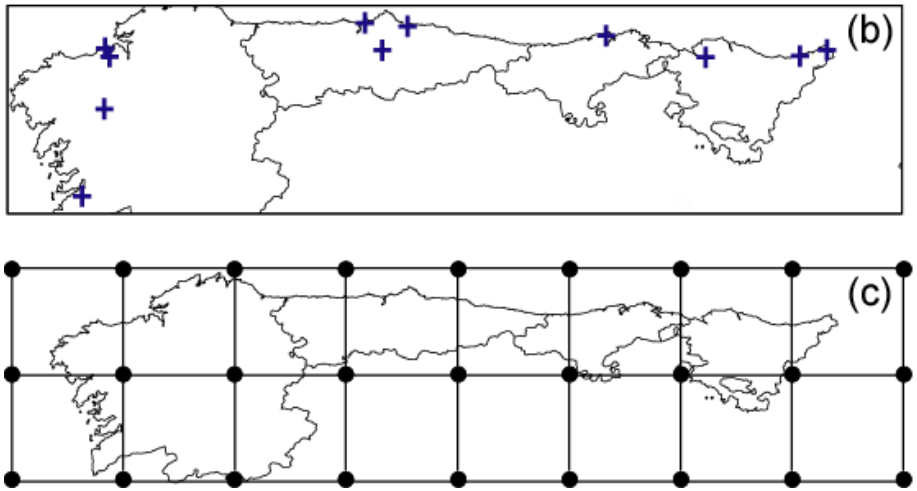
and update the neural weight according to these values

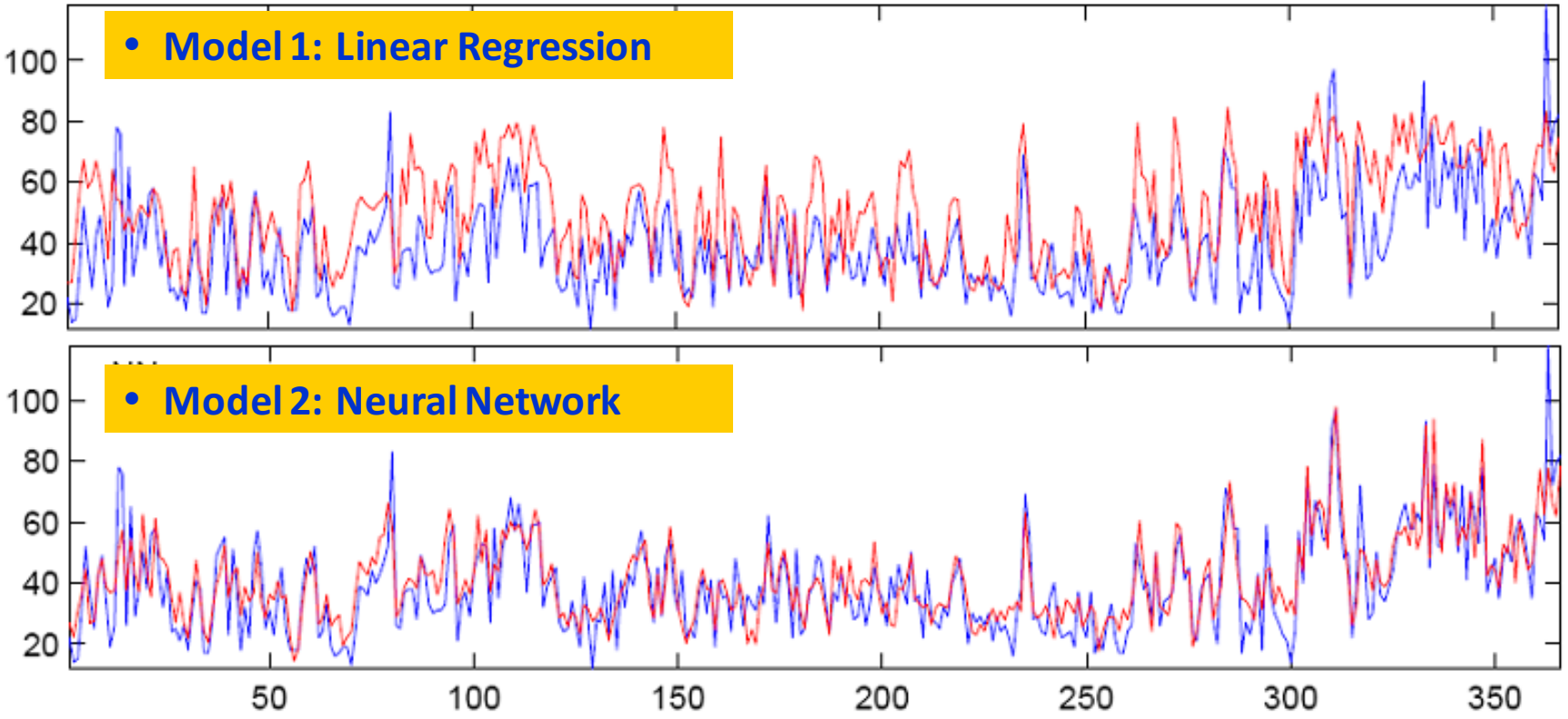
Wind Speed ➔ [0,∞)

Observations from **1977- 2002.**

ERA40 over 27 grid points for the same period

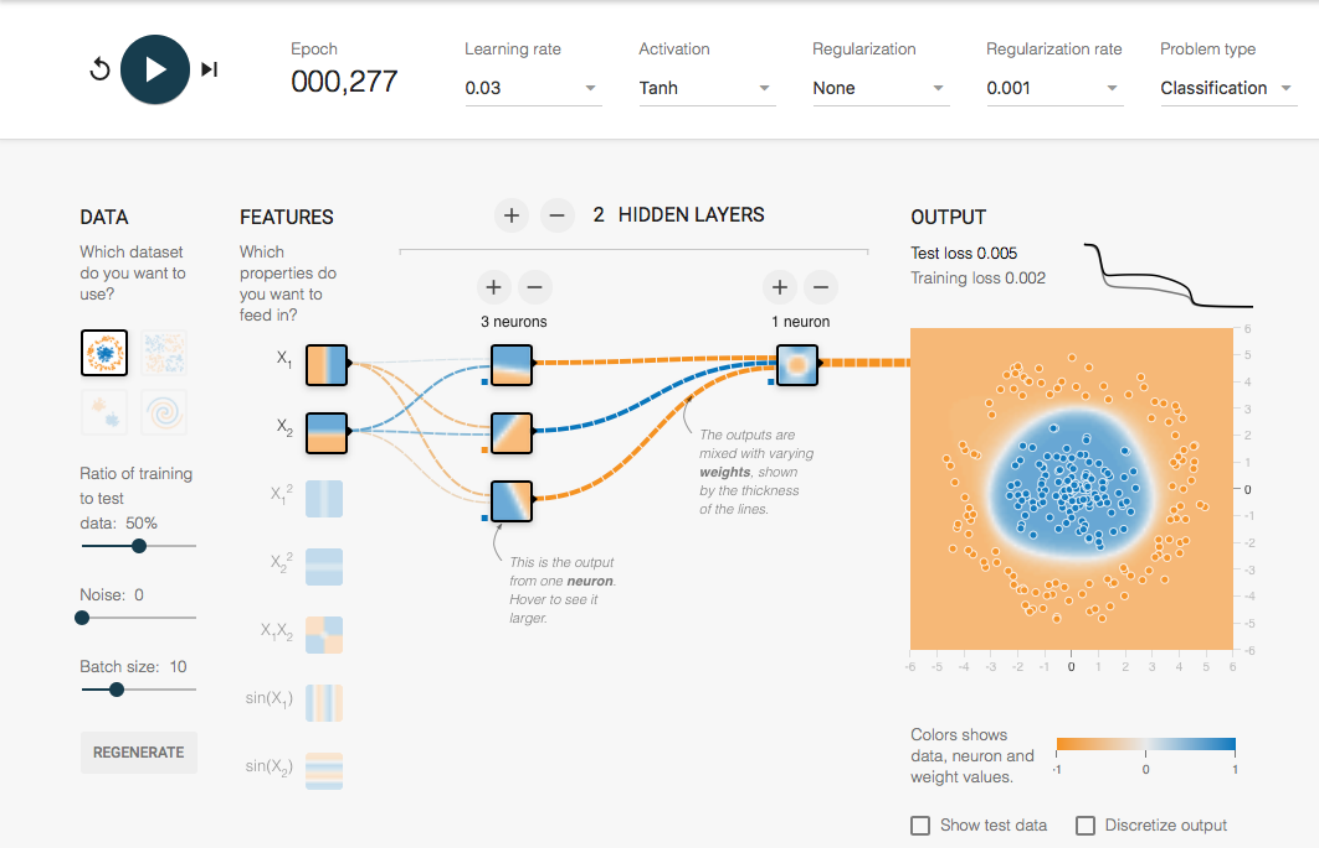
60% for trainning and
40% for validation





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

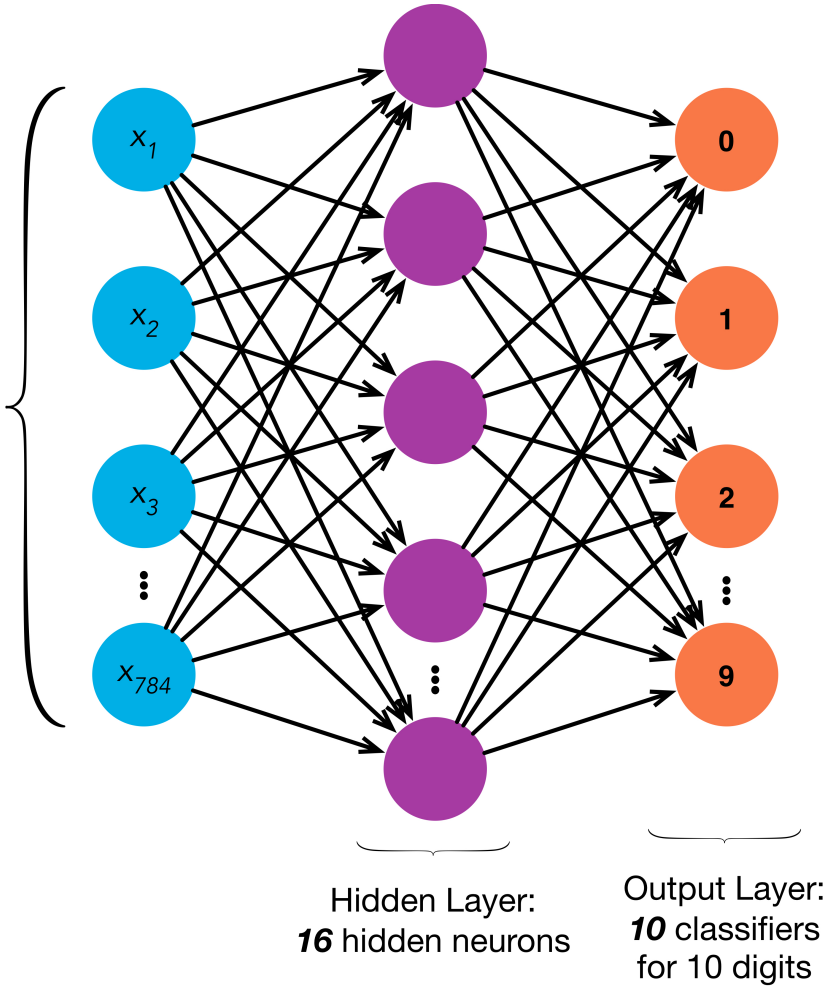
Introductory video: https://www.youtube.com/watch?v=aircAruvnKk



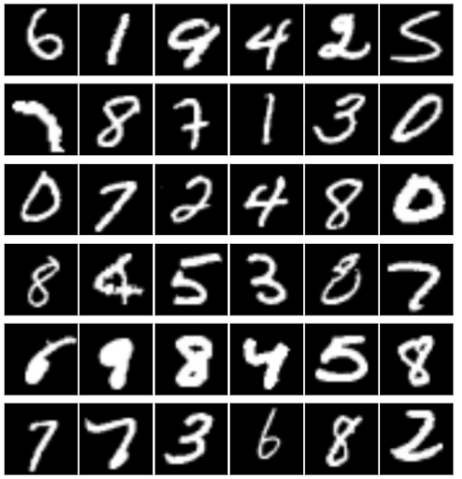
http://playground.tensorflow.org/

1. Usar RSSN para entrenar un modelo para MNIST.

- Calcular cuanto tiempo tarda cada época completa.



http://yann.lecun.com/exdb/mnist/
60000+10000 images 32x32
Labeled as {0,...,9}



Después de haber realizado el Lab01 (uso de RSNNS).