

# Aprendizaje Profundo por Refuerzo

## 02. Aprendizaje profundo

Dr. Juan Gómez Romero  
*Investigador Senior*  
Departamento de Ciencias de la Computación e Inteligencia Artificial  
Universidad de Granada



**UBA**  
Universidad de Buenos Aires  
*Argentina virtus robur et studium*



UNIVERSIDAD  
DE GRANADA



# Aprendizaje profundo

## Bibliografía



**F. Berzal** (2019) Redes Neuronales & Deep Learning. <https://sites.google.com/view/redes-neuronales>

**E. Stevens, L. Antiga** (2019) Deep Learning with PyTorch. Manning.

**I. Goodfellow, Y. Bengio, A. Courville** (2016) Deep Learning. MIT Press. <http://www.deeplearningbook.org>

**A. Trask** (2018) Grokking Deep Learning. Manning.

**M. Nielsen** (2017) Neural Networks & Deep Learning. <http://neuralnetworksanddeeplearning.com>

**DeepLearningAI:** <https://www.deeplearning.ai>

**FastAI:** <https://course.fast.ai>

**A. Géron** (2017) Hands-on Machine Learning with scikit-learn and TensorFlow. O'Reilly.

**F. Chollet** (2017) Deep Learning with Python. Manning.

# Índice

1. Redes neuronales
2. Entrenamiento
3. Implementación

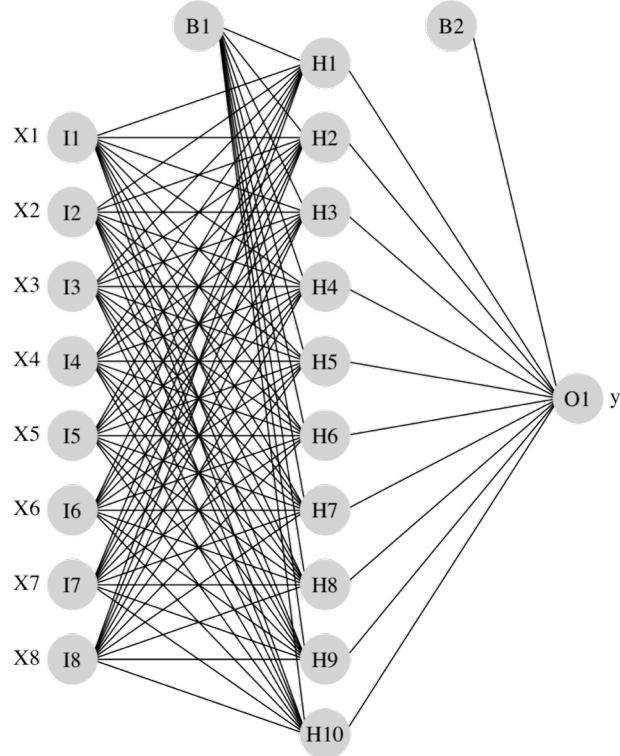
**1**

# Redes neuronales

FUNDAMENTOS

# Redes neuronales

Modelo



<https://youtu.be/MRlv2lwFTPg>

Partes 1, 2



UNIVERSIDAD  
DE GRANADA

**Entrada ( $n$ )**

$$x = (x_1, \dots, x_n)$$

**Salidas capa oculta ( $m$ )**

$$H_j = h\left(\sum_{i=1}^n x_i w_{ij} + B_1\right)$$



producto escalar

$$x \cdot w = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} [w_1 \quad \dots \quad w_n]$$

**Salidas finales ( $k$ )**

$$O_k = h'\left(\sum_{j=1}^m H_j w_{jk} + B_2\right)$$

# Redes neuronales

Funciones de activación y salida

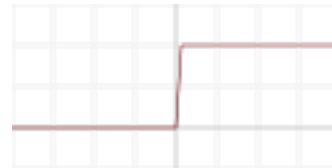


UNIVERSIDAD  
DE GRANADA

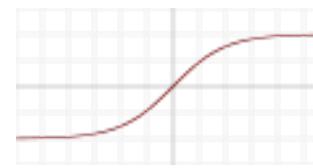
## Función de activación $h$



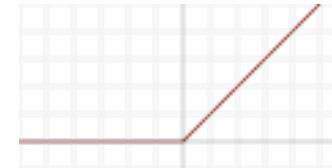
lineal



salto



sinusoidal  
/ sigmoide



rectificador lineal (ReLU)

## Funciones de salida $h'$

$$f_i(\vec{o}) = \frac{e^{o_i}}{\sum_{j=1}^{|o|} e^{o_j}}$$

softmax

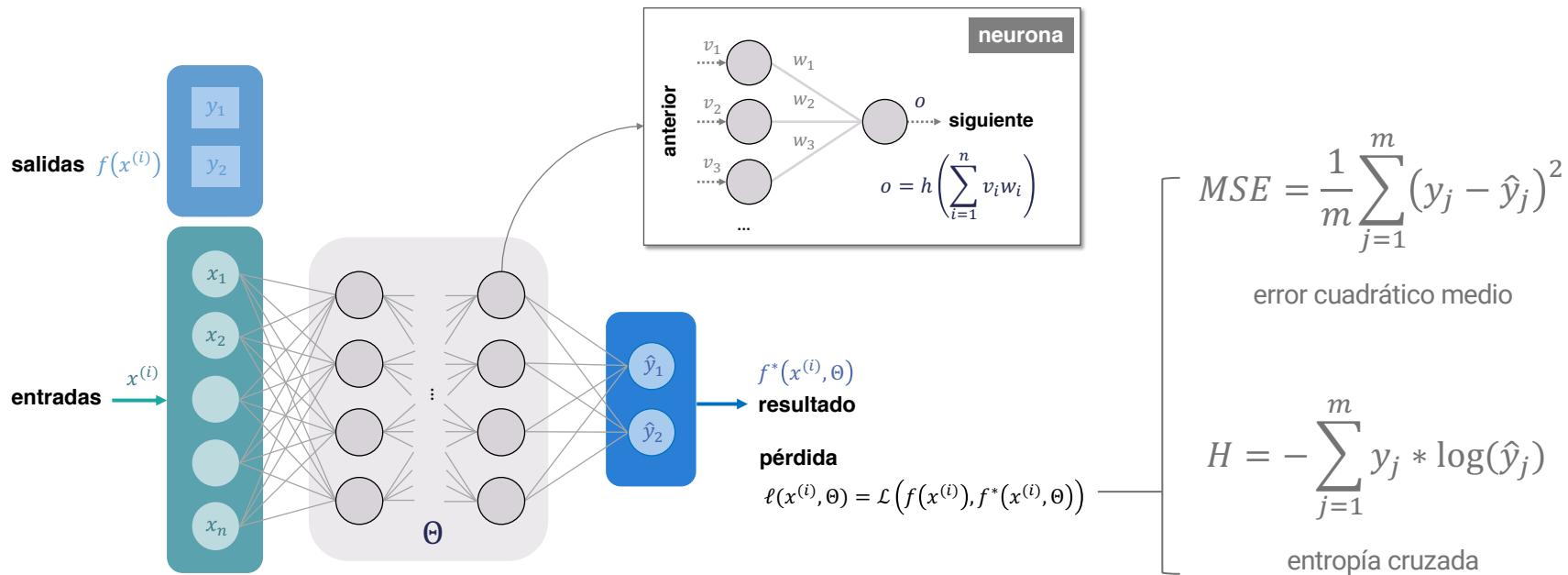
$$f_i(\vec{o}) = \max_i o_i$$

máximo

# Redes neuronales

Error de salida

## Función de pérdida



# Redes neuronales

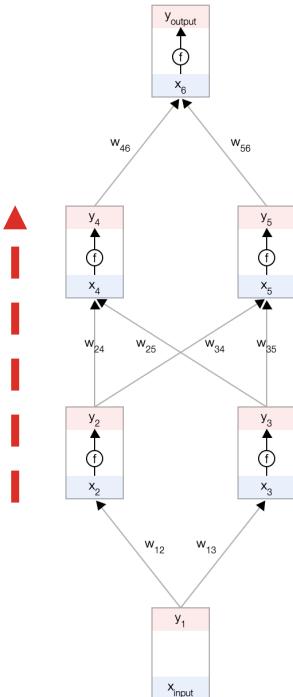
Error de salida



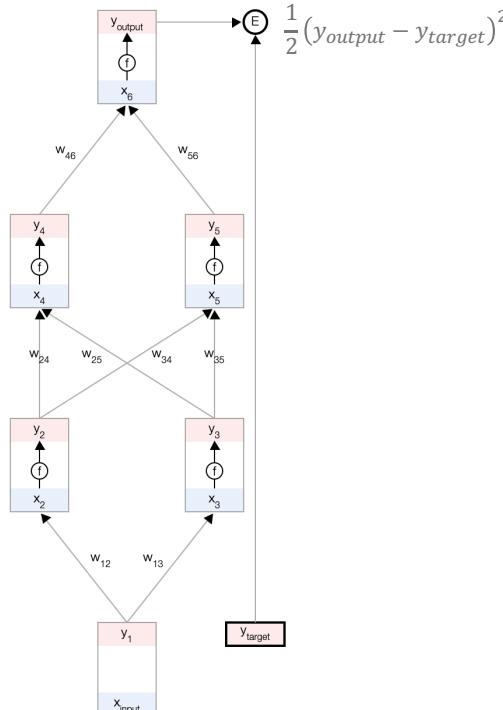
Introducción al aprendizaje automático  
<http://sl.ugr.es/0atf>



UNIVERSIDAD  
DE GRANADA



cálculo de la salida (*forward pass*)



cálculo del error

# Redes neuronales

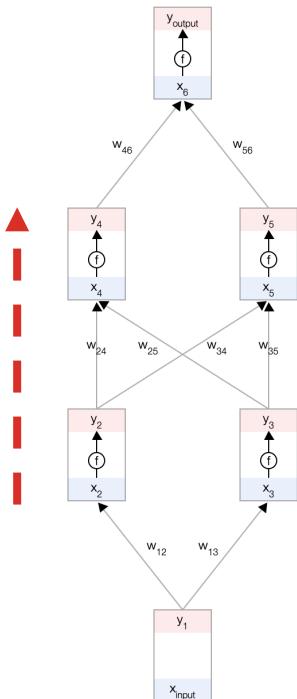
Error de salida



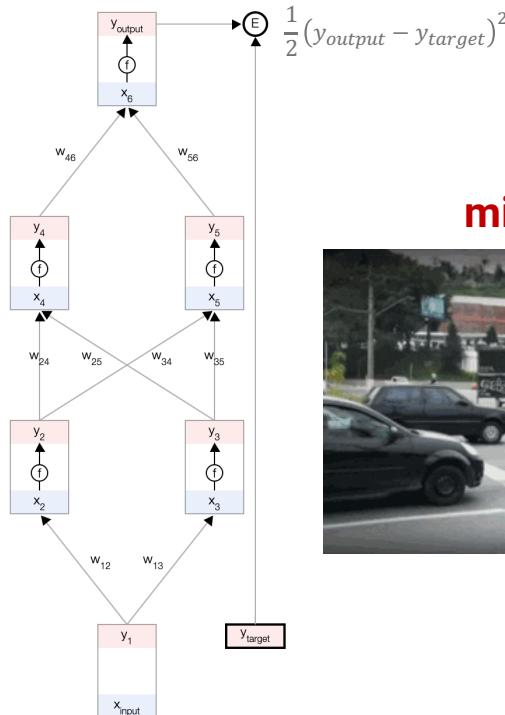
Introducción al aprendizaje automático  
<http://sl.ugr.es/0atf>



UNIVERSIDAD  
DE GRANADA



cálculo de la salida (*forward pass*)



cálculo del error

minimizar error



# 2 Entrenamiento

OPTIMIZACIÓN, *BACKPROPAGATION*, GRADIENTE DESCENDENTE

# Entrenamiento

Hold out

datos históricos

	$x_1$	$\dots$	$x_n$	$y_1$	$\dots$	$y_m$
$x^{(1)}$						
$x^{(2)}$						
$\dots$						
$x^{(K)}$						

random

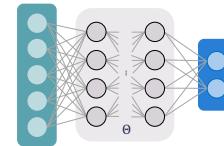
datos entrenamiento [80%]

	$x_1$	$\dots$	$x_n$	$y_1$	$\dots$	$y_m$
$x^{(i)}$						
$\dots$						
$x^{(K')}$						



$$\Theta = \arg \min_{\Theta} \frac{1}{K'} \sum_{i=1}^{K'} \ell(x^{(i)}, \Theta)$$

optimización



datos validación [20%]

	$x_1$	$\dots$	$x_n$	$y_1$	$\dots$	$y_m$
$x^{(i)}$						
$\dots$						
$x^{(K')}$						

evaluación



accuracy  
sensitivity – specificity  
...

# Entrenamiento

*Hold out*

datos históricos

	$x_1$	$\dots$	$x_n$	$y_1$	$\dots$	$y_m$
$x^{(1)}$						
$x^{(2)}$						
$\dots$						
$x^{(K)}$						

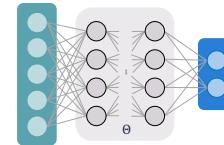
*random*

datos entrenamiento [80%]

	$x_1$	$\dots$	$x_n$	$y_1$	$\dots$	$y_m$
$x^{(i)}$						
$\dots$						
$x^{(K')}$						

$$\theta = \arg \min_{\theta} \frac{1}{K'} \sum_{i=1}^{K'} \ell(x^{(i)}, \theta)$$

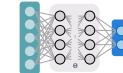
optimización



datos validación [20%]

	$x_1$	$\dots$	$x_n$	$y_1$	$\dots$	$y_m$
$x^{(i)}$						
$x^{(K'')}$						

evaluación



*accuracy*  
*sensitivity – specificity*  
...

datos novedosos

$x$	$x_1$	$\dots$	$x_n$

test

	$\hat{y}_1$	$\dots$	$\hat{y}_m$

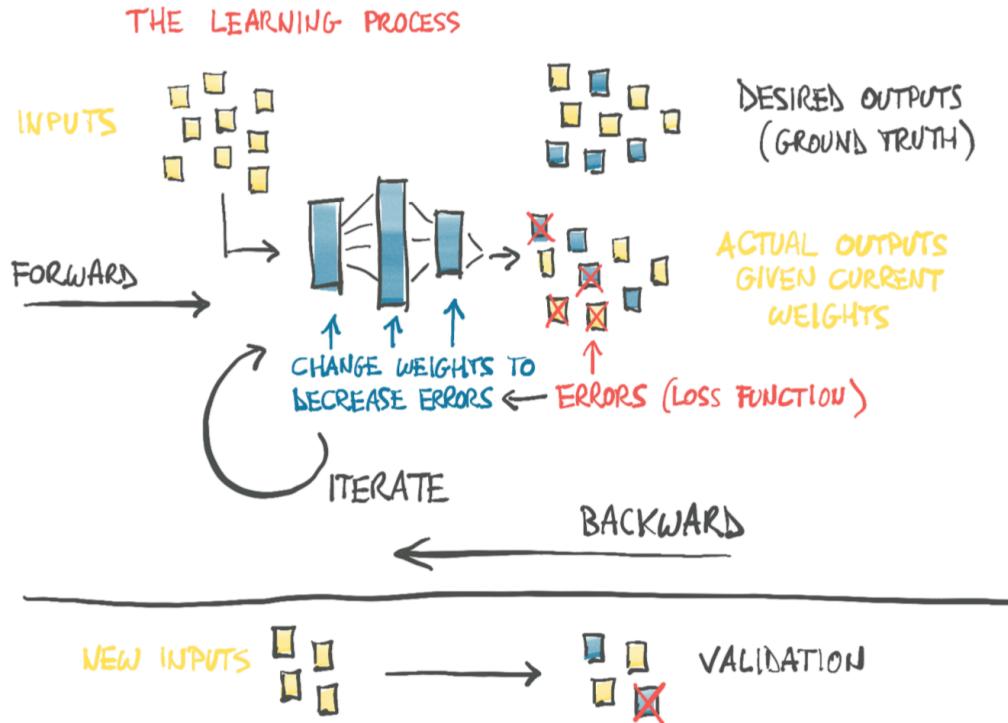
# Entrenamiento

Aprendizaje

optimización



UNIVERSIDAD  
DE GRANADA

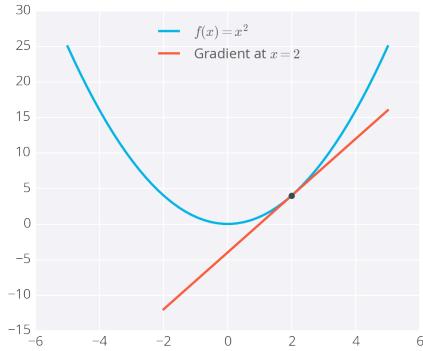


# Entrenamiento

Optimización

## Gradiente descendente

La actualización depende del gradiente de la función de error



$$\nabla E = \frac{\partial f(w)}{\partial w}$$

$$\Delta w_{ij} \propto -\frac{\partial E}{\partial w_{ij}}$$

$$\boxed{\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}}}$$



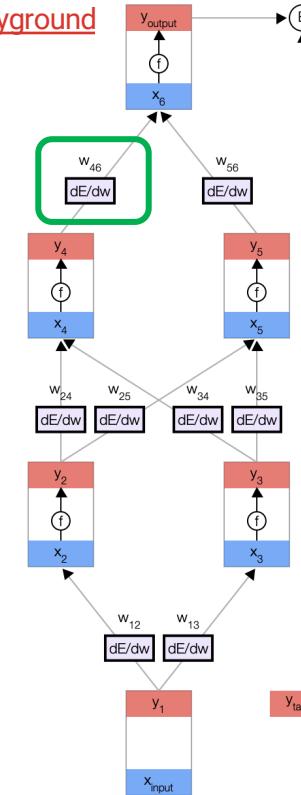
[https://youtu.be/eNlqz\\_noix8](https://youtu.be/eNlqz_noix8)  
Partes 3, 3.5



<http://sl.ugr.es/tfplayground>



UNIVERSIDAD  
DE GRANADA



$$\begin{aligned}\frac{\partial E}{\partial w_{ij}} &= \frac{\partial \left( \frac{1}{Z} (y - \hat{y}(w))^2 \right)}{\partial w_{ij}} \\ &= -(y - \hat{y}) h'(x_j) y_i\end{aligned}$$

## Backpropagation

Actualiza los pesos de la red propagando los errores desde la capa de salida hasta la capa inicial



# Entrenamiento

Aprendizaje

## Algoritmos de optimización

En cada iteración...

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}}$$

Gradiente descendente por lotes

calcula gradientes para todo el conjunto de entrenamiento

Gradiente descendente estocástico

calcula gradientes para una instancia del conjunto de entrenamiento

Gradiente descendente con mini-lotes

calcula gradientes para un subconjunto de instancias del conjunto de entrenamiento

...y actualiza pesos

Tasas de aprendizaje adaptativas:

Adagrad, Adadelta, RMSProp, Adam, Nadam



S. Ruder (2016) *An overview of gradient descent optimization algorithms*. <https://arxiv.org/abs/1609.04747>



# Entrenamiento

Aprendizaje

## Selección de arquitectura de la red e hiperparámetros

### Arquitectura de la red

Número de capas ocultas

Número de neuronas por capa

Funciones de activación

Función de pérdida

...

### Tasa de aprendizaje

> Uso de otros optimizadores

### Tamaño de lotes

> DataLoader

...

### Estrategias y herramientas

<https://blog.floydhub.com/guide-to-hyperparameters-search-for-deep-learning-models/>



M. Jaderberg (2017) *Population Based Training of Neural Networks*.

<https://arxiv.org/abs/1711.09846>

# Entrenamiento

Aprendizaje



<https://youtu.be/7-6X3DTt3R8>

Partes 1, 2



UNIVERSIDAD  
DE GRANADA

## Sobreaprendizaje (*overfitting*)

Un modelo es capaz de predecir las salidas para los valores de entrenamiento, pero no para los de validación

### El modelo no tiene capacidad de generalización

Imagina que tenemos un robot al que queremos enseñar a reconocer fotos de gatos. Para ello, le mostramos 100 fotos de gatos y 100 fotos de perros (*entrenamiento*).

Una vez estudiadas las fotos, el robot dice que ya es capaz de identificar gatos. Si le mostramos una foto de las que ya conoce, seguramente la clasificará correctamente. Sin embargo, si queremos saber si el robot es bueno en su trabajo, deberíamos mostrarle otras fotos que no ha visto (*validación*).

¡El robot podría haber guardado en su memoria las fotos que le hemos pasado y simplemente limitarse a buscar si tiene memorizada una IGUAL!

# Entrenamiento

Aprendizaje



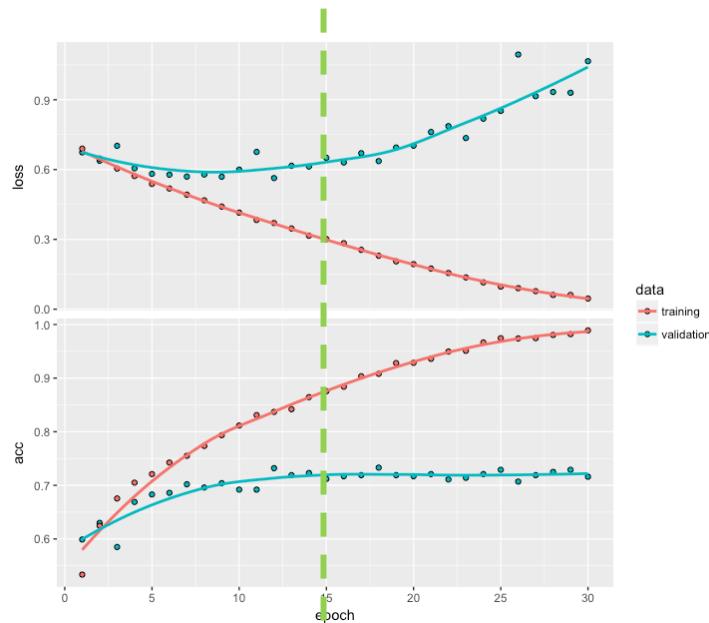
<https://youtu.be/7-6X3DTt3R8>

Partes 1, 2



UNIVERSIDAD  
DE GRANADA

## Sobreaprendizaje (*overfitting*)



sobreaprendizaje



VS



[https://github.com/jgromero/sige2019/blob/master/prácticas/04.%20Modelos%20avanzados/dogs\\_cats.R](https://github.com/jgromero/sige2019/blob/master/prácticas/04.%20Modelos%20avanzados/dogs_cats.R)

# Entrenamiento

Aprendizaje



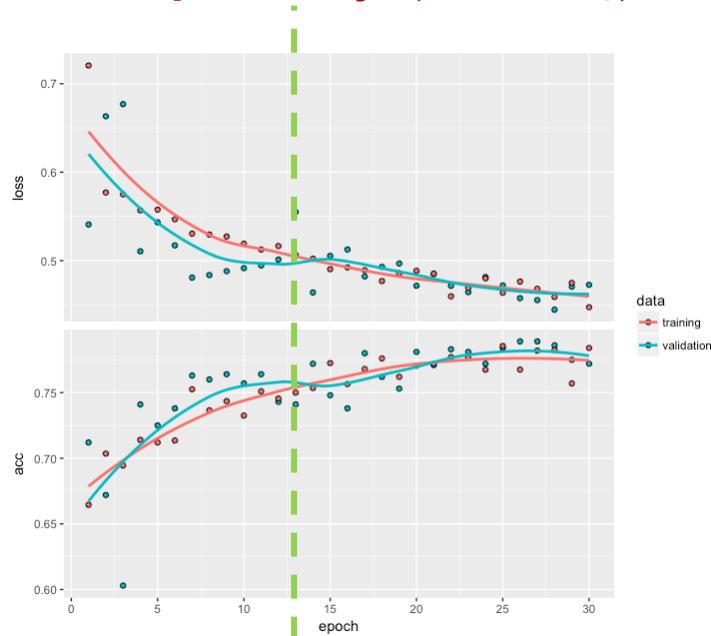
<https://youtu.be/7-6X3DTt3R8>

Partes 1, 2



UNIVERSIDAD  
DE GRANADA

## Sobreaprendizaje (*overfitting*)



sin sobreaprendizaje



VS



[https://github.com/jgromero/sige2019/blob/master/prácticas/04.%20Modelos%20avanzados/dogs\\_cats.R](https://github.com/jgromero/sige2019/blob/master/prácticas/04.%20Modelos%20avanzados/dogs_cats.R)



# Entrenamiento

Aprendizaje

## Sobreaprendizaje (*overfitting*)

### *Early-stopping*

Detener el entrenamiento cuando el error de validación comienza a aumentar

### Selección de datos de entrenamiento y aumento de datos

Incrementar el número de datos disponibles y asegurar que son representativos

### Regularización

Modificar los forma en que se actualizan los pesos de la red durante el entrenamiento para que mantengan valores bajos > Se añade un 'coste adicional' a la función de coste si los pesos son altos

*L1 regularization:* El coste añadido es proporcional al valor absoluto de los pesos

*L2 regularization:* El coste añadido es proporcional al cuadrado de los valores de los pesos

### *Dropout*

Poner a 0 pesos de la red para aumentar resistencia al aprendizaje > Se seleccionan aleatoriamente con probabilidad  $p$ , normalmente entre 0.2 y 0.5

# 3 Implementación

PYTORCH

# Implementación

Google Colaboratory



UNIVERSIDAD  
DE GRANADA



<https://colab.research.google.com>

Entorno gratuito de *Jupyter Notebook* que no requiere configuración y que se ejecuta completamente en la nube.

Permite escribir, ejecutar y guardar código Python utilizando GPUs, de forma gratuita desde el navegador.

El código se escribe en un cuaderno. Un cuaderno puede contener celdas de código (en Python) y celdas de texto (en Markdown).

El cuaderno puede ejecutarse completamente o por celdas.

<https://colab.research.google.com/notebooks/welcome.ipynb>

Google Colaboratory ya tiene preinstalado PyTorch.

```
!pip3 install torch torchvision
```

# Implementación

PyTorch

## PyTorch <https://pytorch.org/>

Biblioteca en Python para el desarrollo de modelos de aprendizaje automático basados en redes neuronales

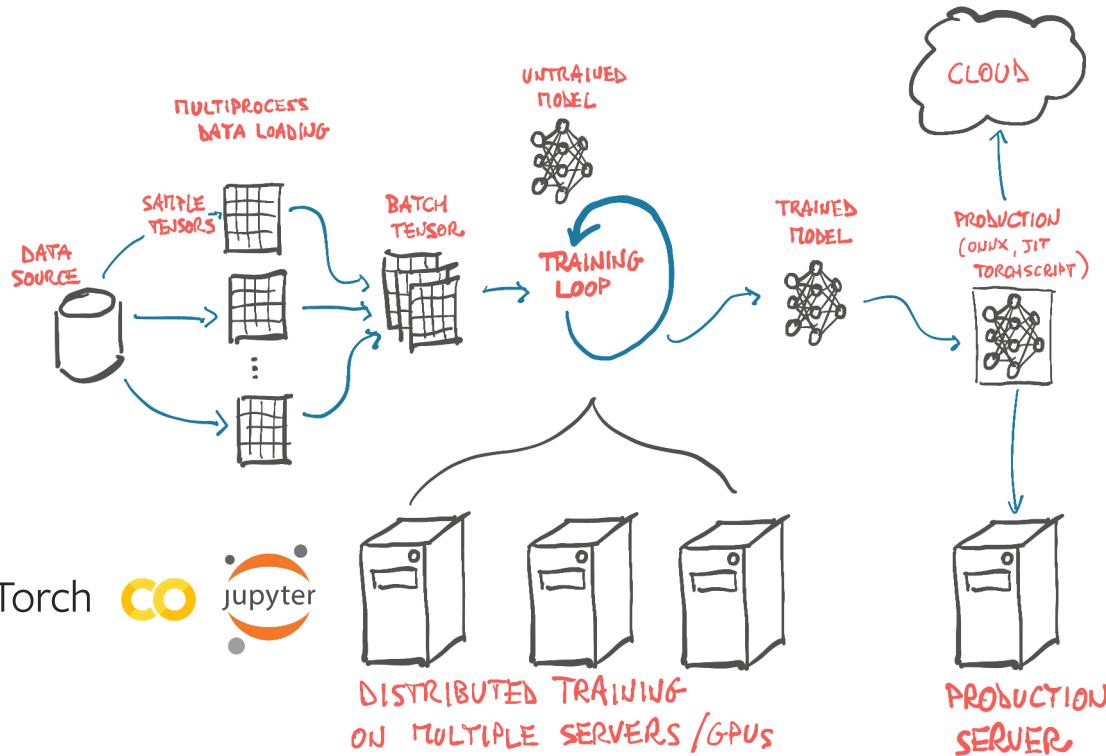
- *Arrays multidimensionales: Tensor*
- *Tensor API*
- Computación mediante grafos de operaciones (modos *eager y static*)
- Diferenciación automática
- Optimización
- Computación distribuida
- Repositorio con modelos predefinidos y pre-entrenados
- Recursos, herramientas, etc.
- Eficiencia: C++, CUDA

# Implementación

PyTorch

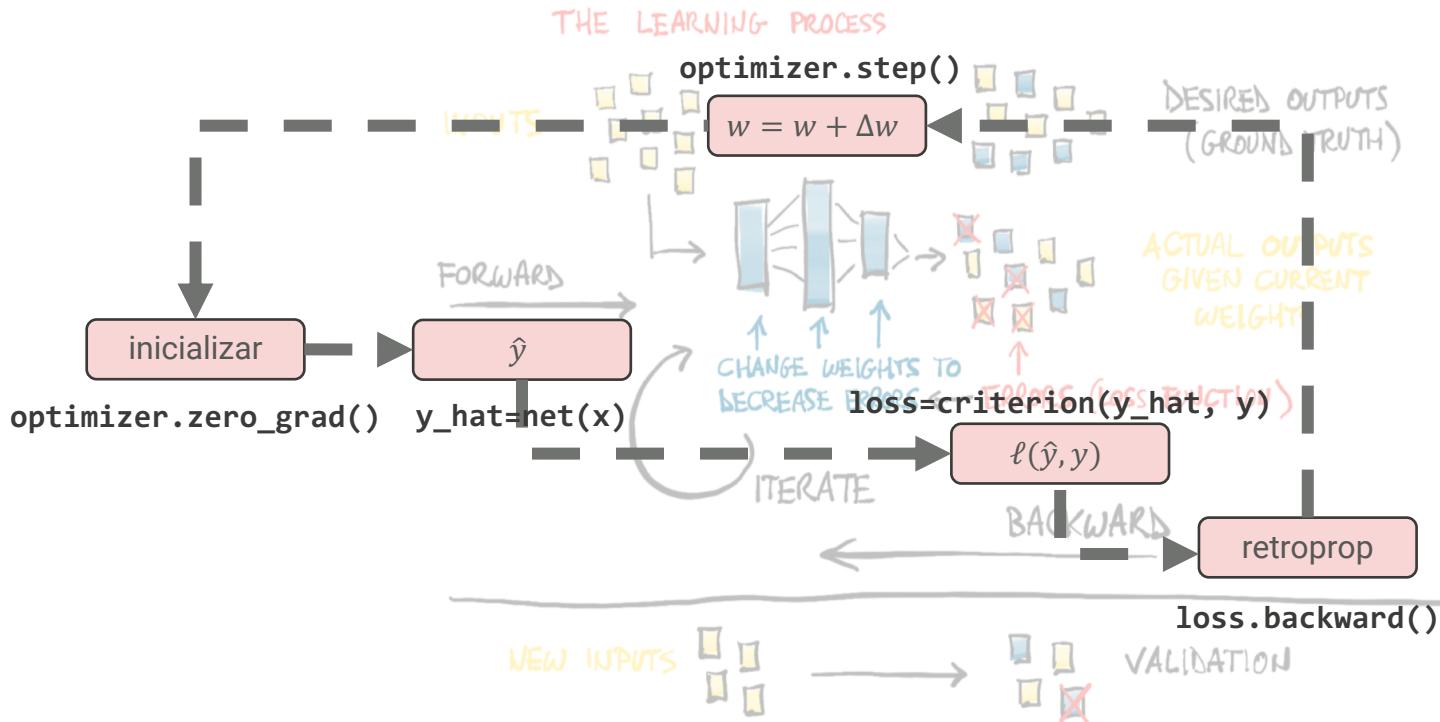


UNIVERSIDAD  
DE GRANADA



# Entrenamiento

Entrenamiento con PyTorch





# Implementación

PyTorch

## Clasificación de flores *iris*

[https://scikit-learn.org/stable/auto\\_examples/datasets/plot\\_iris\\_dataset.html](https://scikit-learn.org/stable/auto_examples/datasets/plot_iris_dataset.html)

Predecir tipo de flor a partir de la anchura y longitud de pétalos y sépalos.

Conjunto de datos clásico en aprendizaje automático

R. Fisher (1936) *The use of multiple measurements in taxonomic problems*

### Datos

Tres especies de flores: iris setosa, iris virginica, iris versicolor

50 muestras de cada especie

4 características: longitud y anchura de pétalos y sépalos (cm)

### Objetivo

Obtener la clase de una flor (desconocida) dados los valores de las 4 características

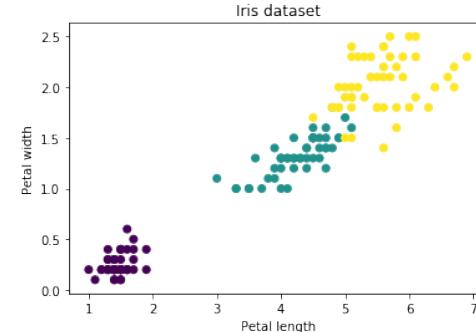
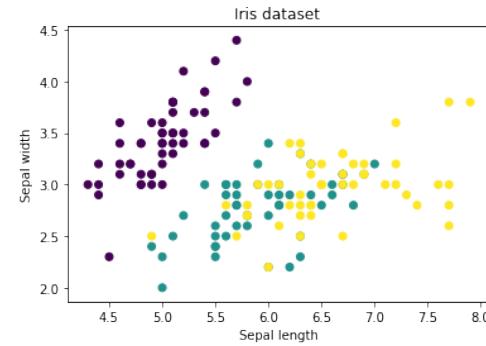
# Implementación

PyTorch

## Clasificación de flores *iris*

[https://scikit-learn.org/stable/auto\\_examples/datasets/plot\\_iris\\_dataset.html](https://scikit-learn.org/stable/auto_examples/datasets/plot_iris_dataset.html)

sepal length	sepal width	petal length	petal width	class
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3.0	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
4.6	3.1	1.5	0.2	Iris-setosa
5.0	3.6	1.4	0.2	Iris-setosa
7.0	3.2	4.7	1.4	Iris-versicolor
6.4	3.2	4.5	1.5	Iris-versicolor
6.9	3.1	4.9	1.5	Iris-versicolor
5.5	2.3	4.0	1.3	Iris-versicolor
6.5	2.8	4.6	1.5	Iris-versicolor
6.7	2.5	5.8	1.8	Iris-virginica
7.2	3.6	6.1	2.5	Iris-virginica
6.5	3.2	5.1	2.0	Iris-virginica
6.4	2.7	5.3	1.9	Iris-virginica
6.8	3.0	5.5	2.1	Iris-virginica
...				



# Implementación

PyTorch

## Clasificación de flores *iris*

[https://scikit-learn.org/stable/auto\\_examples/datasets/plot\\_iris\\_dataset.html](https://scikit-learn.org/stable/auto_examples/datasets/plot_iris_dataset.html)

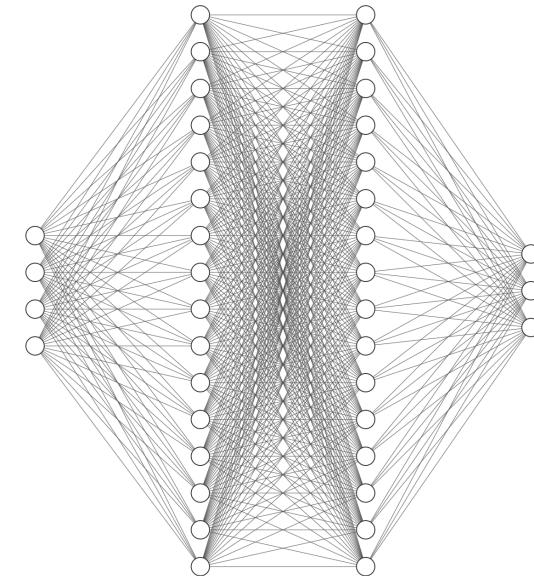
1. Cargar datos *iris*
2. Definir la arquitectura de la red
3. Separar conjuntos de entrenamiento y validación
4. Entrenar red
5. Obtener resultados con conjunto de validación

### Hiperparámetros

*batch size*: 50

número de *epochs*: 1000

**Solución:** 03. iris-nn.ipynb



# Índice

1. Redes neuronales
2. Entrenamiento
3. Implementación



# Aprendizaje profundo

Temas avanzados

## Clasificación no *balanceada*

Predicción cuando en un problema número de instancias de una clase es mucho mayor que el de otra.

## Redes neuronales convolucionales (*convolutional neural networks – CNN*)

Extracción automática de características de imágenes

## Redes neuronales recurrentes

Implementan memoria, de forma que una salida depende de la salida anterior

## Otros *frameworks*

Tensorflow + Keras

## Meta-aprendizaje

Aprendizaje de la arquitectura y de los hiperparámetros de la red