



Saturdays.AI

#1 Neural Networks: Basics

by Saturdays.AI

Saturdays.AI Donostia
Deep Learning



Week 1



Schedule

State of the course

Lesson 1 Review

Challenge

Notebook + resources



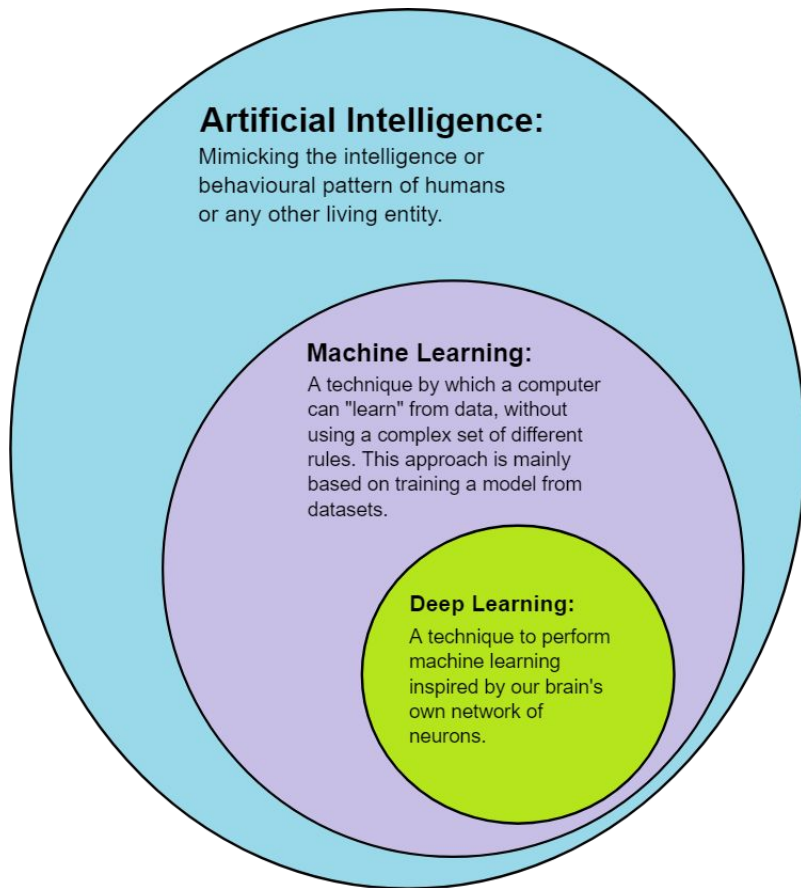
State of the course



Neural Networks

- Deep Learning
- Definitions
- Gradient Descent
- Backpropagation
- Activation functions
- Regularization
- Keras & Other Tools!

Machine Learning → Deep Learning

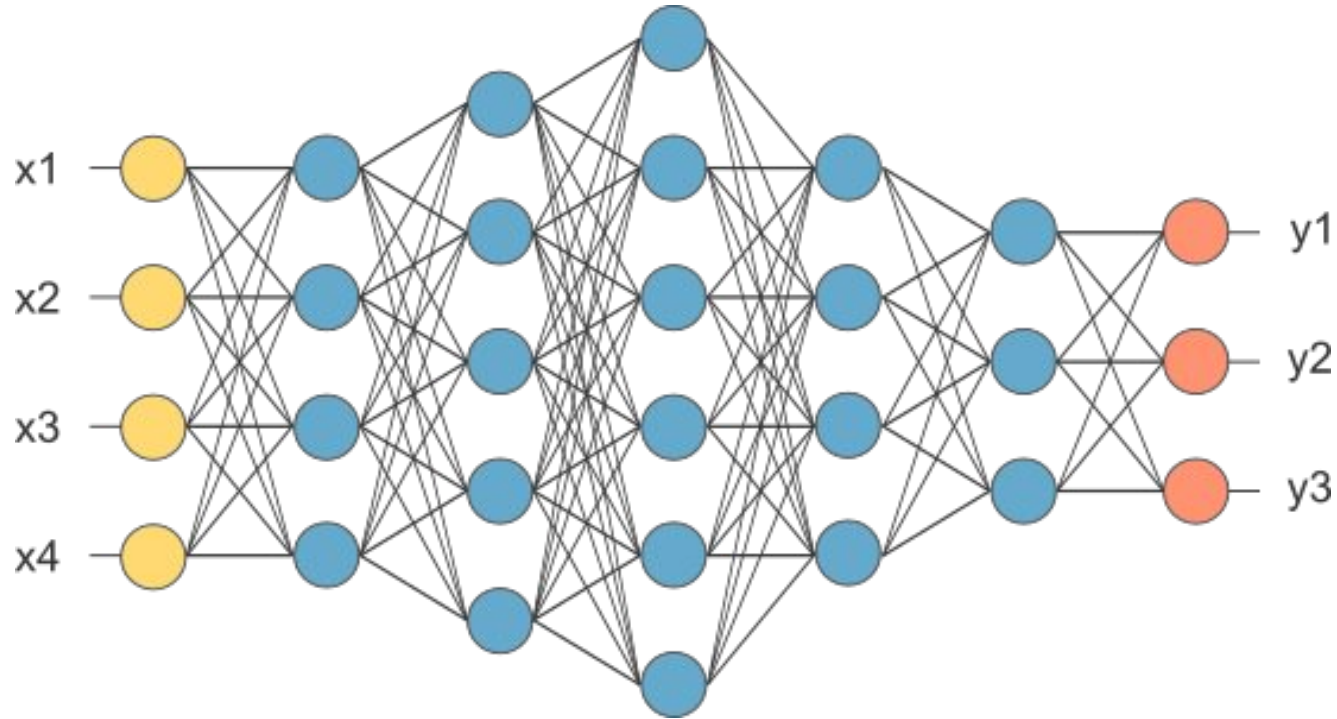


¡Lo aprendido en Machine Learning **SÍ importa** para Deep Learning!

Deep Learning: Proceso de Trabajo



Deep Learning: Red Neuronal



¿Para qué se puede usar una NN?

Deep Learning: Algunas aplicaciones

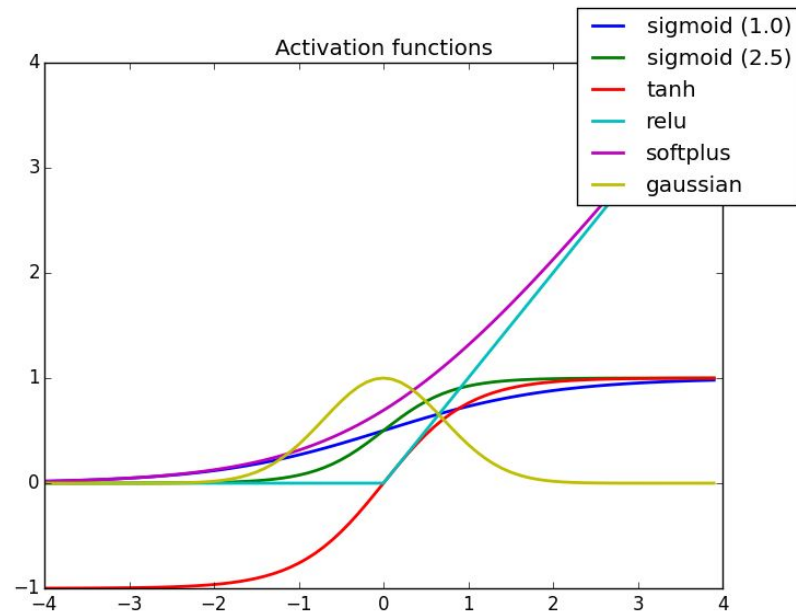
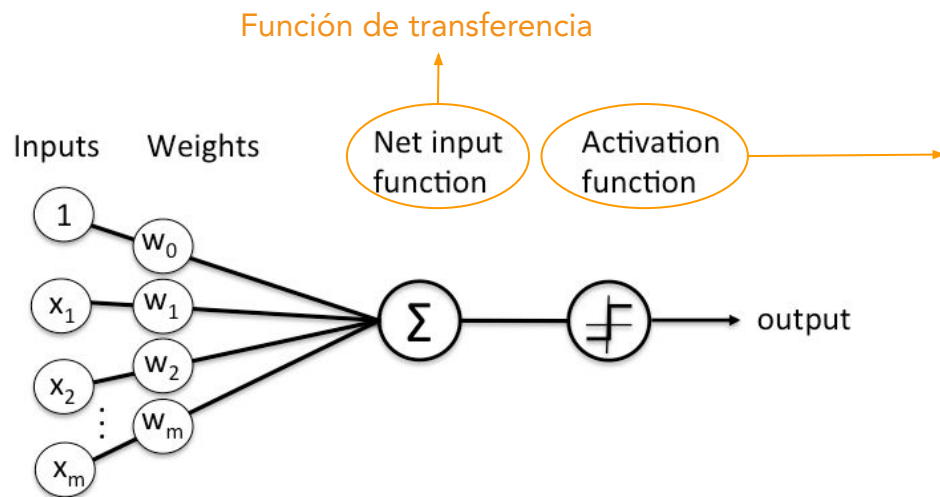
- Clasificación: ¿Tiene o no tiene cáncer?
- Reconocimiento del habla
- Procesamiento de lenguaje natural (NLP)
- Reconocimiento de texto
- Generación de texto
- Reconocimiento de imágenes: ¿Es un gatito o un tigre?
 - Ejemplo particular: Reconocimiento óptico de caracteres (OCR)
- Traducción automática
- Análisis de series temporales
- Composición musical
- Predicción de vídeos

Nota: Todas no son óptimas para todo! Pero hoy comenzamos con la base para entenderlas todas

Pero... ¿Qué componentes tiene una NN?

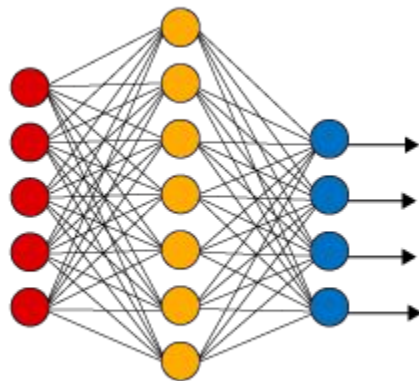
Deep Learning: Perceptrón // Neurona

Número dentro de cada neurona: **Activación**

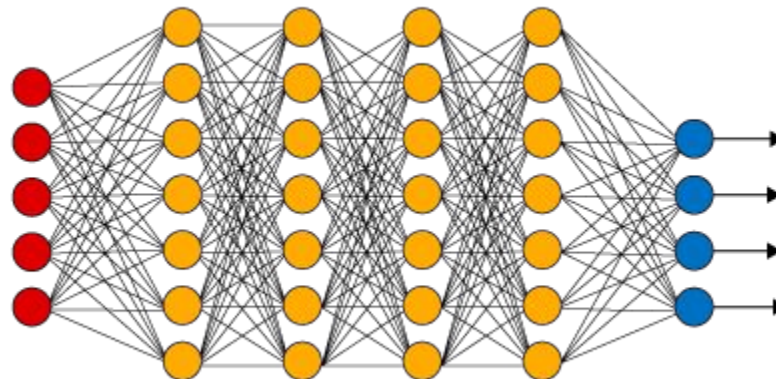



Deep Learning: Architectures and Variants

Simple Neural Network



Perceptrón Multicapa
Deep Learning Neural Network



 Input Layer

 Hidden Layer

 Output Layer



Neurona (función de
activación)



Peso

+ Bias / Sesgo

$$(y = ax + b)$$

Deep Learning: Architectures and Variants

A mostly complete chart of Neural Networks

©2019 Fjodor van Veen & Stefan Lejnen asimovinstitute.org

- Input Cell
- Backfed Input Cell
- Noisy Input Cell
- Hidden Cell
- Probabilistic Hidden Cell
- Spiking Hidden Cell
- Capsule Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- Gated Memory Cell
- Kernel
- Convolution or Pool

Perceptron (P)



Feed Forward (FF)



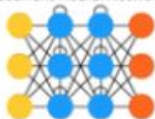
Radial Basis Network (RBF)



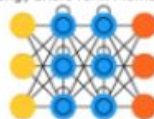
Deep Feed Forward (DFF)



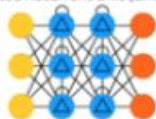
Recurrent Neural Network (RNN)



Long / Short Term Memory (LSTM)



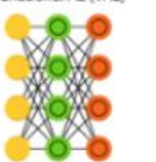
Gated Recurrent Unit (GRU)



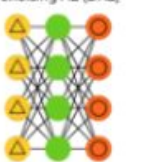
Auto Encoder (AE)



Variational AE (VAE)



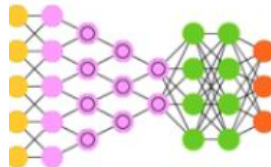
Denoising AE (DAE)



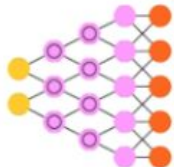
Sparse AE (SAE)



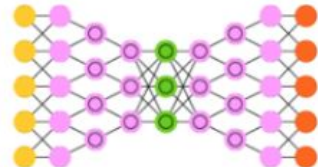
Deep Convolutional Network (DCN)



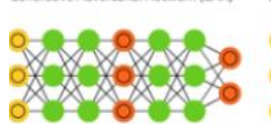
Deconvolutional Network (DN)



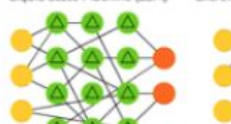
Deep Convolutional Inverse Graphics Network (DIGIN)



Generative Adversarial Network (GAN)



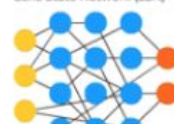
Liquid State Machine (LSM)



Extreme Learning Machine (ELM)



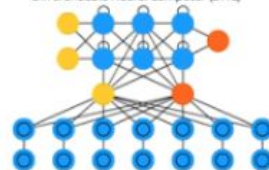
Echo State Network (ESN)



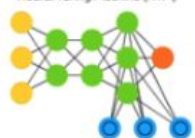
Deep Residual Network (DRN)



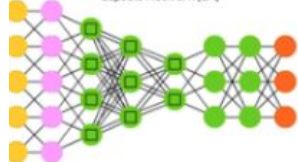
Differentiable Neural Computer (DNC)



Neural Turing Machine (NTM)



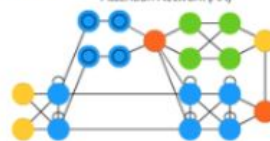
Capsule Network (CN)



Kohonen Network (KN)



Attention Network (AN)



Markov Chain (MC)



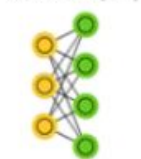
Hopfield Network (HN)



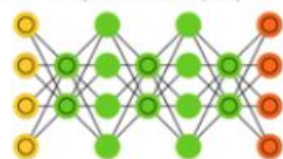
Boltzmann Machine (BM)



Restricted BM (RBM)

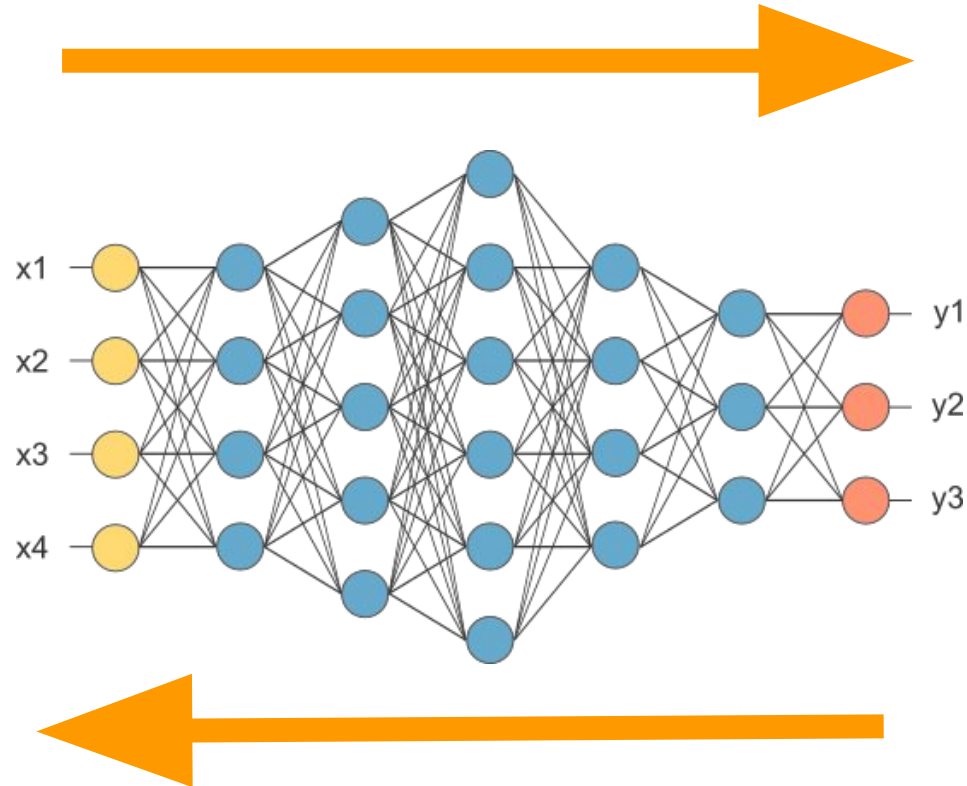


Deep Belief Network (DBN)

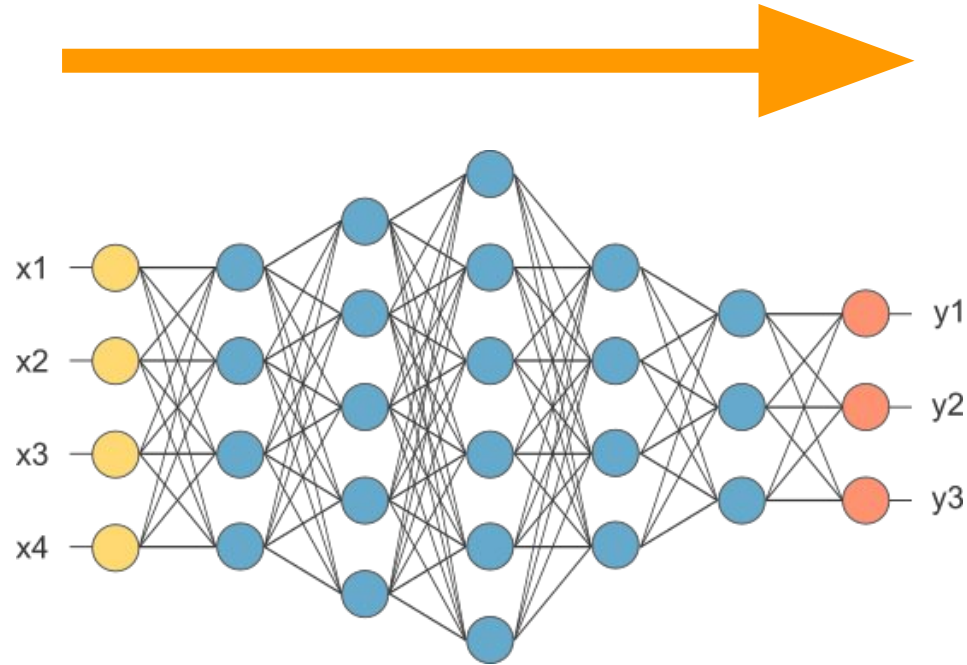


¿Y cómo funciona esto?

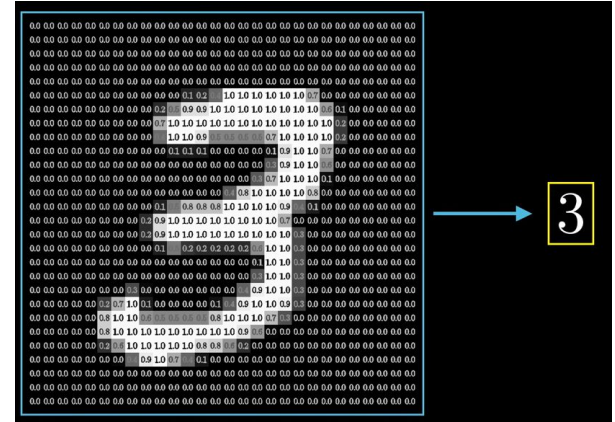
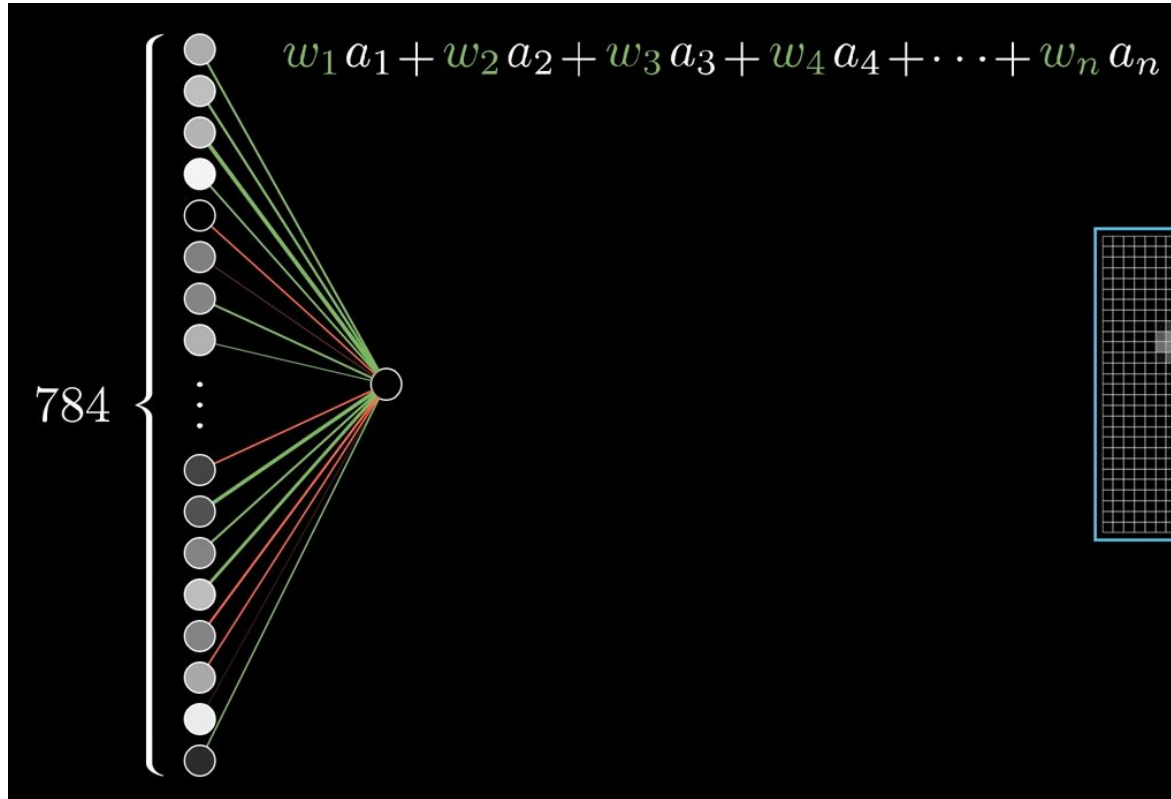
Deep Learning: Forward & Back Propagation



Deep Learning: Forward Propagation (I)

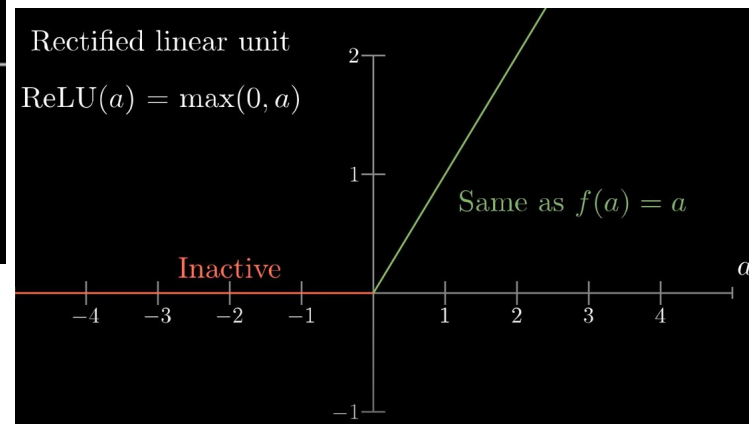
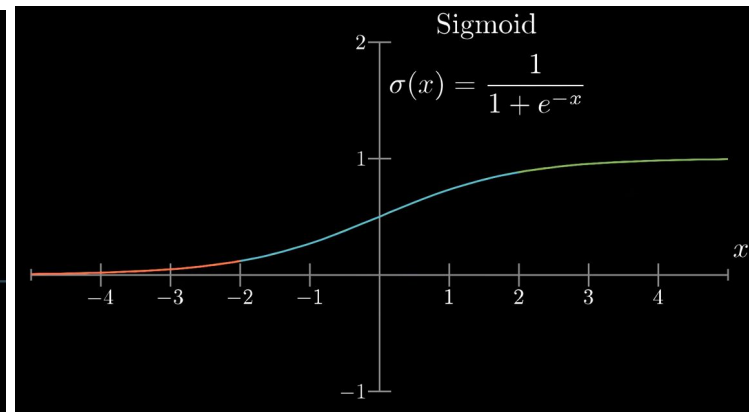
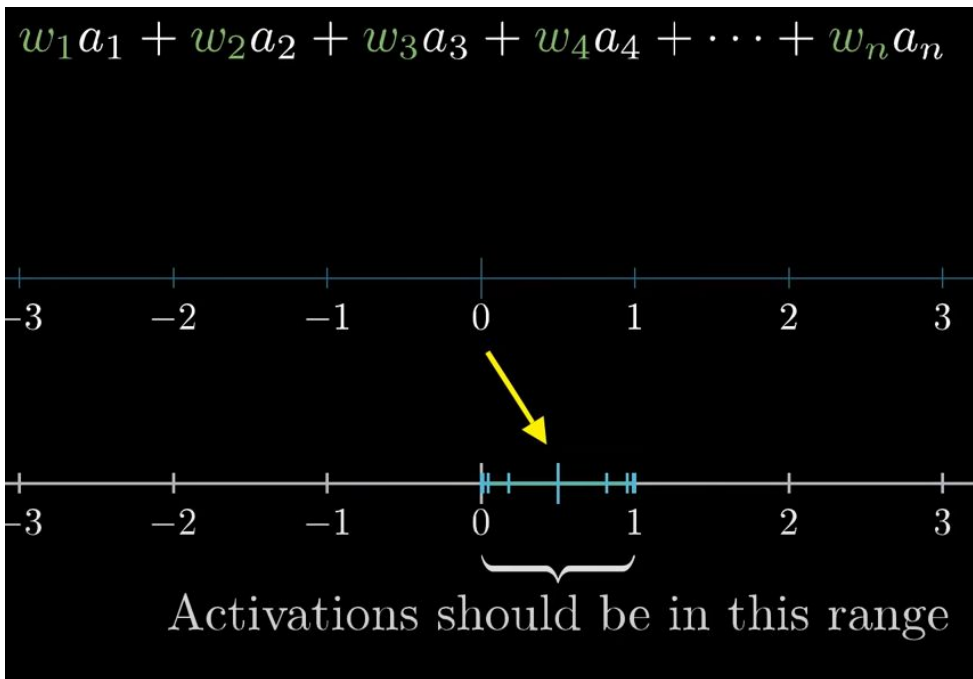


Deep Learning: Forward Propagation (II)



Fuente: 3Blue1Brown

Deep Learning: Forward Propagation (III)



Deep Learning: Forward Propagation (IV)

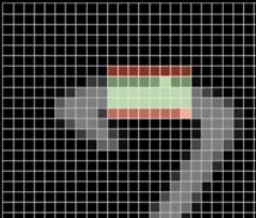
Sigmoid

How positive is this?

$\sigma(w_1a_1 + w_2a_2 + w_3a_3 + \dots + w_na_n \boxed{-10})$

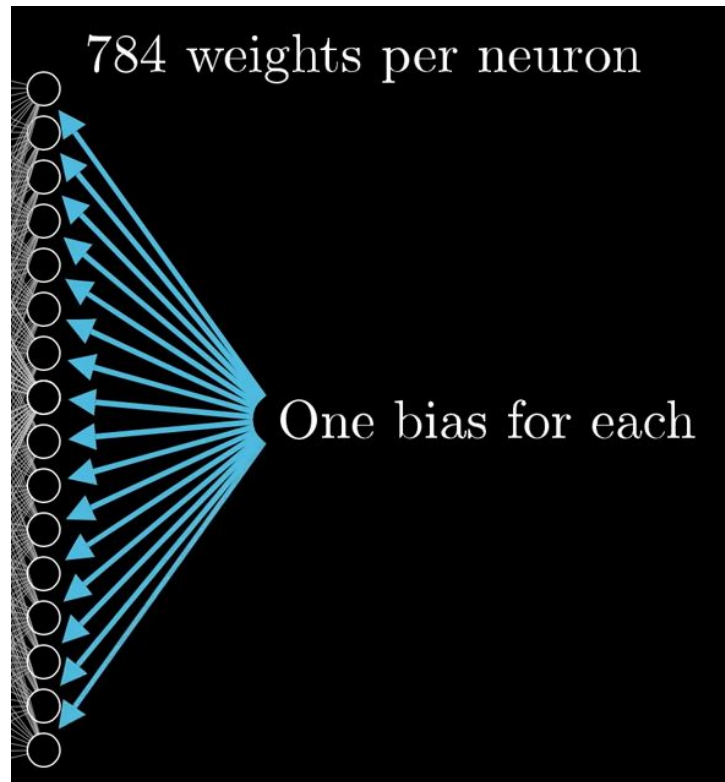
“bias”

Only activate meaningfully
when **weighted sum** > 10



Analogía:
($y = ax + b$)

Fuente: 3Blue1Brown



Deep Learning: Forward Propagation (V)

$$784 \times 16 + 16 \times 16 + 16 \times 10$$

weights

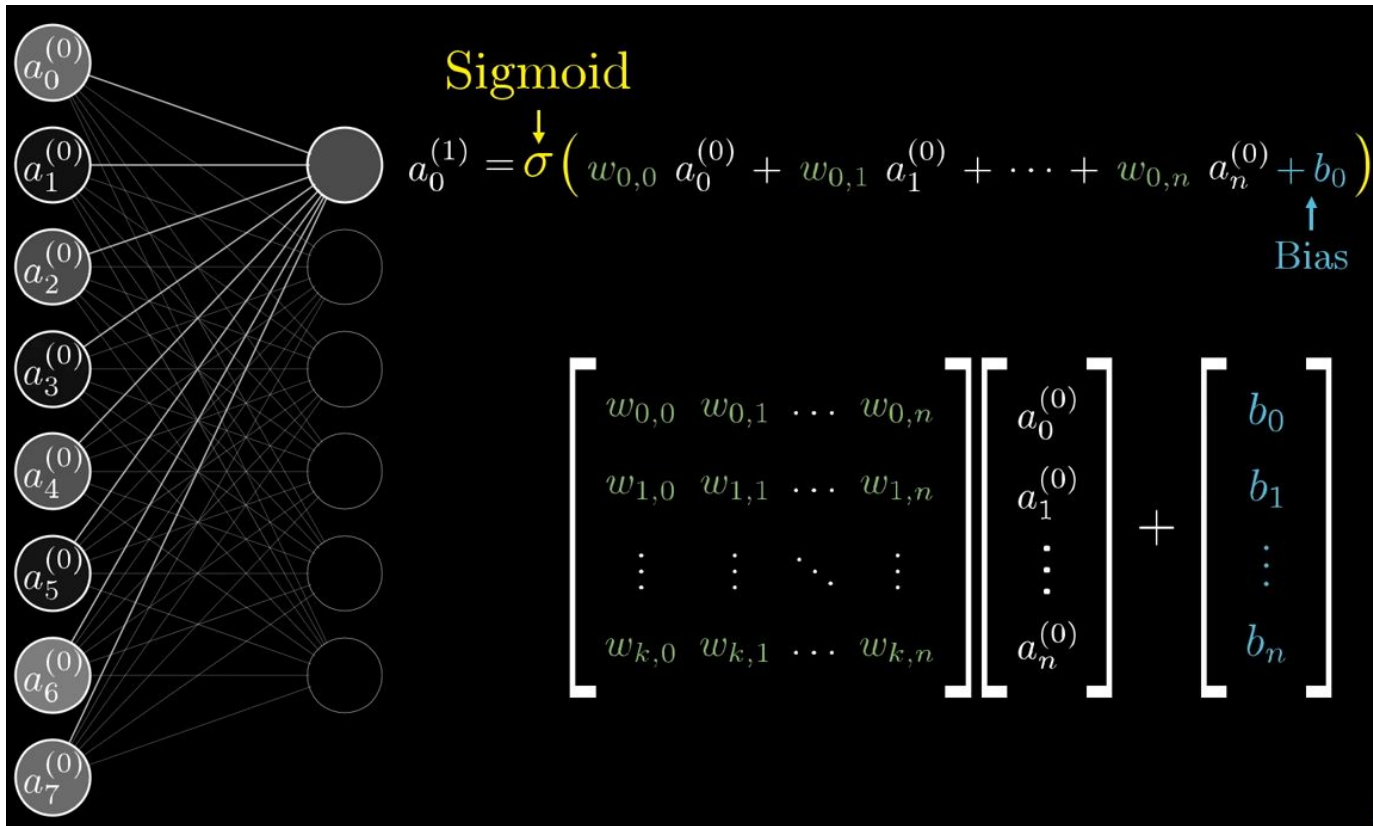
$$16 + 16 + 10$$

biases

13,002

Learning → Finding the right
weights and biases

Deep Learning: Forward Propagation (VI)



Deep Learning: Forward Propagation (VII)

$$\mathbf{a}^{(1)} = \sigma(\mathbf{W}\mathbf{a}^{(0)} + \mathbf{b})$$

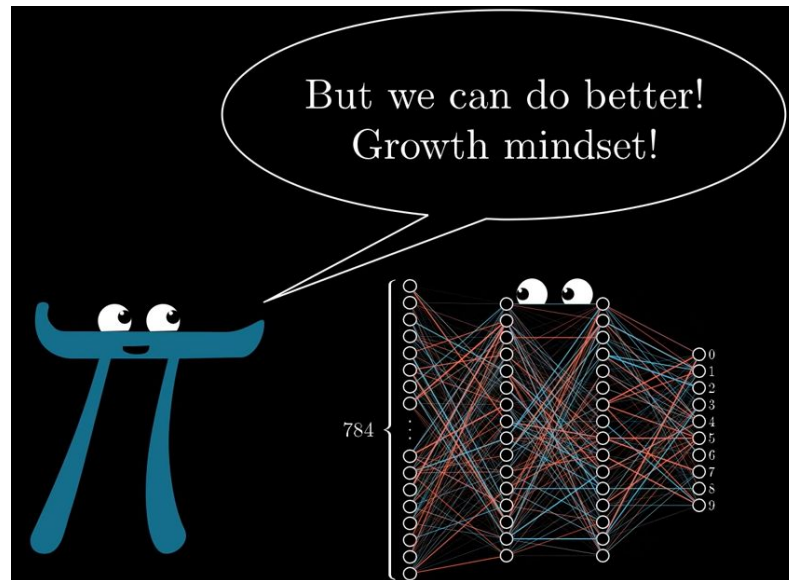
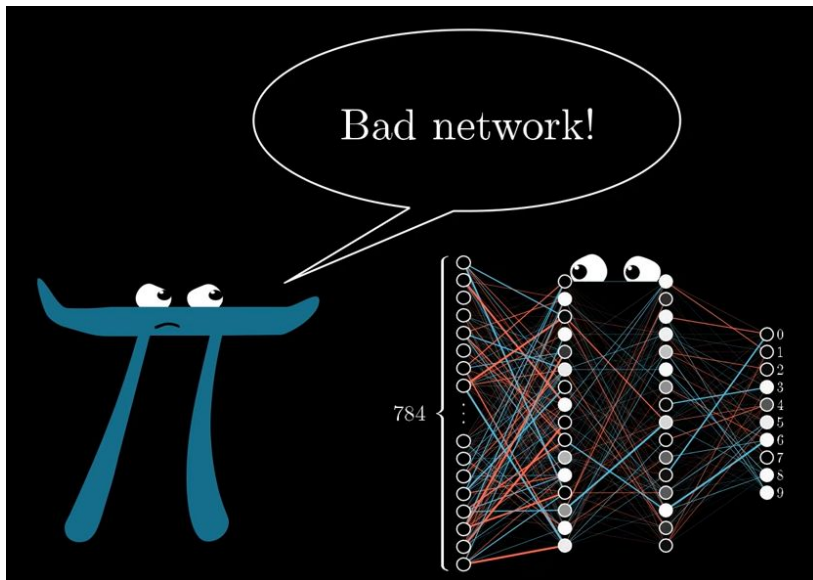
$$\sigma \left(\begin{bmatrix} w_{0,0} & w_{0,1} & \dots & w_{0,n} \\ w_{1,0} & w_{1,1} & \dots & w_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,0} & w_{k,1} & \dots & w_{k,n} \end{bmatrix} \begin{bmatrix} a_0^{(0)} \\ a_1^{(0)} \\ \vdots \\ a_n^{(0)} \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix} \right)$$

Cálculo
optimizado
(Vectorización)

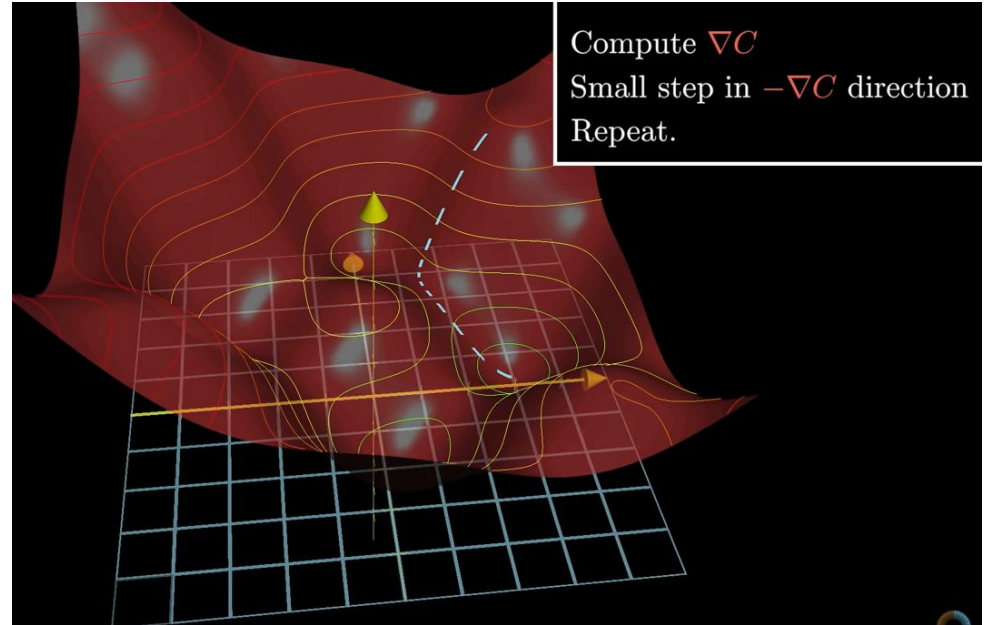
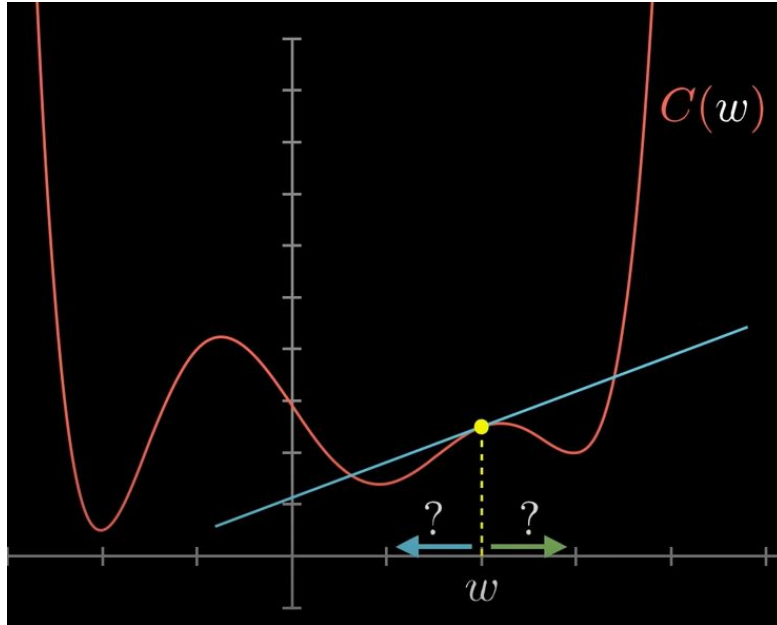
Fuente: 3Blue1Brown

¿Y cómo indicamos a la red que lo está haciendo “bien” o “mal”?

Optimizador: Gradient Descent (I)



Gradient Descent (II)



NOTA: Esto ya lo estabais usando para optimizar diversos modelos (Reg. Lineal, Reg. Logística, SVM)

Gradiente = "Pendiente" (derivada) en múltiples direcciones

Fuente: 3Blue1Brown

Gradient Descent (III)

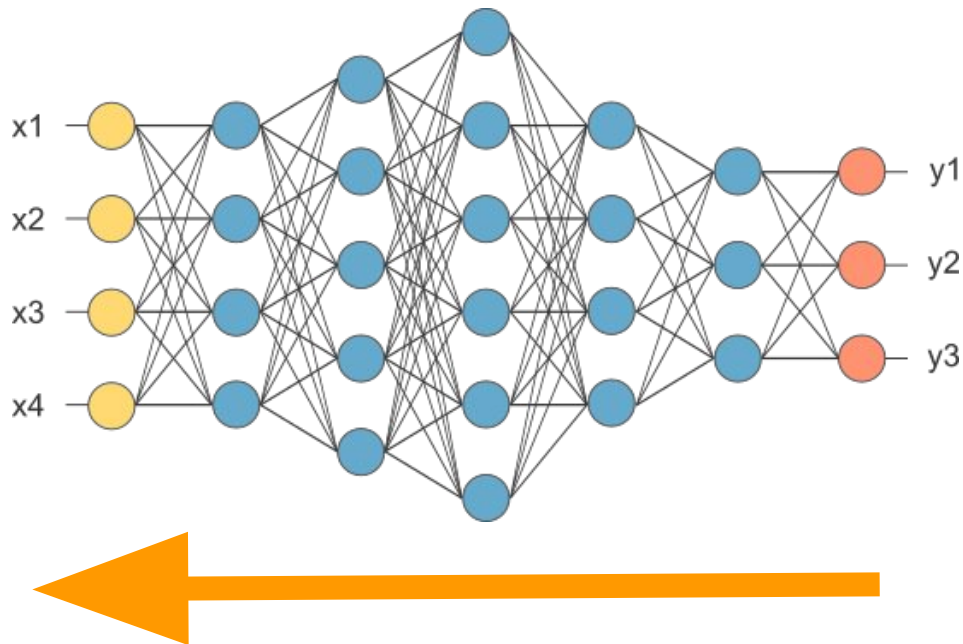
$$-\nabla C(\vec{\mathbf{W}}) = \begin{bmatrix} 0.31 \\ 0.03 \\ -1.25 \\ \vdots \\ 0.78 \\ -0.37 \\ 0.16 \end{bmatrix}$$

w_0	should increase somewhat
w_1	should increase a little
w_2	should decrease a lot
$w_{13,000}$	should increase a lot
$w_{13,001}$	should decrease somewhat
$w_{13,002}$	should increase a little

¿Y cómo se calcula el Gradient Descent en NN?

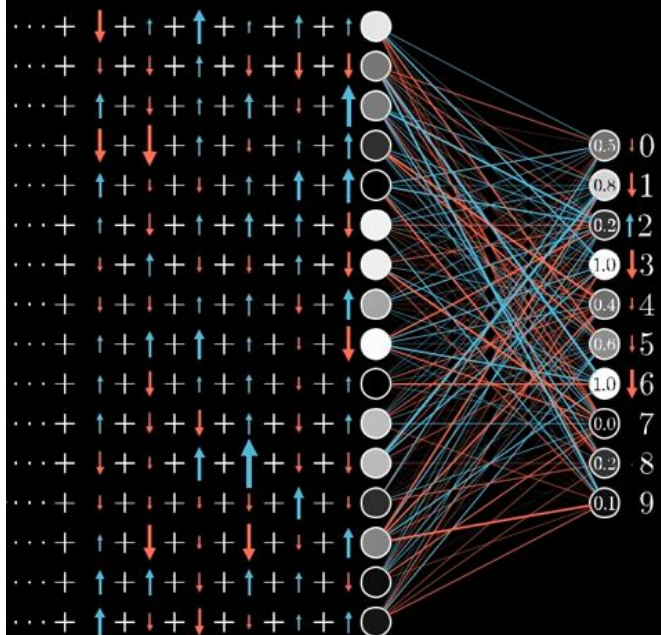
Deep Learning: BackPropagation (I)

Mediante el proceso de backpropagation, se ajustan **pesos y sesgos (weights & biases)** para **optimizar** (en este caso, **minimizar**) la función de coste (por ejemplo, **cross-entropy**) de la red neuronal.



Deep Learning: BackPropagation (II)

Propagate backwards



ALGORITMO (“COMO PARA ANDAR POR CASA”):

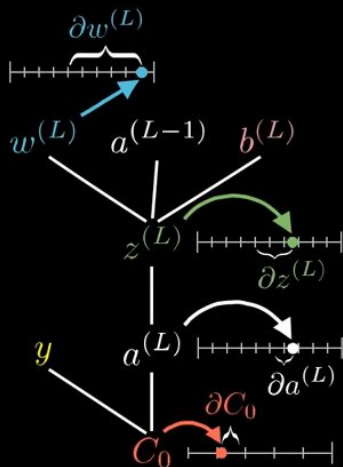
1. Cada neurona calcula la variación que le afecta, ya sea de manera “positiva” (es decir, la predicción es la que se busca en esa neurona) o “negativa” (es una predicción “indeseada”, o en la cual el output no se ha de activar).
2. Recursivamente, se hace el mismo cálculo en cada capa de neuronas, llevando atrás el efecto de cada neurona en el output final y actualizando los pesos.
3. Una vez llegado a la capa de input, se toma la siguiente observación.
4. Vuelve al Paso 1 hasta que tu *training set* se haya utilizado completamente para el entrenamiento.

¿CÓMO? CON **OPTIMIZADORES (GD, SGD, etc.)**

Deep Learning: BackPropagation (III)

$$\frac{\partial C_0}{\partial w^{(L)}} = \frac{\partial z^{(L)}}{\partial w^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C_0}{\partial a^{(L)}}$$

Chain rule

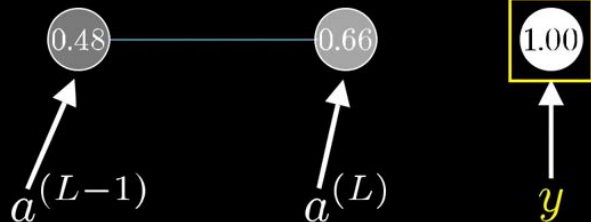


$$C_0(\dots) = (a^{(L)} - y)^2$$

$$z^{(L)} = w^{(L)} a^{(L-1)} + b^{(L)}$$

$$a^{(L)} = \sigma(z^{(L)})$$

Desired output



Deep Learning: BackPropagation (IV)

$$z^{(2)} = XW^{(1)} \quad (1)$$

$$a^{(2)} = f(z^{(2)}) \quad (2)$$

$$z^{(3)} = a^{(2)}W^{(2)} \quad (3)$$

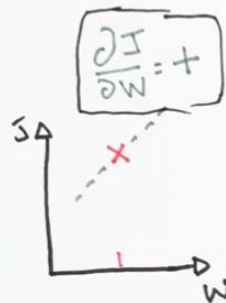
$$\hat{y} = f(z^{(3)}) \quad (4)$$

$$J = \sum \frac{1}{2} (y - \hat{y})^2 \quad (5)$$

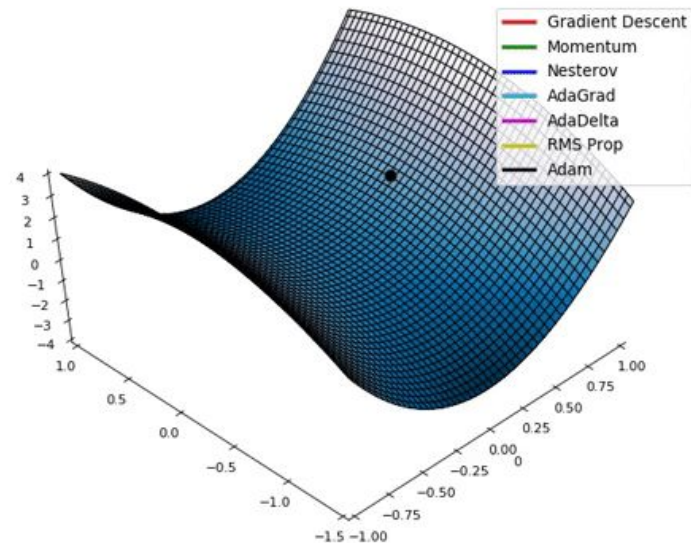
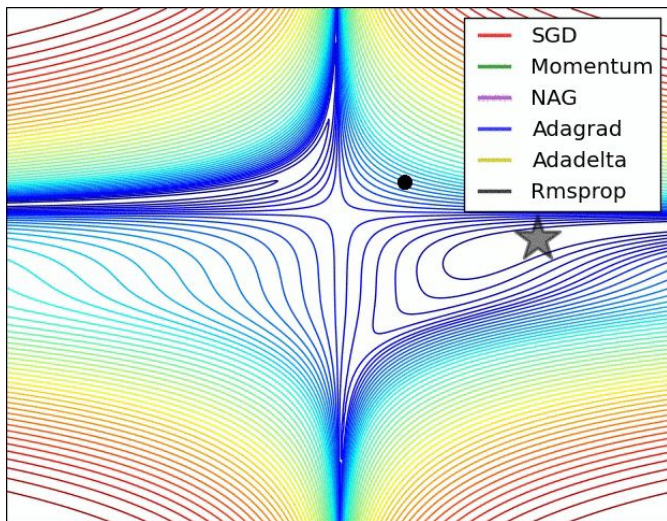
$$J = \sum \frac{1}{2} (y - f(f(XW^{(1)})W^{(2)}))^2$$

↑ HOW DOES THIS CHANGE
IF I CHANGE THESE?

$$\frac{\partial J}{\partial W}$$

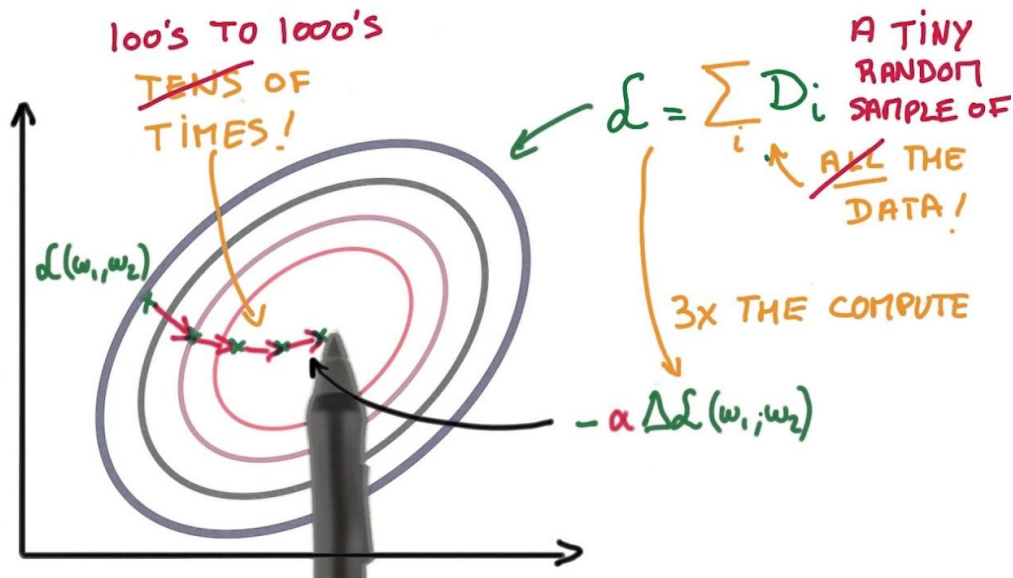


Optimizers



Problema de Gradient Descent: El PC muere... :- (→ SGD (o Adam, SGD “mejorado” con medias móviles)

Stochastic Gradient Descent (SGD) (I)



NEURAL NETWORK USING STOCHASTIC GRADIENT DESCENT

ALGORITMO:

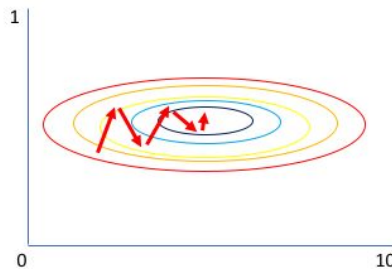
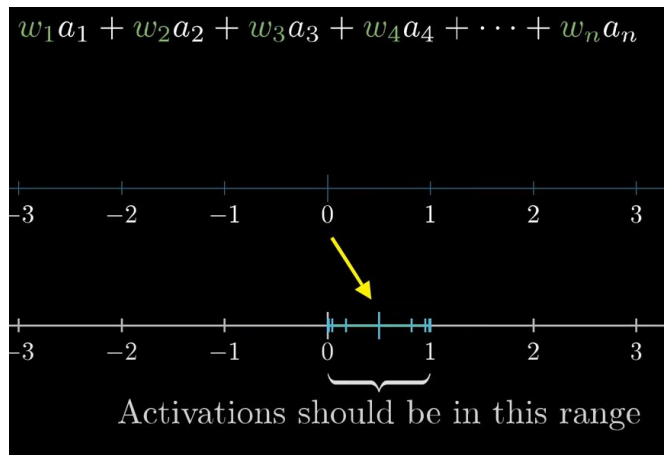
1. Se hace un barajeo aleatorio o *random shuffling* de los datos.
2. Se divide el dataset en cachos o *mini-batches*.
3. Procesa el Gradient Descent de un *mini-batch* para calcular el gradiente promedio.
4. Usa el gradiente promedio del *mini-batch* para actualizar la red, y vuelve al Paso 3.
5. Cuando hayas terminado, profit! (๐ ͡ ๐)

VENTAJA:

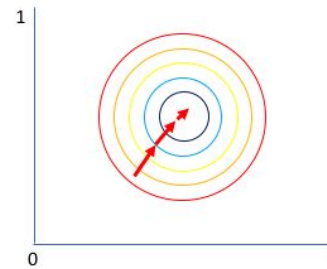
Al dividir el cálculo en mini-batches se puede **vectorizar** (y por lo tanto, paralelizar), así que el entrenamiento es mucho más rápido que en el GD convencional.

SGD (III)

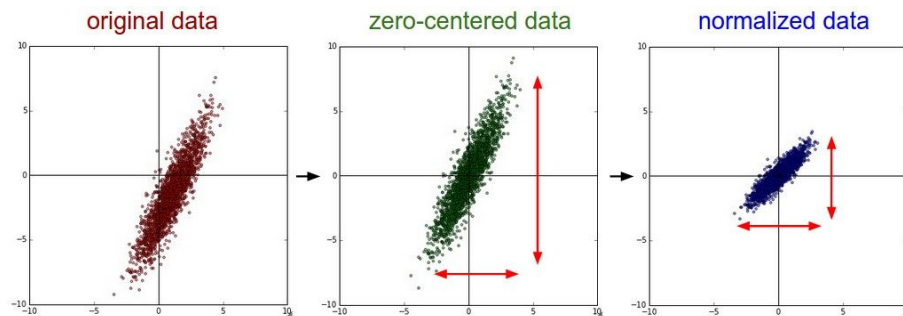
Why normalize?



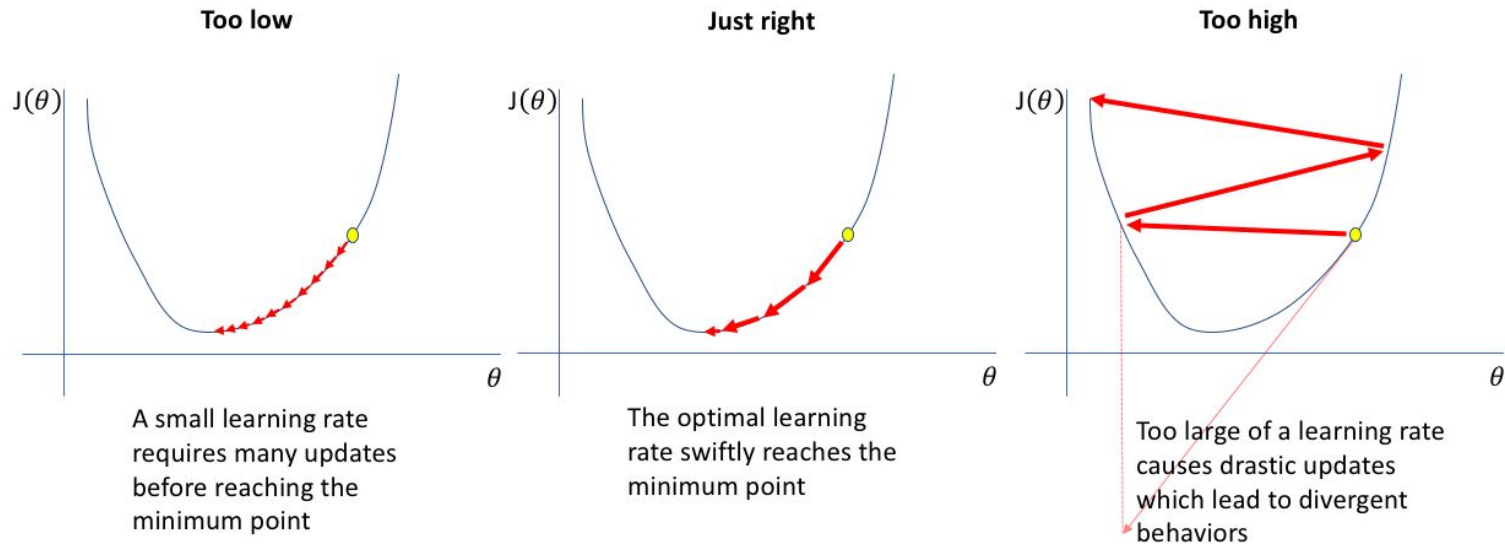
Gradient of larger parameter dominates the update



Both parameters can be updated in equal proportions



Learning Rates (alpha)



Understand the Impact of Learning Rate on Neural Network Performance

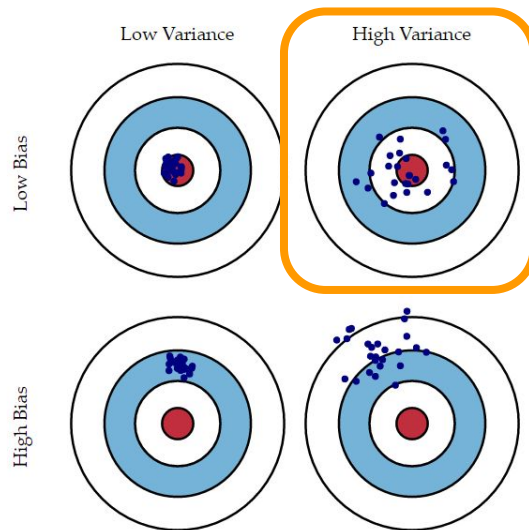
Y si esto funciona como el ML... ¿puede overfittear una NN?

EN EFECTO

Regularización: Métodos (I)

NOTA: Al resolver overfitting, buscamos corregir un escenario donde tenemos mucha varianza y poco sesgo (bias) en los datos.

No se trata por lo tanto de optimizar la función de coste.



Regularización: Métodos (II)

1) Dropout

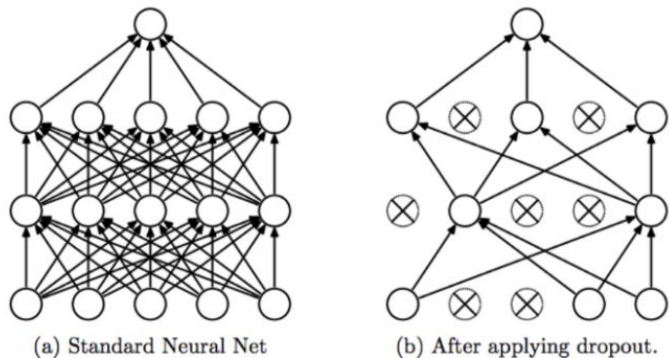
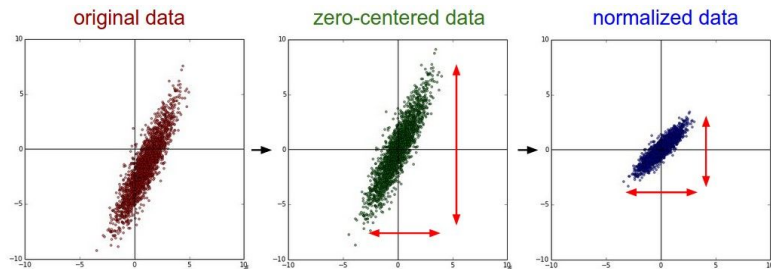


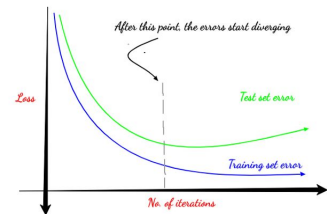
Image source: Google

2) Batch Normalization



3) **Data Augmentation** (p. ej. voltear imágenes)

4) **Early Stopping** (no “sobreentrenar”)



5) **Regularización L1** (restar efecto a gradiente)

6) **Regularización L2** (sumar componente a loss para suavizar pequeñas actualizaciones de pesos)

¿Y qué herramientas se usan para construir NNs?

 Keras **vs.**  PyTorch



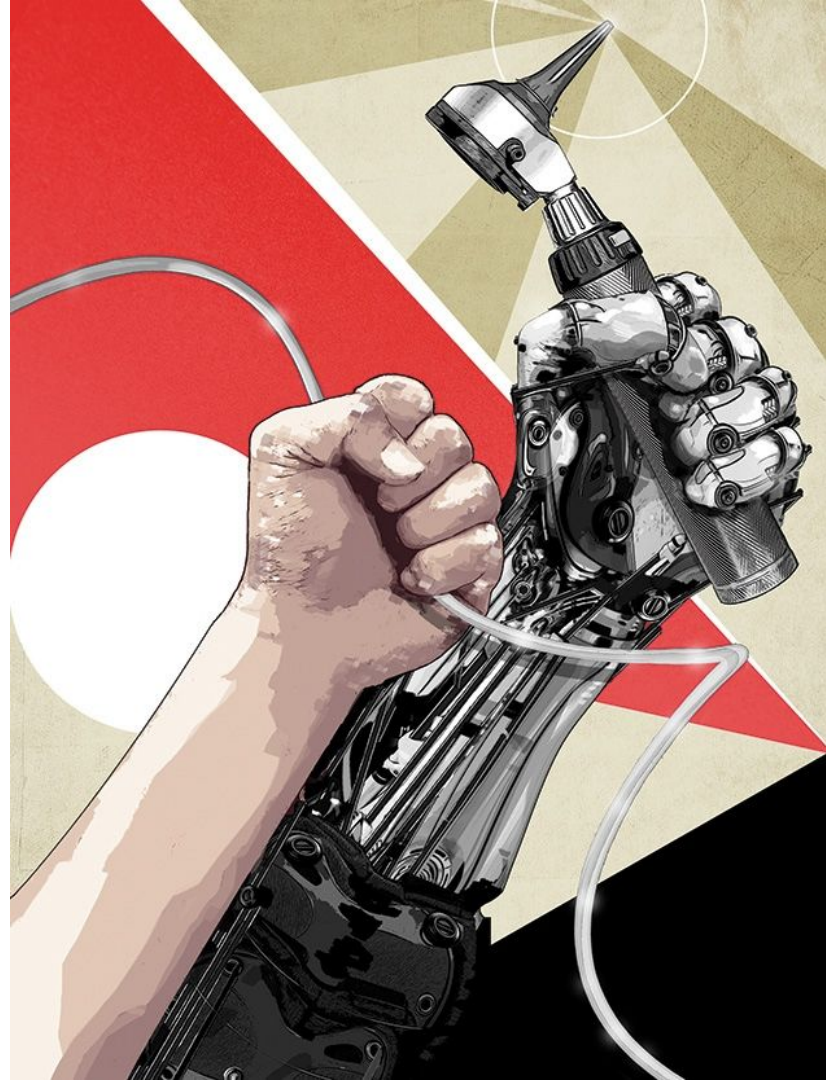
theano



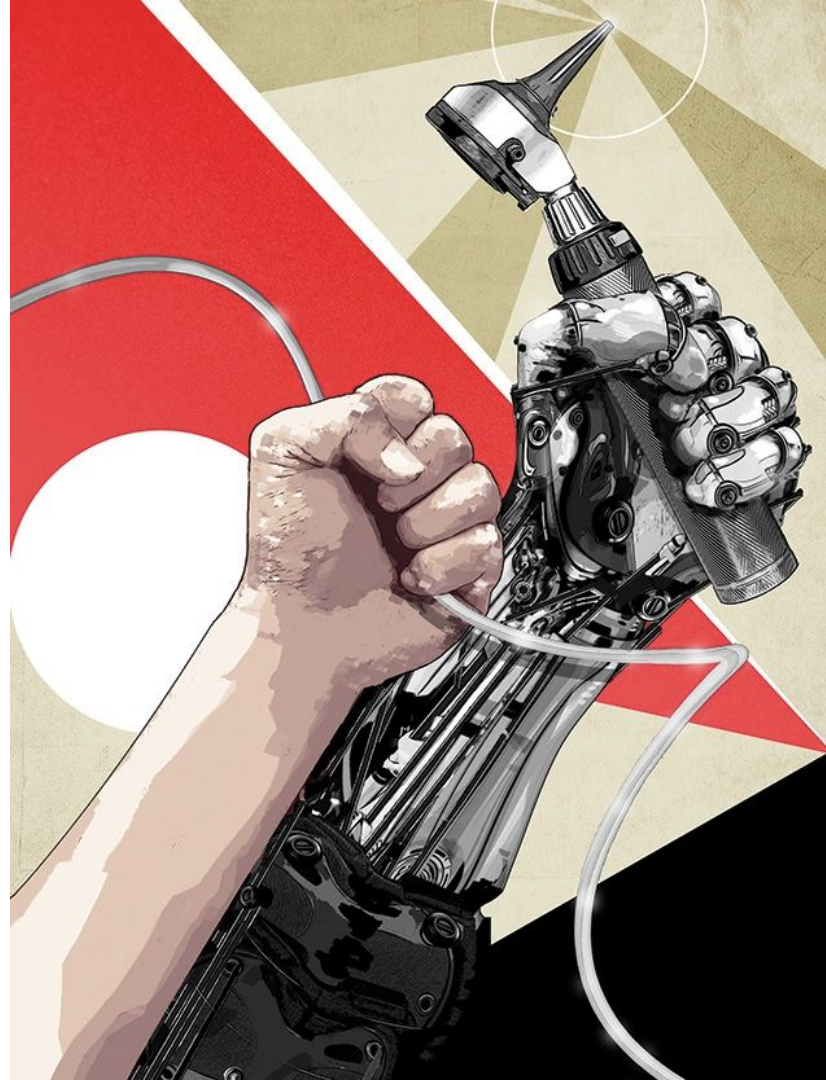
(aka CNTK)

Keras vs Pytorch for Deep Learning

Practice - ANN!



Challenge!



Bibliografía

/1./ /Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow/

/2./ /Fast.AI - Introduction to Machine Learning for Coders/

/3./ /MLCourse.AI/

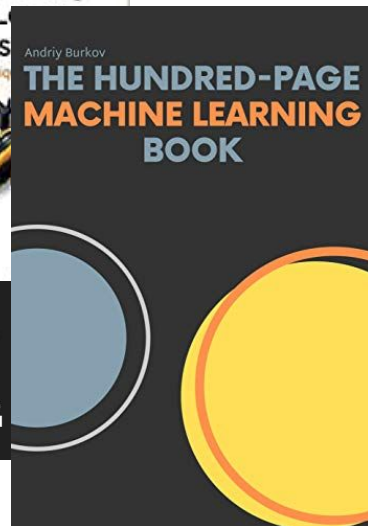
/4./ /DeltaAnalytics/

/5./ /The Hundred-page Machine Learning Book/

/6./ /Machine Learning for Humans (Vishal Maini)/

/7./ /Datacamp/

/8./ /DataQuest/



Partners y Equipo Organizador

Agradecemos a nuestros partners por confiar en **nosotros** para facilitar la formación en **IA** de cara a la 4ª Revolución Industrial.





Saturdays.AI

#1 Neural Networks: Basics

by Saturdays.AI

Saturdays.AI Donostia
Deep Learning

