



# Deep Learning con TensorFlow

por Antón Pastoriza

23 OCT 2018

# Machine Learning

- Identificar e implementar sistemas y algoritmos para que las máquinas puedan aprender
- El proceso aprendizaje es por comparación de datos

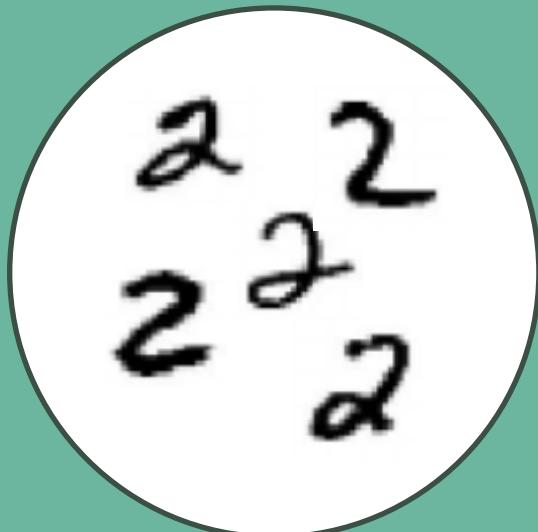


Los algoritmos nos permiten crear patrones que nos ayuden a comparar

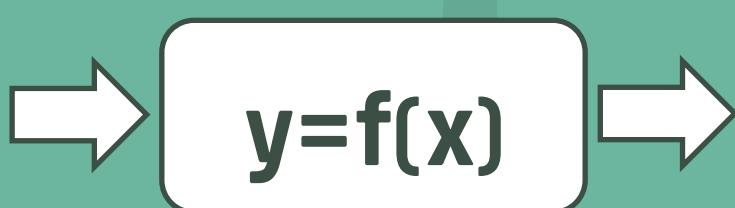
# Machine Learning

## El proceso de aprendizaje (Entrenar...)

Datos de  
entrenamiento



Algoritmos



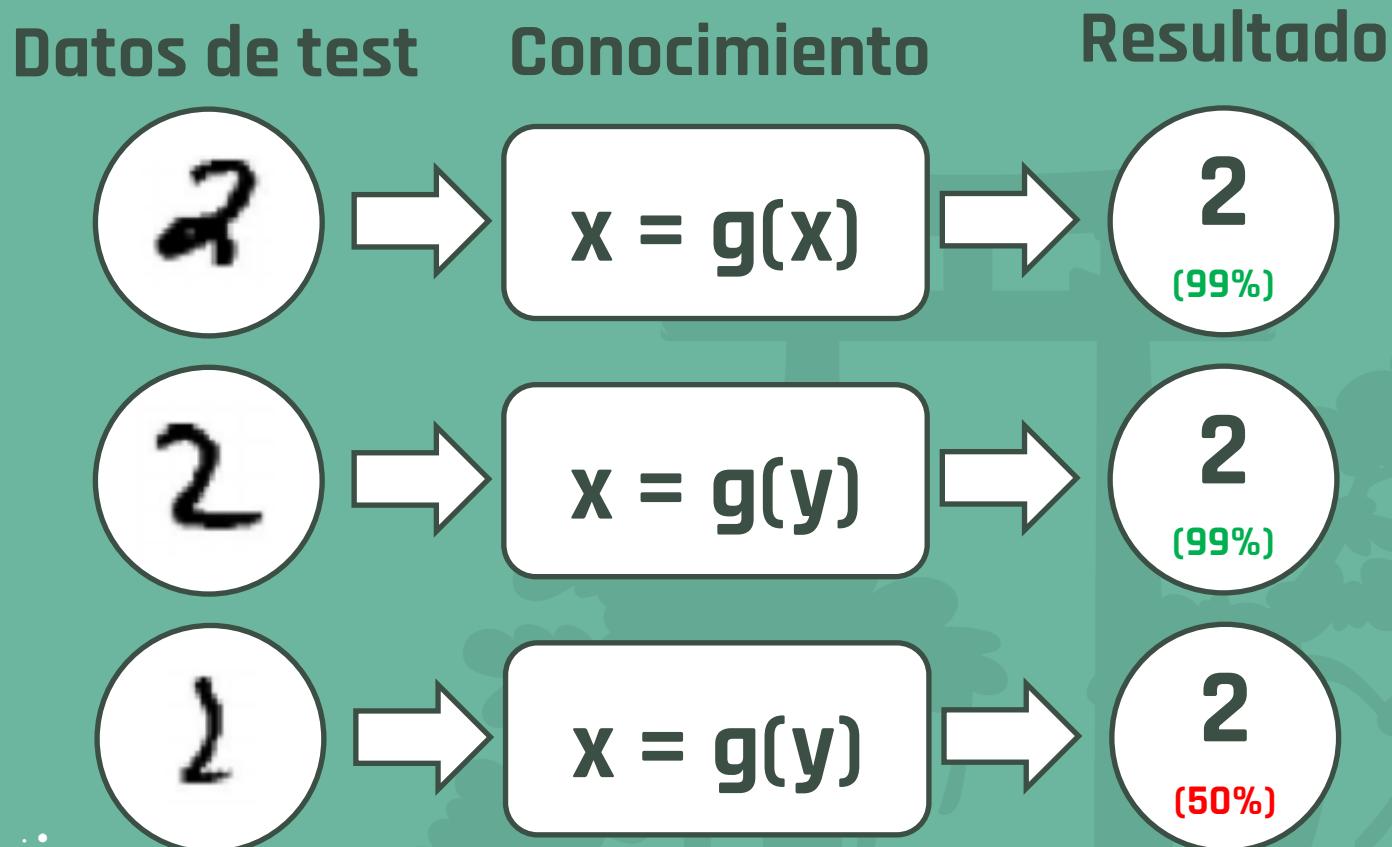
Patrón resultado



Crear un set de datos de entrenamiento y aplicar algoritmos de aprendizaje. Cuanto más completo sea el set, mejores serán los resultados obtenidos

# Machine Learning

## El proceso de aprendizaje (...y Validar)



Crear un set de datos de test y validar el aprendizaje.  
El set de pruebas debe ser 1/3 del set de entrenamiento.

# Machine Learning

## Paradigmas de aprendizaje

- **Supervisado.** Clasificar datos, hacer regresión sobre datos, ordenar datos, rankings,...
- **No supervisado.** Asociar datos, segmentar datos, aplicar reducciones,...
- **Por refuerzo.** Retroalimentar, ajustar parámetros en tiempo de ejecución, algoritmos de fuerza bruta y Montecarlo

# Deep Learning

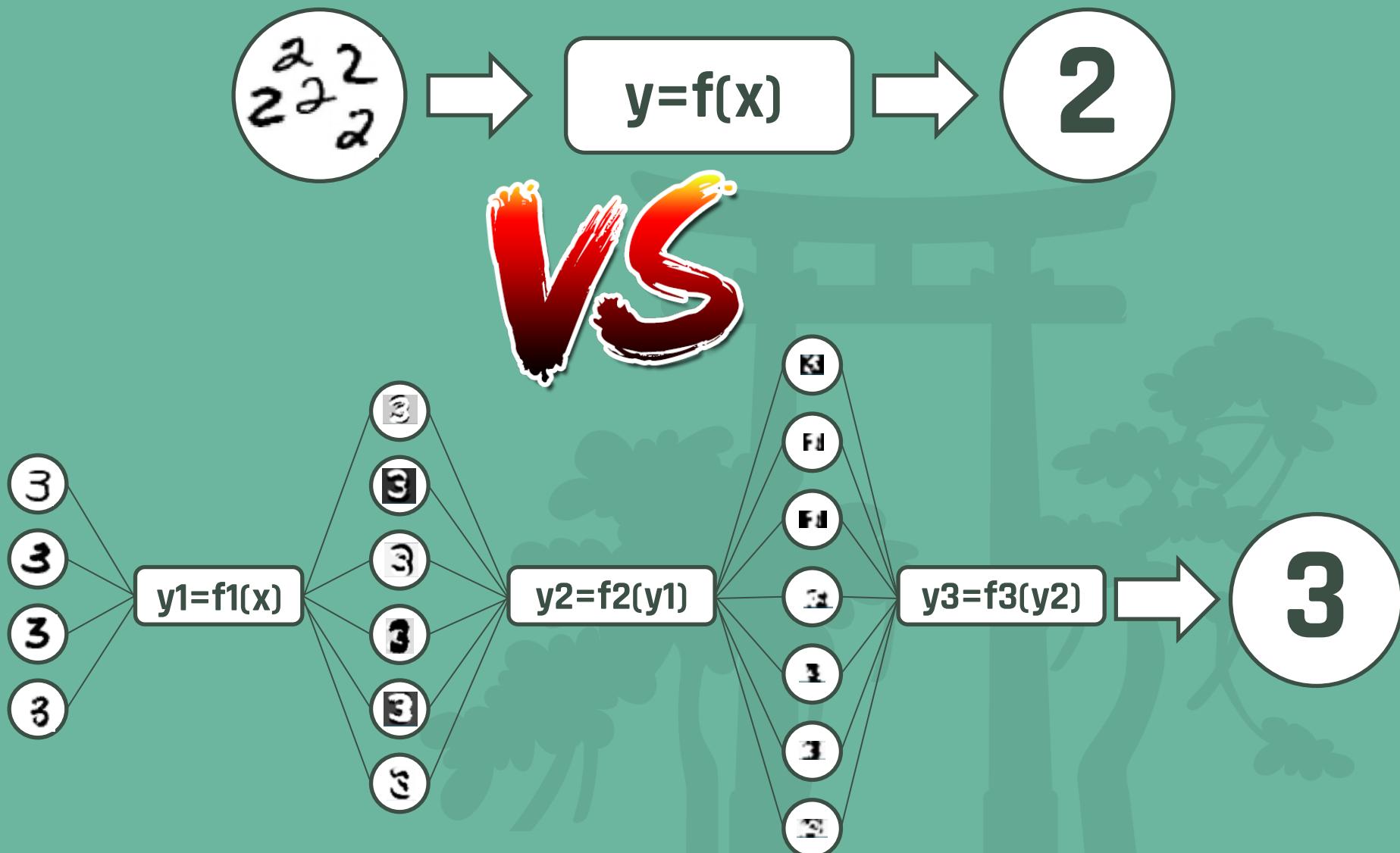
## Introducción

- Es una disciplina dentro del Machine Learning
- Aprendizaje por capas, creando redes
- Divide y vencerás: Varios algoritmos antes de llegar al resultado final
- Permite validar resultados parciales



Cuantas capas y algoritmos son necesarias para que el aprendizaje se considere Deep Learning? Dos? Tres? No hay un consenso

# Deep Learning vs Machine Learning



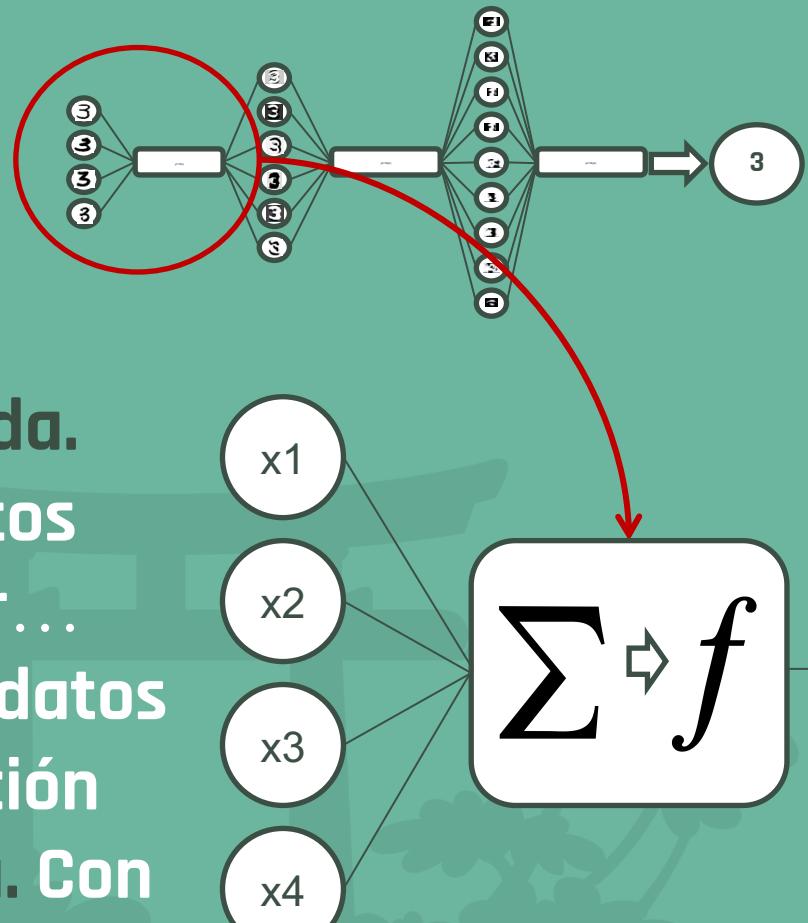
# Deep Learning

## Estructura de capas

- Una ó más variables de entrada. Con el objetivo de recoger datos externos ó de la capa anterior...
- Un algoritmo. Que agrega los datos de entrada y ejecuta una función
- Una ó mas variables de salida. Con el objetivo de obtener resultados ó ser entrada de la siguiente capa...

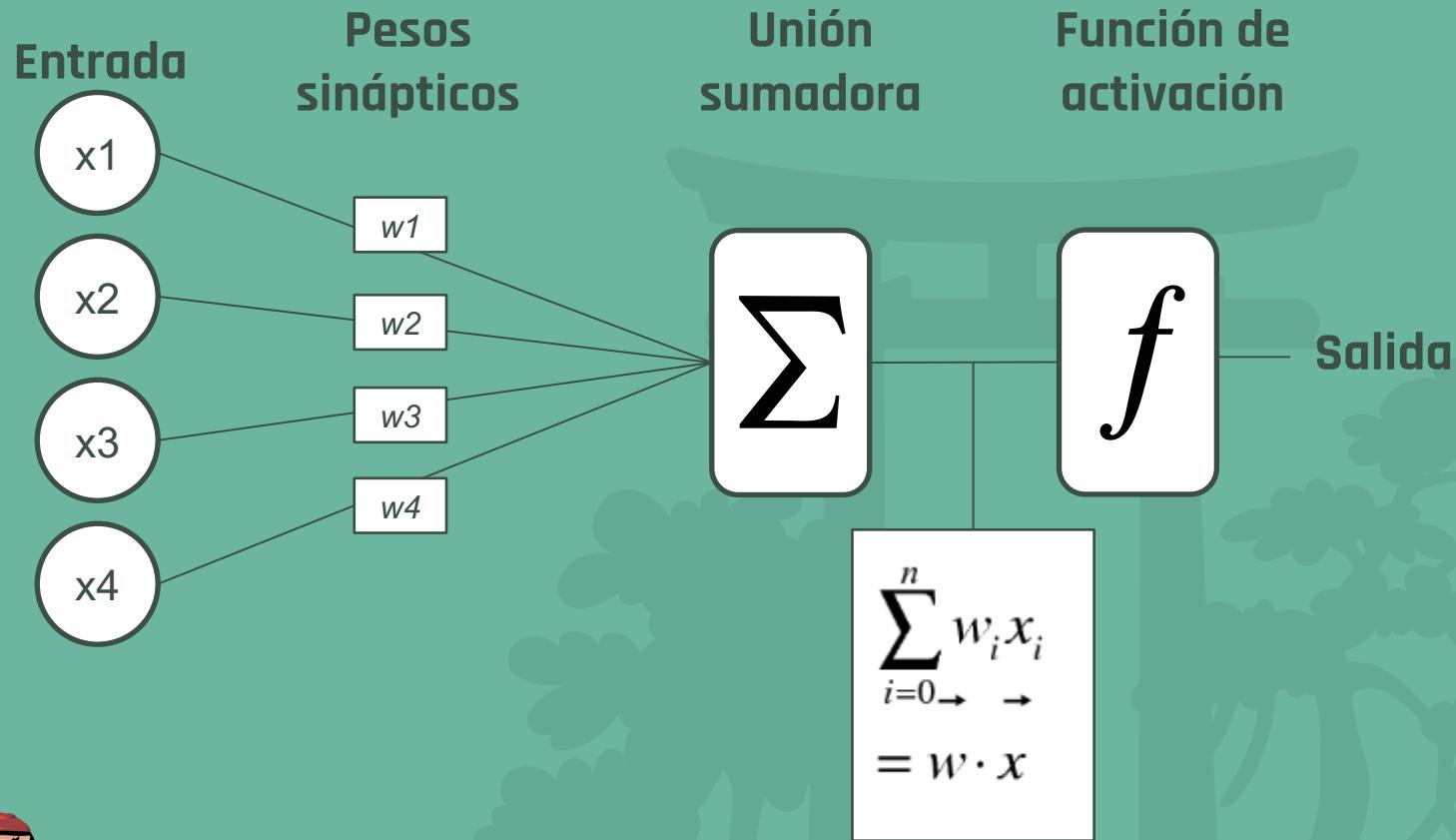


La estructura de capas del Deep Learning es, de facto, una red neuronal.



# Deep Learning

## Estructura de una neurona básica ó perceptrón



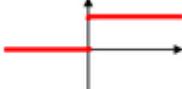
Un perceptrón es una neurona que consta de un sumatorio y una función de activación.



# Deep Learning

## Las funciones de activación más comunes

### Unit step (Heaviside)

$$\phi(z) = \begin{cases} 0, & z < 0, \\ 1, & z \geq 0, \end{cases}$$


Aplicada en modelos binarios. Suele combinarse con la función Linear dando lugar a la función ReLU.

### Linear

$$\phi(z) = z$$


Activa cualquier valor sin modificarlo. La salida es equivalente a la función sumatoria.

### Sigmoidal

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


Permite normalizar las salidas. Aplicada en modelos de clasificación.



TensorFlow cuenta con 11 funciones de activación. Una de ellas es la función dropout, que consiste en asignar siempre el valor 0.

# Deep Learning

## El aprendizaje: ejemplo básico de entrenamiento

- Aprende a realizar la función binaria con tres entradas.
- Función activación: Heaviside

$$\phi(z) = \begin{cases} 0, & z < 0.5, \\ 1, & z > 0.5, \end{cases}$$

- Tasa de aprendizaje ( $r$ ): 0.1
- Conjunto de entrenamiento, que consiste en cuatro muestras:

Entrada			Salida deseada
Valores de sensor			
$x_0$	$x_1$	$x_2$	$y$
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

# Deep Learning

## El aprendizaje: ejemplo básico de entrenamiento (y 2)

Entrada				Pesos iniciales			Salida					Error	Corrección	Pesos finales		
Valores de sensor		Salida deseada					Sensor			Suma	Activación					
$x_0$	$x_1$	$x_2$	$y$	$w_0$	$w_1$	$w_2$	$c_0$	$c_1$	$c_2$	$z$	$n$	$e$	$d$	$w_0$	$w_1$	$w_2$
							$x_0 * w_0$	$x_1 * w_1$	$x_2 * w_2$	$c_0 + c_1 + c_2$	Heaviside	$y - n$	$r * e$	$\Delta(x_0 * d)$	$\Delta(x_1 * d)$	$\Delta(x_2 * d)$
1	0	0	1	0	0	0	0	0	0	0	0	1	+0.1	0.1	0	0
1	0	1	1	0.1	0	0	0.1	0	0	0.1	0	1	+0.1	0.2	0	0.1
1	1	0	1	0.2	0	0.1	0.2	0	0	0.2	0	1	+0.1	0.3	0.1	0.1
1	1	1	0	0.3	0.1	0.1	0.3	0.1	0.1	0.5	0	0	0	0.3	0.1	0.1
1	0	0	1	0.3	0.1	0.1	0.3	0	0	0.3	0	1	+0.1	0.4	0.1	0.1
1	0	1	1	0.4	0.1	0.1	0.4	0	0.1	0.5	0	1	+0.1	0.5	0.1	0.2
1	1	0	1	0.5	0.1	0.2	0.5	0.1	0	0.6	1	0	0	0.5	0.1	0.2
1	1	1	0	0.5	0.1	0.2	0.5	0.1	0.2	0.8	1	-1	-0.1	0.4	0	0.1
1	0	0	1	0.4	0	0.1	0.4	0	0	0.4	0	1	+0.1	0.5	0	0.1
1	0	1	1	0.5	0	0.1	0.5	0	0.1	0.6	1	0	0	0.5	0	0.1
1	1	0	1	0.5	0	0.1	0.5	0	0	0.5	0	1	+0.1	0.6	0.1	0.1
1	1	1	0	0.6	0.1	0.1	0.6	0.1	0.1	0.8	1	-1	-0.1	0.5	0	0

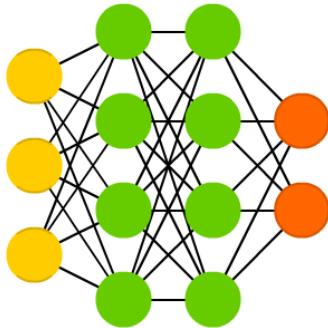


1	0	0	1	0.7	-0.2	-0.2	0.7	0	0	0.7	1	0	0	0.7	-0.2	-0.2
1	0	1	1	0.7	-0.2	-0.2	0.7	0	-0.2	0.5	0	1	+0.1	0.8	-0.2	-0.1
1	1	0	1	0.8	-0.2	-0.1	0.8	-0.2	0	0.6	1	0	0	0.8	-0.2	-0.1
1	1	1	0	0.8	-0.2	-0.1	0.8	-0.2	-0.1	0.5	0	0	0	0.8	-0.2	-0.1
1	0	0	1	0.8	-0.2	-0.1	0.8	0	0	0.8	1	0	0	0.8	-0.2	-0.1
1	0	1	1	0.8	-0.2	-0.1	0.8	0	-0.1	0.7	1	0	0	0.8	-0.2	-0.1

# Deep Learning

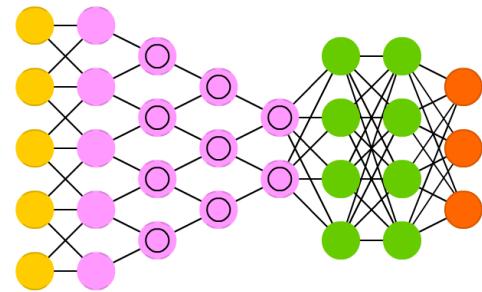
## Arquitectura de redes más comunes

### Multilayer perceptron



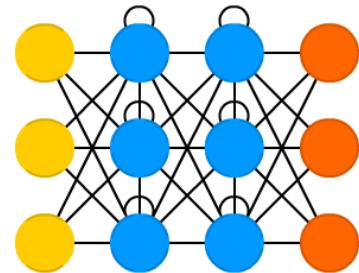
- Arquitectura generalista
- Todos las neuronas conectados a la siguiente capa.
- Dos ó más capas ocultas.

### Convolutional Neural Networks



- Aplican kernels y operaciones de convolución previas a las funciones de red
- Orientadas al reconocimiento y análisis de imágenes.

### Recurrent neural networks



- Las neuronas reciben feedback de su propia salida.
- Comunes en aprendizaje no supervisado: Sistemas de control, reconocimiento de voz,...



Combinando perceptrones, se construyen el resto de redes neuronales. En <http://www.asimovinstitute.org/neural-network-zoo/> mantienen un inventario de todas las tipologías de red.

### Leyenda

Capa de entrada	Capa función convolución
Capa Oculta	Capa función kernel
Capa de salida	Capa recursiva

# Deep Learning

## Frameworks de referencia código abierto

### Tensorflow



**TensorFlow**

Sponsorizado por Google, es una de las suites más completas

Tiene soporte para GPUs y para dispositivos móviles.

### Torch



Es el framework con la curva de acceso más sencilla.

Muy bien documentado.

Soporta GPUs

### Caffe

# Caffe

Framework pionero en dar soporte a Deep Learning.

Facebook fue uno de sus principales sponsor.

### Cognitive Toolkit



Framework de Microsoft bajo Licencia Open Source.

Actualmente es el segundo framework más popular de la industria



Esta lista es el criterio del presentador. PyTorch, Keras, Theano, DeepLearning4J, Intel Data Analytics Acceleration Library...

# Tensorflow

## Introducción

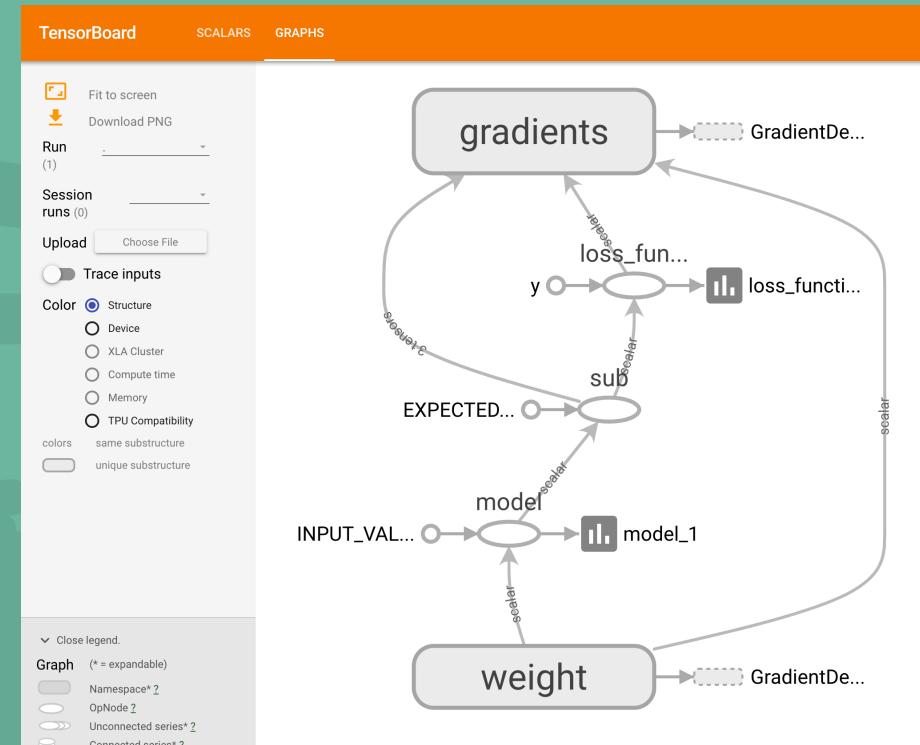
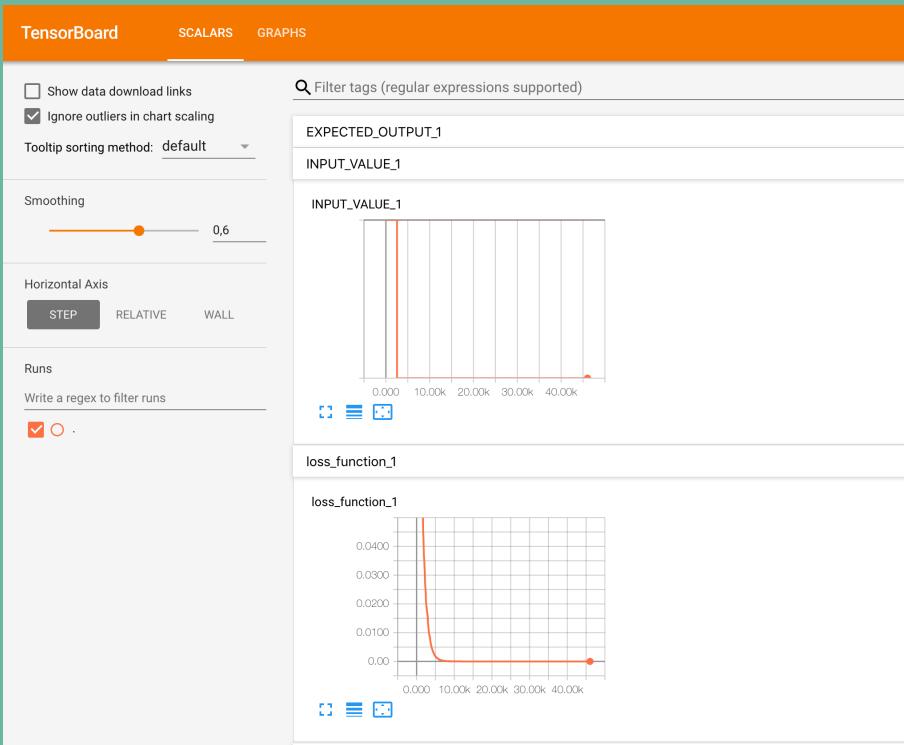
- Es un **framework de cálculo distribuido** orientado al Machine Learning
- Todos los **elementos se representan como grafos (data flow graph)**, lo que facilita trazar los flujos de datos
- Principales tipos de grafos: Datos de entrada/salida (**tensores**) y operaciones (p.e. **funciones de activación**)



Los cálculos en Tensorflow son stateful

# Tensorflow

## Representación del ejemplo básico de entrenamiento



### Evolución de los valores



Tensorflow cuenta con una herramienta nativa de monitorización: Tensorboard. Tensorboard es extensible mediante APIs

### Representación del grafo

# Tensorflow

## Tensores

- Las operaciones en TensorFlow, soportan arrays multidimensionales de datos.
- Estos arrays multidimensionales son referidos como **tensores**, pues así se conocen en la definición matemática clásica.

Orden	Representación matemática	Representación TensorFlow
Tensores de orden cero: escalares	100	<code>tf.constant(100)</code>
Tensores de orden uno: vectores y covectores	{1, 2, 3, 4, 5}	<code>tf.constant([1, 2, 3, 4, 5])</code>
Tensores de orden dos: matrices y formas cuadráticas	[{1, 2, 3}, {4, 5, 6}]	<code>tf.constant([[1, 2, 3], [4, 5, 6]])</code>
Tensores de orden $m$ generalizados	[{(1), (2), (3)}, {(4), (5), (6)}, {(7), (8), (9)}]	<code>tf.constant([   [     [1], [2], [3]   ],   [     [4], [5], [6]   ],   [     [7], [8], [9]   ] ])</code>



Tipos de tensores: **constant** (Valor constante), **placeholder** (sin valor definido) ó **Variable** (valor cambiante)

# Tensorflow

## Modelo de ejecución y programación declarativa

1. Definición del modelo (data flow graph)
2. Creación de sesión y ejecución del modelo
3. Obtención de resultados

En este punto, se define el modelo y las operaciones. La suma y multiplicación no se ejecutarán

Aquí se crea la sesión. Con la llamada a `session.run` se ejecutan las operaciones y obtiene el resultado

```
# coding=utf-8

import tensorflow as tf

from tfninja.utils import loggerfactory

logger = loggerfactory.get_logger(__name__)

CONSTANT_A = tf.constant([100.0])
CONSTANT_B = tf.constant([200.0])
CONSTANT_C = tf.constant([10.0])
add_operation = tf.add(CONSTANT_A, CONSTANT_B)
multiply_operation = tf.multiply(CONSTANT_A, CONSTANT_C)

def run_session():

    with tf.Session() as session:
        result = session.run([add_operation, multiply_operation])
        logger.info(result)

if __name__ == '__main__':
    run_session()
```

# Tensorflow API stack

Pre-made Estimators

APIs de *Model-in-a-box*. Abstraen de conceptos básicos. Define entradas, salidas y Tensorflow intenta inferir el modelo y los pesos que necesita la ejecución.

Estimator

Keras Model

Layers

Datasets

APIs base. Se definen el modelo en base a estándares de la industria.

Python Frontend

C++

Java

Go

...

Engine APIs. Orientadas a la gestión infraestructura en la que va a correr el modelo modelo, p. e. repartir cargas entre puestos de escritorio y GPUs.

TensorFlow Distributed Execution Engine

CPU

GPU

TPU

Android

XLA

iOS

...

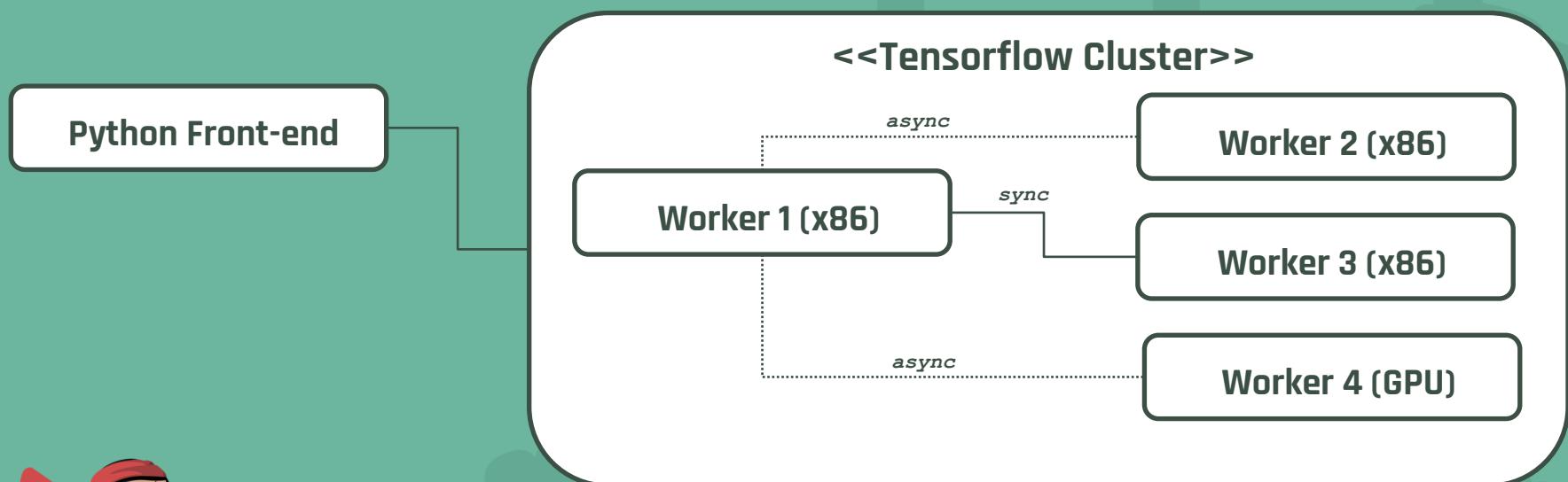


Keras es más lento que TensorFlow nativo pero es portable a otros engines “Keras-friendly” como Microsoft Cognitive Toolkit.

# Tensorflow

## Modelo de ejecución: Distribuido

- Toda ejecución en TensorFlow sucede en un entorno distribuido.
- Cuando interviene más de un engine, debe definirse un Cluster en el que cada engine asume el rol de worker.
- Uno de los workers, asumirá el rol primario y distribuirá las cargas de trabajo dentro del cluster.



Aunque hay iniciativas como kubeflow y nvidia-docker, crear un entorno distribuido de TensorFlow es muy complejo.

# Tensorflow

## Ejemplos y Demos

Todas las demos que vamos a ver están disponibles en  
<https://github.com/apastoriza/tf-ninja>



Arigato!

THE  
NINJA PROJECT