

# Introducción a PyTorch

Enrique Escalante-Notario

## Outline

- D Regresión y varianza.
- D Instalación
- D Manejo básico de PyTorch
- D Ejemplo
  - Red de convolución de imágenes.

# PyTorch

From Wikipedia, the free encyclopedia

**PyTorch** is an [open source](#) machine learning library based on the [Torch](#) library,<sup>[3][4][5]</sup> used for applications such as computer vision and natural language processing,<sup>[6]</sup> primarily developed by Facebook's AI Research lab (FAIR).<sup>[7][8][9]</sup> It is free and open-source software released under the [Modified BSD license](#). Although the [Python](#) interface is more polished and the primary focus of development, PyTorch also has a [C++](#) interface.<sup>[10]</sup>

A number of pieces of Deep Learning software are built on top of PyTorch, including Tesla Autopilot<sup>[11]</sup>, Uber's Pyro,<sup>[12]</sup> HuggingFace's Transformers,<sup>[13]</sup> PyTorch Lightning<sup>[14][15]</sup>, and Catalyst.<sup>[16][17]</sup>

PyTorch provides two high-level features:<sup>[18]</sup>

- Tensor computing (like [NumPy](#)) with strong acceleration via [graphics processing units \(GPU\)](#)
- Deep neural networks built on a tape-based [automatic differentiation](#) system

# PyTorch

- ▷ Es un framework Python 3.X (X=7)
- ▷ Alguno de sus puntos "Tensorflow Killer"
- ▷ Basado en numpy
- ▷ Es usado en el Deep Learning;  
"yo" se programó flexible y sencillo.

# Alternatives to PyTorch

- ③ TensorFlow {
  - calculates neurons
  - mediante graph
- ④ Theano {
  - calcula simbolica
  - de forma.
- ⑤ Keras {
  - Rede neural

## Interés a lo largo del tiempo

Google Trends

● PyTorch ● TensorFlow ● Scikit-learn



# ¿Por qué PyTorch?

- ▷ Python
- ▷ Rápido crecimiento
- ▷ Un mundo de soporte para la ciencia
- ▷ Sencillas implementaciones
- ▷ Trabajar con gráficos dinámicos
- ▷ Soporte para CUDA

## Instalaciones

▷ Python 3.x ✓

▷ Pip, cardo o un instalador de paquetes  
de Python ✓

(1) X Si, tenues una GPU se pueblan procesos

(-1) X Podemos usar Pytorch sobre C++/Java.

▷ Podemos elegir entre la rama de  
desarrollo o estable ✓

## Instalaciones

- ▷ Python 3.x ✓
- ▷ Pip, conda o un instalador de paquetes de Python ✓
- ▷ Puedes elegir entre la rama de desarrollo o estable ✓

```
> sudo pip3 install torch torchvision torchaudio  
matplotlib numpy utils
```

## Instalación

- ▷ Python 3.x ✓
  - ▷ Pip, conda o una instalación de paquetes de Python ✓
  - ▷ Podemos elegir entre la rama de desarrollo o estable ✓
- > sudo pip3 install jupyter
- Alternativo, pero útil para aprender

### Terminal - python3

```
→ ~ python3
Python 3.8.5 (default, Jul 28 2020, 12:59:40)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import torch
>>> import torchvision
>>> x = torch.rand(5,3)
>>> print(x)
tensor([[0.5280, 0.9970, 0.7704],
       [0.1556, 0.6320, 0.7765],
       [0.2484, 0.7608, 0.1847],
       [0.1539, 0.1815, 0.6031],
       [0.4293, 0.2727, 0.5217]])
>>> torch.cuda.is_available()
/usr/local/lib/python3.8/dist-packages/torch/cuda/__init__.py:52: UserWarning: CUDA initialization: Found no NVIDIA driver on your system. Please check that you have an NVIDIA GPU and installed a driver from http://www.nvidia.com/Download/index.aspx (Triggered internally at /pytorch/c10/cuda/CUDAFunctions.cpp:100.)
    return torch._C._cuda_getDeviceCount() > 0
False
>>> █
```

Voy a ver en libreta de Jupyter.

```
Terminal - jupyter notebook
→ ~ jupyter notebook
[I 10:31:47.082 NotebookApp] Serving notebooks from local directory: /home/enrique
[I 10:31:47.083 NotebookApp] Jupyter Notebook 6.1.4 is running at:
[I 10:31:47.083 NotebookApp] http://localhost:8888/?token=a17b7562d0f2e42d46e381
1817285d6636d8d1a1008fbefd
[I 10:31:47.083 NotebookApp] or http://127.0.0.1:8888/?token=a17b7562d0f2e42d46
e3811817285d6636d8d1a1008fbefd
[I 10:31:47.083 NotebookApp] Use Control-C to stop this server and shut down all
kernels (twice to skip confirmation).
[C 10:31:47.133 NotebookApp]

To access the notebook, open this file in a browser:
file:///home/enrique/.local/share/jupyter/runtime/nbserver-337117-open.h
tml
Or copy and paste one of these URLs:
http://localhost:8888/?token=a17b7562d0f2e42d46e3811817285d6636d8d1a1008
fbefd
or http://127.0.0.1:8888/?token=a17b7562d0f2e42d46e3811817285d6636d8d1a1008
fbefd
```

Home Page - Select or create a new notebook

localhost:8888/tree

# jupyter

Files    Running    Clusters

Duplicate    Rename    Move    Download    View    Edit   

Upload    New   

	Name	Last Modified	File size
<input type="checkbox"/>	Descargas	hace 2 minutos	
<input type="checkbox"/>	Documentos	hace 16 horas	
<input type="checkbox"/>	Downloads	hace 5 días	
<input type="checkbox"/>	Escritorio	hace 5 días	
<input type="checkbox"/>	Imágenes	hace un minuto	
<input type="checkbox"/>	Plantillas	hace 2 meses	
<input type="checkbox"/>	snap	hace 10 días	
<input type="checkbox"/>	2020-11-27-Note-16-05.pdf	hace un minuto	5.81 MB
<input type="checkbox"/>	2020-11-27-Note-17-28.xoj	hace un minuto	726 kB
<input checked="" type="checkbox"/>	orbi2ml	hace un mes	384 B

Values a copy for.

# Evolución de las redes neuronales

## Etapas I

1958 - Perceptron

1965 - Multilayer perceptron.

1980

- Neuronas sigmoidales
- Redes Feed forward
- Fully connected

$$d(z) = \frac{1}{1 + e^{-z}}$$

Biases      ||  
Pesos      ||  
Manojo      ||  
Multicapa      ||  
Difícil de      ||  
calcular

faltantes      ||  
Primeras funciones  
de activación      ||  
- Sigmoidal  
- Bias      ||  
Control      ||  
los estados en      ||  
aproximadas personas

## Etapas 2

1986 - Back propagation

No hay loops !!  
Todos los  
neuronas ↴  
Conectadas.

Rodar  
entrena de  
modo supervisado  
a través de  
prior

1989 - Convolutional Neural Network

{ Una capa extenderá info y luego clasificarán.  
Se reduce el número de features manteniendo la info más importante.  
Las últimas capas tiene nodos para cada clasificación.

1997 Learning short term  
Memory / Recurrent  
Neural Network

Conexión hetero  
áreas ~ Señal temporal  
Alta tasa de  
niveles de  
memoria

### Etapas 3

Deep Learning

2006 Deep Belief Network.

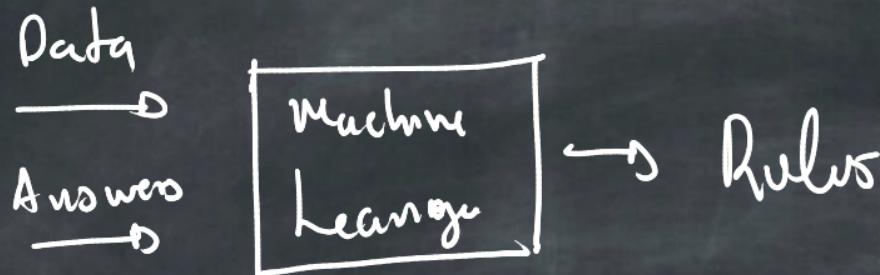
Autoencoder con Restricted  
Boltzmann Machine

No tiene el  
poder de generalización  
Presentando  
no supervisado  
Evitar caer en  
minimos locales  
por Gradi Decent-

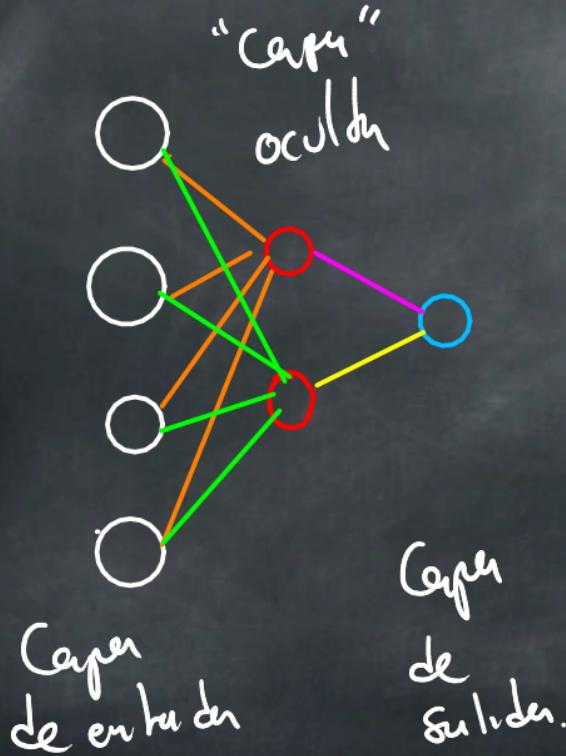
2014 Generative Adversarial  
Network

Se enthalten 2  
verb. Gnerende,  
Pizivimverder.

# Machine learning.



Una red neuronal **no es un program**, es un modelo para  
resolver problemas, un sistema que procesa información.



Hay conexiones fuertes  
y débiles, se determinan  
por la info.

La red se caracteriza  
por

- Su organización
- Su aprendizaje

- Arquitectura;

① Configuración de conexiones.

- + feed forward
- + Recurrent
- + Multilayers
- + Single layers

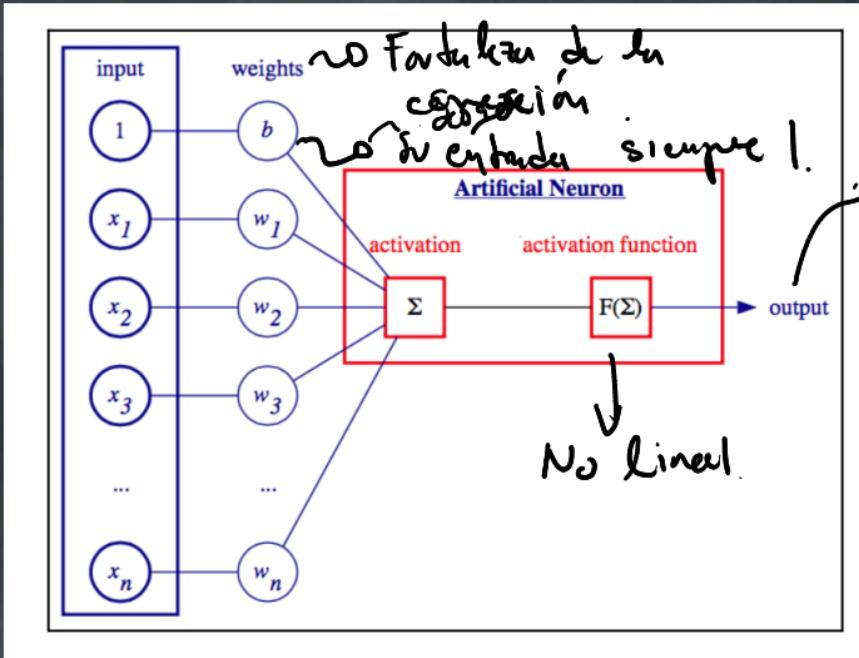
② Número de neuronas en cada capa.

- Aprendizaje ~ entrenamiento

El más popular: Gradient descent  
y Back propagation.

# Partes de un neurón.

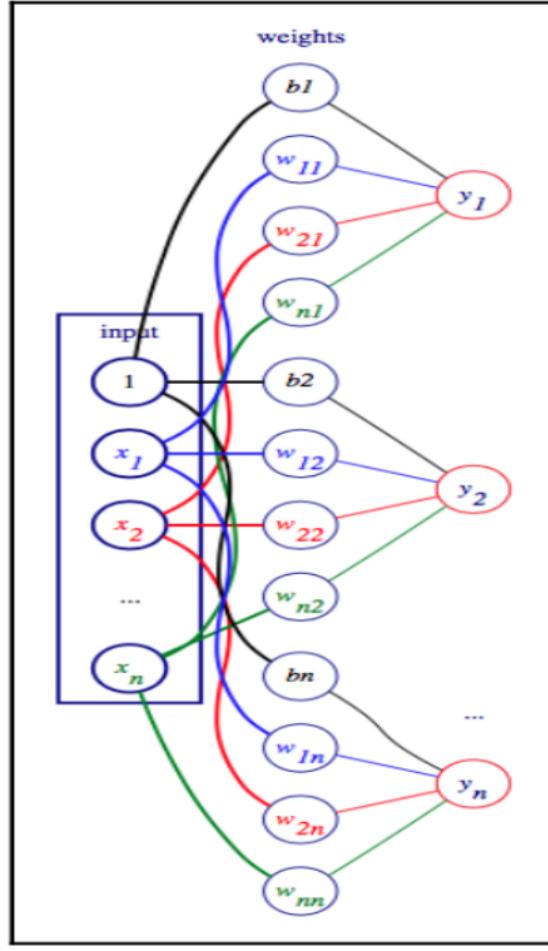
b m  
de podr  
de reproducción.



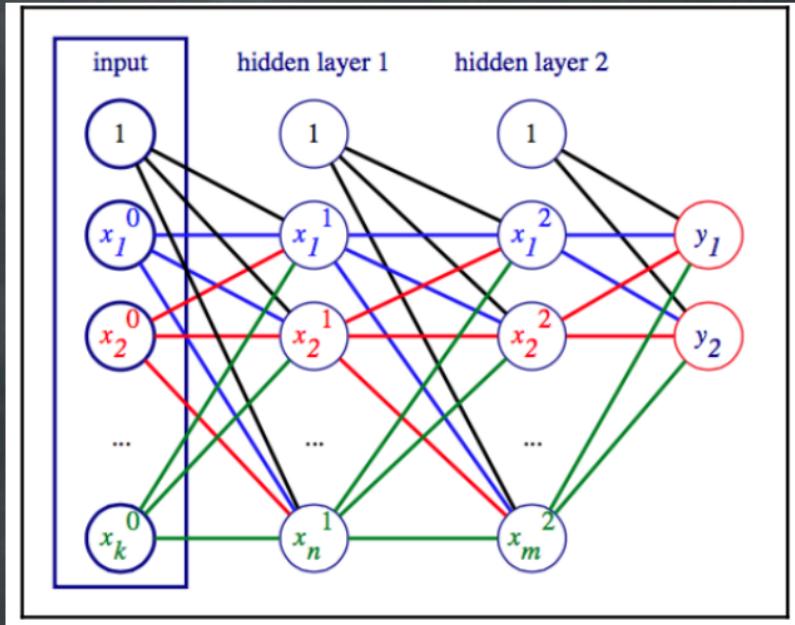
$$y = f(\sum x_i w_i + b)$$

Cafes

Red  
unicapra

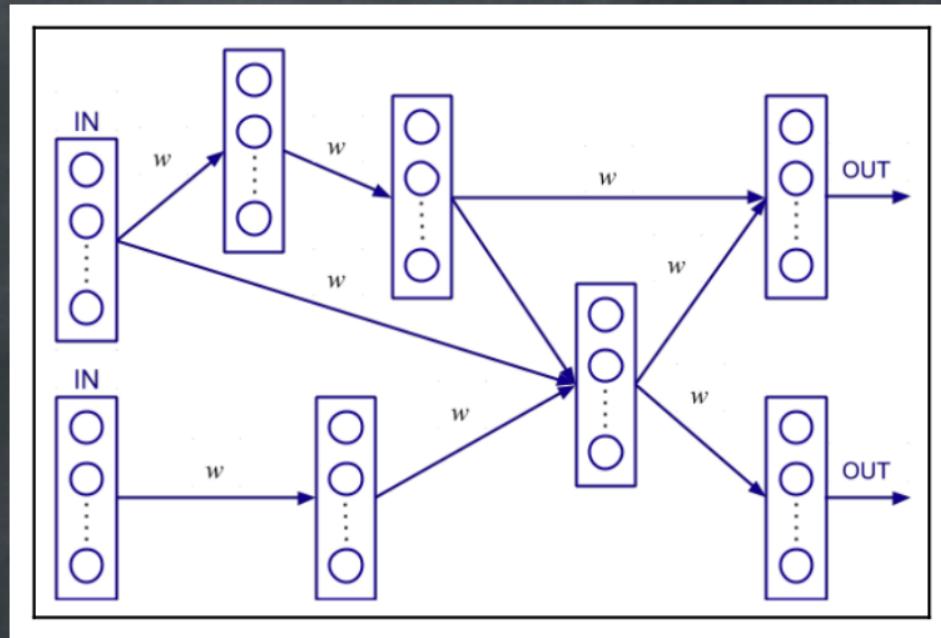


3-layered  
fully  
connected



No loops

Convolve  
sequentially



## Funció de activació

- Són clars el model genèric línies  
~ Regressió línies.
- Generalment, totes les neurones en un corpi tenen la mateixa funció, però difereixen en difèrents cossos.

Los más comunes

- Identidad

$$f(a) = a$$

- Escalón

$$f(a) = \begin{cases} 1 & \text{si } a > 0 \\ 0 & \text{si } a \leq 0 \end{cases} \quad (\text{Binario})$$

más común • Logistic

$$f(a) = \frac{1}{1 + \exp(-a)} \quad (\text{sigmoid})$$

- Bipolar  
sigmoid

$$f(a) = \frac{2}{1 + \exp(-a)} - 1$$

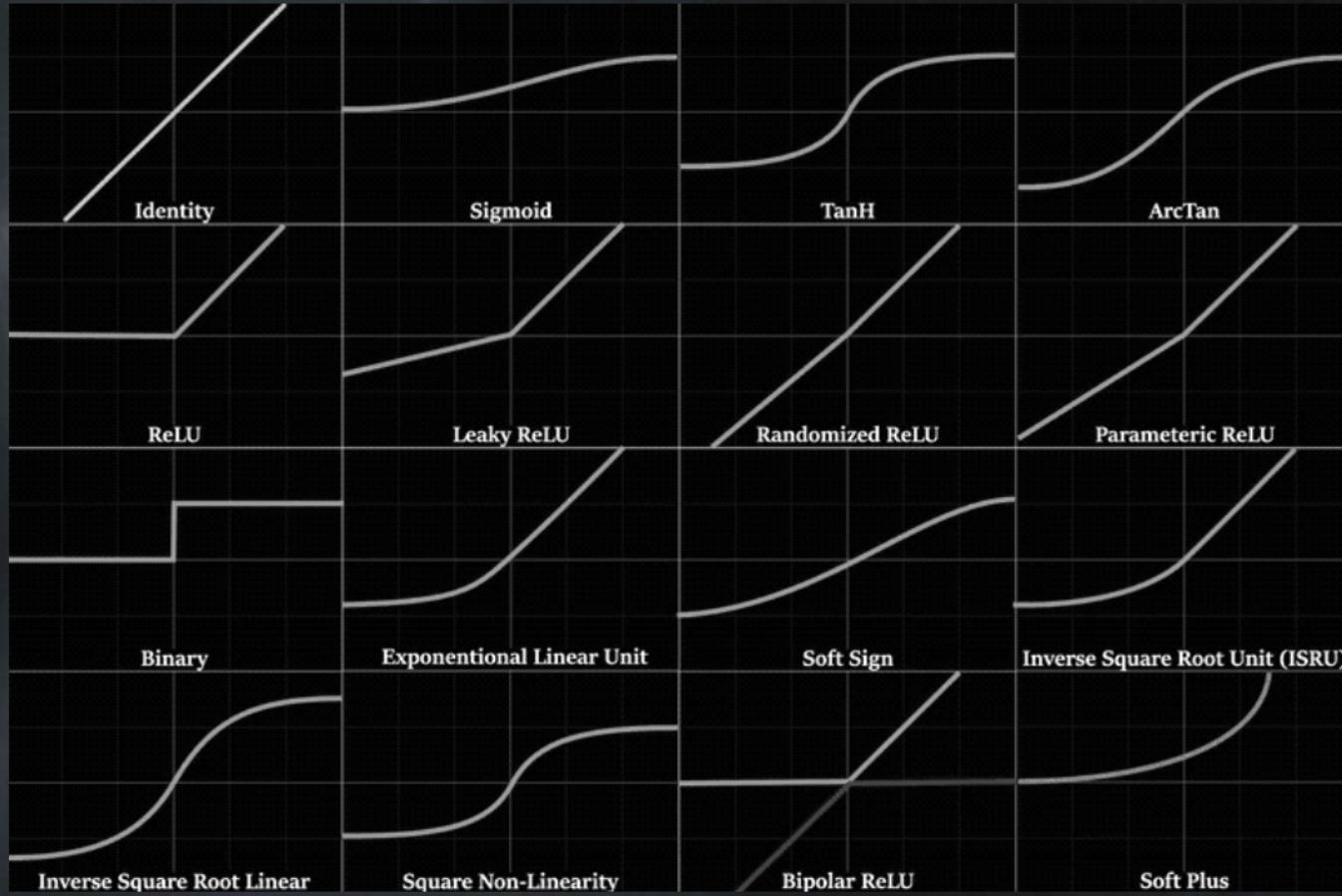
• Tanh

$$f(a) = \frac{1 - \exp(-2a)}{1 + \exp(-2a)}$$

• ReLU

Rectified Linear Unit

$$f(a) = \begin{cases} a & \text{si } a > 0 \\ 0 & \text{si } a \leq 0 \end{cases}$$



## Entrenamiento (Izq → Derecha)

- Necesitas los pesos para determinar los costos informando de cada neurona en la red.
- Necesitas un optimizador ~ gradient descent

"Todas red es una aproximación de una función"

$$F(x_i) \approx N(x_i) + D$$

~ Error.

## Função de custo (erro).

$$\frac{\partial J}{\partial w_{ij}} = \frac{\partial J}{\partial y_i} \frac{\partial g_i}{\partial a_i} \frac{\partial a_j}{\partial w_{ij}}$$

-  $w_{ij}$  peso de la i-esima neurona da la capa l  
y la j-esima neurona da la capa  $l+1$

-  $y_i$  Es la cantidad de la siguiente capa;  
(función de activación)

-  $a_i$  es la actividad  $\vec{h} \circ \vec{w}$

- $\frac{\partial g_{ij}}{\partial w_{ij}} = g_i \leftarrow o_j$
- $\frac{\partial g_i}{\partial g_j}$  es la derivada de los pesos de los activos
- $\frac{\partial J}{\partial g_i}$  es la derivada con respecto a los pesos de los activos de la "suma de" error.

- Calcularon otros derivadas para las otras capas.
- Podemos calcular para la siguiente

$$\frac{\partial J}{\partial y_i} = \sum_j \frac{\partial J}{\partial y_j} \frac{\partial g_j}{\partial a_i}$$

$y_i$  salida "prima"  
 $y_j$  salida "segunda"

Feed Forward

$y_i$  depende  
de  $TODAS$  las  
salidas "anteriores"

 $\leftarrow = \sum_j \frac{\partial J}{\partial y_j} \frac{\partial g_j}{\partial a_j} \frac{\partial a_j}{\partial y_i}$ 

Contribuyen al error BP

Počítač se kalkulu

- $\frac{\partial y_i}{\partial a_j}$
- $\frac{\partial a_j}{\partial y_i} = w_{ij}$
- $\frac{\partial J}{\partial y_j} \rightarrow \frac{\partial J}{\partial y_i} \rightarrow \frac{\partial J}{\partial w_{ij}}$

En venner:

Si tener un vector de capas

$$y_i \rightarrow y_j \rightarrow y_k.$$

Hay 2 caminos fundamentales

$$\frac{\partial J}{\partial w_{i,j}} = \frac{\partial J}{\partial y_j} \frac{\partial y_j}{\partial a_j} \frac{\partial a_j}{\partial w_{i,j}}$$

$$\frac{\partial J}{\partial y_j} = \sum_k \frac{\partial J}{\partial y_k} \frac{\partial y_k}{\partial y_j}$$

Usando esto calcular la derivada de  
función de costo en cada capa.

Error  
en cada  
nervio  
 $y_j$

$$\delta_j = \frac{\partial J}{\partial y_j} \frac{\partial y_j}{\partial a_j}$$

Variación de la  
función de costo  
en respecto al valor  
de activación

Recorbanos

$$\begin{aligned}\frac{\partial J}{\partial y_i} &= \sum_j \frac{\partial J}{\partial y_j} \frac{\partial y_j}{\partial a_i} = \sum_j \frac{\partial J}{\partial y_j} \frac{\partial y_j}{\partial a_j} \frac{\partial a_j}{\partial y_i} \\ &= \sum_j \delta_j w_{ij}\end{aligned}$$

Entrenar

$$\delta_i = \left( \sum_j \delta_j w_{ij} \right) \frac{\partial y_i}{\partial a_i}$$

BP altrno, hay una variación de  $J$  con respecto  
al valor de activación.

Tener en

$$\delta_j = \frac{\partial J}{\partial y_j} \frac{\partial y_j}{\partial a_j}$$

$$\delta_i = \left( \sum_j \delta_j w_{ij} \right) \frac{\partial y_i}{\partial a_i}$$

Combinador

$$\frac{\partial J}{\partial w_{i,j}} = \delta_j \frac{\partial a_j}{\partial w_{i,j}} = \delta_j y_i$$

Las reglas para actualizar los pesos de cada capa

$$w_{i,j} \rightarrow w_{i,j} - \eta \delta_j y_i$$

$\eta$ : learning rate  $\left\{ \begin{array}{l} \text{grande: Rápido pero impreciso} \\ \text{pequeño: Lento y no puede terminar.} \end{array} \right.$

Reynolds a Superstar