Introducción a TensorFlow

...

Alejandro Solano - Málaga Python

MÁLAGA
PYTHON

TensorFlow


TensorFlow

gato

input

target

??

gato

TensorFlow

input

target

sin

\+

X

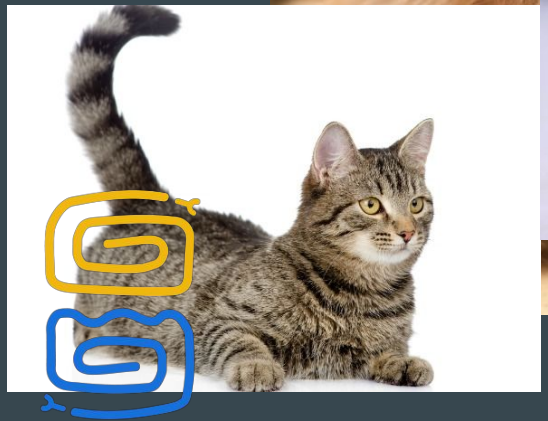
log

exp

gato

TensorFlow

input



target

sin

\+

x

log

exp

gato

TensorFlow

Deep Learning

¿Qué es TensorFlow?

- TensorFlow es una librería open-source para Deep Learning.
- Desarrollado por el equipo de Google Brain y liberado en noviembre de 2015.
- La versión 1.0.0 se lanzó en febrero de 2017.

Instalación

Instalar Tensorflow (Linux y OS X)

- Descargar Anaconda
- Crear entorno con las librerías imprescindibles:

```
$ conda create -n tensorflow python=3.5
$ source activate tensorflow
$ conda install pandas matplotlib jupyter notebook scipy scikit
$ pip install tensorflow
```

Instalar Tensorflow (Windows)

- Descargar Anaconda
- Crear entorno con las librerías imprescindibles:

```
$ conda create -n tensorflow python=3.5
$ activate tensorflow
$ conda install pandas matplotlib jupyter notebook scipy scikit
$ pip install tensorflow
```

Conceptos

gato

TensorFlow

MODEL

gato

TensorFlow

MODEL

no gato

TensorFlow

MODEL

predicción

no gato

target

gato

input

TensorFlow

MODEL

no gato

COST

gato

TensorFlow

MODEL

no gato

C
O
S
T

error

gato

TensorFlow

MODEL

no gato

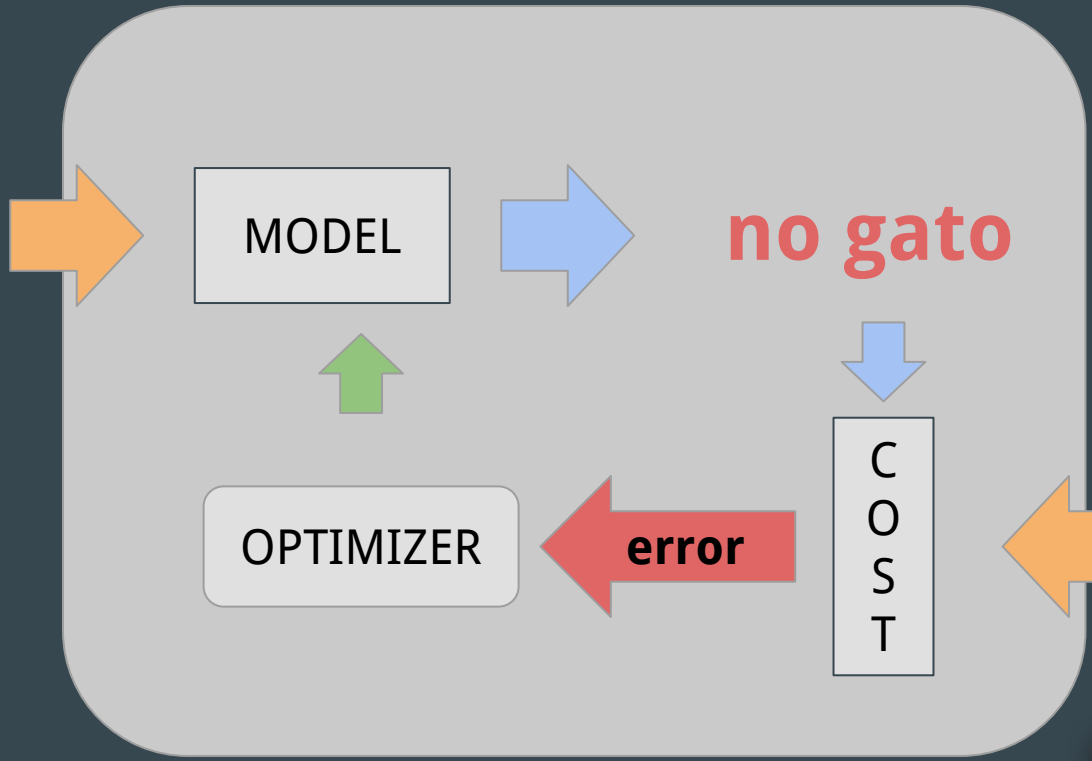
COST

gato

error

OPTIMIZER

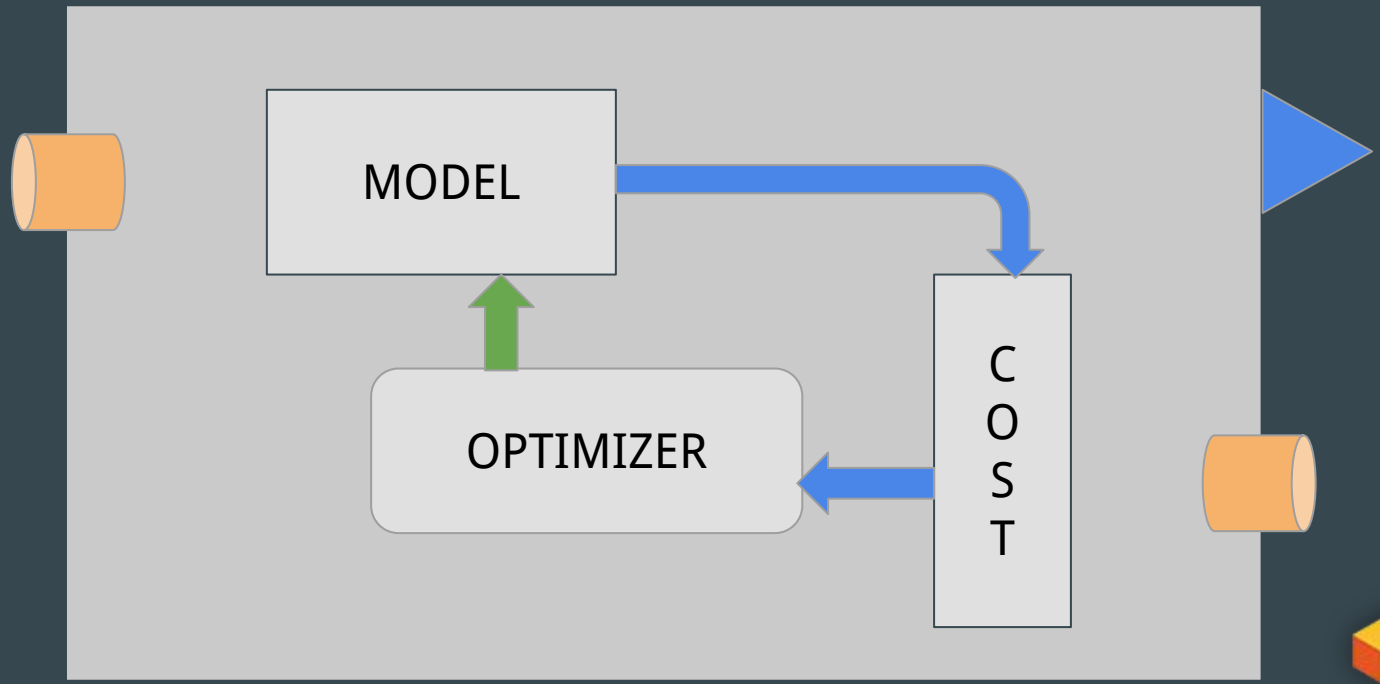
TensorFlow



Graph

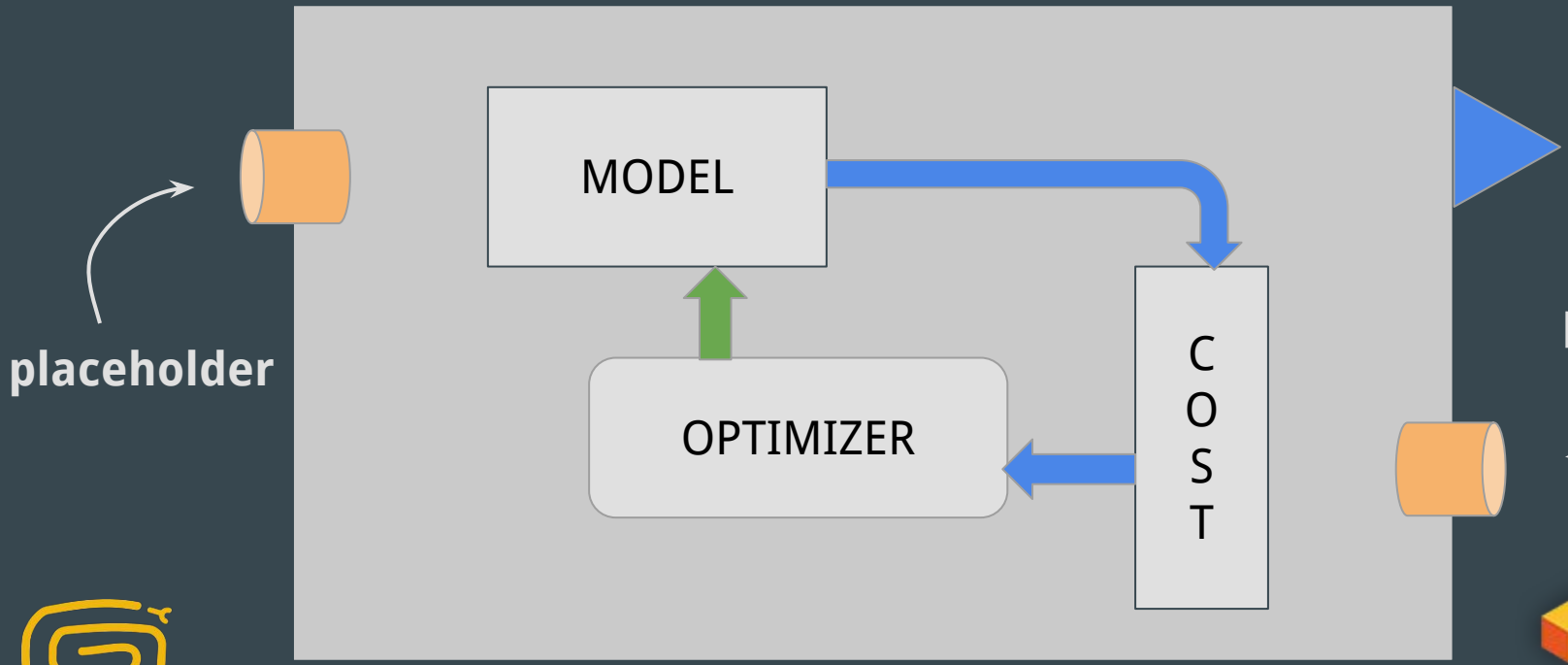
TensorFlow

Grafo



TensorFlow

Grafo

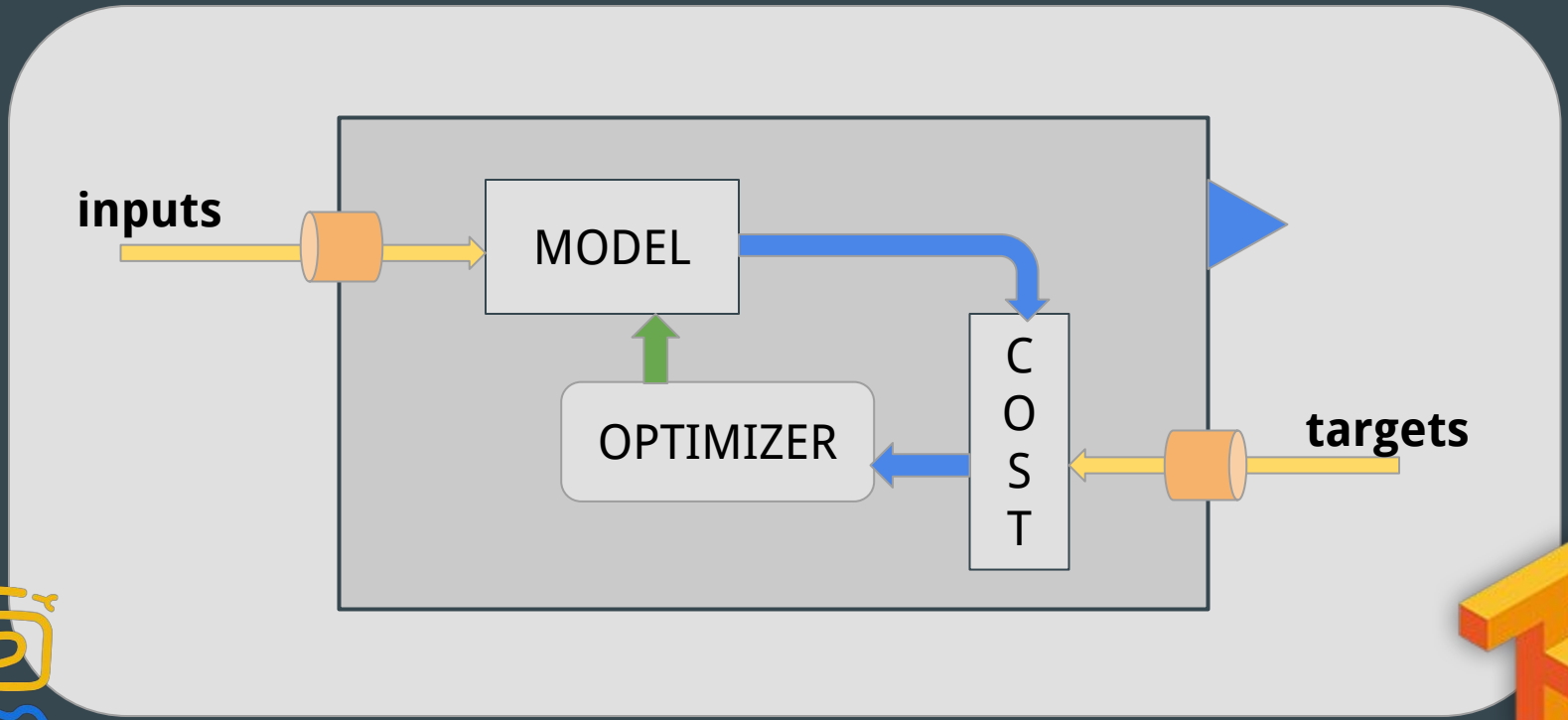


TensorFlow

Grafo

- **Placeholders:** puertas por donde se introducen los ejemplos
- **Modelo:** realiza las predicciones. Conjunto de **variables** y operaciones.
- **Función de coste:** función que calcula el error del modelo
- **Optimizador:** algoritmo que optimiza las variables para que el coste sea cero.

Sesión: Grafo + Datos

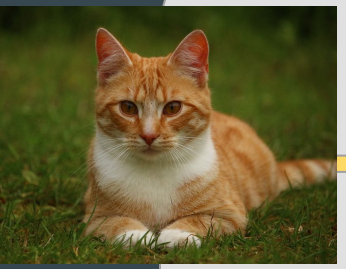


TensorFlow

Grafo, Datos y Sesión

- **Grafo:** lo de antes (?). Independiente de los datos.
- **Datos:** ejemplos con los que se entrena la red neuronal. Se dividen en dos: entradas y objetivos.
- **Sesión:** donde ocurre todo. Aquí **alimentamos** el grafo con los datos.

Sesión: Grafo + Datos



MODEL

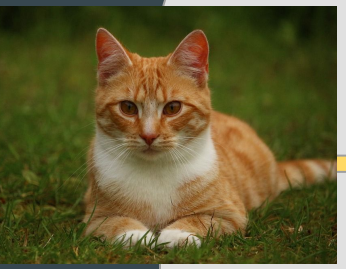
OPTIMIZER

COST

gato

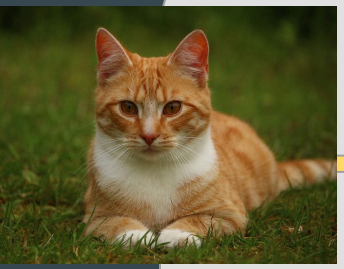
TensorFlow

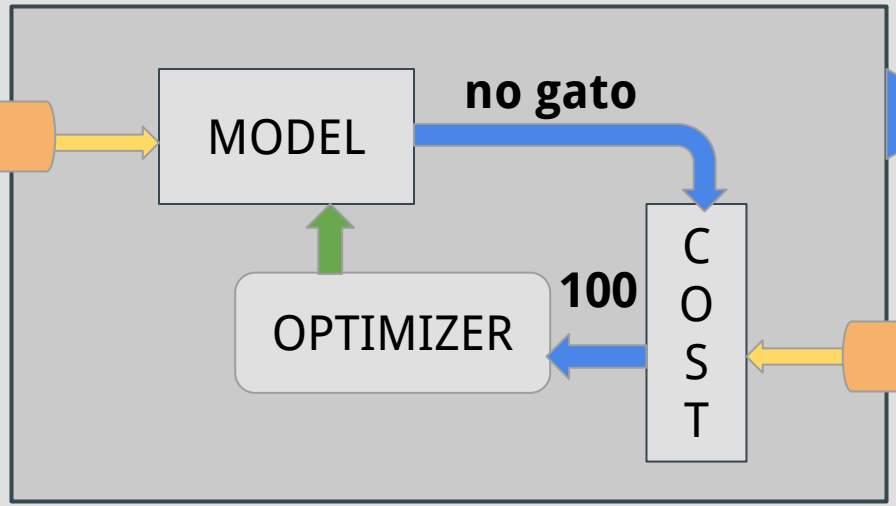
Sesión: Grafo + Datos



TensorFlow

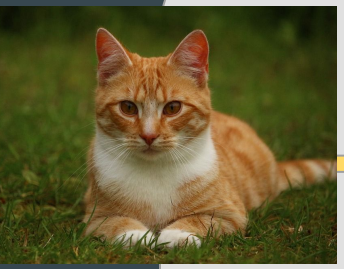
Sesión: Grafo + Datos





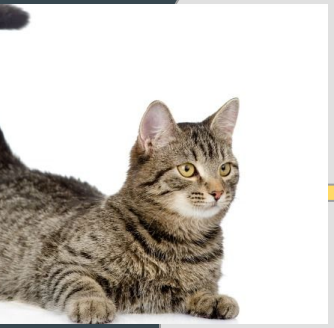
TensorFlow

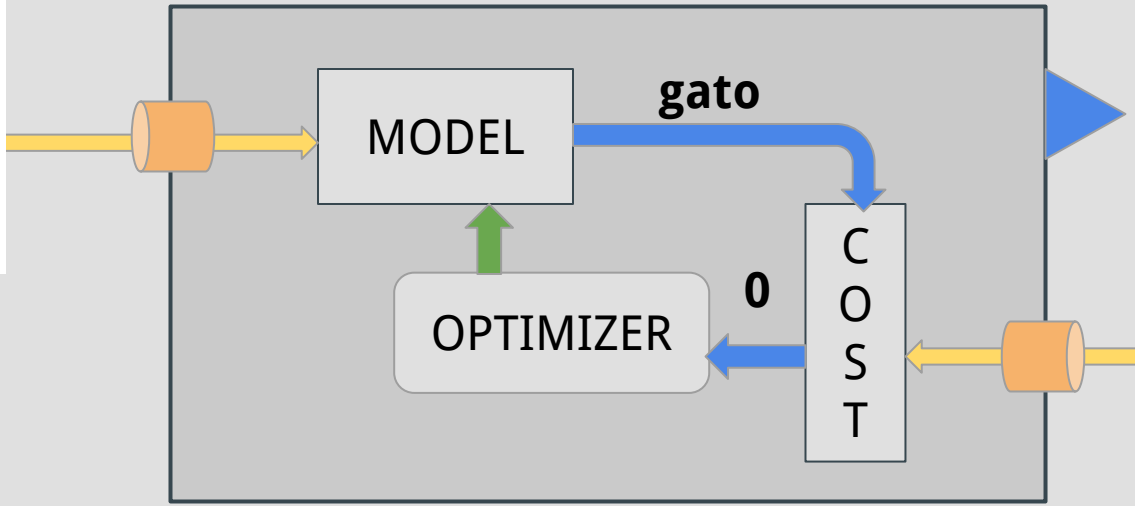
Sesión: Grafo + Datos



TensorFlow

Sesión: Grafo + Datos





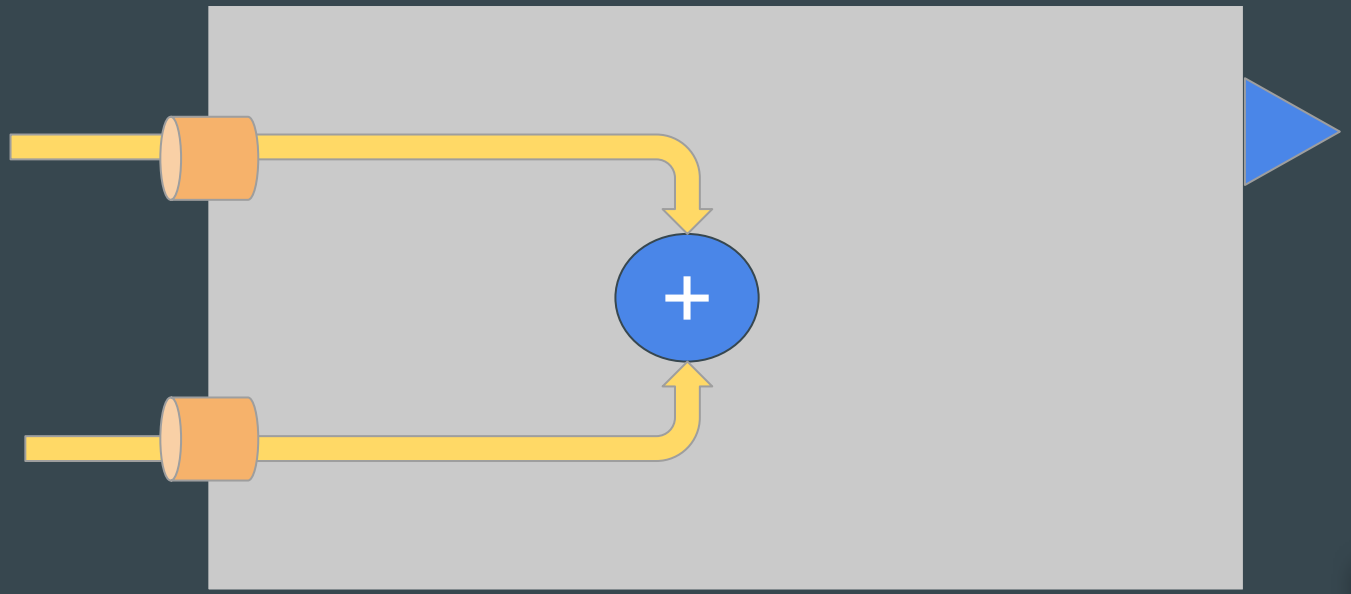
TensorFlow

Hello world!

Hello world!: Suma de dos enteros

```
import tensorflow as tf
```

Hello world!: Suma de dos enteros



Hello world!: Suma de dos enteros

```
##### GRAPH #####
a = tf.placeholder(tf.int32)
b = tf.placeholder(tf.int32)
suma = tf.add(a, b)

##### DATA #####
num1 = 3
num2 = 8
```

Hello world!: Suma de dos enteros

```
##### SESSION #####
with tf.Session() as sess:
    suma_resultado = sess.run(suma, feed_dict={
        a: num1,
        b: num2
        })
```

Vamos al notebook

Regresión

TensorFlow para Regresión: aprender a sumar

- Misión: aprender a sumar a través de 10,000 ejemplos.

$$x_1 + x_2 = y$$

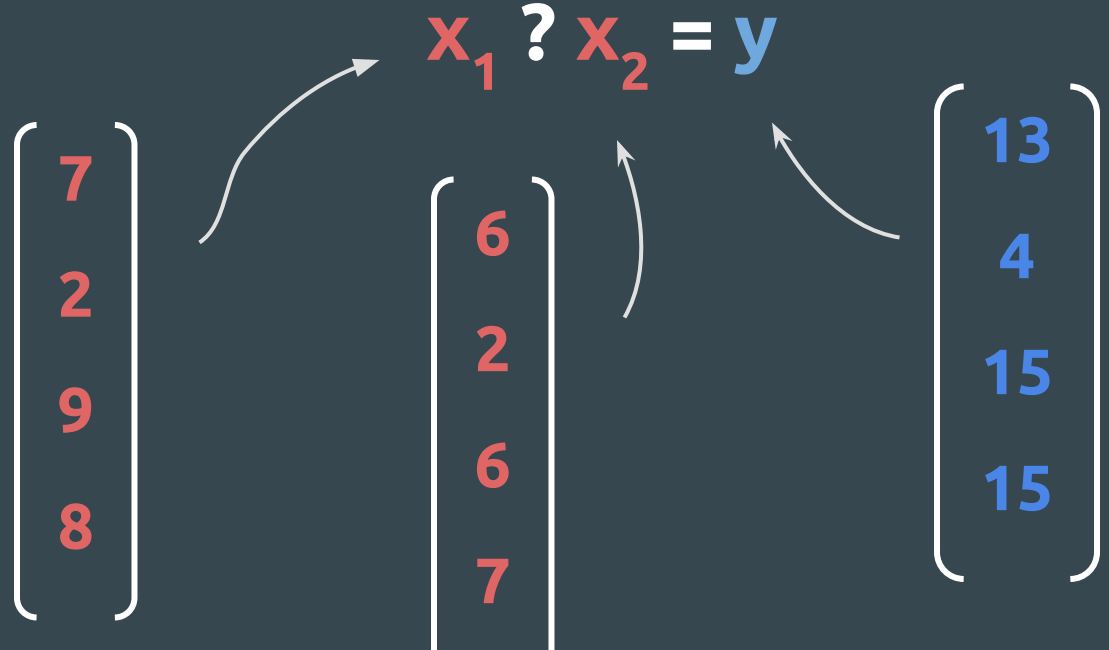
TensorFlow para Regresión: aprender a sumar

- Misión: aprender a sumar a través de 10,000 ejemplos.

$$x_1 \; ? \; x_2 = y$$

TensorFlow para Regresión: aprender a sumar

- Misión: aprender a sumar a través de 10,000 ejemplos.



TensorFlow para Regresión: aprender a sumar

- Misión: aprender a sumar a través de 10,000 ejemplos.

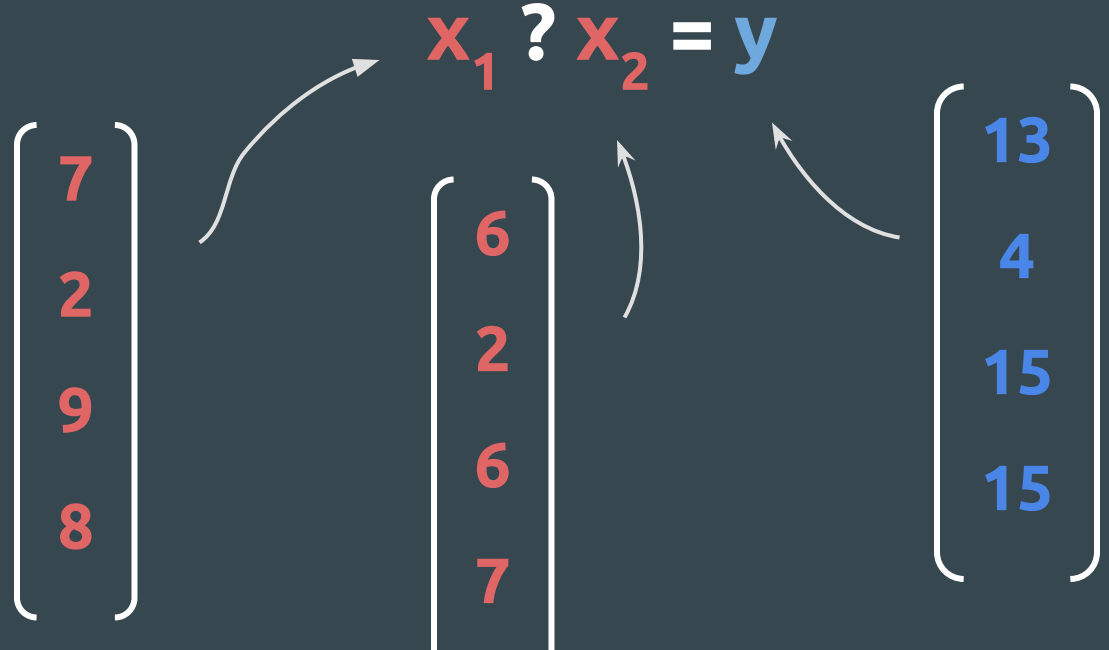
$x_1 \; ? \; x_2 = y$

250 m

TensorFlow

TensorFlow para Regresión: aprender a sumar

- Misión: aprender a sumar a través de 10,000 ejemplos.



TensorFlow para Regresión: aprender a sumar

- Misión: aprender a sumar a través de 10,000 ejemplos.

$$x_1 \; ? \; x_2 = y$$

- Partimos de que la relación entre **x** e **y** es una función lineal.

$$x \cdot W + b = y$$

TensorFlow para Regresión: aprender a sumar

- Misión: aprender a sumar a través de 10,000 ejemplos.

$$x_1 \; ? \; x_2 = y$$

- Partimos de que la relación entre **x** e **y** es una función lineal.

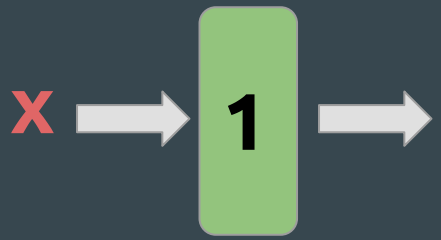
$$x \cdot W + b = y$$

variables a aprender

Red neuronal

$$x \cdot W_1 + b_1 = y$$

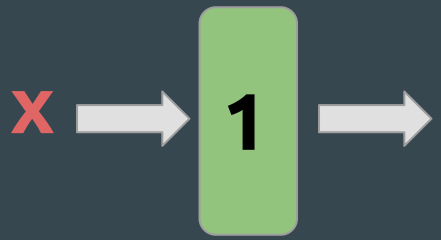
número de capa



TensorFlow

Red neuronal

$$x \cdot W_1 + b_1 = y$$

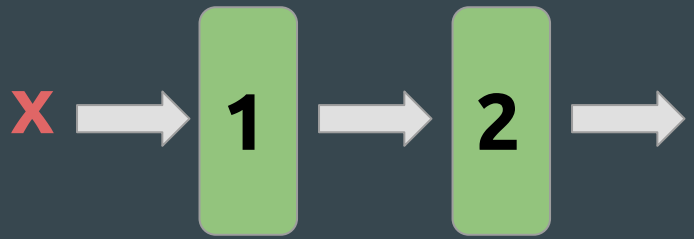


TensorFlow

Red neuronal

$$x \cdot W_1 + b_1 = y$$

$$(x \cdot W_1 + b_1) \cdot W_2 + b_2 = y$$



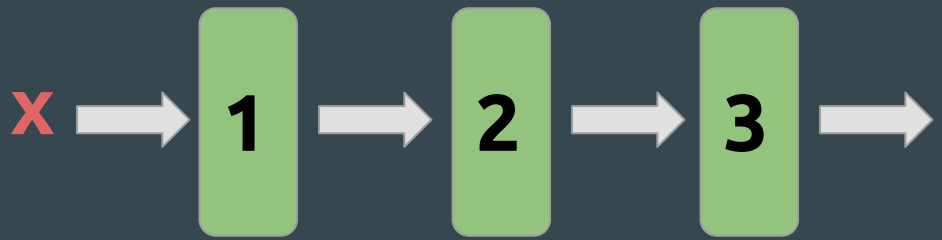
TensorFlow

Red neuronal

$$x \cdot W_1 + b_1 = y$$

$$(x \cdot W_1 + b_1) \cdot W_2 + b_2 = y$$

$$((x \cdot W_1 + b_1) \cdot W_2 + b_2) \cdot W_3 + b_3 = y$$



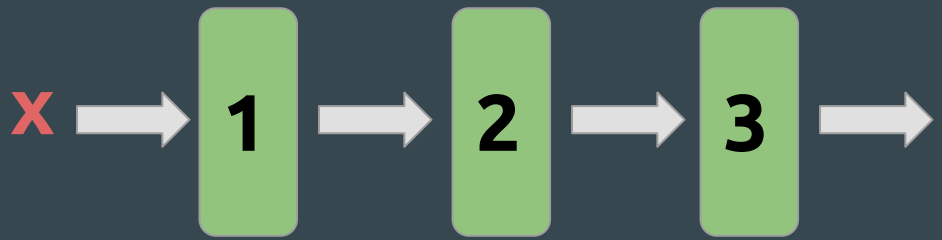
TensorFlow

Red neuronal

$$x \cdot W_1 + b_1 = y$$

$$(x \cdot W_1 + b_1) \cdot W_2 + b_2 = y$$

$$\tanh(\sigma(x \cdot W_1 + b_1) \cdot W_2 + b_2) \cdot W_3 + b_3 = y$$



TensorFlow

aquí lo que quiero es que se vea las dimensiones de W

tengo que explicar que las w1, w2, b las va a definir el optimizador

explicar simple

no dar detalle

tengo que explicar la idea y que quede clara

TensorFlow para Regresión: aprender a sumar

```
# PLACEHOLDERS
x = tf.placeholder(tf.float32, [None, 2])
y = tf.placeholder(tf.float32, [None, 1])
```

TensorFlow para Regresión: aprender a sumar

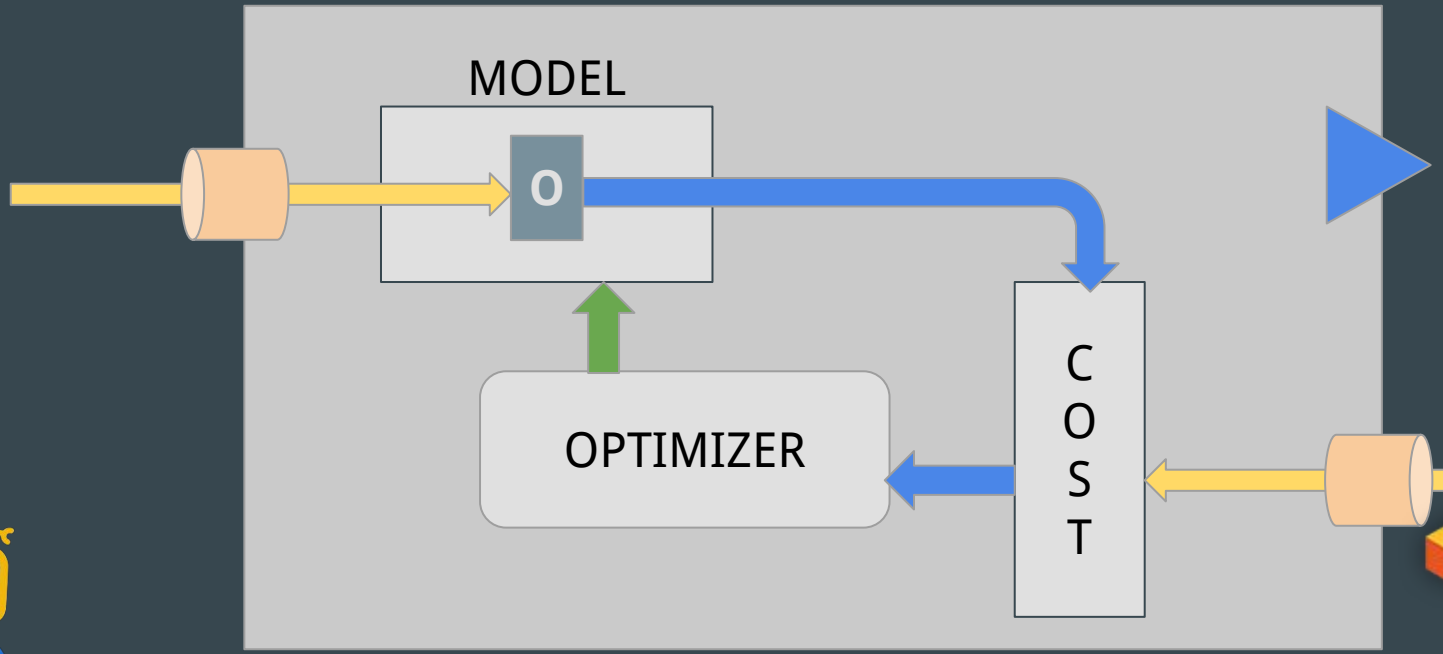
```
# PLACEHOLDERS
x = tf.placeholder(tf.float32, [None, 2])
y = tf.placeholder(tf.float32, [None, 1])
```

(no sabemos cuántos ejemplos tendremos, pero sí que cada uno tiene 2 datos de entrada y 1 de salida)

TensorFlow para Regresión: aprender a sumar

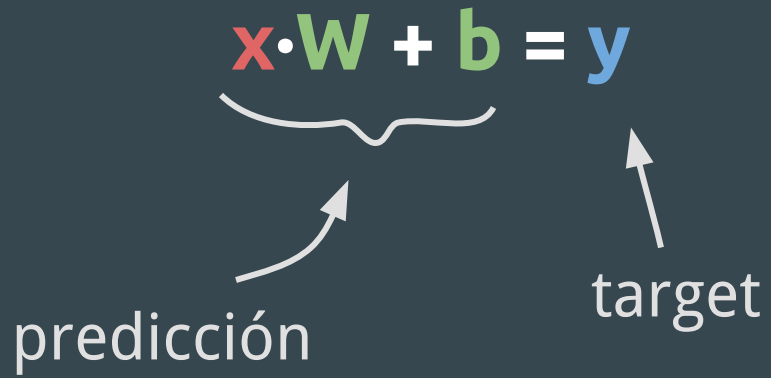
```
# MODEL
W = tf.Variable(tf.truncated_normal([2, 1], stddev=0.05))
b = tf.Variable(tf.random_normal([1]))

output = tf.add(tf.matmul(x, W), b)
```

TensorFlow para Regresión: aprender a sumar



TensorFlow

Cost (loss) function



Cost (loss) function

$$y - (x \cdot W + b)$$

Cost (loss) function

$$[\, y - (\, x \cdot W + b \,) \,]^2$$

Cost (loss) function

$$\Sigma[\, y_i - (\, x_i \cdot W + b \,)\,]^2$$

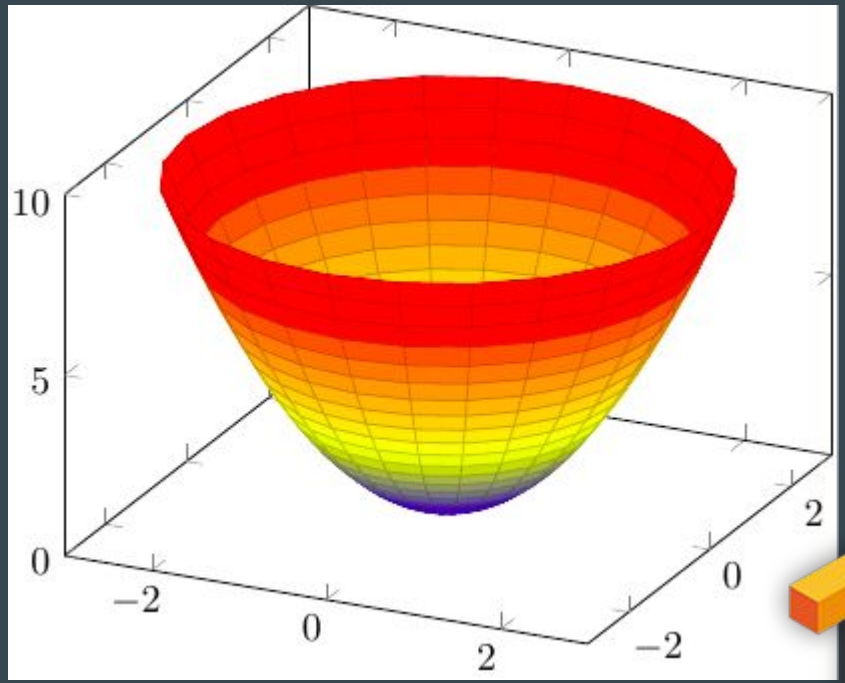
TensorFlow para Regresión: aprender a sumar

```
cost = tf.reduce_sum(tf.square(output - y))
```

Gradient Descent

función de coste

```
cost = f(w1, w2, b)
```

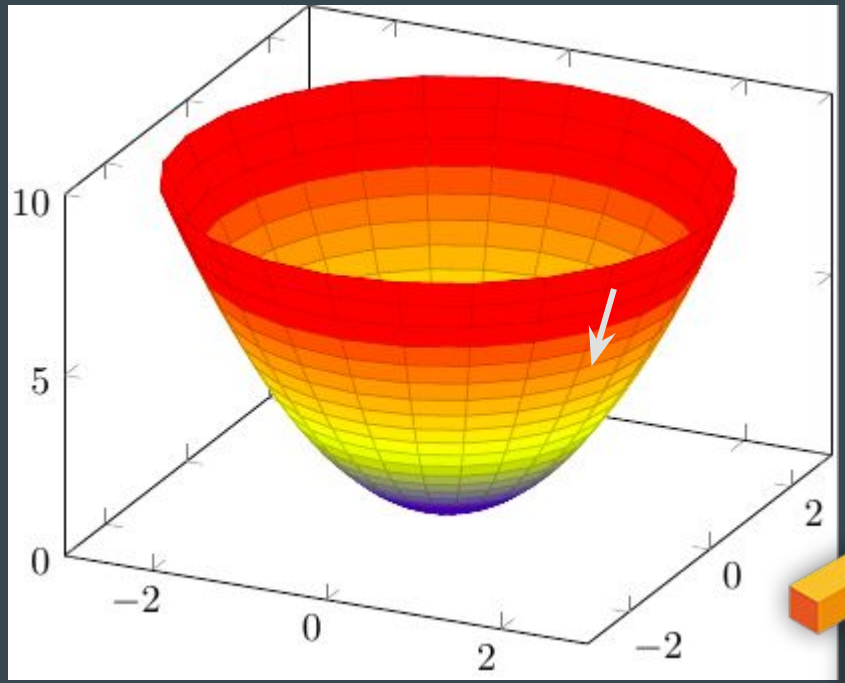


TensorFlow

Gradient Descent

función de coste

```
cost = f(w1, w2, b)
```

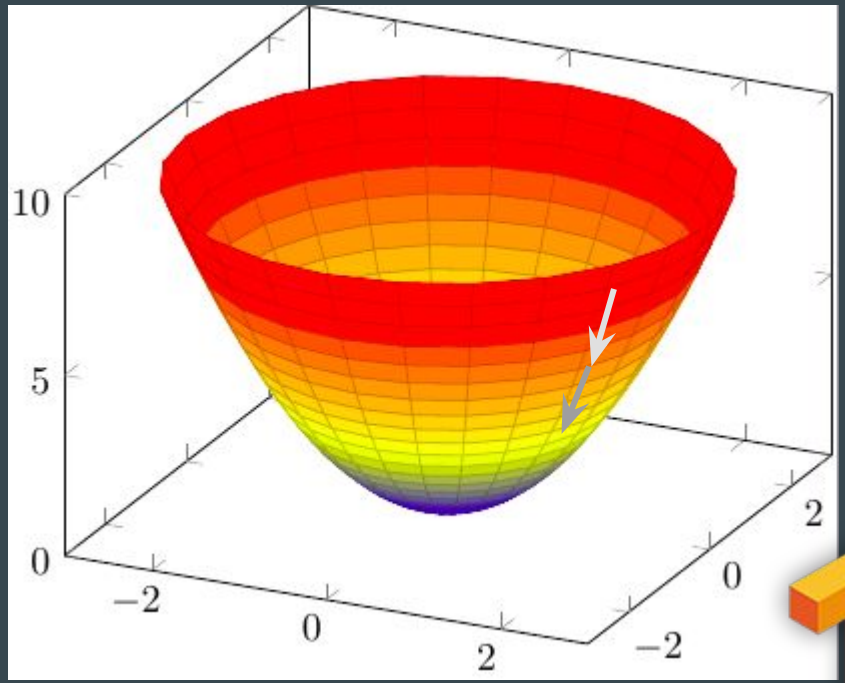


TensorFlow

Gradient Descent

función de coste

```
cost = f(w1, w2, b)
```

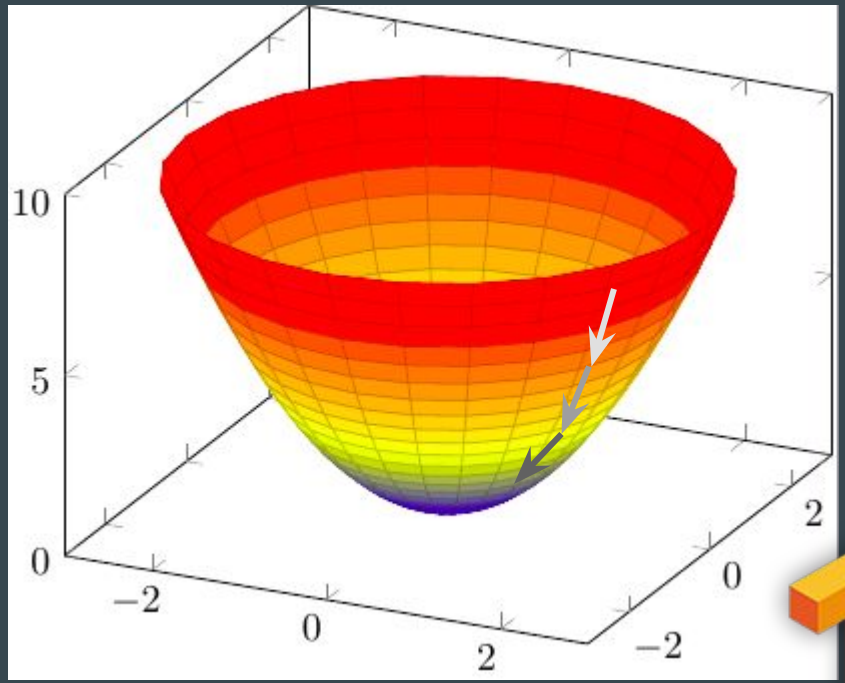


TensorFlow

Gradient Descent

función de coste

```
cost = f(w1, w2, b)
```

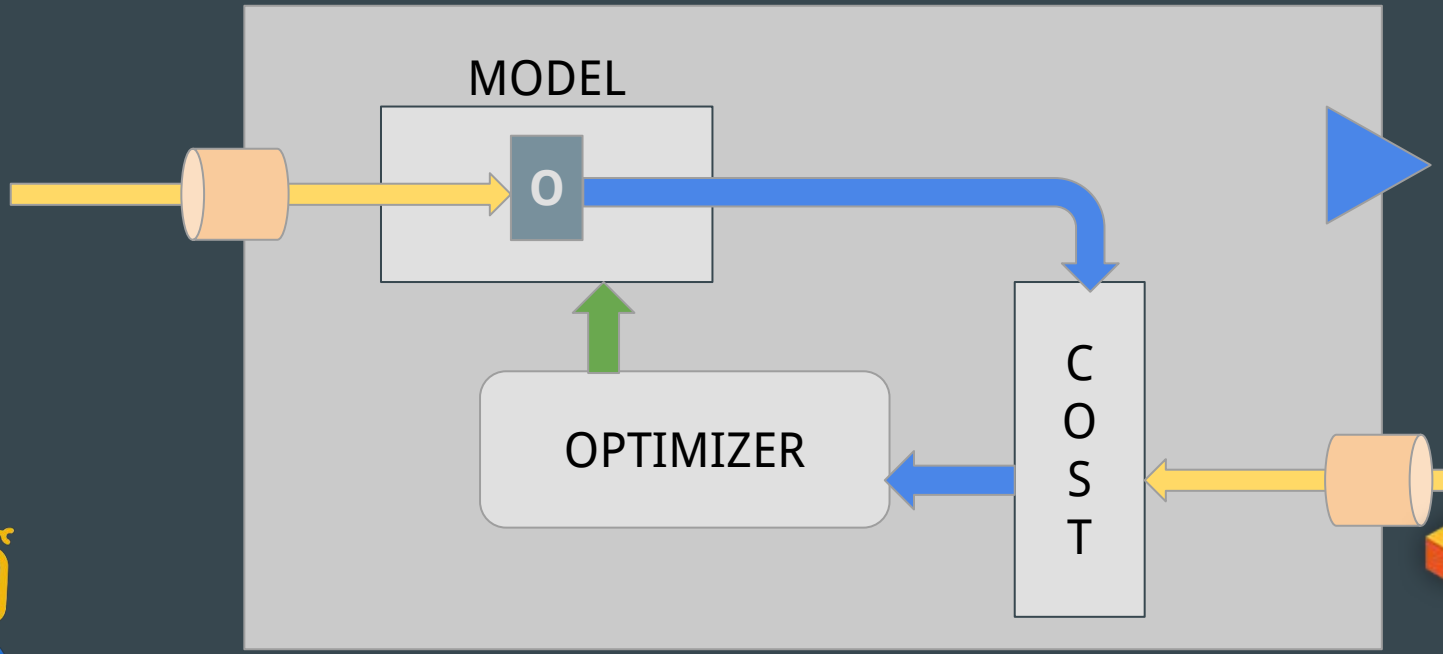


TensorFlow

TensorFlow para Regresión: aprender a sumar

```
optimizer =
tf.train.GradientDescentOptimizer(learning_rate=0.00001)

optimizer = optimizer.minimize(cost)
```

TensorFlow para Regresión: aprender a sumar

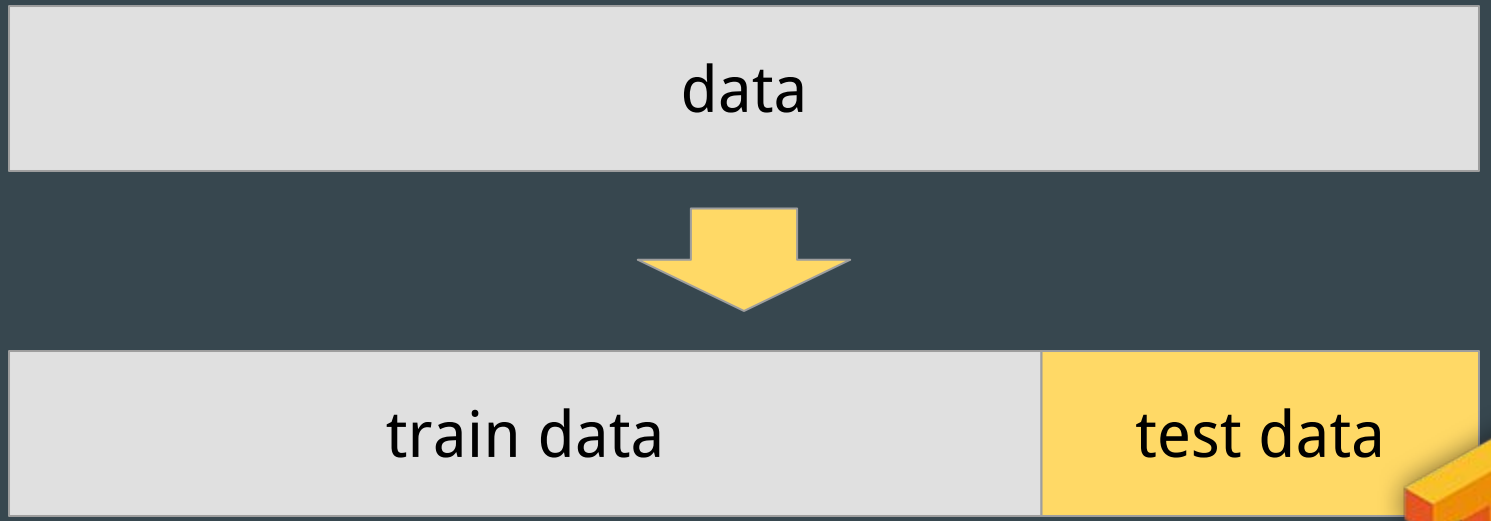


TensorFlow

Data split

data

Data split

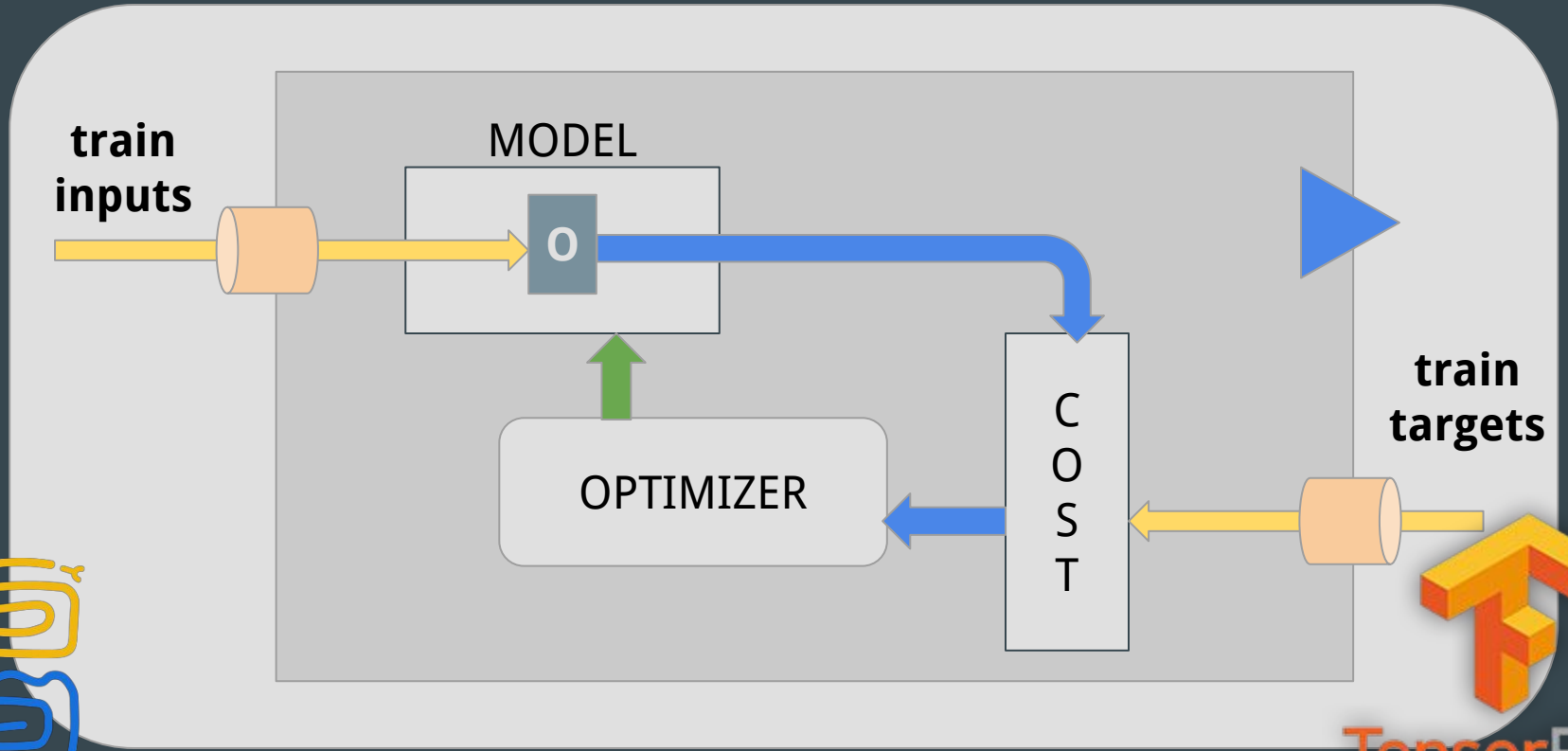


TensorFlow

TensorFlow para Regresión: aprender a sumar

```
from helper import get_data, split_data

# DATA
inputs, targets = get_data(max_int=10, size=10000)

# split train and test data
train_inputs, test_inputs, train_targets, test_targets =
split_data(inputs, targets)
```

TensorFlow para Regresión: aprender a sumar



TensorFlow para Regresión: aprender a sumar

```
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())

    for epoch in range(epochs):
        sess.run(optimizer, feed_dict={
                        x: train_inputs,
                        y: train_targets
                        })
```

Vamos al notebook

he dividido, además, los datos en cachitos, tomados de forma aleatoria

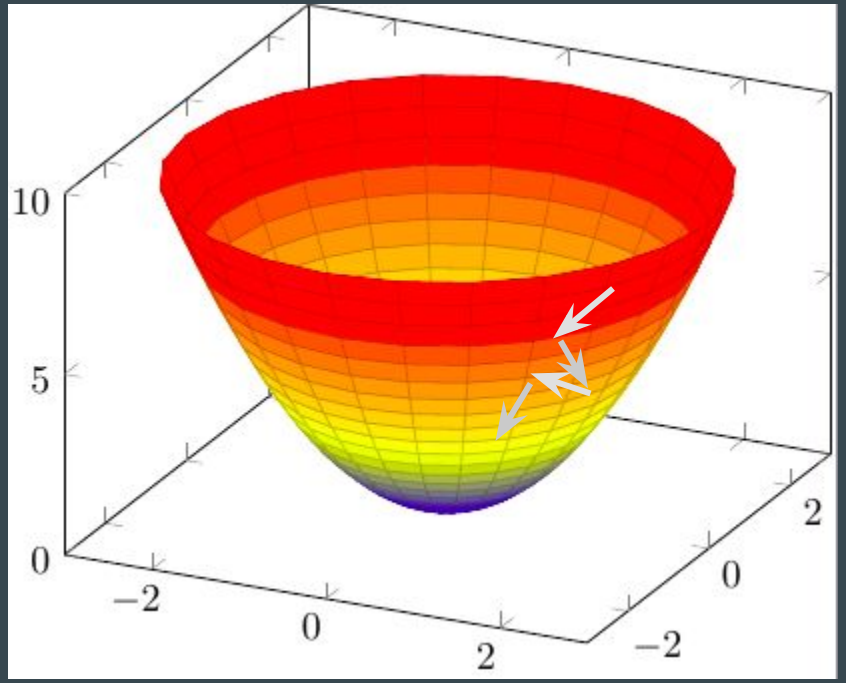
estos cachitos se llaman batches

al calcular el gradiente solo teniendo en cuenta un cachito de los datos, el gradiente no se calcula bien. es una aproximación tirando a mala

esto se llama stochastic gradient descent

al final se consigue que los resultados sean mejores

stochastic gradient descent



Clasificación

gato

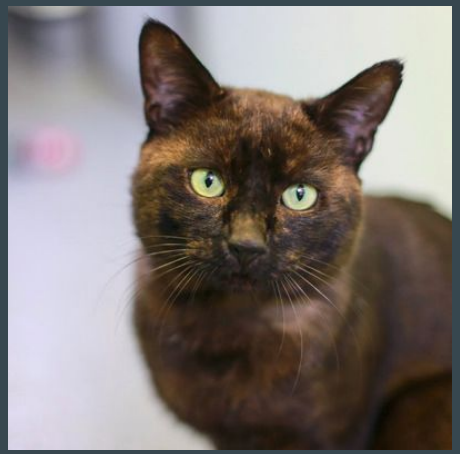
no gato

TensorFlow

[0, 1]

[1, 0]

TensorFlow



MODEL

[0.175,

0.825]

TensorFlow

MODEL

[0.457,

0.543]

TensorFlow

MODEL

[0.457,
+ 0.543]
1.000

TensorFlow

TensorFlow para Clasificación

- Misión: aprender si la suma de dos números es mayor que 10.

$$\text{if } (x_1 + x_2 > 10) \text{ then } y = [0; 1]$$

$$\text{else } y = [1; 0]$$

TensorFlow para Clasificación

- Misión: aprender si la suma de dos números es mayor que 10.

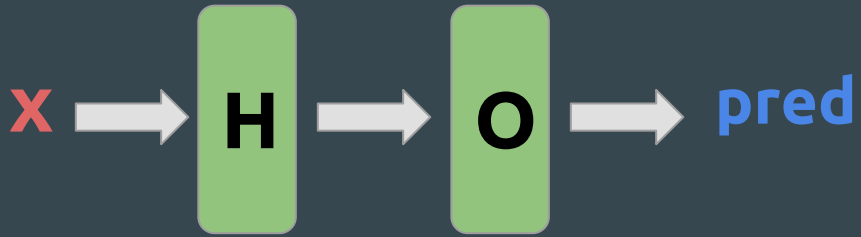
$$x_1 \;??\; x_2 = y$$

TensorFlow para Clasificación

- Misión: aprender si la suma de dos números es mayor que 10.

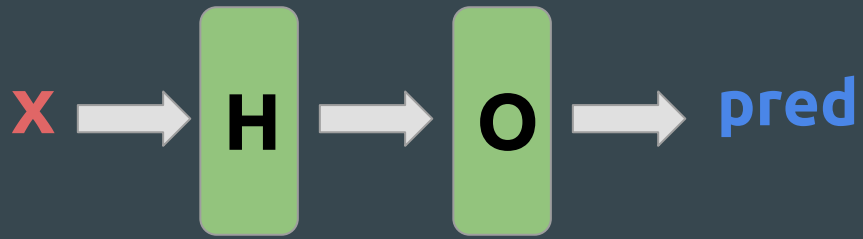
$$x_1 \;??\; x_2 = y$$

- Más complejidad: añadimos una nueva capa.



Redes neuronales: intuición

Las primeras capas extraen información más básica, y las siguientes trabajan a partir de esa información.



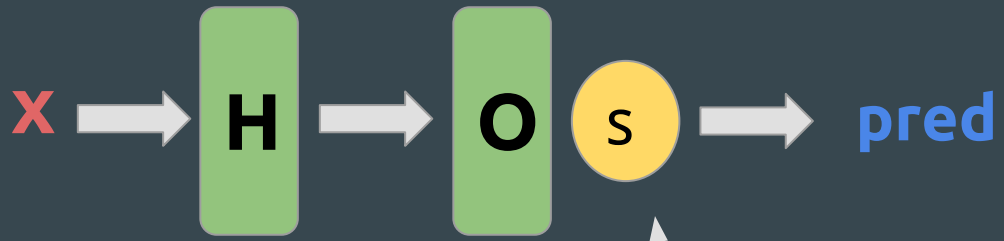
hace la suma

clasifica la suma

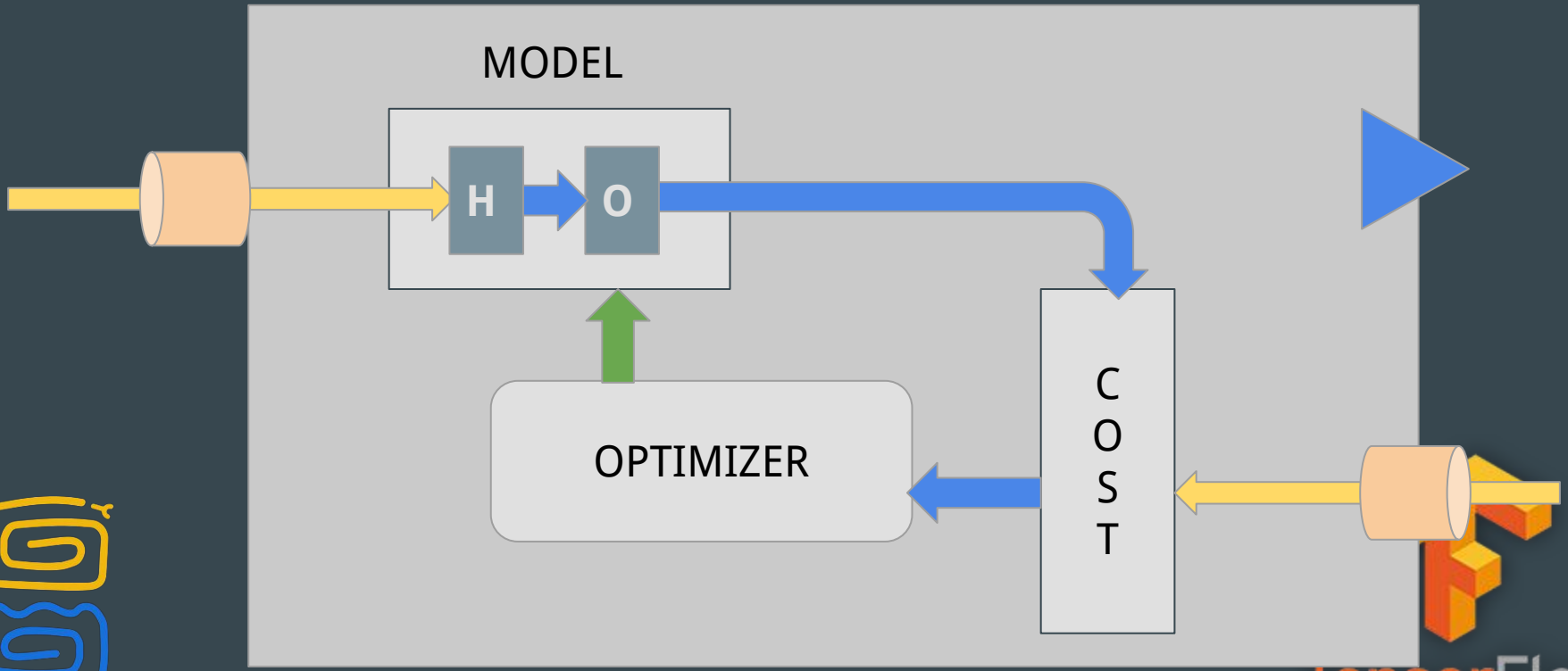
TensorFlow

Redes neuronales: intuición

Las primeras capas extraen información más básica, y las siguientes trabajan a partir de esa información.



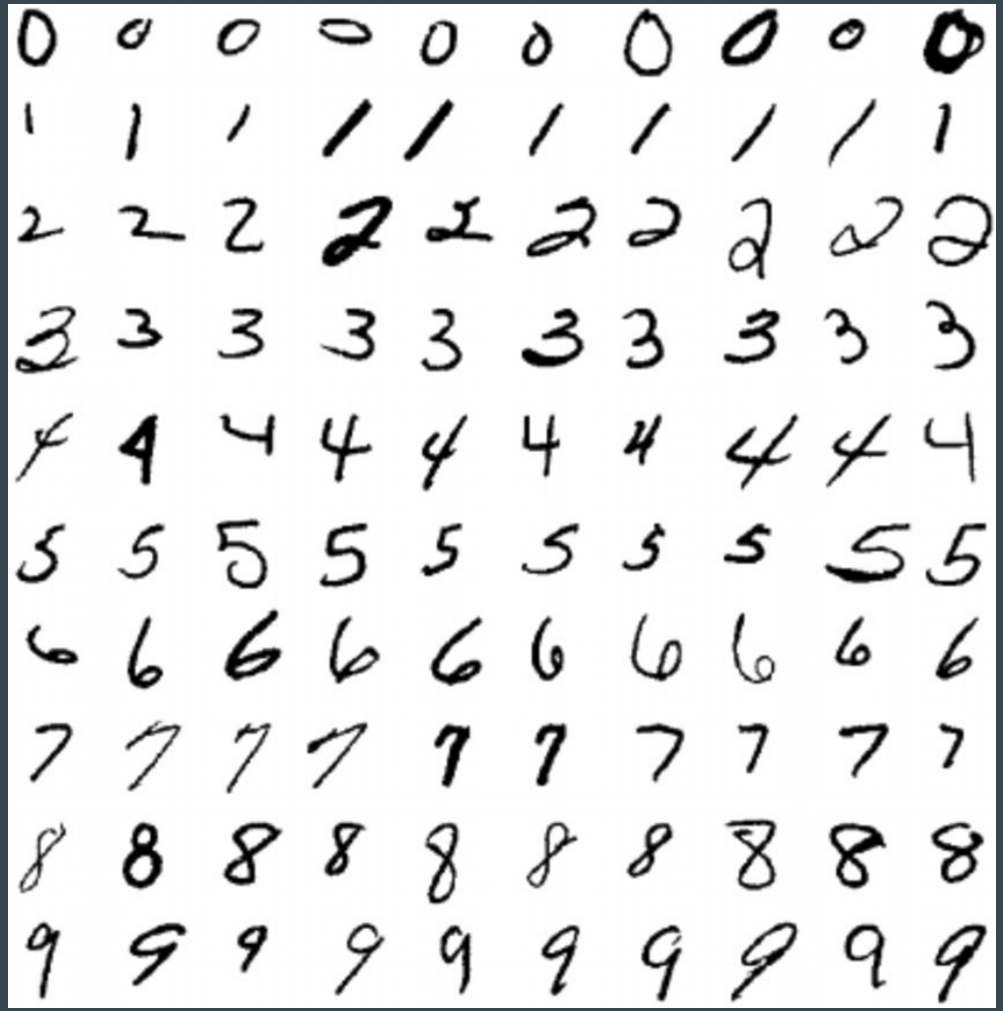
TensorFlow


MODEL
H
O
OPTIMIZER
C
O
S
T
TensorFlow

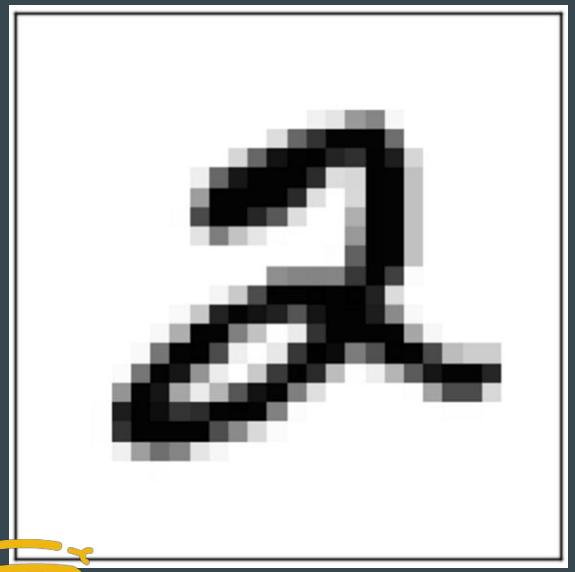
Vamos al notebook

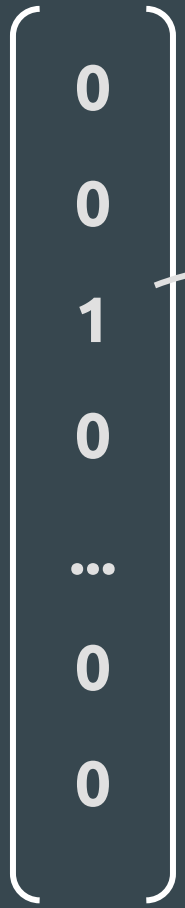
MNIST

MNIST dataset



TensorFlow

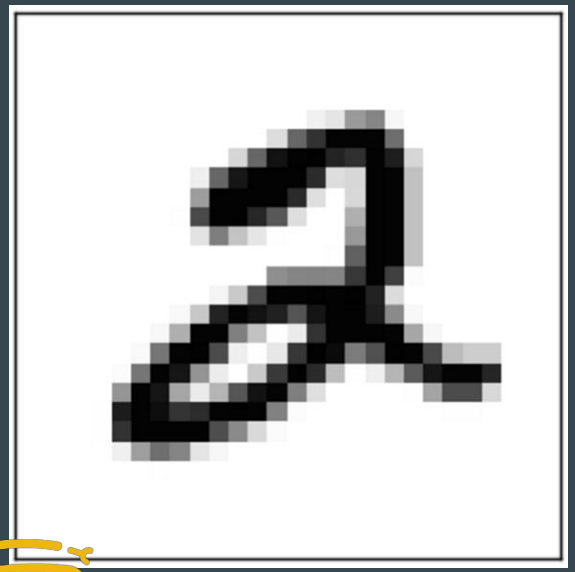




argmax

2

TensorFlow



MODEL

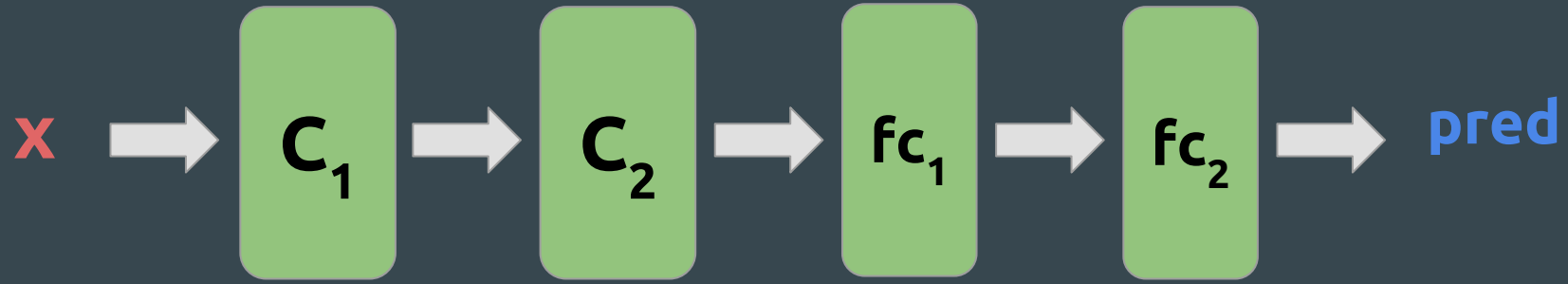
0
0
1
0
...
0
0

argmax

2

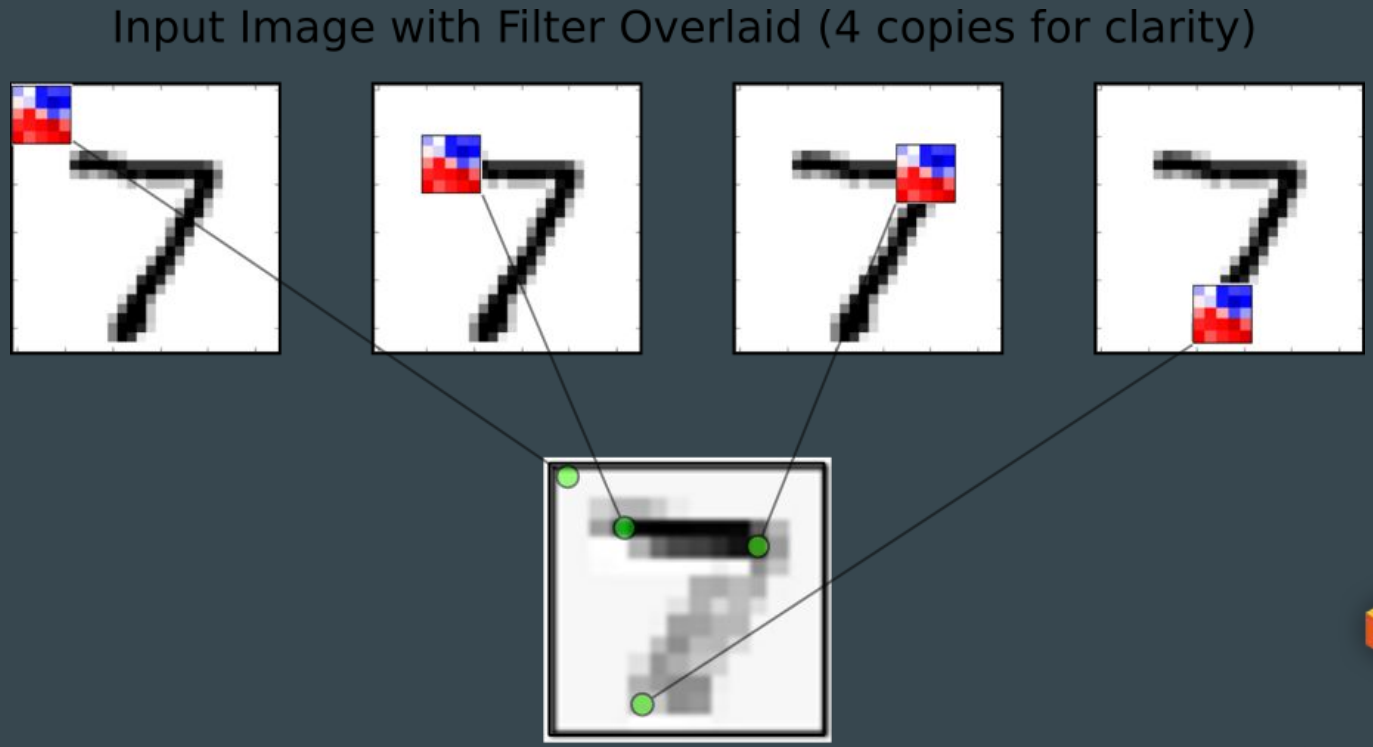
TensorFlow

el modelo



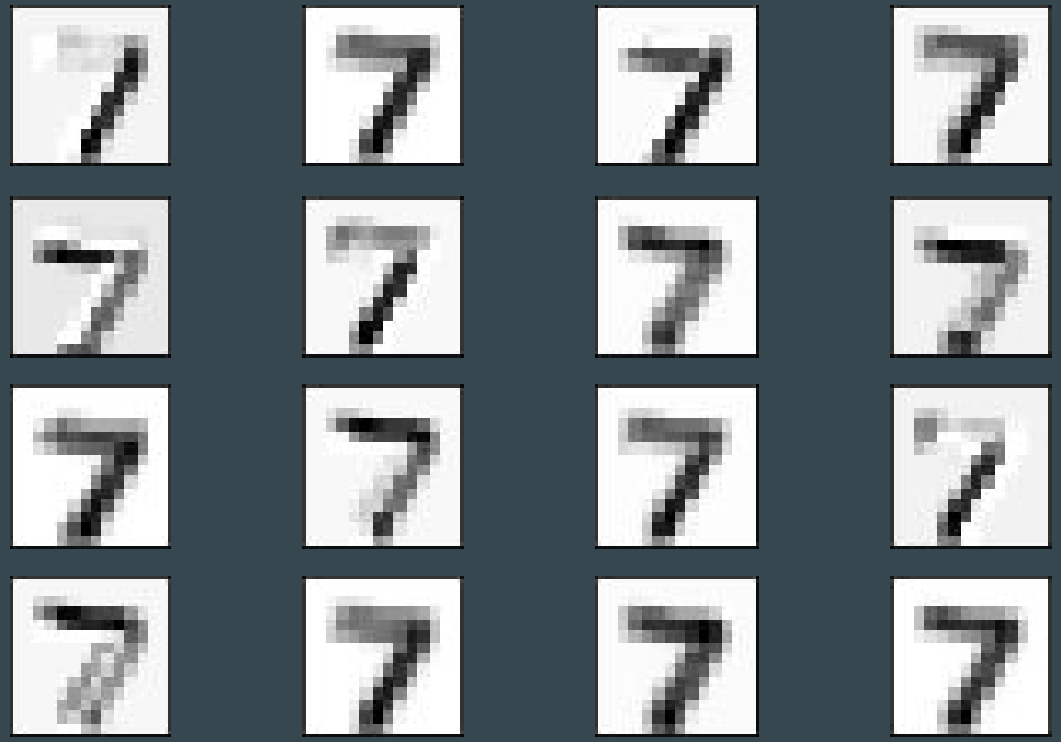
TensorFlow

convolución



TensorFlow

primera capa: información básica



TensorFlow

Vamos al notebook

Para saber más

Deep learning

- Neural Networks and Deep Learning - **Michael Nielsen**
- Stanford's CS231n - **Andrej Karpathy**

Tensorflow

- Tensorflow Tutorials - **Hvass Laboratories**
- Deep Learning Foundations Nanodegree - **Udacity**

Para empezar a saber más

Basics

- Intro to Data Science - Udacity
- Intro to Machine Learning - Udacity

alesolano/mastering_tensorflow