
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

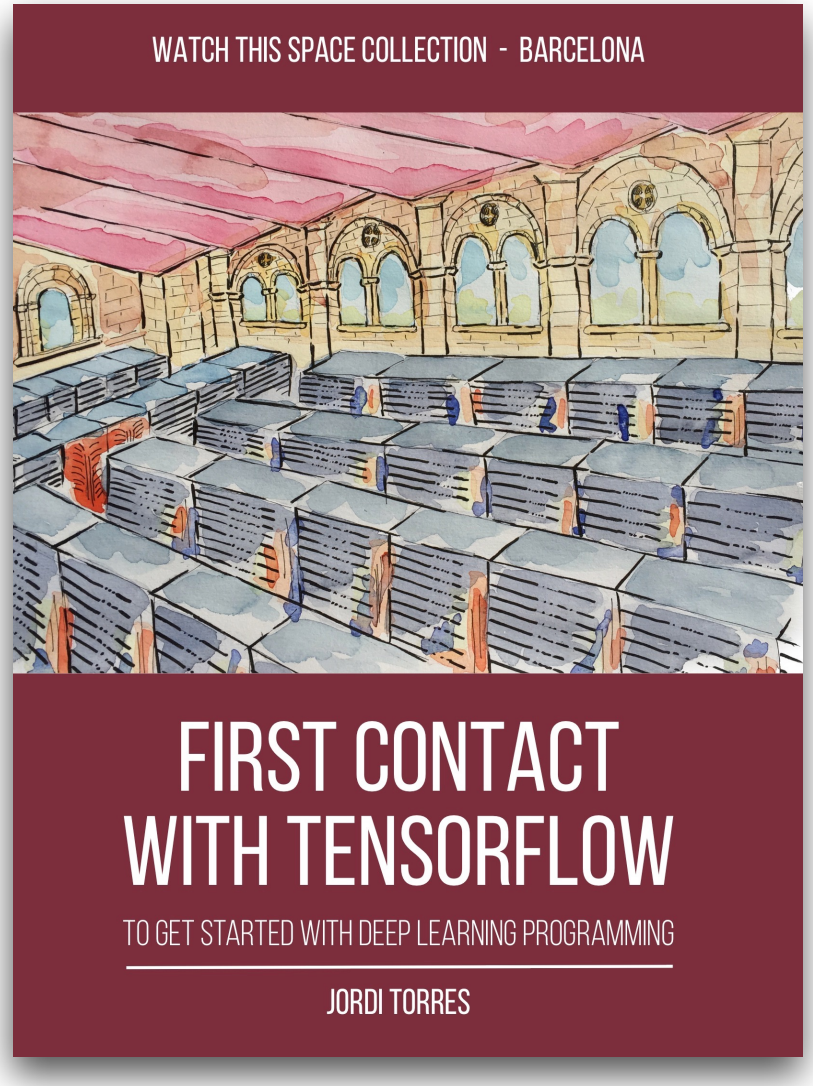

# First Contact with TensorFlow Part 1: Basics

PRACE Advanced Training Centre Course: Big Data Analytics

Autonomic Systems and eBusiness Platforms
Computer Science Department
Jordi Torres

10/02/2016

# Complete Course Content:




## Part1: Basics

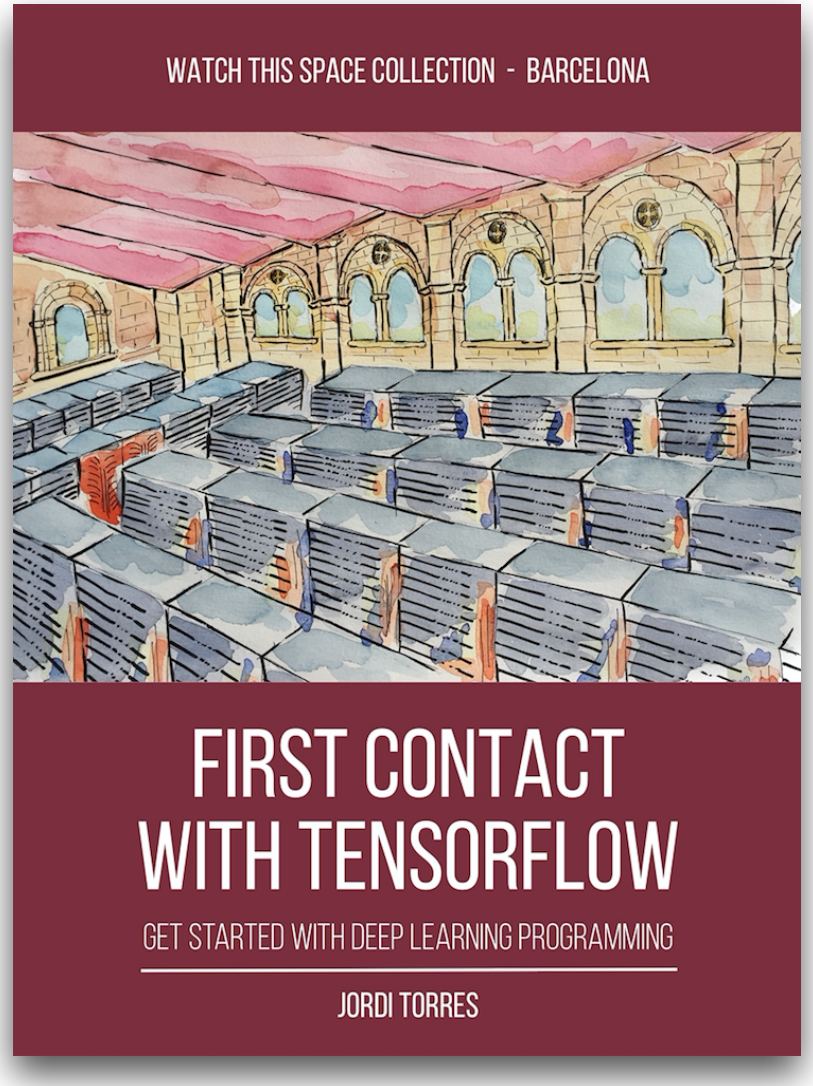
Introduction
1. TensorFlow Basics
2. Linear Regression
3. Clustering

## Part2: Advanced

4. Single Layer Neural Network
5. Multi-layer Neural Networs
6. TensorFlow and GPUs

Closing

# Today:

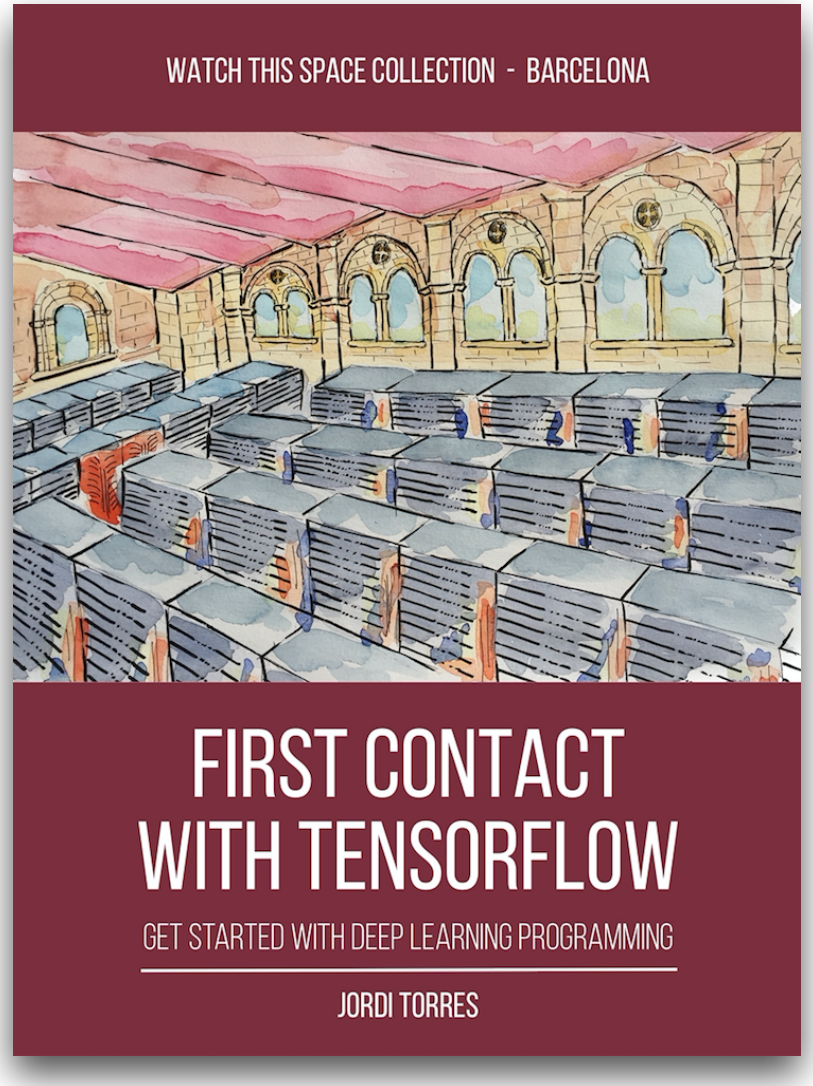



## Part1: Basics

Introduction

1. TensorFlow Basics
2. Linear Regression
3. Clustering

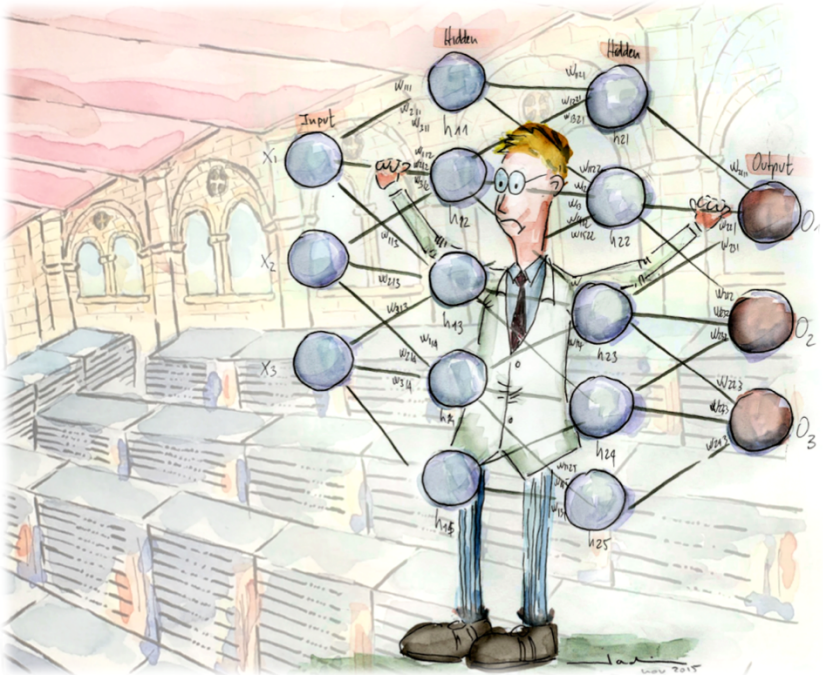
# Content:




"The Fourth Industrial Revolution, which includes developments in previously disjointed fields such as artificial intelligence and machine-learning, robotics, nanotechnology, 3-D printing, and genetics and biotechnology, will cause widespread disruption not only to business models but also to labour markets over the next five years, with enormous change predicted in the skill sets needed to thrive in the new landscape."

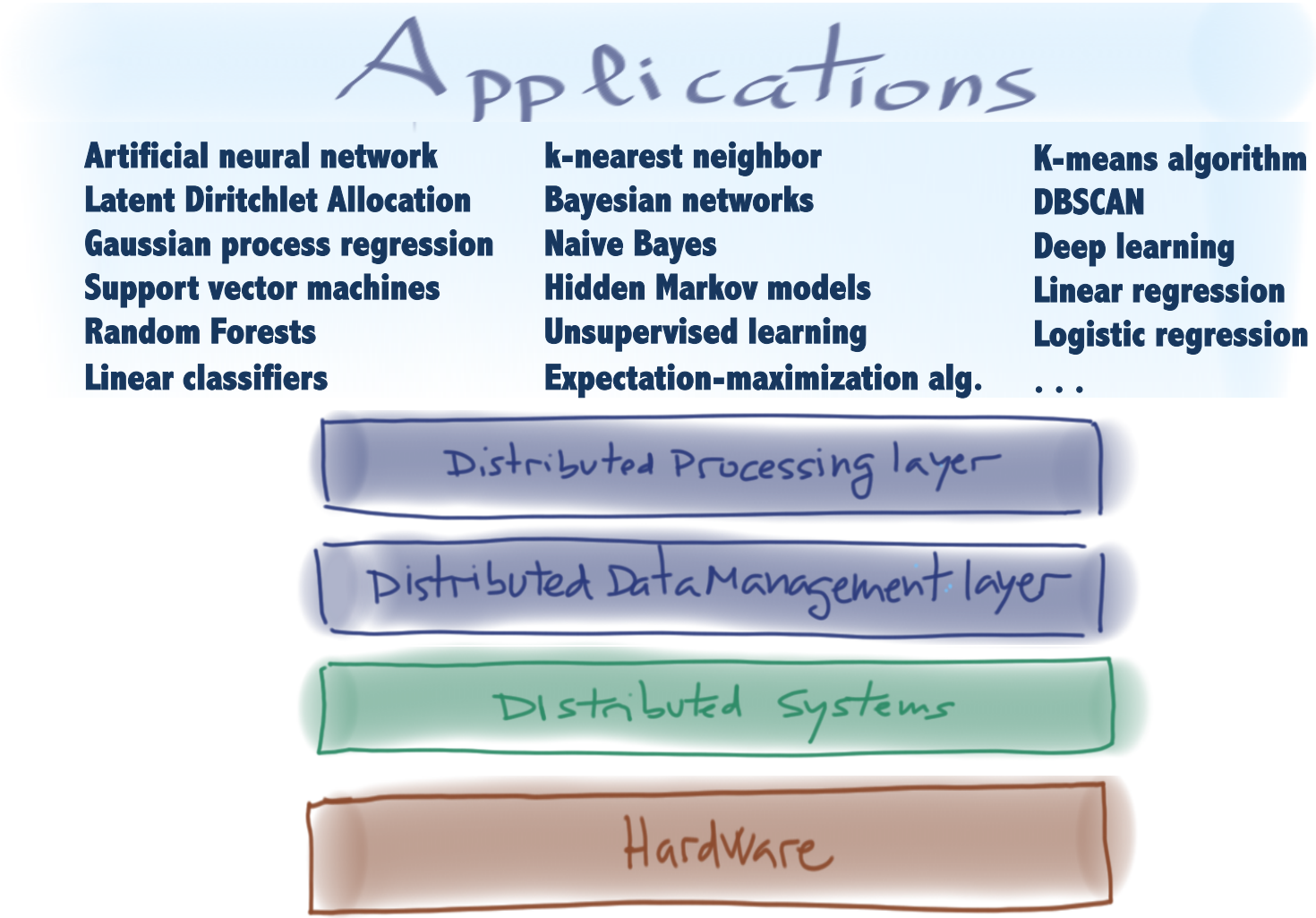
# ~~Future?~~ Present!

## New type of Computing is required

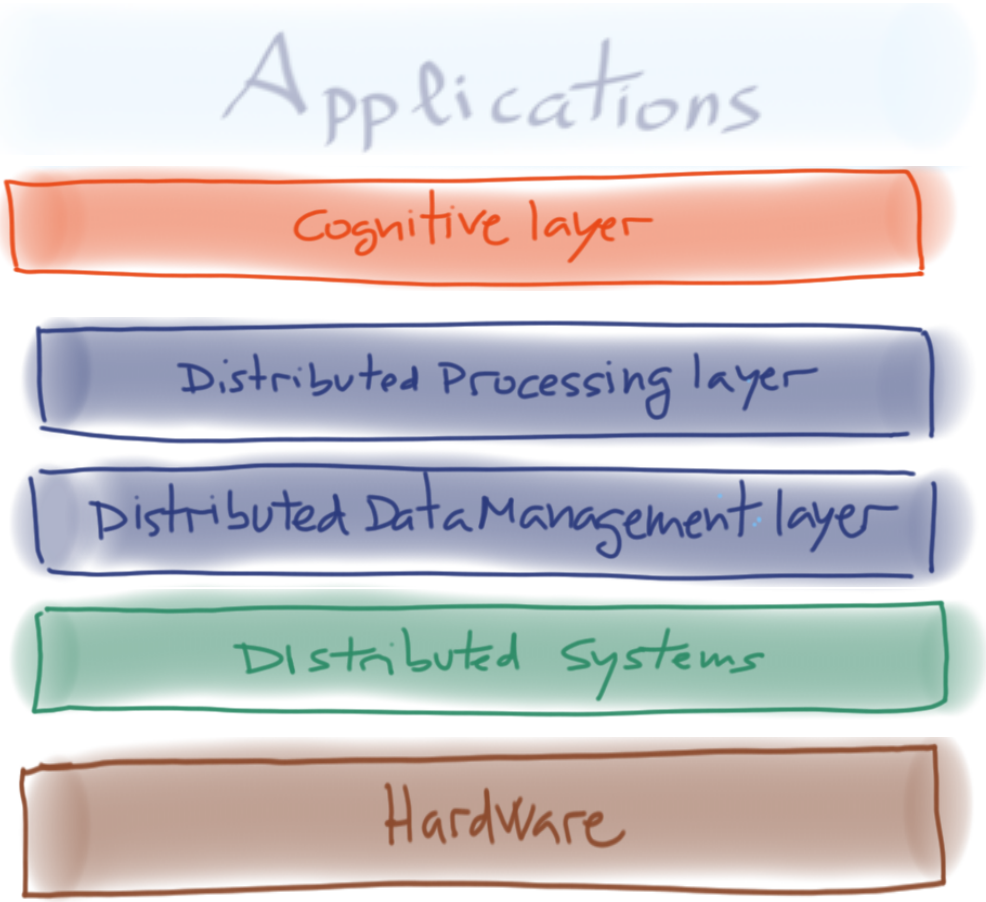



Giving computers a greater ability to understand information, and to learn, to reason, and act upon it

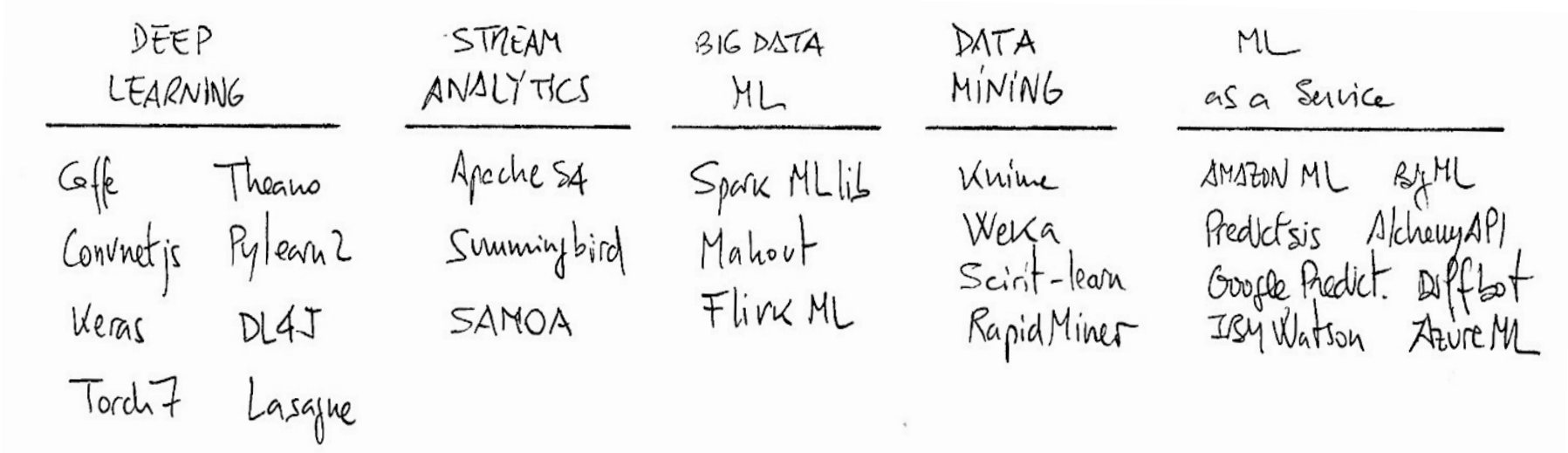
# Data Analytics Today



# Advanced Analytics will be provided by the “system”



# Meanwhile …




# … and TensorFlow!

# Attendance background?

“The course has a practical nature, and therefore I reduced the theoretical part as much as possible, assuming that the attendance has some basic understanding about Machine Learning”

# Machine Learning?

- Field of Computer Science that evolved from the study of pattern recognition and computational learning theory into Artificial Intelligence.
- Its goal is to give computers the ability to learn without being explicitly programmed.
- For this purpose, Machine Learning uses mathematical/ statistical techniques to construct models from a set of observed data rather than have specific set of instructions entered by the user that define the model for that set of data.

# Why Deep Learning?

- Conventional Machine Learning techniques were limited in their ability to process natural data in their raw form;

  for example, to identify objects in images or transcribe speech into text.

- Increasingly, these applications make use of a class of techniques called Deep Learning.

  These methods are based on “deep” multi-layer neural networks with many hidden layers that attempt to model high-level abstractions in data.

- Right now, a research in diverse types of Deep Learning’s networks is being developed

  Deep Convolutional Nets have brought breakthroughs in processing images and speech: we will consider this type in this course.

# Why is Deep Learning taking off now?

- It is know that many of the ideas used in Deep Learning have been around for decades.
- One of the key drivers of its current progress is clearly the huge deluge of data available today. Thanks to the advent of Big Data these models can be “trained” by exposing them to large data sets that were previously unavailable.
- But another not less important driver is the computation power available today. As an example, due the raising of GPUs, Deep Learning’s community started shifting to GPUs (TensorFlow works with GPUs).

# Data deluge


SOURCE: http://www.cray.com/Assets/Images/urika/edge/analytics-infographic.html

# Evolution of Computers

**FLOP/second**

E P T G

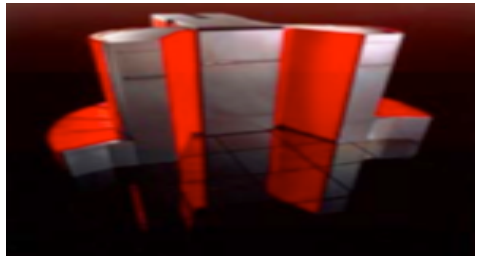
1000000000000000000

~2019
? ($1x10^7$ processadors

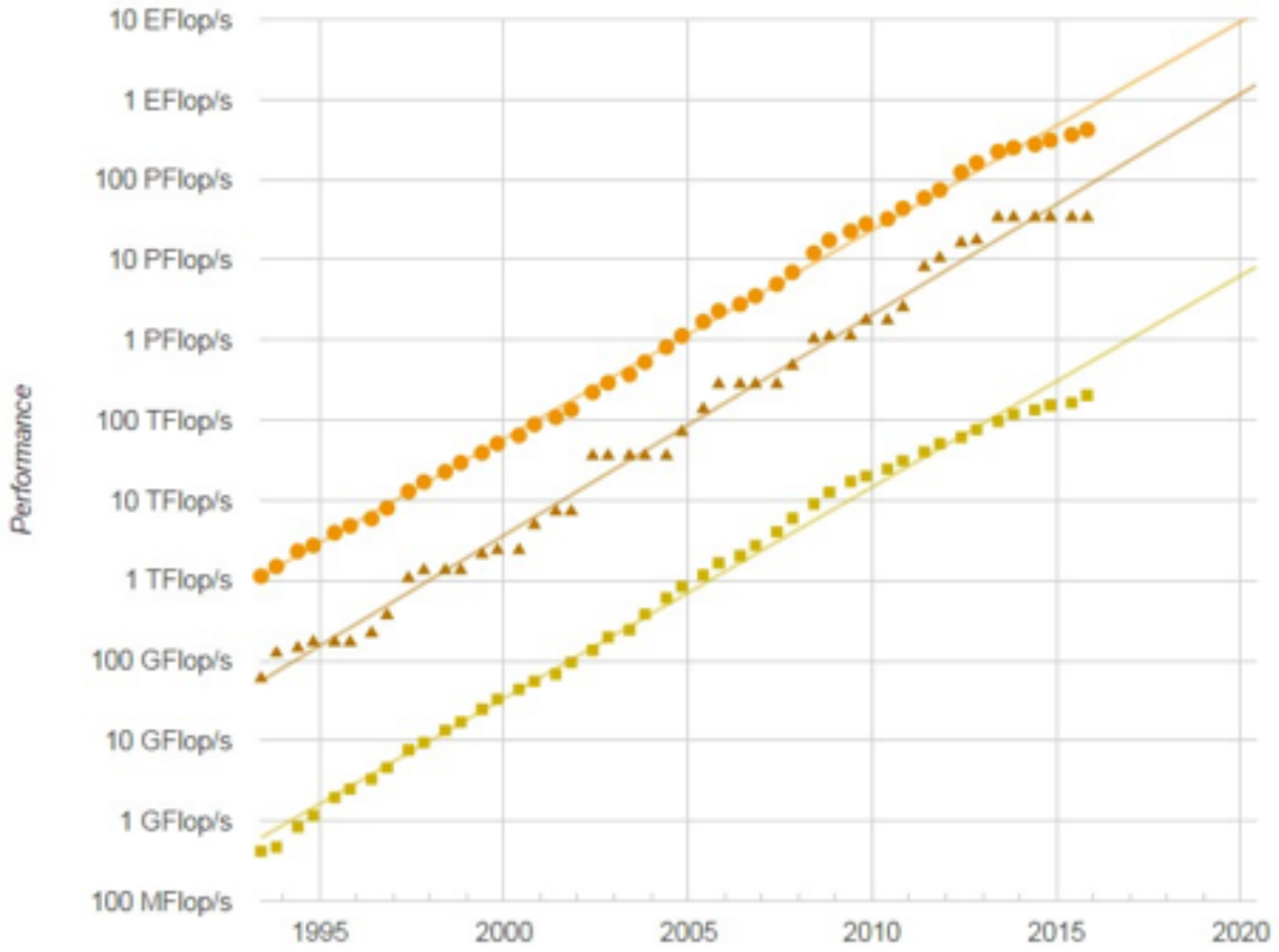
2008
Cray XT5 (15000 processadors)

1998
Cray T3E (1024 processadors)

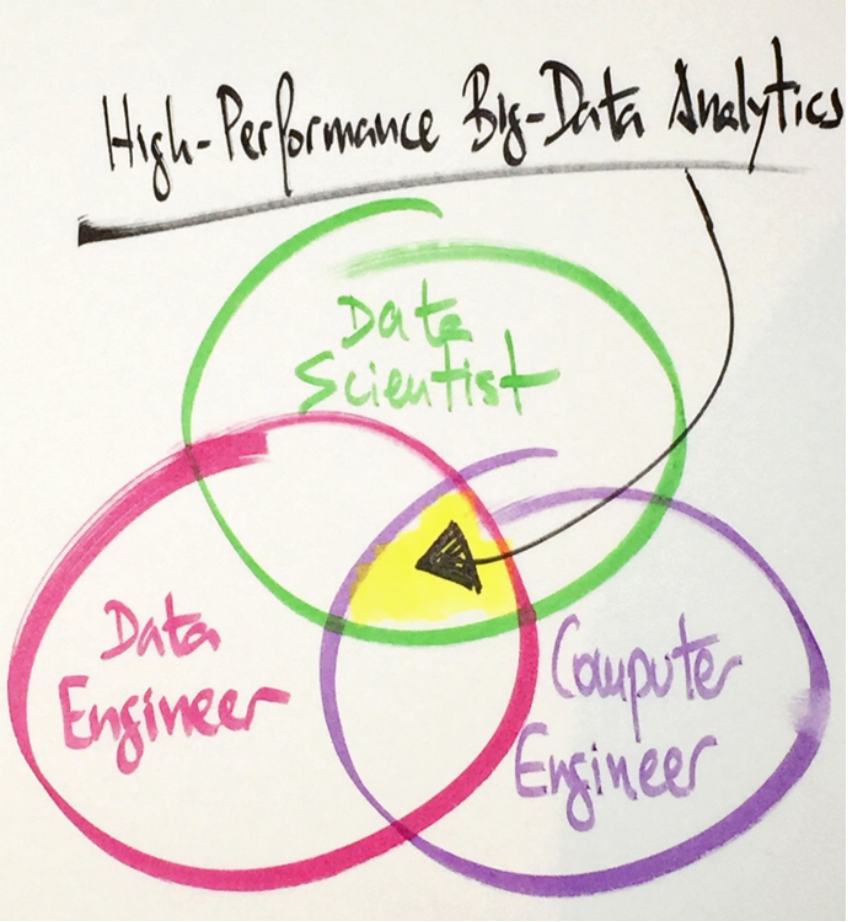
1988
Cray Y-MP (8 processadors)



# Top500 list


10 EFlop/s
1 EFlop/s
100 PFlop/s
10 PFlop/s
1 PFlop/s
100 TFlop/s
10 TFlop/s
1 TFlop/s
100 GFlop/s
10 GFlop/s
1 GFlop/s
100 MFlop/s
Performance
1995
2000
2005
2010
2015
2020

# New requirements!



“I would like to thank the author, whom I have the pleasure of knowing, the effort to disseminate this technology. He had written this book (first Spanish version) in record time, two months after it was announced the open source project.”

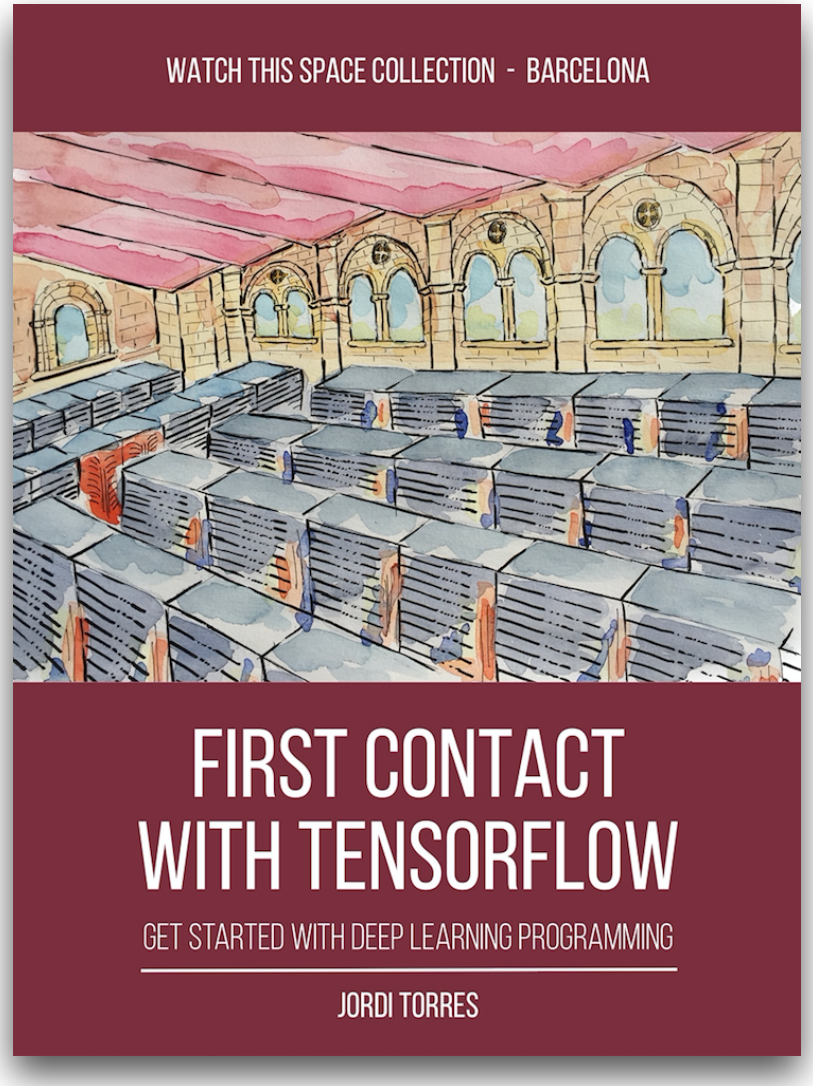
*Foreword:* ***Oriol Vinyals***
*Research Scientist en Google Brain*

# Learn by doing!

*"Tell me and I forget.*
*Teach me and I remember.*
*Involve me and I learn"*

- Benjamin Franklin

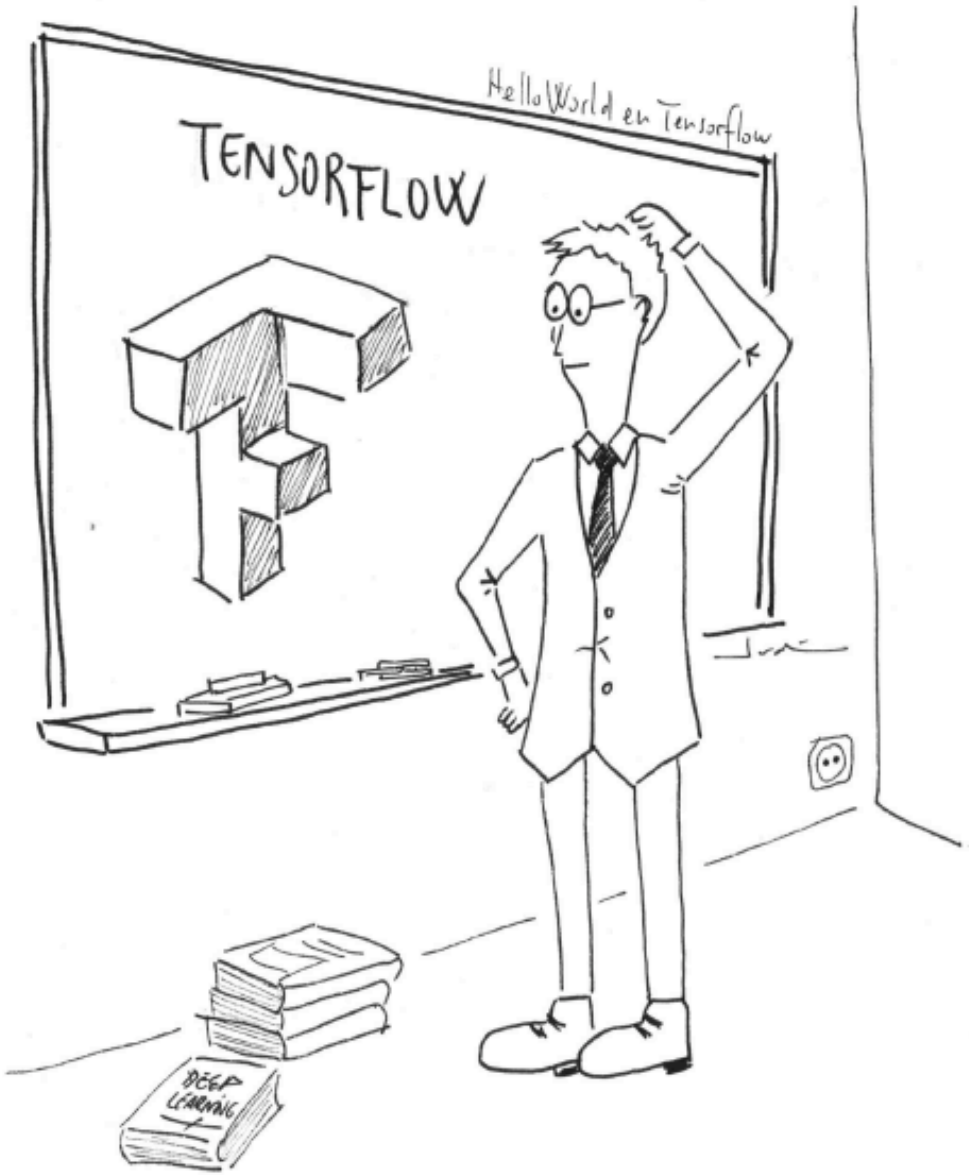
# Content:




Introduction

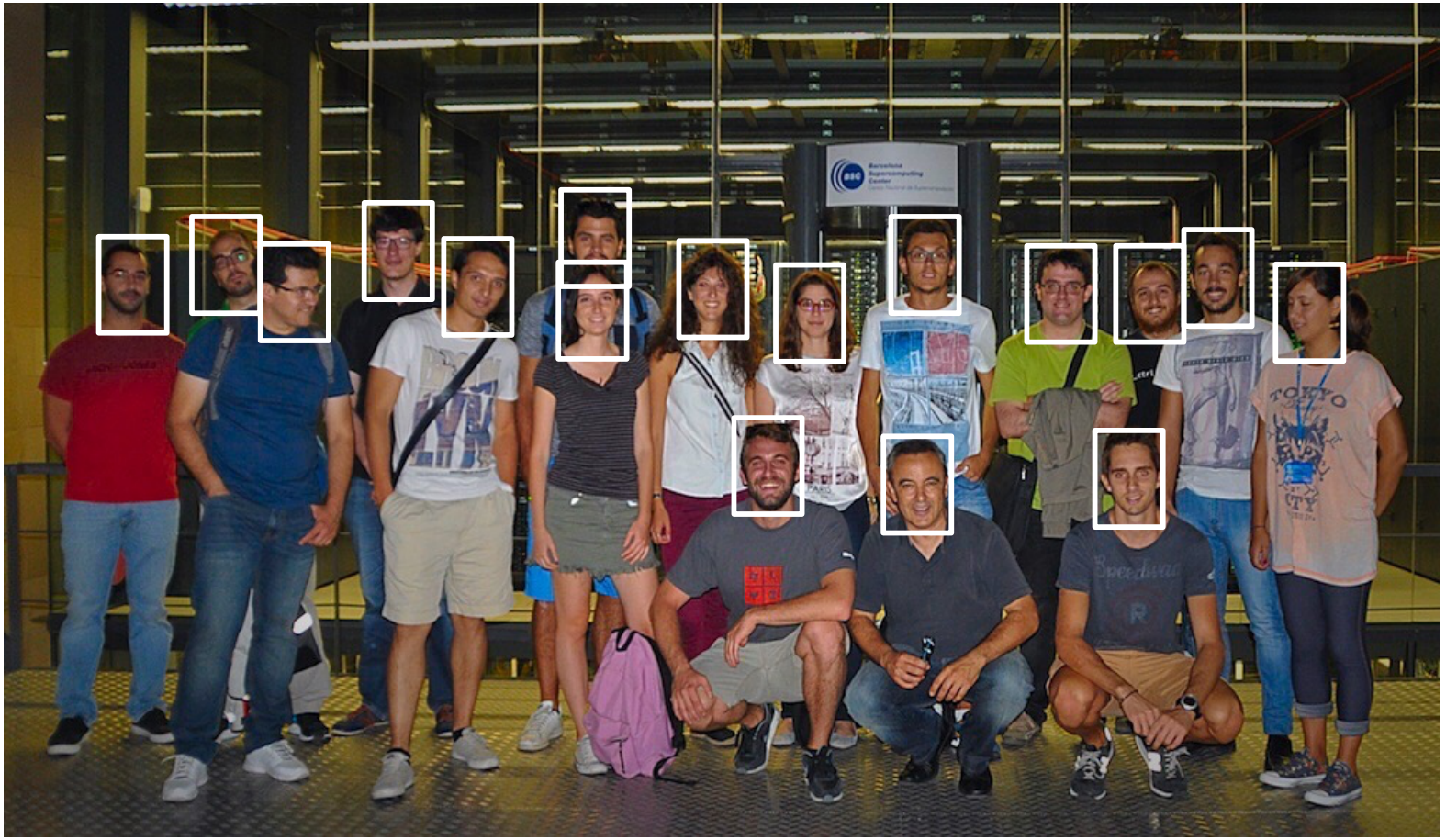
1. TensorFlow Basics
2. Linear Regression
3. Clustering
4. Single Layer Neural Network
5. Multi-layer Neural Networs
6. TensorFlow and GPUs

Closing


HelloWorld en Tensorflow
TENSORFLOW
DEEP LEARNING

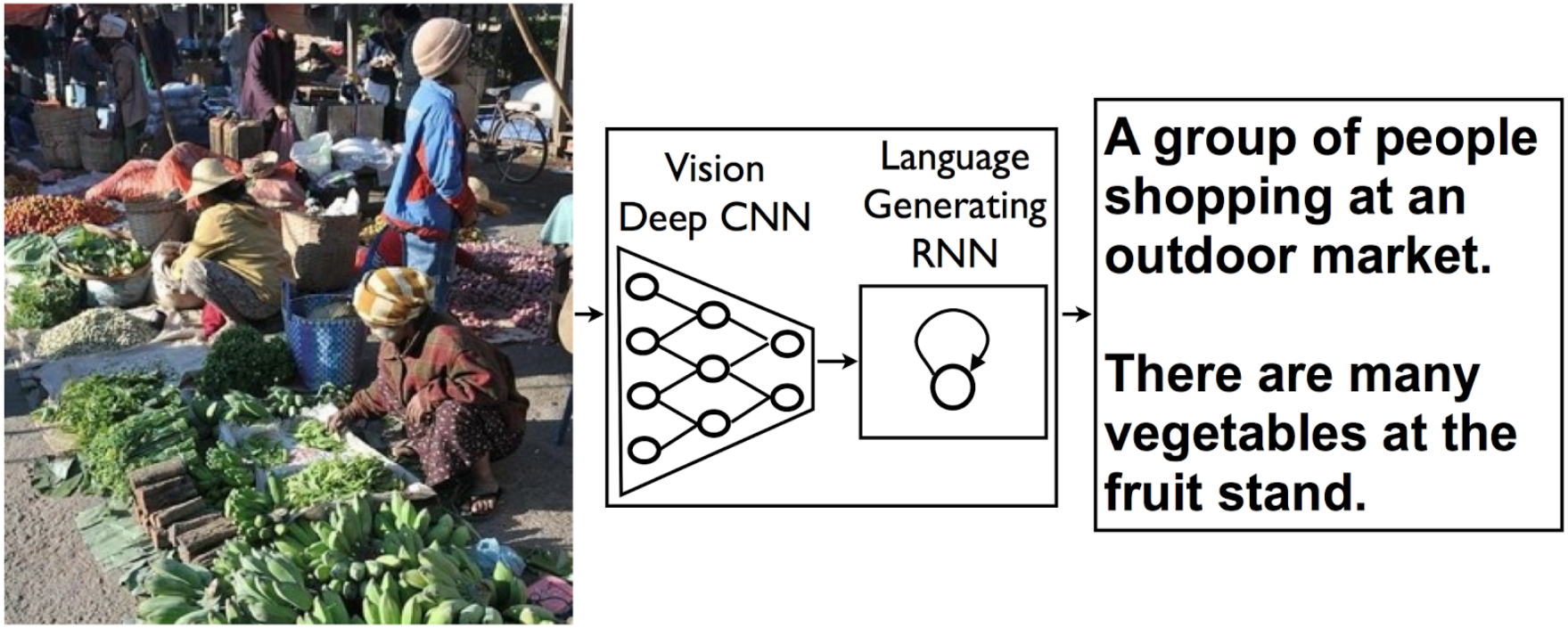
# A few years ago …

A breakthrough in ML suddenly enabled computers to recognize objects shown in photographs with unprecedented accuracy.



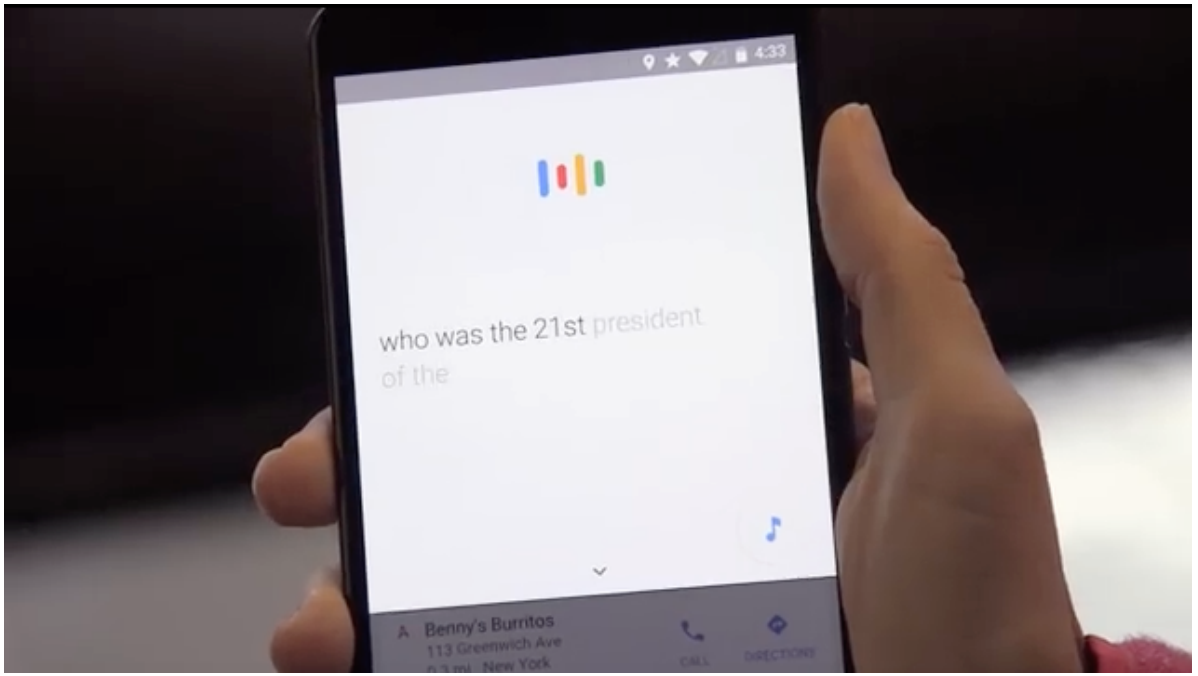
# The question now is …

Whether machines can make another leap, by learning to make sense of what's actually going on in such images.

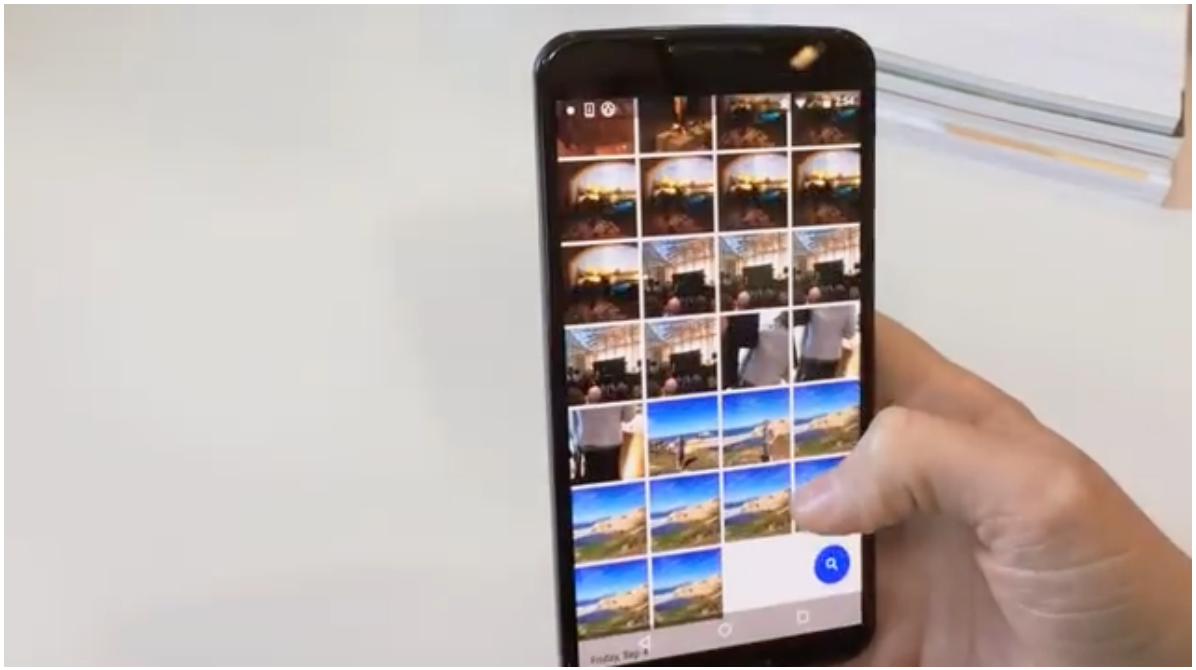



Source: Oriol Vinyals – *Research Scientist* at *Google Brain*

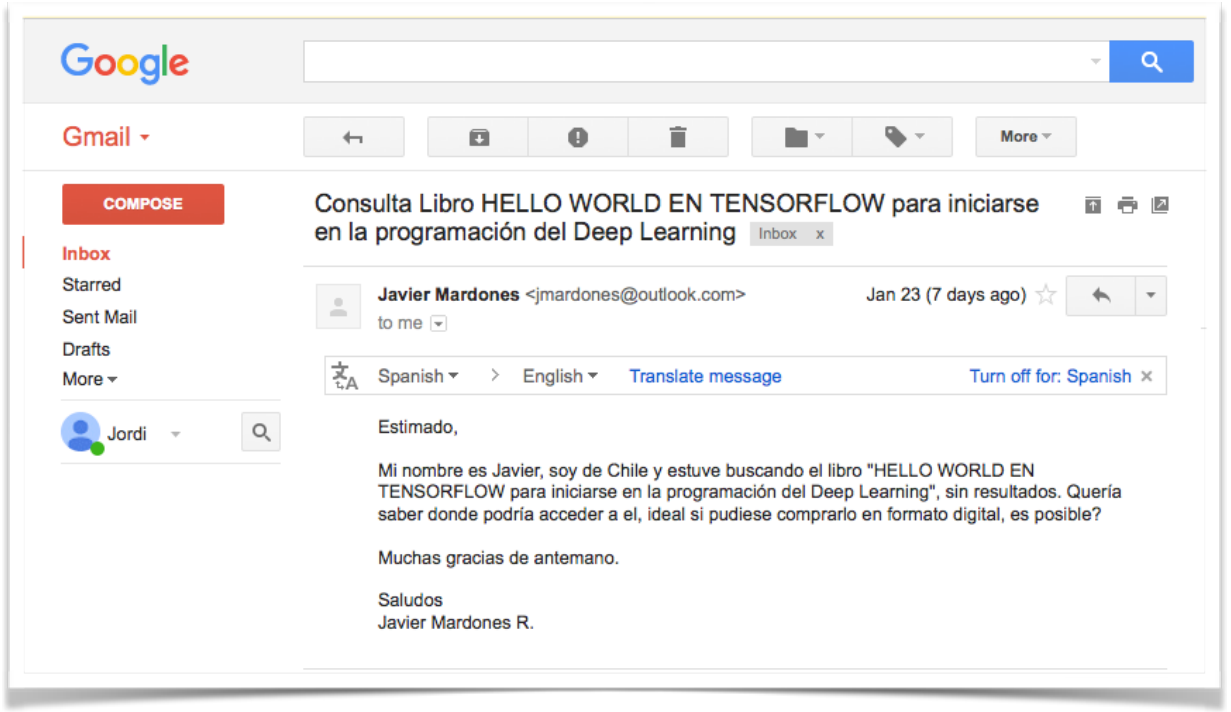
# TensorFlow at Google



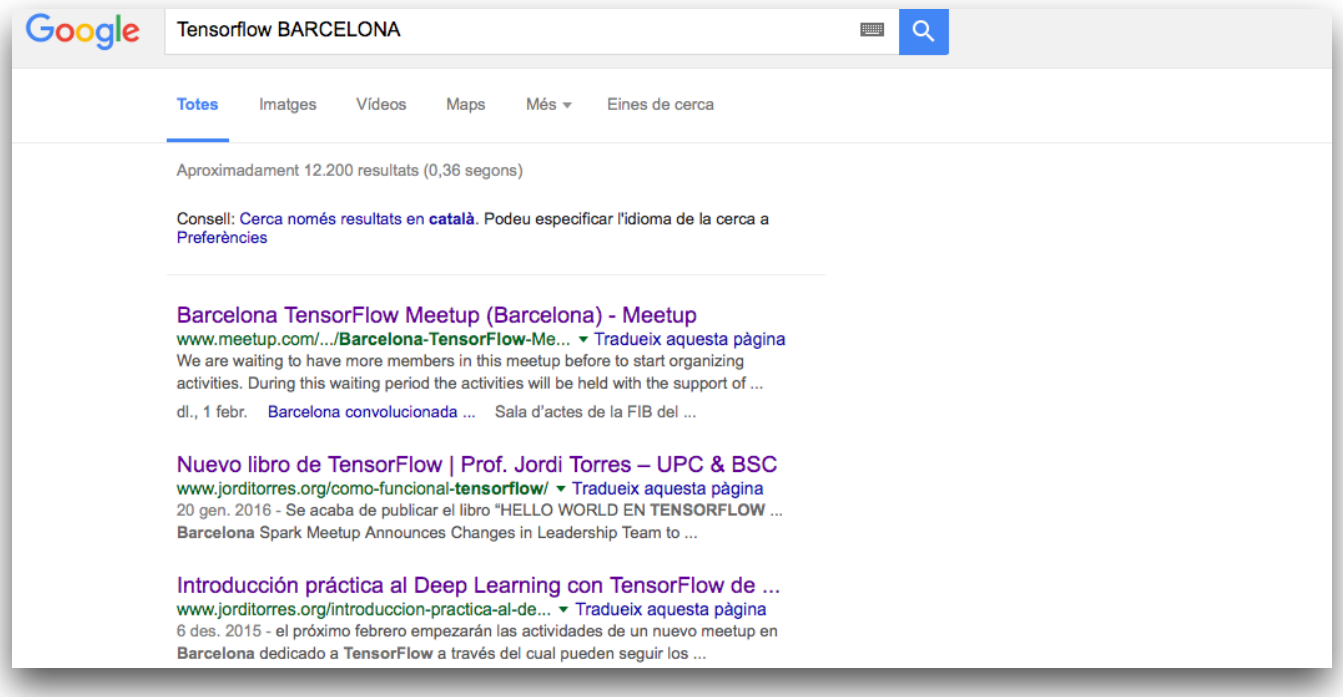
# TensorFlow at Google



# TensorFlow at Google



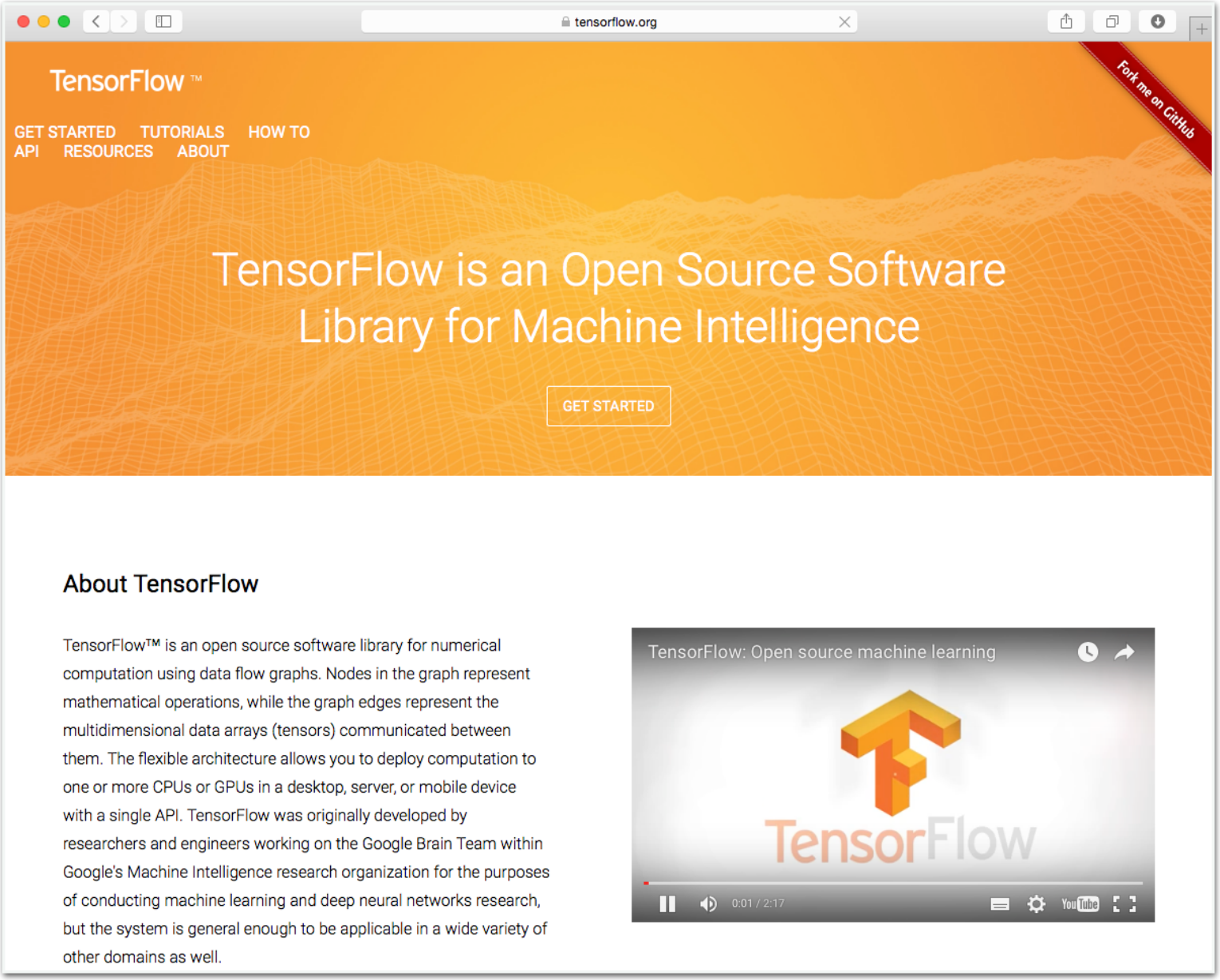
# TensorFlow at Google



# TensorFlow?



source: https://www.youtube.com/watch?v=CMdHDHEuOUE


tensorflow.org
TensorFlow ™
GET STARTED TUTORIALS HOW TO
API RESOURCES ABOUT
Fork me on GitHub
TensorFlow is an Open Source Software Library for Machine Intelligence
GET STARTED
About TensorFlow
TensorFlow™ is an open source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them. The flexible architecture allows you to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API. TensorFlow was originally developed by researchers and engineers working on the Google Brain Team within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well.
TensorFlow: Open source machine learning
TensorFlow
0:01 / 2:17

# About TensorFlow

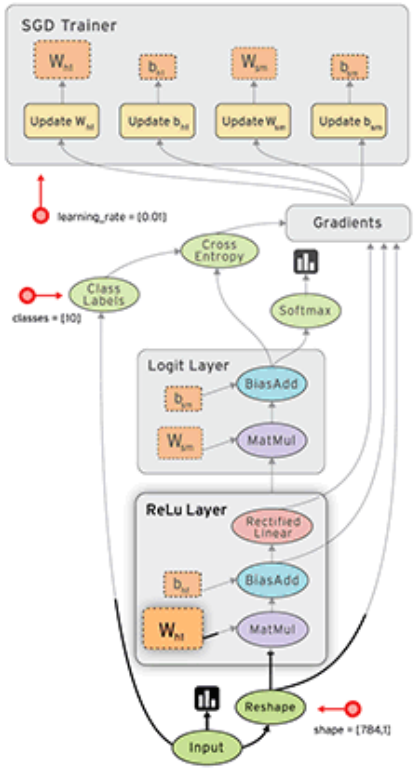
- TensorFlow was originally developed by the Google Brain Team within Google's Machine Intelligence research organization **for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well.**

- Open source software library for numerical computation using data flow graphs.

- The flexible architecture allows you to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API.

# Data Flow Graph?

- Describe mathematical computation with a directed graph of nodes & edges.
  - ✦ Nodes in the graph represent mathematical operations,
  - ✦ Edges describe the i/o relationships between nodes.
  - ✦ Data edges carry dynamically-sized multidimensional data arrays, or **tensors**.
- The flow of tensors through the graph is where TensorFlow gets its name. Nodes are assigned to computational devices and execute asynchronously and in parallel once all the tensors on their incoming edges becomes available.






31

# Before to start: Python basics

- Line indentation

  Python has no mandatory statement termination characters and blocks are specified by indentation
  Statements that expect an indentation level end in a colon (:).
  The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount.
  Python will advise you if there is a unclear line indentation with the following warning:

  ```
  IndentationError: unexpected indent
  ```

- Comments

  Start with the pound (#) sign and are single-line

# Before to start: Python basics

- Variables, operators and data types

```
>>> IntegerVar = 10
>>> IntegerVar += 10
>>> print IntegerVar
20
>>> StringVar = “Welcome”
>>> StringVar += ” to Barcelona”
>>> print StringVar
Welcome to Barcelona
```

# Before to start: Python basics

- Variables, operators and data types (cont.)

```
>>> IntegerVar, StringVar = StringVar,
IntegerVar
>>> print IntegerVar
Welcome to Barcelona
>>> print StringVar
20
>>> help (StringVar)
Help on int object:
class int(object)
 | int(x=0) -> int or long
 | int(x, base=10) -> int or long
 |
```

# Before to start: Python basics

- Data Structures available in Python are lists, tuples and dictionaries.

  Lists are like one-dimensional arrays and we can also have lists of other lists or tuples. There are number of methods for lists (methods follow the list name). Tuples are immutable one-dimensional array.

```
>>> sampleList = [1,2,3,4,5,6,7,8]
>>> print (sampleList[1])
2
>>> sampleTuple = (1,2,3,4,5,6,7,8)
>>> print (sampleTuple)
(1, 2, 3, 4, 5, 6, 7, 8)
>>> print sampleList
[1, 2, 3, 4, 5, 6, 7, 8]
>>> sampleList.append(9)
>>> print sampleList
[1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> sampleTuple.append(9)
Traceback (most recent call last):
  File “<stdin>”, line 1, in <module>
AttributeError: ‘tuple’ object has no attribute ‘append’
```

# Before to start: Python basics

- Data Structures available in Python are lists, tuples and dictionaries (cont.)

  Python dictionaries are associative arrays that has keys to obtain the elements of the dictionary. Dictionaries aren't exactly based on an index, there is no particular order in dictionaries (we could add a key: value and, it will appear in random places). A big caution here is that you cannot create different values with the same key (Python will just overwrite the value of the duplicate keys).

```
>>> myDicc = {'someItem': 2, 'otherItem': 20}
>>> print(myDicc['otherItem'])
20
```

# Before to start: Python basics

- Data Structures available in Python are lists, tuples and dictionaries (cont.)

  Python lists/dictionaries/tuples can be of any type, so you can mix them in. Variables can point to functions. The index of the first element is 0 and negative numbers count from the end towards the beginning ( -1 is the last element).

```
>>> Example = [1, “BCN”, [“another”, “list”], {“and”,”a”,”tuple”}]
>>> print Example
[1, ‘BCN’, [‘another’, ‘list’], set([‘a’, ‘and’, ‘tuple’])]
>>> myfunction = len
>>> print myfunction (Example)
4
>>> print Example[-1]
set([‘a’, ‘and’, ‘tuple’])
>>> print Example[3]
set([‘a’, ‘and’, ‘tuple’])
```

# Before to start: Python basics

- Flow control statements: **if**

  Often partnered with the if statement are else if and else.
  However, Python's else if is shortened into elif.
  After every conditional we have a colon.
  Next, we could proceed to a new line with number of spaces to tell Python we only want this code to be run when the previous conditional is satisfied.

```
>>> a = 20
>>> if a >= 22:
…     print(“Paris”)
… elif a >= 21:
…     print(“Londres”)
… else:
…     print(“Barcelona”)
…
Barcelona
>>>
```

# Before to start: Python basics

- Flow control statements: **for, while**

  Loops in Python are very versatile and simple.

```
>>> for a in range(1,4):
…         print (a)
…
1
2
3
>>> a = 1
>>> while a < 4:
…         print (a)
…         a+=1
…
1
2
3
>>> a = 20
```

# Before to start: Python basics

- Functions

  We can define a function by using the keyword def before your function name. Optional arguments are set in the function declaration after the mandatory arguments by being assigned a default value. Functions can return a tuple (and using tuple unpacking you can effectively return multiple values).

```
>>> def someFunction():
…         print(“Barcelona”)
…
>>> someFunction()
Barcelona
>>>
>>> def fib(n):    # write Fibonacci series up to n
…          a, b = 0, 1
…          while b < n:
…              print b,
…              a, b = b, a+b
…
>>> fib(1000)
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
```

# Before to start: Python basics

- Lambda Functions

  Python supports the creation of anonymous functions (i.e. functions that are not bound to a name) at runtime, using a construct called "lambda". This is not exactly the same as lambda in functional programming languages

```
>>> fibonacci = (lambda  x: 1 if  x <= 2 else
        fibonacci(x-1) + fibonacci(x-2))
>>> fibonacci(10)
55
>>> foo = [2, 18, 9, 22, 17, 24, 8, 12, 27]
>>> for  i in filter(lambda  x: x % 3 == 0, foo):
print (i)
18
9
24
12
27
```

# Before to start: Python basics

- Classes

  Python supports a limited form of multiple inheritance in classes.

```
>>> class Calculator(object):
…     #define class to simulate a calculator
…     def __init__ (self):
…         #start with zero
…         self.current = 0
…     def add(self, amount):
…         #add number to current
…         self.current += amount
…     def getCurrent(self):
…         return self.current
…
>>> myCalc = Calculator() # make myCalc into a Calculator object
>>> myCalc.add(2) #use myCalc's new add method derived from Calculator class
>>> print(myCalc.getCurrent())
2
>>> myCalc.add(2)
>>> print(myCalc.getCurrent())
```

*In the previous example the first part defines a Class. With def __init__ (self) it is created a new instance of this class. The second part shows how to use this class in Python.*

# Before to start: Python basics

- Importing

  External libraries are used with the `import [libname]` keyword. We can also use `from [libname] import [funcname]` for individual functions.

```
>>> from random import randint
>>> randomint = random.randint(1, 100)
>>> print randomint
84
>>> randomint = random.randint(1, 100)
>>> print randomint
44
```

# Before to start: Python basics

- Read/Write files

Python uses the following sintax for read/write files:

```
>>> f = open(“test.txt”,”w”) #opens file with name of “test.txt”
>>> f.write(“Barcelona, “)
>>> f.write(“is the best city of the world.”)
>>> f.write(“With an excellent weather.”)
>>> f.close()
>>>
>>> f = open(“test.txt”,”r”) #opens file with name of “test.txt”
>>> print(f.read())
Barcelona, is the best city of the world.With an excellent weather.
>>> f.close
```

# Before to start: Python basics

- Iterators

  Python define a object type for taking each item of something, one after another. Any time you use a loop, explicit or implicit, to go over a group of items, that is iteration
  An iterable is an object that has an __iter__ method which returns an iterator, or which defines a __getitem__ method that can take sequential indexes starting from zero.
  An iterator is an object with a next (Python 2) or __next__ (Python 3) method.

```
>>> vec = [1, 2, 3]
>>> it = iter (vec)
>>> next (it)
1
>>> next (it)
2
>>> next (it)
3
>>> type (vec)
<class  'list'>
>>> type (it)
<class  'list_iterator'>
```

  *In this example vec iterable and it es a iterador.*

# Before to start: Python basics

- More information if necessary?

  **https://docs.python.org**

# Let's start!

```
# Ubuntu/Linux 64-bit
$ sudo apt-get install python-pip python-dev python-virtualenv

# Mac OS X
$ sudo easy_install pip
$ sudo pip install --upgrade virtualenv
```

Available at: https://www.tensorflow.org/versions/master/get_started/os_setup.html#download-and-setup [Accessed: 16/12/2015].

# Virtualenv

```
$ virtualenv --system-site-packages ~/tensorflow

$ source ~/tensorflow/bin/activate # si se usa bash
$ source ~/tensorflow/bin/activate.csh # si se usa csh
(tensorflow)$
```

Available at: https://www.tensorflow.org/versions/master/get_started/os_setup.html#download-and-setup [Accessed: 16/12/2015].

# How to start?

```
# Ubuntu/Linux 64-bit, CPU only:
(tensorflow)$ pip install --upgrade https://storage.googleapis.com/
tensorflow/linux/cpu/tensorflow-0.5.0-cp27-none-linux_x86_64.whl



# Mac OS X, CPU only:
(tensorflow)$ pip install --upgrade https://storage.googleapis.com/
tensorflow/mac/tensorflow-0.5.0-py2-none-any.whl
```

Available at: https://www.tensorflow.org/versions/master/get_started/os_setup.html#download-and-setup [Accessed: 16/12/2015].

# Deactivate

```
(tensorflow)$ deactivate
```

Available at: https://www.tensorflow.org/versions/master/get_started/os_setup.html#download-and-setup [Accessed: 16/12/2015].

# My first code

```
import tensorflow as tf

a = tf.placeholder("float")
b = tf.placeholder("float")

y = tf.mul(a, b)

sess = tf.Session()

print sess.run(y, feed_dict={a: 3, b: 3})
```

Available: https://github.com/jorditorresBCN/TutorialTensorFlow

# Hands-on 1

1. Download the code from:

   https://github.com/jorditorresBCN/TutorialTensorFlow

2. Follow teacher's instructions

# math operations for manipulate the tensors

| Operation | Description |
|---|---|
| tf.add | sum |
| tf.sub | substraction |
| tf.mul | multiplication |
| tf.div | division |
| tf.mod | module |
| tf.abs | return the absolute value |
| tf.neg | return negative value |

# math operations for manipulate the tensors

| | |
|---|---|
| tf.sign | return the sign |
| tf.inv | returns the inverse |
| tf.square | calculates the square |
| tf.round | returns the nearest integer |
| tf.sqrt | calculates the square root |
| tf.pow | calculates the power |
| tf.exp | calculates the exponential |
| tf.log | calculates the logarithm |
| tf.maximum | returns the maximum |
| tf.minimum | returns the minimum |
| tf.cos | calculates the cosine |
| tf.sin | calculates the sine |

# math operations on matrices

| Operation | Description |
|---|---|
| tf.diag | returns a diagonal tensor with a given diagonal values |
| tf.transpose | returns the transposes of the argument |
| tf.matmul | returns a tensor product of multiplying two tensors listed as arguments |
| tf.matrix_determinant | returns the determinant of the square matrix specified as an argument |
| tf.matrix_inverse | returns the inverse of the square matrix specified as an argument |

# Operations & Kernels

- An operation has a name and represents an abstract computation (e.g., “matrix multiply”, or “add”). An operation can have attributes
- A kernel is a particular implementation of an operation that can be run on a particular type of device (e.g., CPU or GPU).
- A TensorFlow binary defines the sets of operations and kernels available

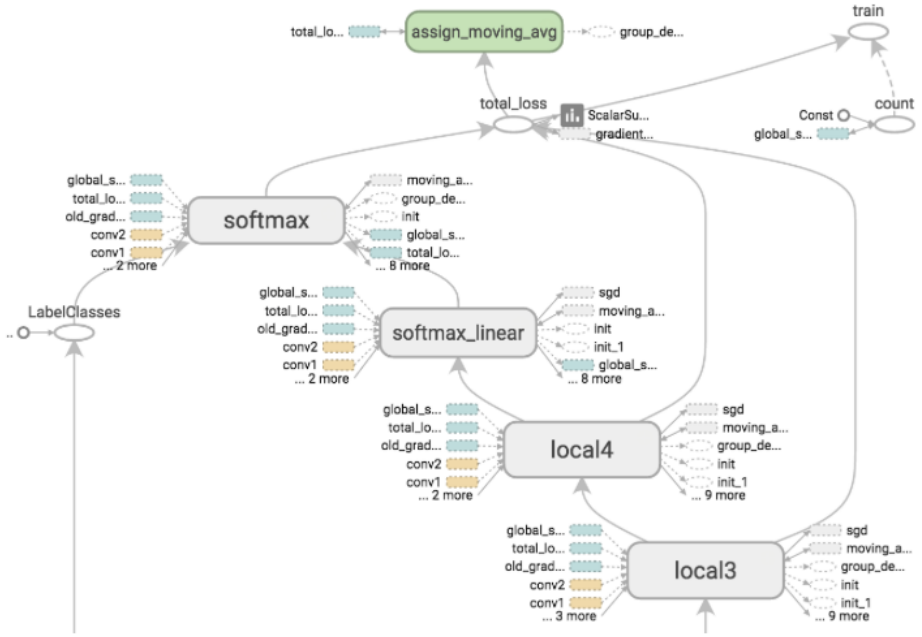
# kernels

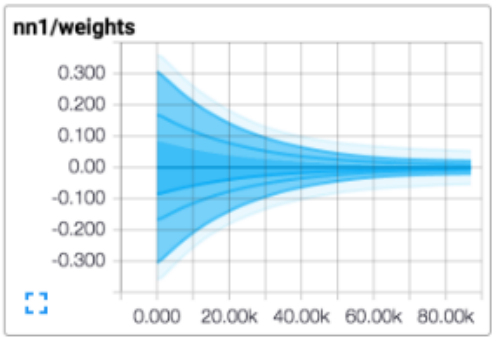
| Operations groups | Operations |
|---|---|
| Maths | Add, Sub, Mul, Div, Exp, Log, Greater, Less, Equal |
| Array | Concat, Slice, Split, Constant, Rank, Shape, Shuffle |
| Matrix | MatMul, MatrixInverse, MatrixDeterminant |
| Neuronal Network | SoftMax, Sigmoid, ReLU, Convolution2D, MaxPool |
| Checkpointing | Save, Restore |
| Queues and sincronizations | Enqueue, Dequeue, MutexAcquire, MutexRelease |
| Flow control | Merge, Switch, Enter, Leave, NextIteration |

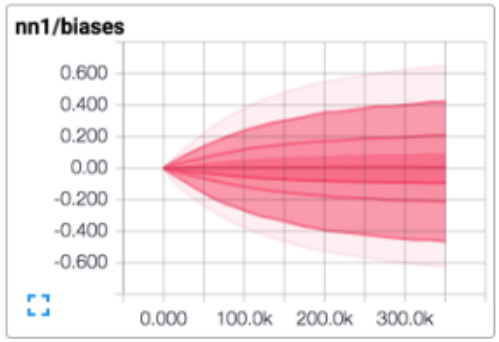
TensorFlow: Large-scale machine learning on heterogeneous systems, (2015). [Online]. Available at: http://download.tensorflow.org/paper/whitepaper2015.pdf [Accessed: 20/12/2015].

# TensorBoard: Visualization of graph structures and summary statistics

- In order to help users understand the structure of their computation graphs and also to understand the overall behavior of machine learning models.



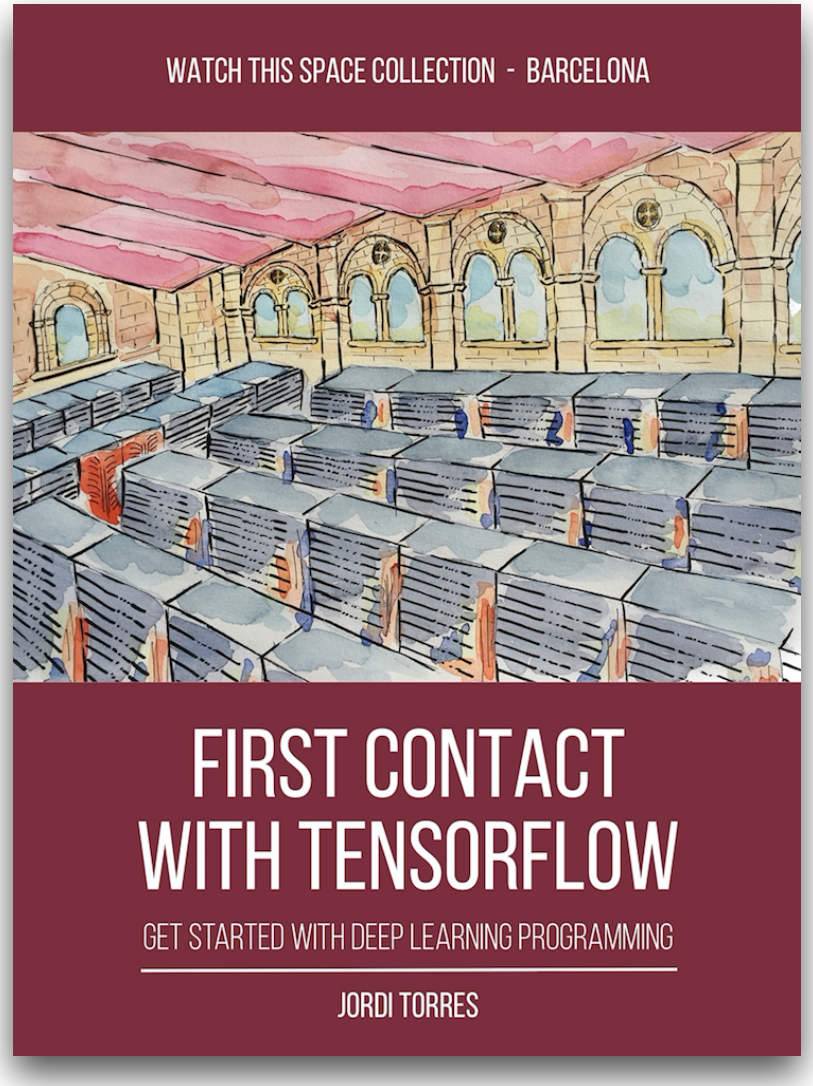
# Tensorboard

- TensorFlow included functions to debug and optimize programs in a visualization tool called TensorBoard.
- The data displayed with TensorBoard module are generated during the execution of TensorFlow and stored in trace files whose data are obtained from the summary operations
- The way we can invoke it is very simple:  http://localhost:6006/

**(out of scope of this presentation)**

- More information: TensorFlow, (2016) TensorBoard: Graph Visualization. [Online]. Available at: https://www.tensorflow.org/versions/master/how_tos/ graph_viz/index.html [Accessed: 02/01/2016].

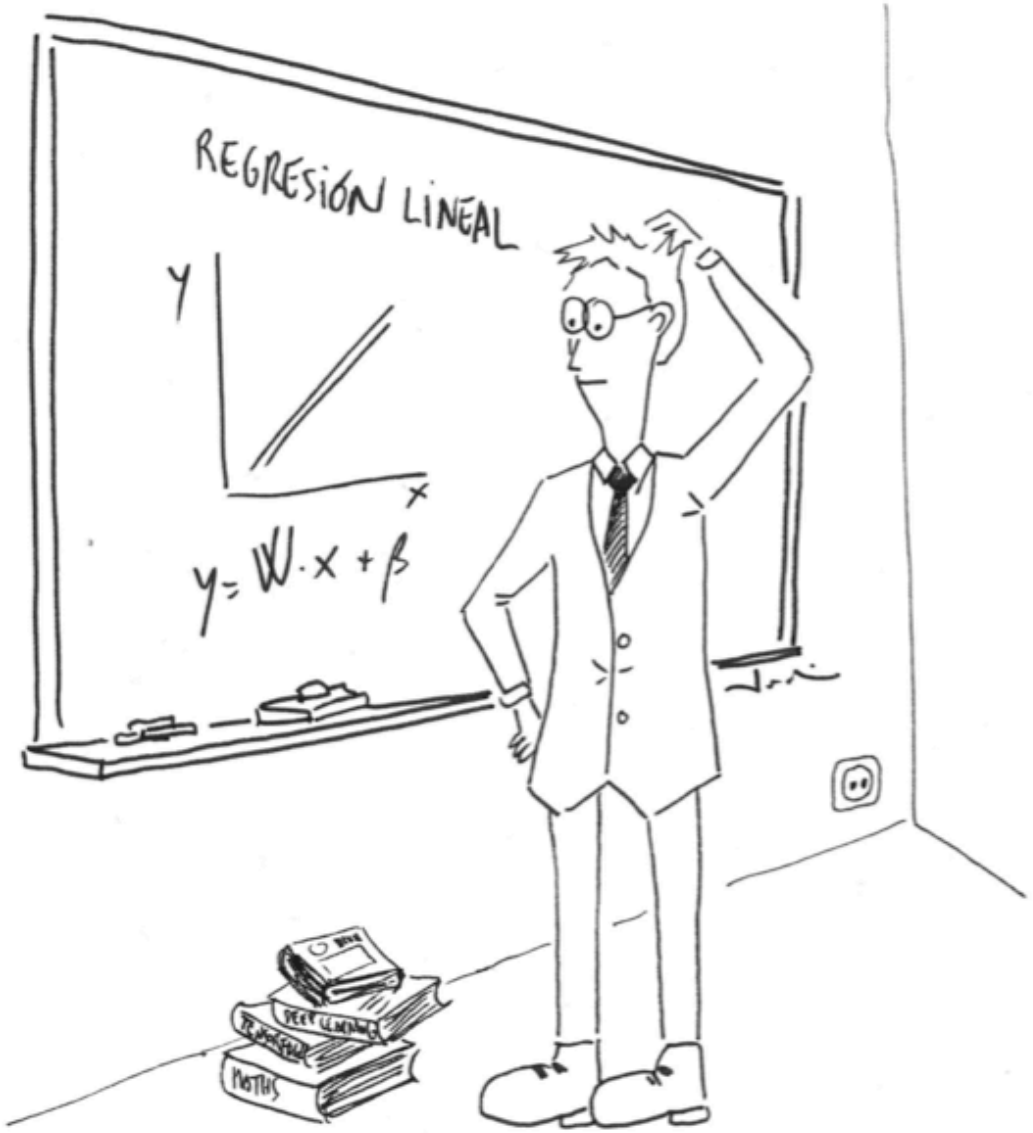
# Content:




Introduction

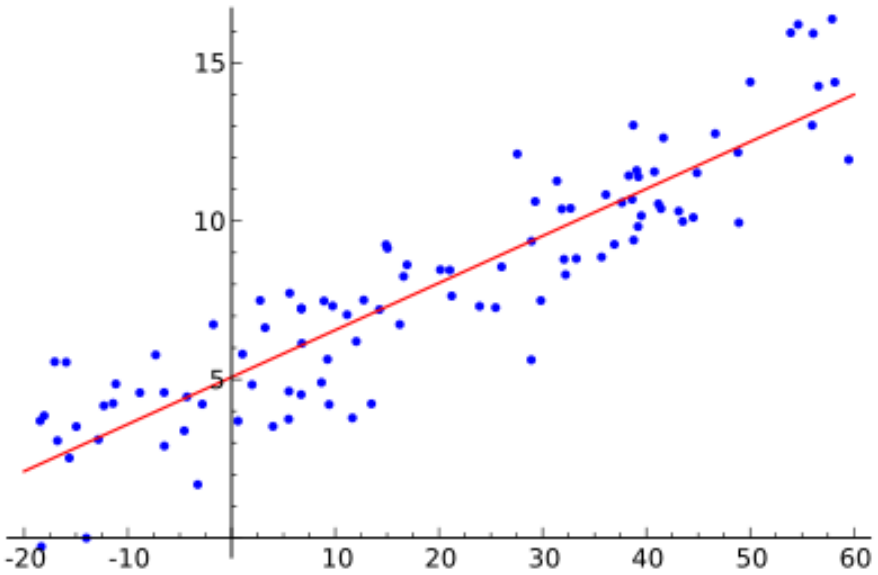
1. TensorFlow Basics
2. Linear Regression
3. Clustering
4. Single Layer Neural Network
5. Multi-layer Neural Networs
6. TensorFlow and GPUs

Closing


REGRESION LINEAL
Y
X
y = W.x + β
MATHS

# Linear Regression

- Wikipedia: "In statistics, **linear regression** is an approach for modeling the relationship between a scalar dependent variable *y* and one or more explanatory variables (or independent variables) denoted *X*. "



# Linear Regression

- Linear regression is a statistical technique used to measure the relationship between variables.
- a simple linear regression can be y = W * x + b.
- Case Study:  y = 0.1 * x + 0.3

# Case study

For this, we use a simple Python program that creates data in two-dimensional space

```
import numpy as np

num_puntos = 1000
conjunto_puntos = []
for i in xrange(num_puntos):
    x1= np.random.normal(0.0, 0.55)
    y1= x1 * 0.1 + 0.3 + np.random.normal(0.0, 0.03)
    conjunto_puntos.append([x1, y1])

x_data = [v[0] for v in conjunto_puntos]
y_data = [v[1] for v in conjunto_puntos]
```

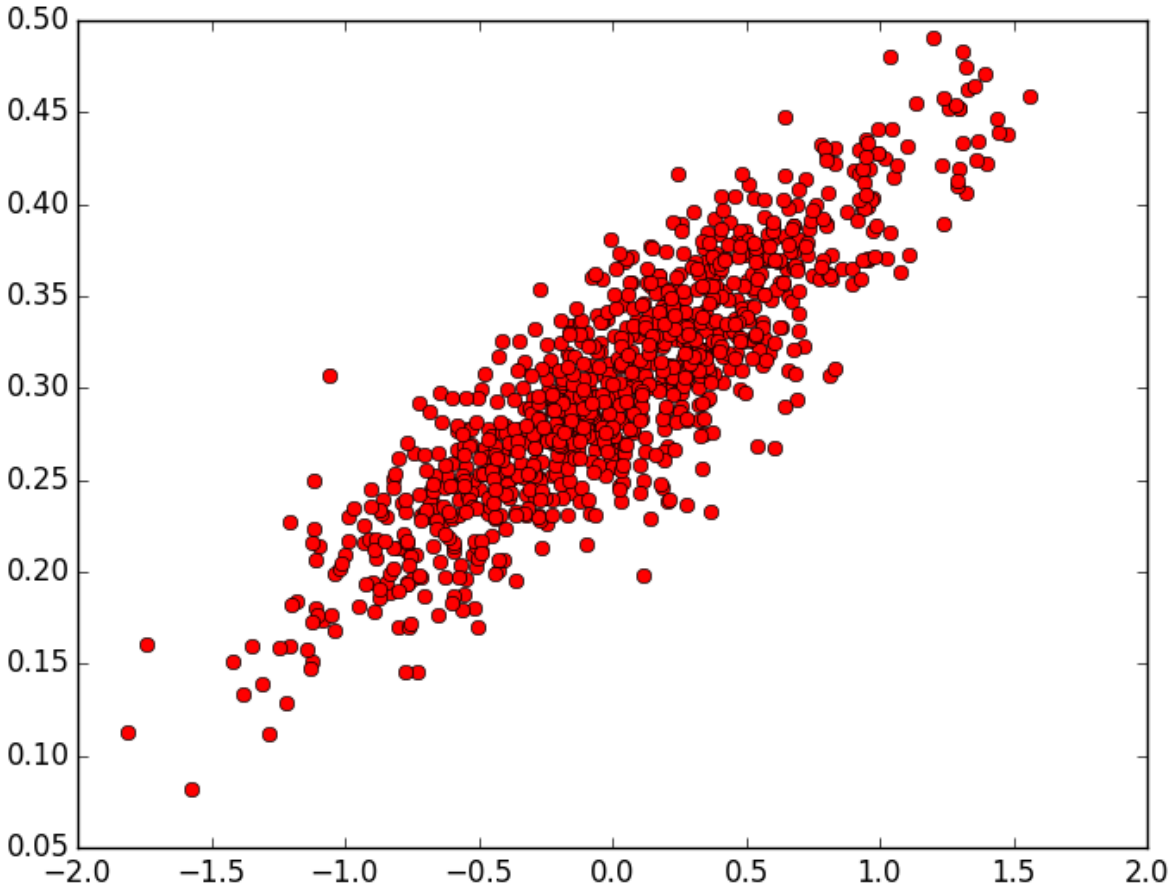
Available: https://github.com/jorditorresBCN/TutorialTensorFlow

# Case study

```
import matplotlib.pyplot as plt

plt.plot(x_data, y_data, 'ro', label='Original data')
plt.legend()
plt.show()
```

Available: https://github.com/jorditorresBCN/TutorialTensorFlow

# Case study



# We will ask TensorFlow looking for the line that best fits these points

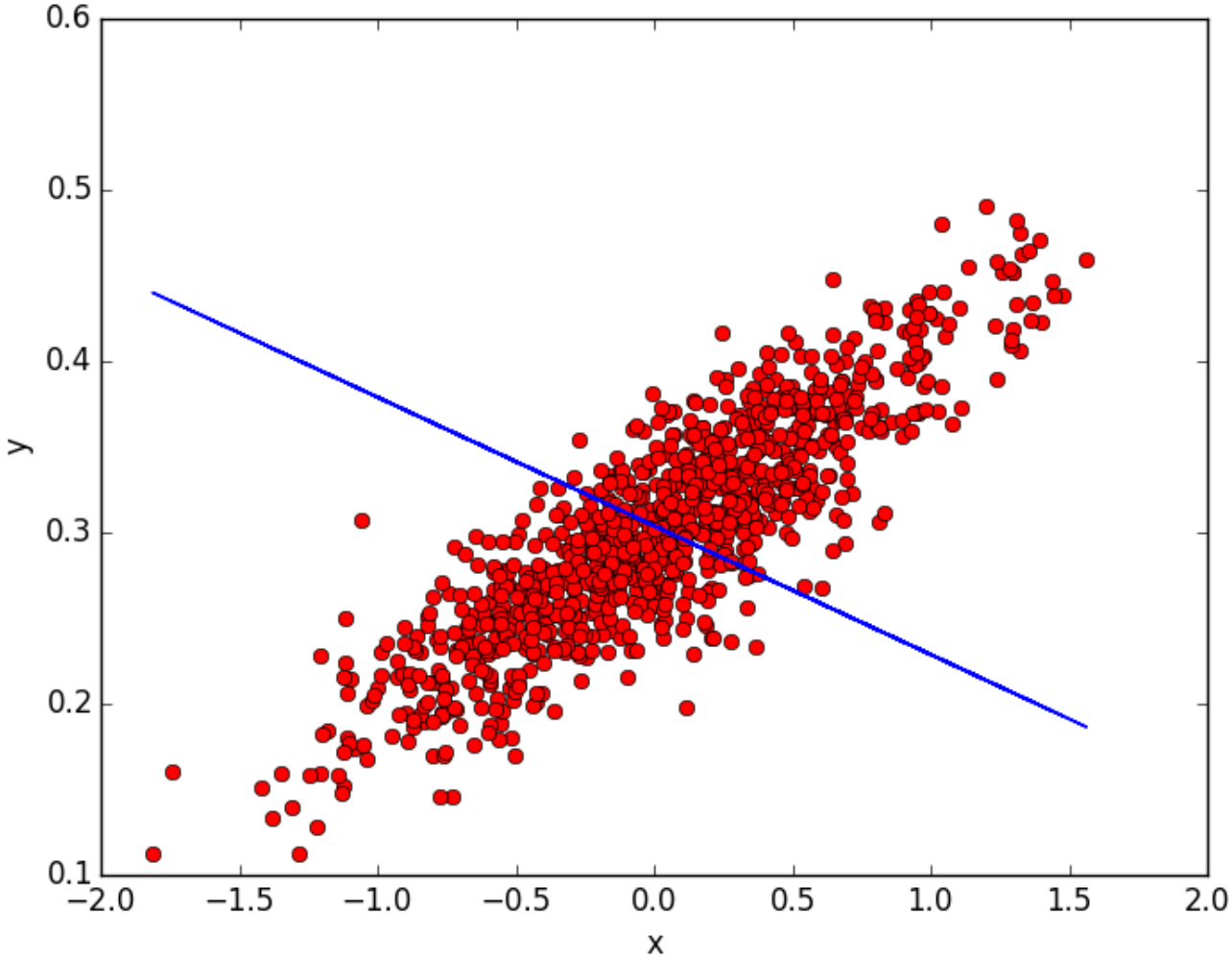
```
import tensorflow as tf

W = tf.Variable(tf.random_uniform([1], -1.0, 1.0))
b = tf.Variable(tf.zeros([1]))
y = W * x_data + b

loss = tf.reduce_mean(tf.square(y - y_data))
optimizer = tf.train.GradientDescentOptimizer(0.5)
train = optimizer.minimize(loss)

init = tf.initialize_all_variables()

sess = tf.Session()
sess.run(init)

for step in xrange(16):
    sess.run(train)
```

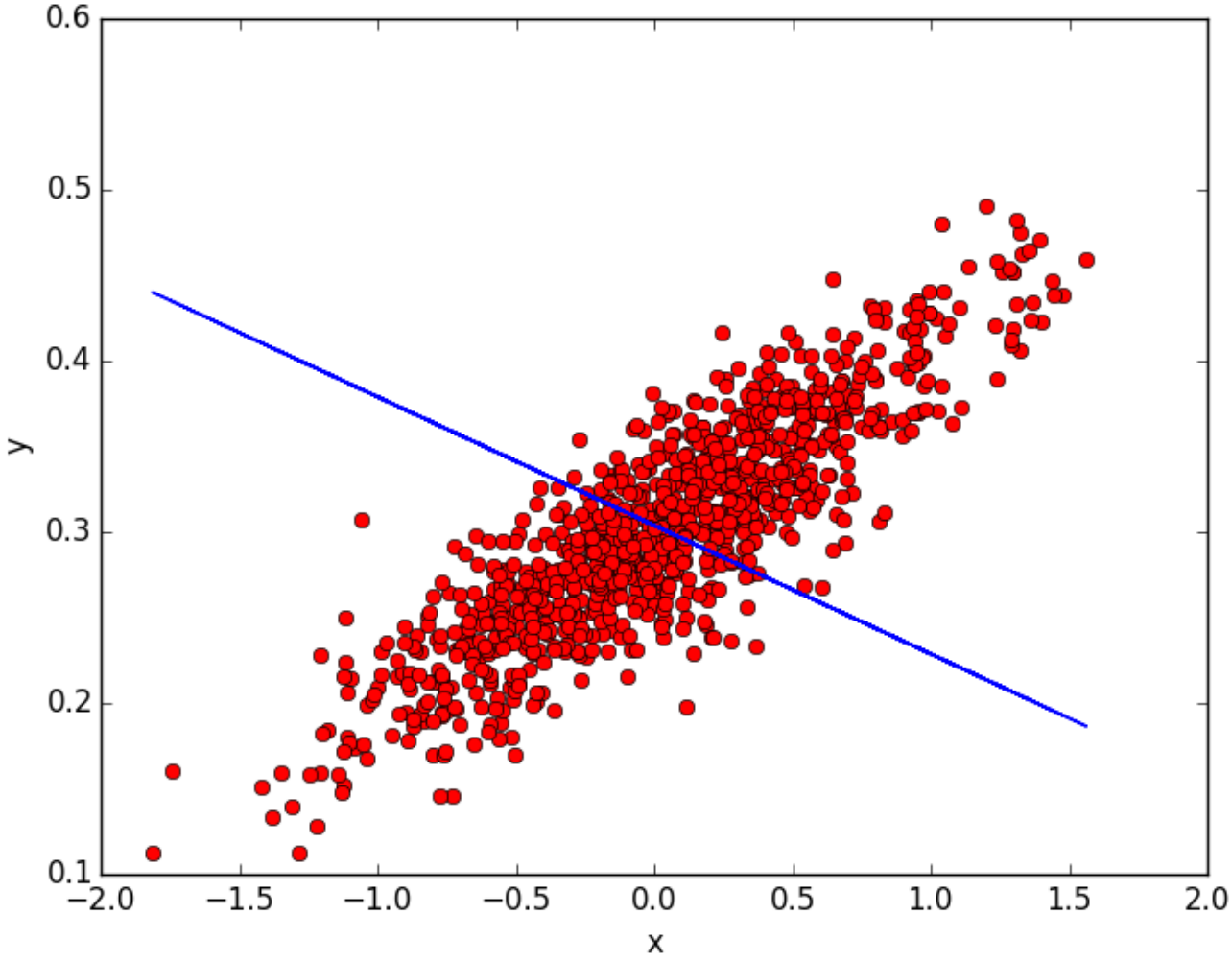
Available: https://github.com/jorditorresBCN/TutorialTensorFlow

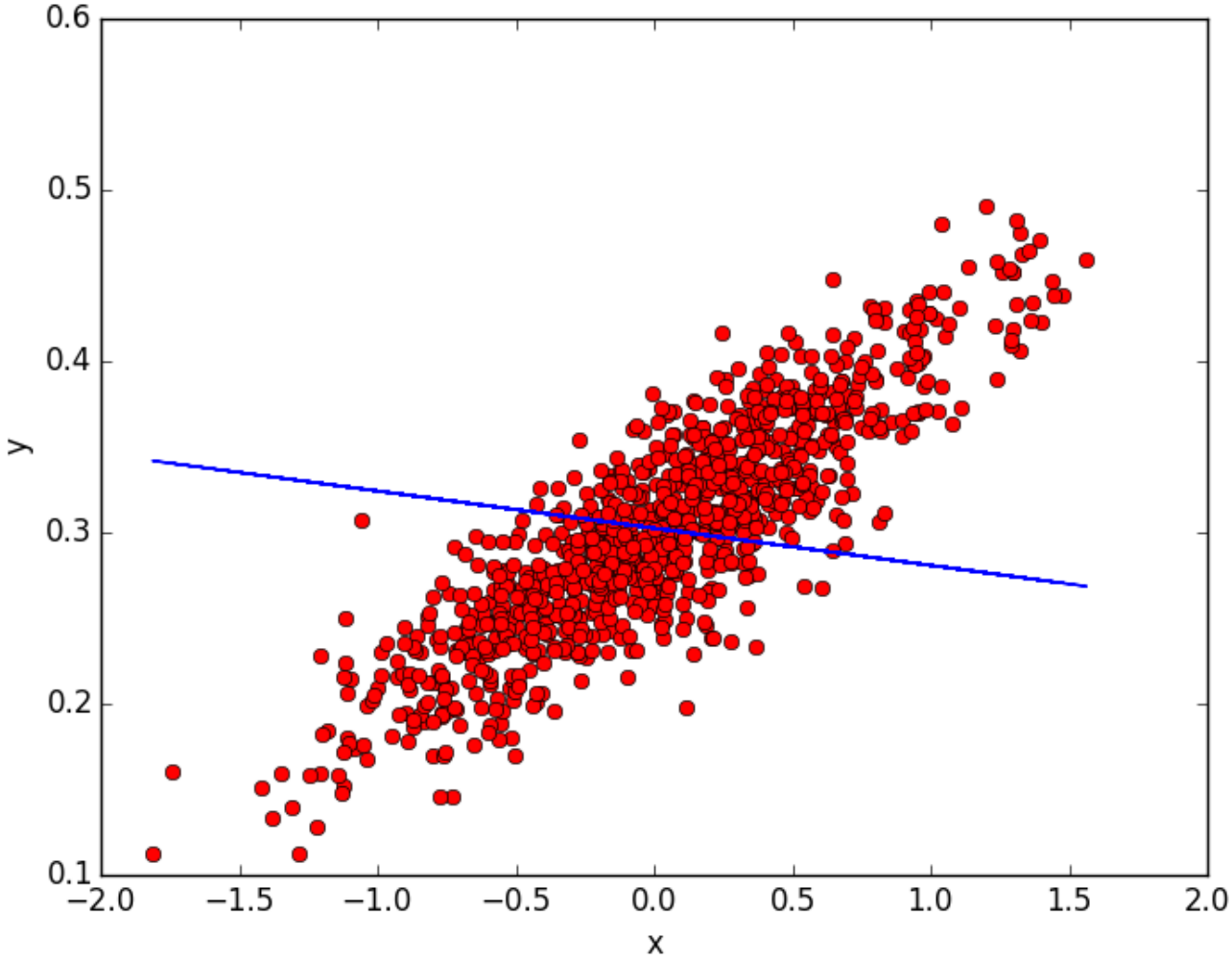
# Case Study: Running the Algorithm

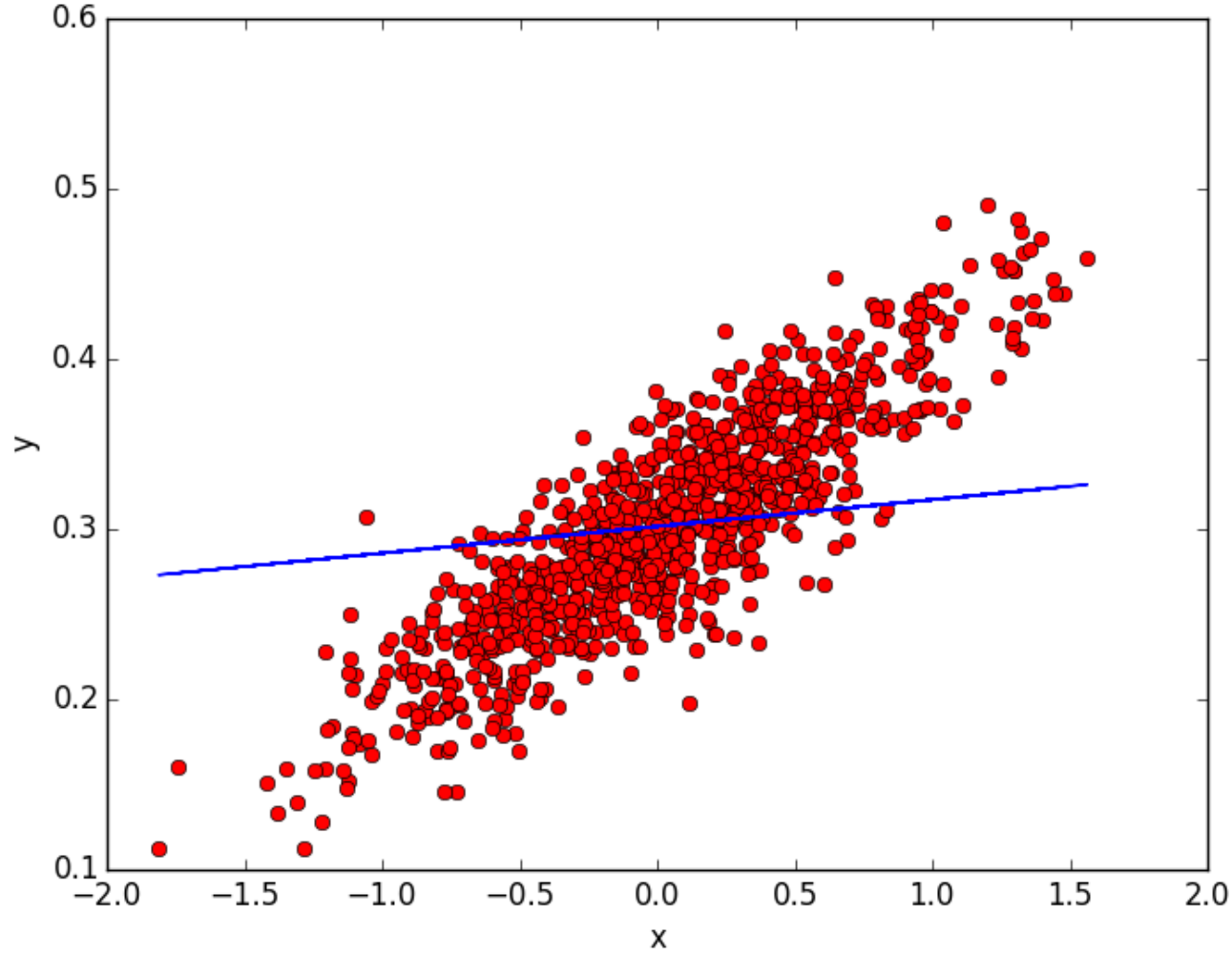
```
for step in xrange(16):
  sess.run(train)

  plt.plot(x_data, y_data, 'ro')
  plt.plot(x_data, sess.run(W) * x_data + sess.run(b))
  plt.xlabel('x')
  plt.xlim(-2,2)
  plt.ylim(0.1,0.6)
  plt.ylabel('y')
  plt.legend()
  plt.show()
```

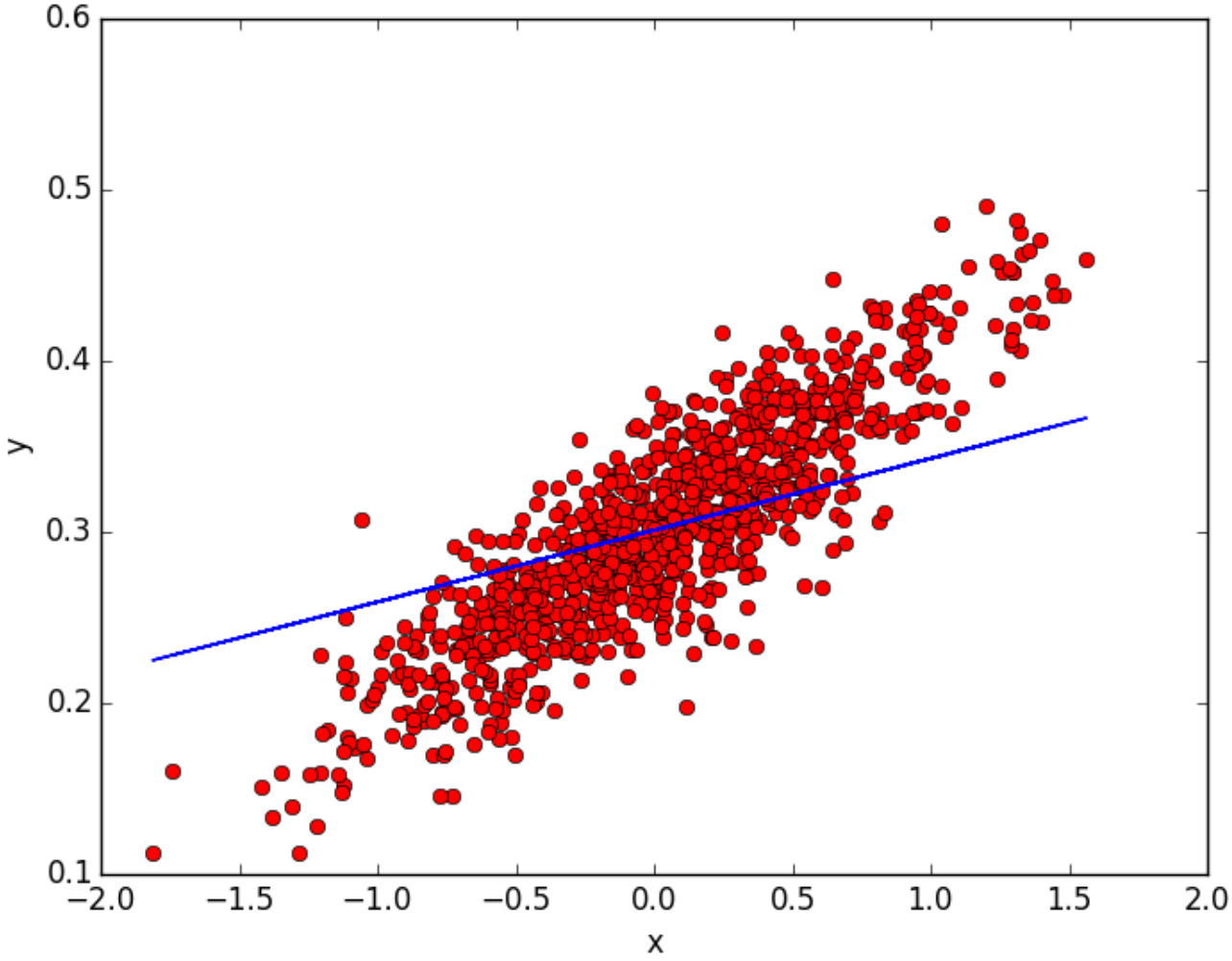
Disponible en: https://github.com/jorditorresBCN/TutorialTensorFlow

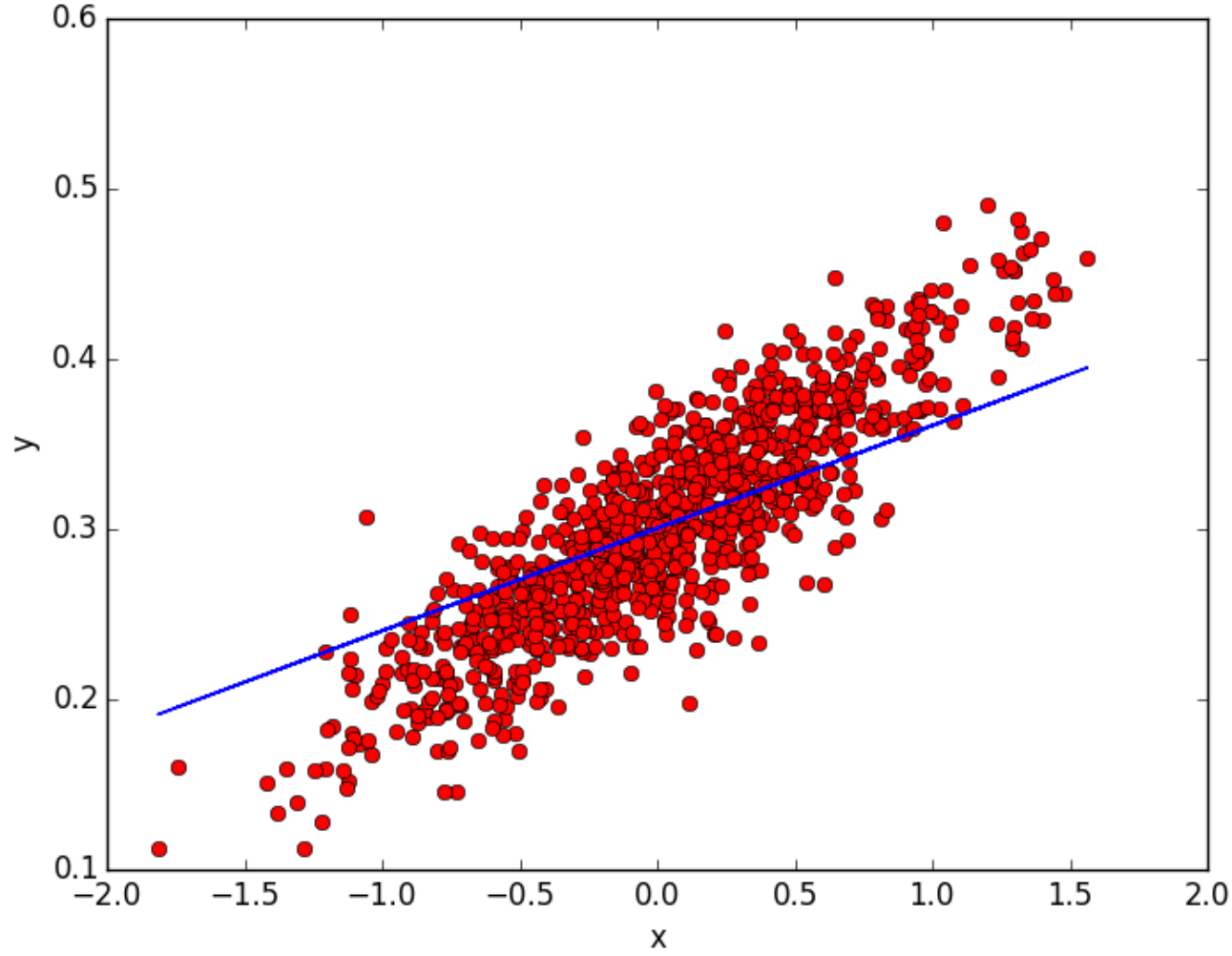

0.6
0.5
0.4
0.3
0.2
0.1
y
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
x

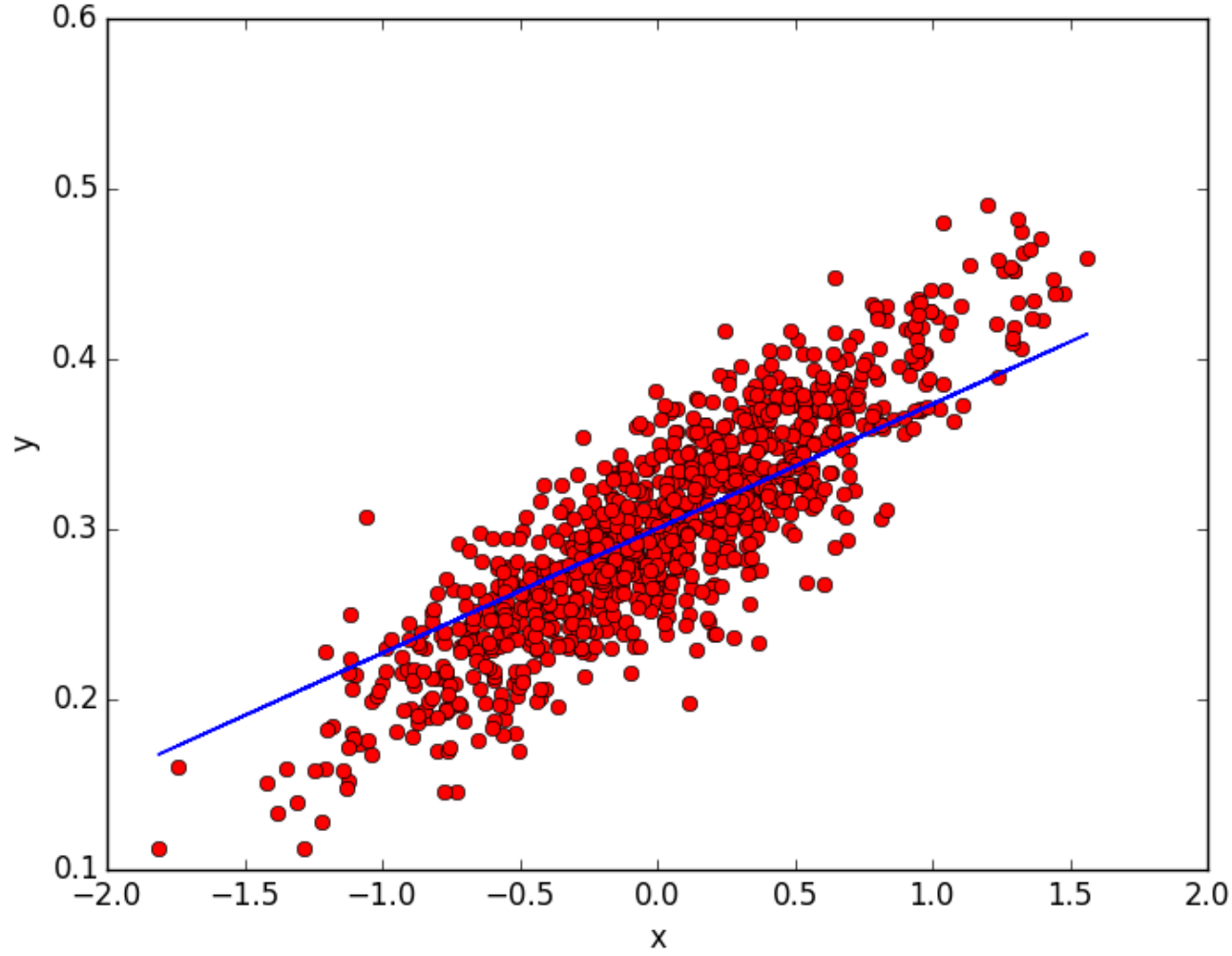

0.6
0.5
0.4
0.3
0.2
0.1
y
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
x

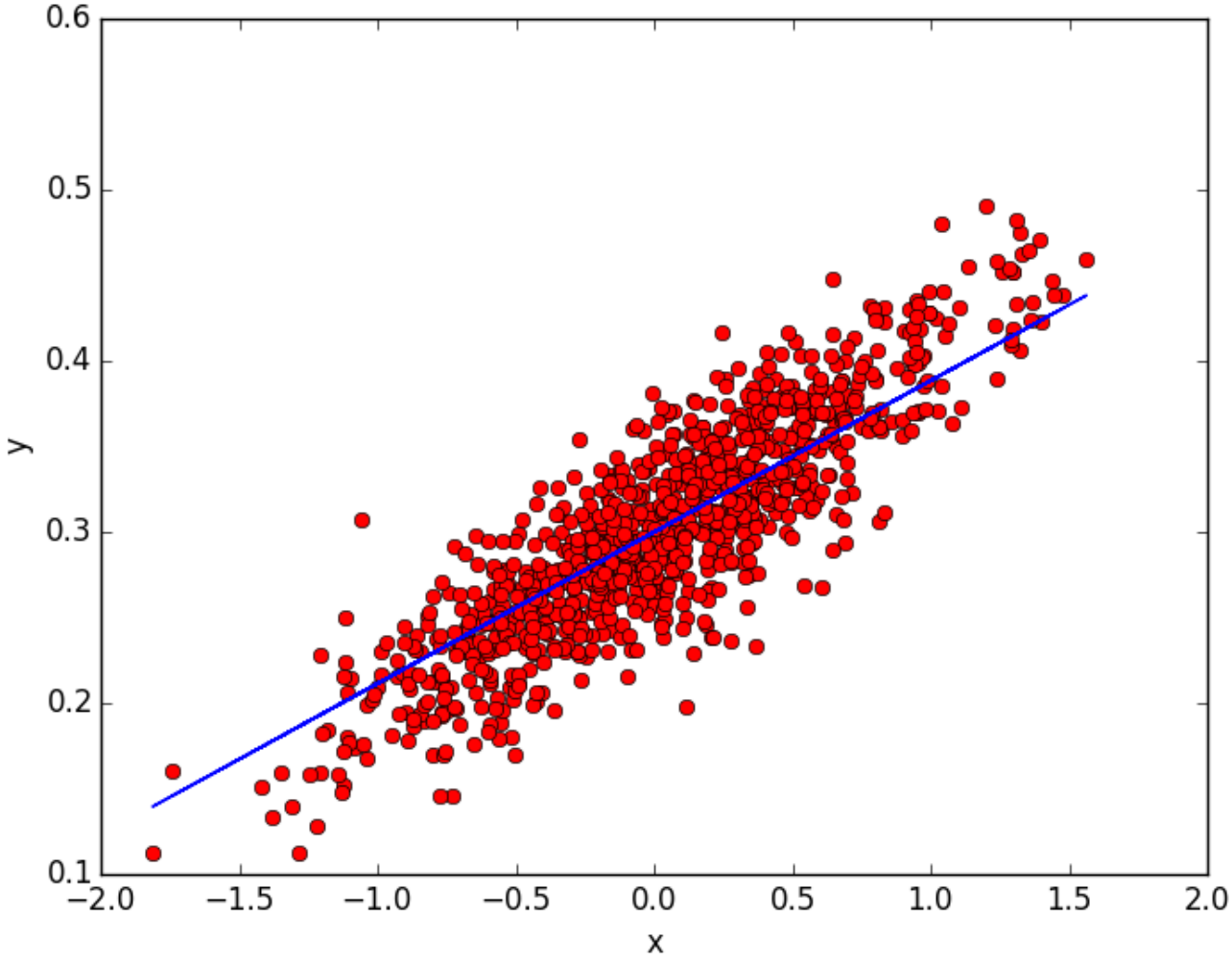

0.6
0.5
0.4
0.3
0.2
0.1
y
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
x

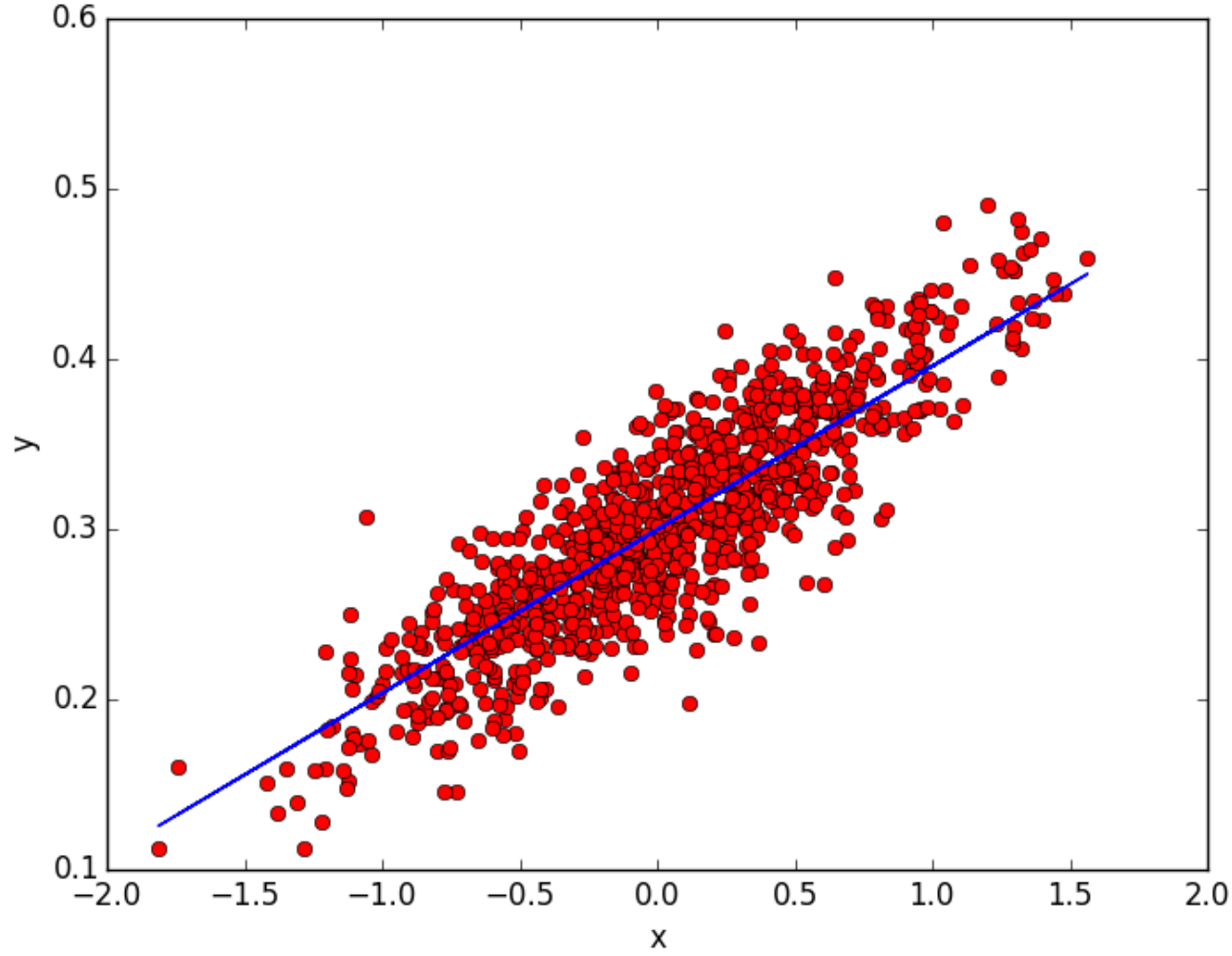

0.6
0.5
0.4
0.3
0.2
0.1
y
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
x

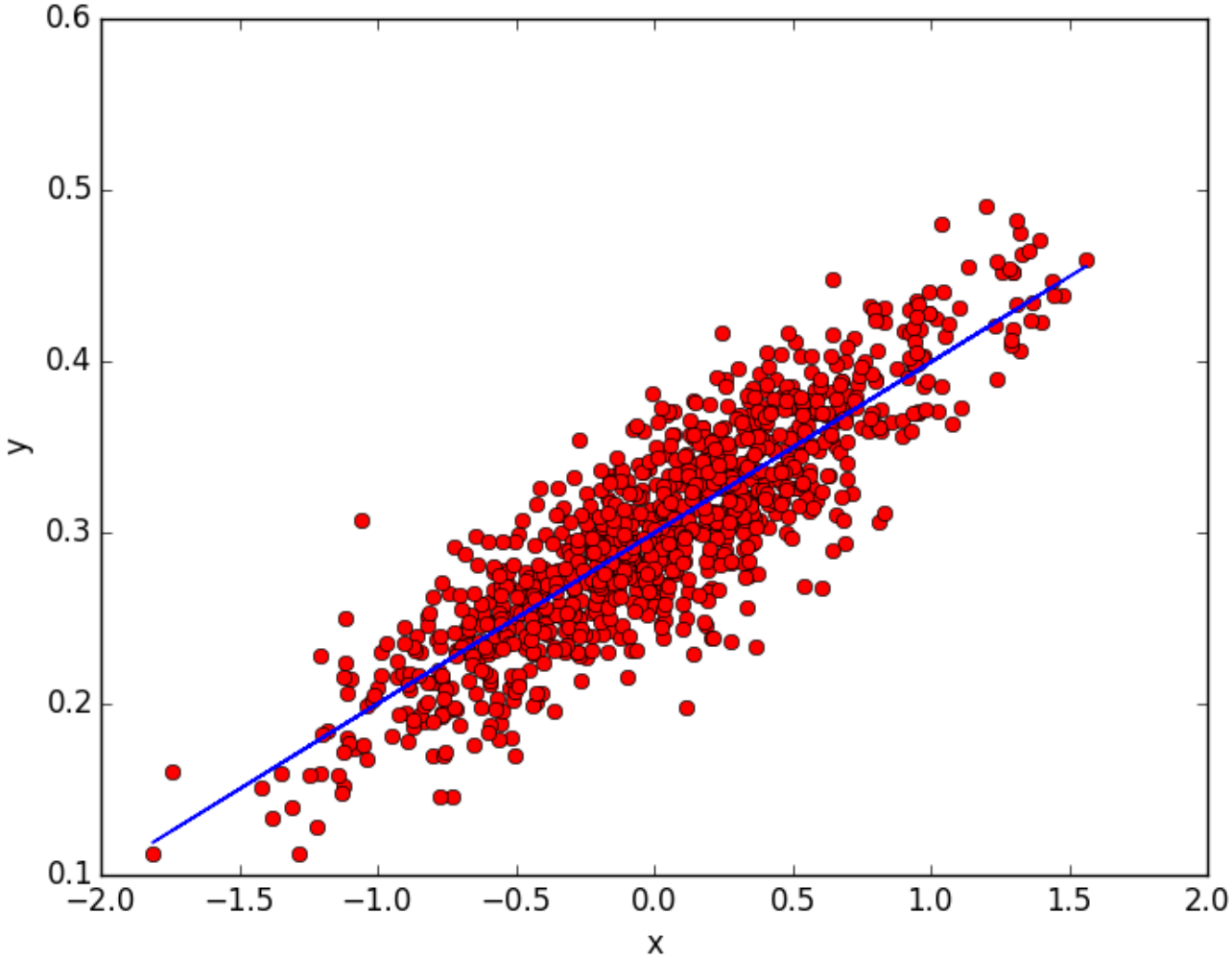

0.6
0.5
0.4
0.3
0.2
0.1
y
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
x

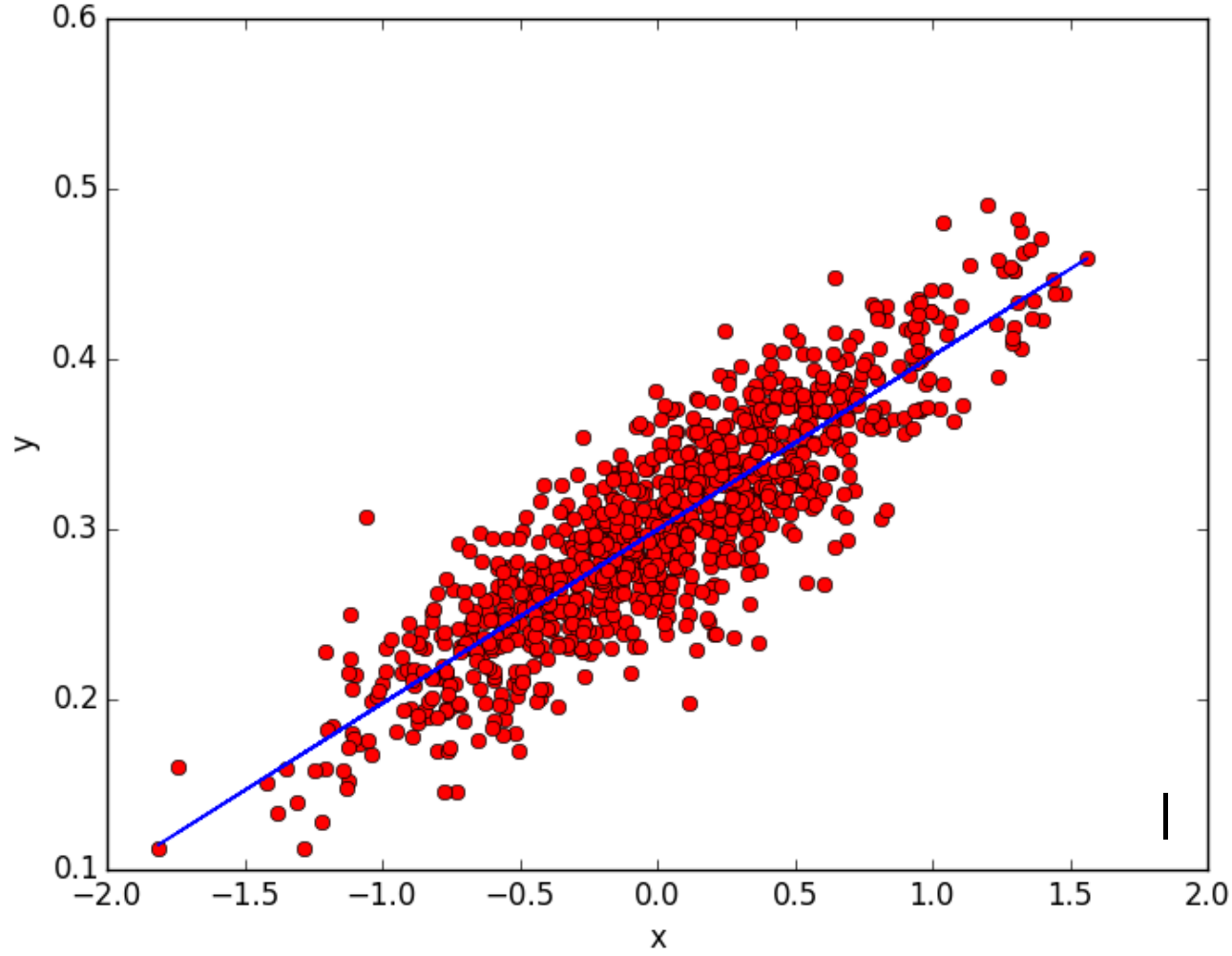

0.6
0.5
0.4
0.3
0.2
0.1
y
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
x


0.6
0.5
0.4
0.3
0.2
0.1
y
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
x


0.6
0.5
0.4
0.3
0.2
0.1
y
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
x


0.6
0.5
0.4
0.3
0.2
0.1
y
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
x


0.6
0.5
0.4
0.3
0.2
0.1
y
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
x


0.6
0.5
0.4
0.3
0.2
0.1
y
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
x

# print(step, sess.run(W), sess.run(b))

(0, array([-0.04841119], dtype=float32), array([ 0.29720169], dtype=float32))

(1, array([-0.00449257], dtype=float32), array([ 0.29804006], dtype=float32))

(2, array([ 0.02618564], dtype=float32), array([ 0.29869056], dtype=float32))

(3, array([ 0.04761609], dtype=float32), array([ 0.29914495], dtype=float32))

(4, array([ 0.06258646], dtype=float32), array([ 0.29946238], dtype=float32))

(5, array([ 0.07304412], dtype=float32), array([ 0.29968411], dtype=float32))

(6, array([ 0.08034936], dtype=float32), array([ 0.29983902], dtype=float32))

(7, array([ 0.08545248], dtype=float32), array([ 0.29994723], dtype=float32))

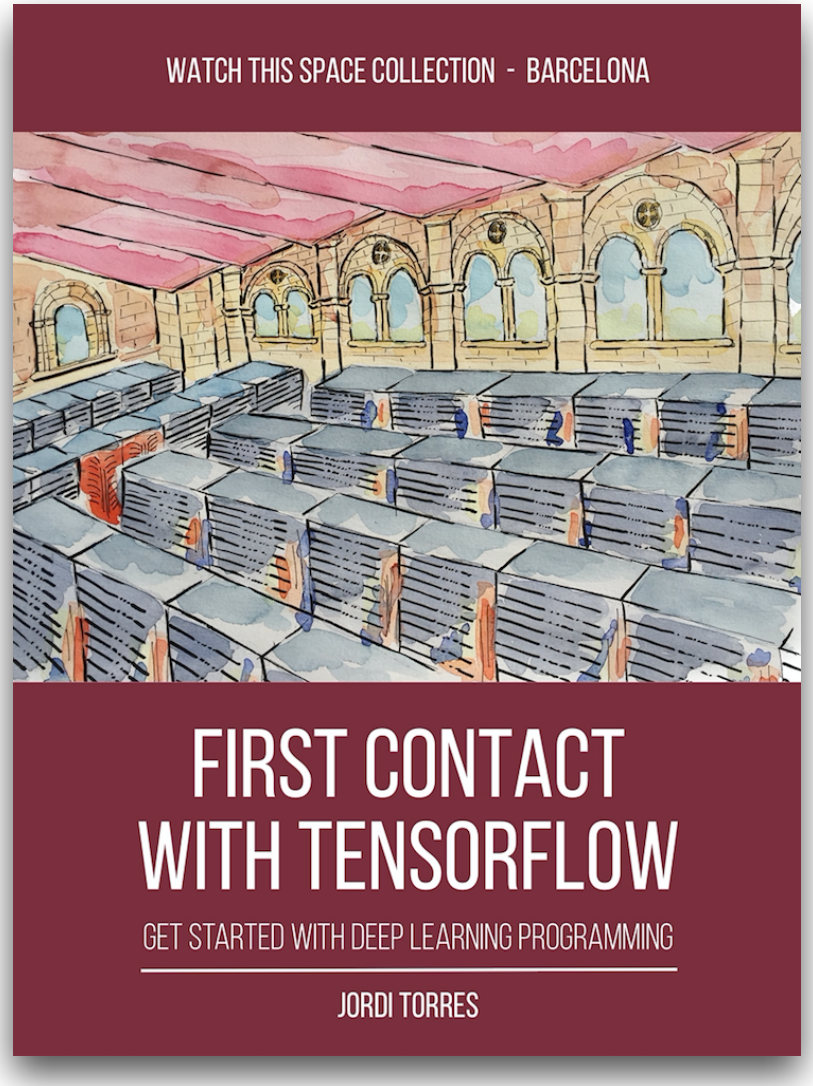
# Hands-on 2

1. Download the code from:

   https://github.com/jorditorresBCN/TutorialTensorFlow

2. Follow teacher's instructions

# Content:




Hello World en Tensorflow
CLUSTERING
8
6
4
2
0 2 4 6

# Basic data type: Tensor

Tensors can be considered a dynamically-sized multidimensional data arrays.

| Type in TensorFlow | Type in Python | Description |
|---|---|---|
| **DT_FLOAT** | tf.float32 | Floating point of 32 bits |
| **DT_INT16** | tf.int16 | Integer of 16 bits |
| **DT_INT32** | tf.int32 | Integer of 32 bits |
| **DT_INT64** | tf.int64 | Integer of 64 bits |
| **DT_STRING** | tf.string | *String* |
| **DT_BOOL** | tf.bool | *Boolean* |

# Tensor Shape, Rank and Dimension

| Shape | Rank | Dimension Number |
|---|---|---|
| **[]** | 0 | 0-D |
| **[D0]** | 1 | 1-D |
| **[D0, D1]** | 2 | 2-D |
| **[D0, D1, D2]** | 3 | 3-D |
| **…** | … | … |
| **[D0, D1, ... Dn]** | n | n-D |

Example: Tensor rank 2

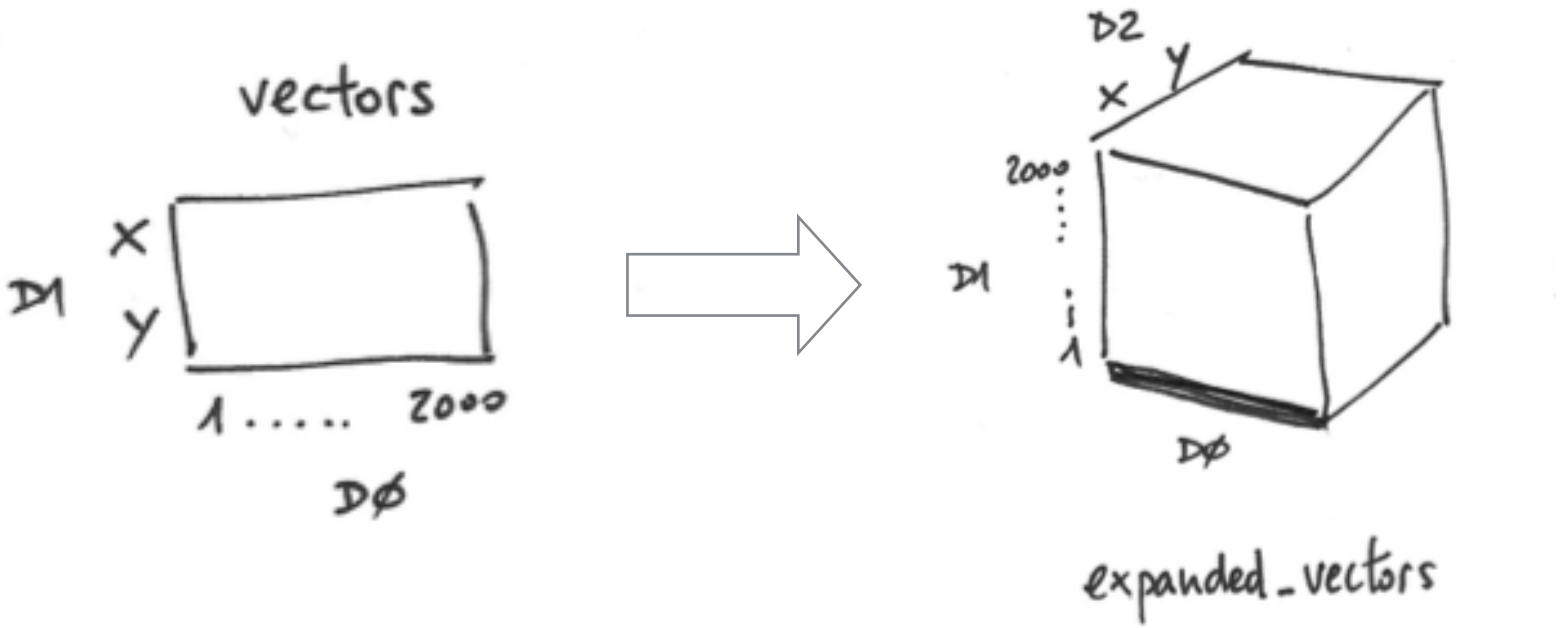
```
t = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

# Tensor Transformations

| Operación | Descripción |
|---|---|
| **tf.shape** | Para saber la forma de un tensor |
| **tf.size** | Para saber el número de elementos de un tensor |
| **tf.rank** | Para saber el rango de un tensor |
| **tf.reshape** | Permite cambiar la forma de un tensor manteniendo los mismos elementos que contenía |
| **tf.squeeze** | Permite borrar del tensor dimensiones de tamaño 1 |
| **tf.expand_dims** | Permite insertar una dimensión al tensor |
| **tf.slice** | Extrae una porción del tensor |
| **tf.split** | Divide el tensor en varios tensores a lo largo de una dimensión |
| **tf.tile** | Crea un nuevo tensor replicando un tensor múltiples veces |
| **tf.concat** | Concatena tensores en una de las dimensiones |
| **tf.reverse** | Invierte una determinada dimensión de un tensor |
| **tf.transpose** | Permuta dimensiones de un tensor |
| **tf.gather** | Recolecta porciones de acuerdo a un índice |

# Tensor Transformation example:

```
vectors = tf.constant(conjunto_puntos)
extended_vectors = tf.expand_dims(vectors, 0)
```



# Data Storage in TensorFlow

1. Data files

   example: https://github.com/jorditorresBCN/LibroTensorFlow/blob/master/input_data.py

2. Memory:
   - Constants `tf.constant(…)`
   - Variables `tf.Variable(…)`

3. Python code : Placeholders
   - `feed_dict` parameter

# Constants generation

| Operation | Description |
|---|---|
| **tf.zeros_like** | Crea un *tensor* con todos los elementos inicializados a 0 |
| **tf.ones** | Crea un *tensor* con todos los elementos inicializados a 1 |
| **tf.ones_like** | Crea un *tensor* con todos los elementos inicializados a 1 |
| **tf.fill** | Crea un *tensor* con todos los elementos inicializados a un valor *scalar* pasado como |
| **tf.constant** | Crea un *tensor* de constantes con los elementos indicados como argumento |

# Tensor random generation

| Operation | Description |
|---|---|
| **tf.random_normal** | Valores *random* con una distribución normal, con una media y desviación estándar indicadas como argumentos |
| **tf.truncated_normal** | Igual que la anterior pero eliminando aquellos valores cuya magnitud es más de 2 veces la desviación estándar. |
| **tf.random_uniform** | Valores *random* con una distribución uniforme de un rango indicado en los argumentos |
| **tf.random_shuffle** | Mezclados aleatoriamente los elementos de un *tensor* en su primera dimensión |
| **tf.set_random_seed** | Establece la semilla aleatoria a nivel de grafo |

# Placeholders

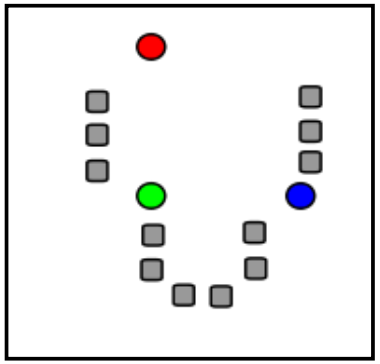
- With calls `Session.run()` or `Tensor.eval()`

```
import tensorflow as tf

a = tf.placeholder("float")
b = tf.placeholder("float")

y = tf.mul(a, b)

sess = tf.Session()

print sess.run(y, feed_dict={a: 3, b: 3})
```

# K-means algorithm

(Example from Wikipedia)

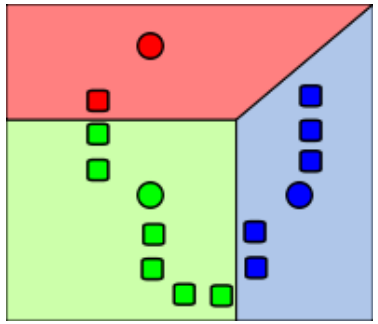
1. *k* initial "means" (in this case *k*=3) are randomly generated within the data domain (shown in color)



# K-means algorithm

(Example from Wikipedia)

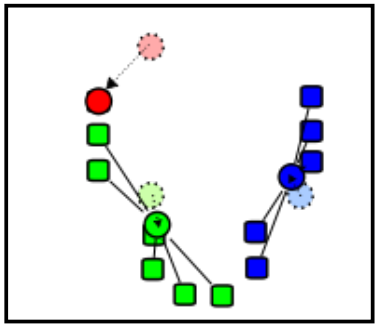
2. *k* clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.



# K-means algorithm

(Example from Wikipedia)

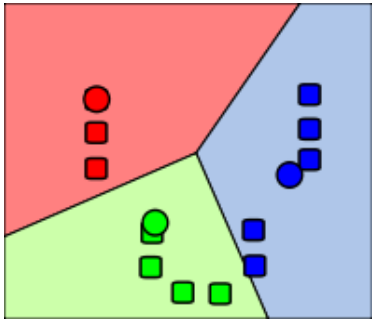
3. The centroid of each of the *k* clusters becomes the new mean.*k* clusters are created by associating every observation with the nearest mean.



# K-means algorithm

(Example from Wikipedia)

4. Steps 2 and 3 are repeated until convergence has been reached.



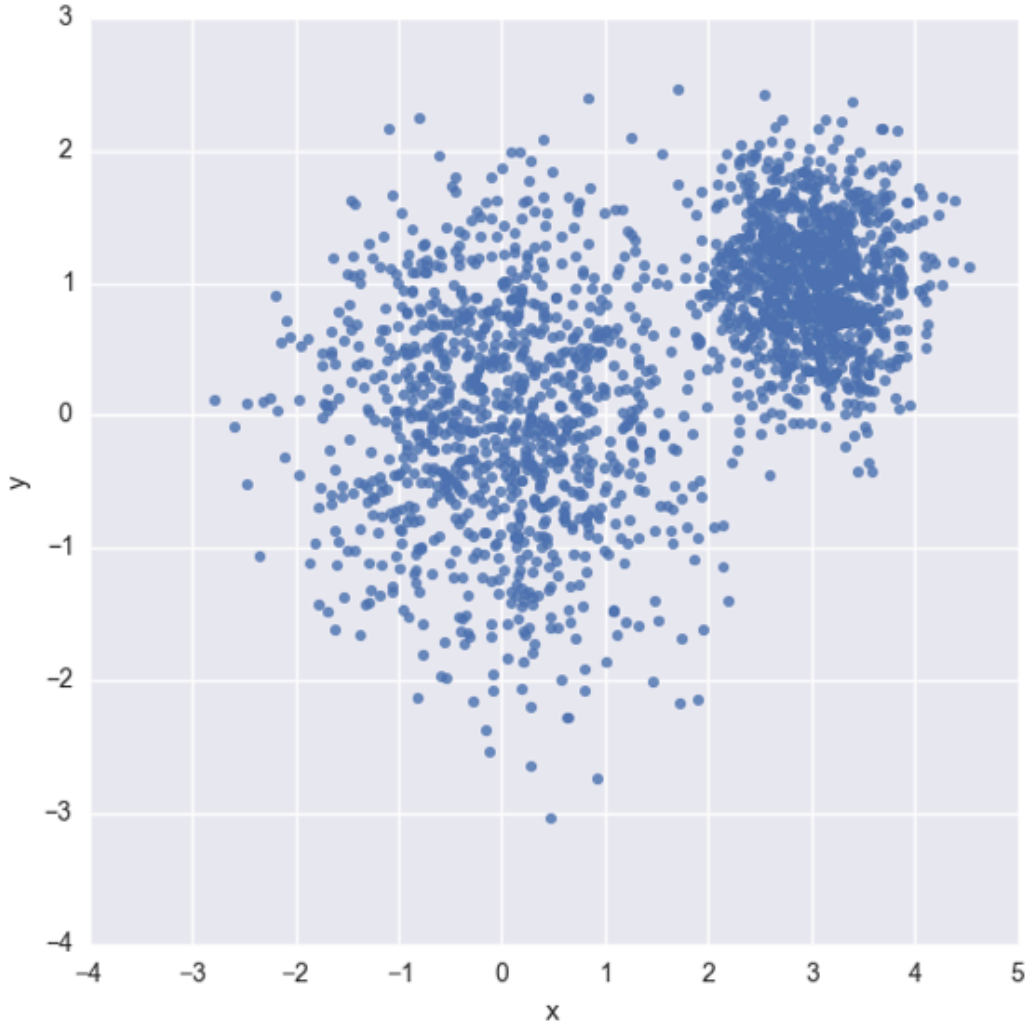
# K-means algorithm

As it is a heuristic algorithm, there is no guarantee that it will converge to the global optimum, and the result may depend on the initial clusters. As the algorithm is usually very fast, it is common to run it multiple times with different starting conditions.

# K-means: Case study

```
num_puntos = 2000
conjunto_puntos = []
for i in xrange(num_puntos):
  if np.random.random() > 0.5:
    conjunto_puntos.append([np.random.normal(0.0, 0.9),
                            np.random.normal(0.0, 0.9)])
  else:
    conjunto_puntos.append([np.random.normal(3.0, 0.5),
                           np.random.normal(1.0, 0.5)])
```

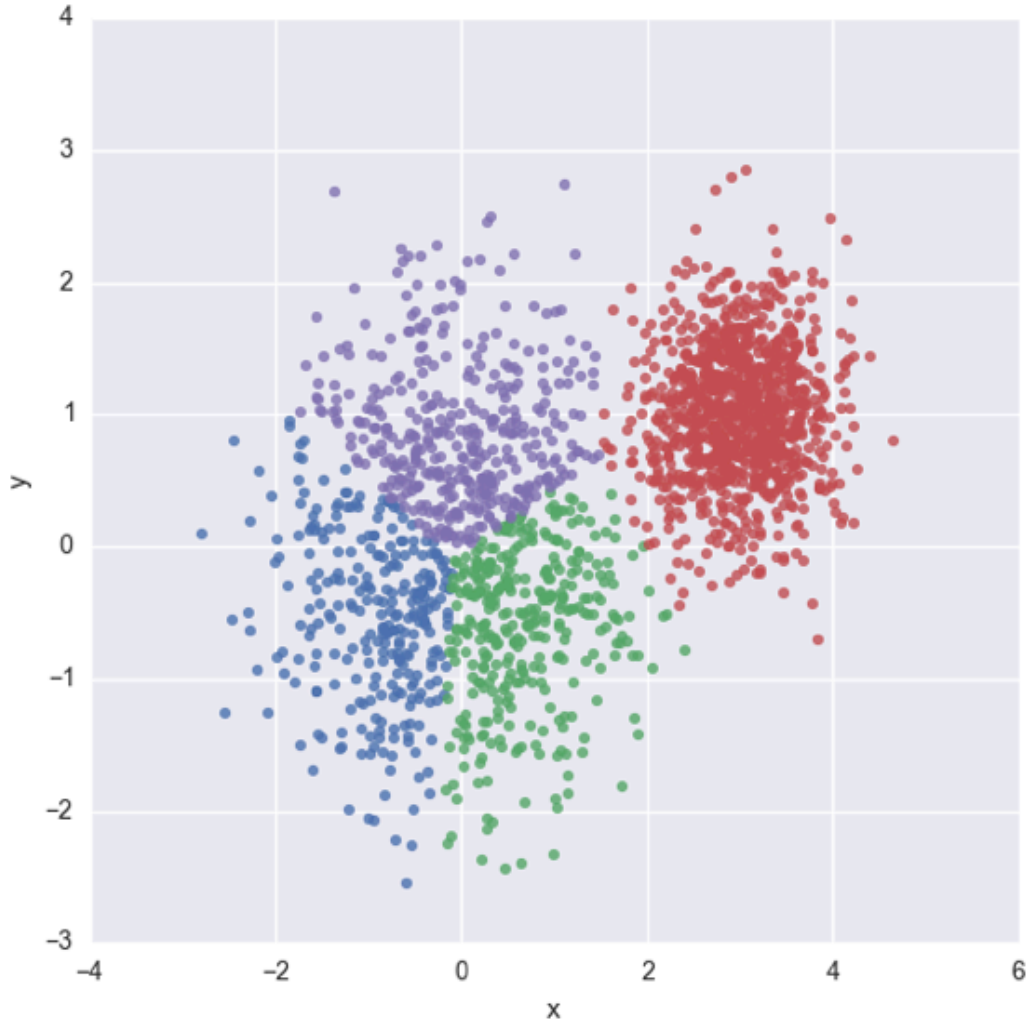
# K-means: Case study



# K-means in TensorFlow

```
vectors = tf.constant(conjunto_puntos)
k = 4
centroides = tf.Variable(tf.slice(tf.random_shuffle(vectors),[0,0],[k,-1]))

expanded_vectors = tf.expand_dims(vectors, 0)
expanded_centroides = tf.expand_dims(centroides, 1)

assignments = tf.argmin(tf.reduce_sum(tf.square(tf.sub(expanded_vectors,
              expanded_centroides)), 2), 0)

means = tf.concat(0, [tf.reduce_mean(tf.gather(vectors,
tf.reshape(tf.where( tf.equal(assignments, c)),[1,-1])),
reduction_indices=[1]) for c in xrange(k)])

update_centroides = tf.assign(centroides, means)

init_op = tf.initialize_all_variables()

sess = tf.Session()
sess.run(init_op)

for step in xrange(100):
   _, centroid_values, assignment_values = sess.run([update_centroides,
                  centroides, assignments])
```

# K-means: Case study



# K-means: Case study

- Plot code

```
data = {"x": [], "y": [], "cluster": []}

for i in xrange(len(assignment_values)):
  data["x"].append(conjunto_puntos[i][0])
  data["y"].append(conjunto_puntos[i][1])
  data["cluster"].append(assignment_values[i])

df = pd.DataFrame(data)
sns.lmplot("x", "y", data=df,
           fit_reg=False, size=6,
           hue="cluster", legend=False)

plt.show()
```

# K-means in TensorFlow: in more detail!

- Create a Tensor

```
vectors = tf.constant(conjunto_puntos)
```

- Create a *k* initial "means"

```
k = 4
centroides = tf.Variable(tf.slice(tf.random_shuffle(vectors),
[0,0],[k,-1]))
```

```
print vectors.get_shape()
print centroides.get_shape()

TensorShape([Dimension(2000), Dimension(2)])
TensorShape([Dimension(4), Dimension(2)])
```

# K-means in TensorFlow: in more detail!

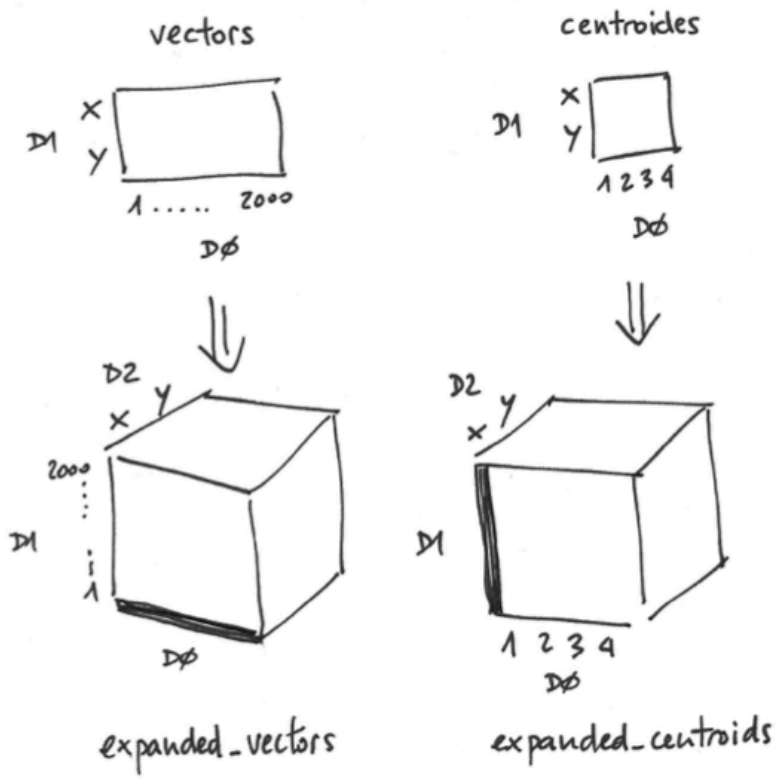
```
expanded_vectors = tf.expand_dims(vectors, 0)
expanded_centroides = tf.expand_dims(centroides, 1)

assignments =
tf.argmin(tf.reduce_sum(tf.square(tf.sub(expanded_vectors,
                expanded_centroides)), 2), 0)
```

- "Squared Euclidian Distance"

$$d^2(\text{vector}, \text{centroide}) = (\text{vector}_x - \text{centroide}_x)^2 + (\text{vector}_y - \text{centroide}_y)^2$$

# K-means in TensorFlow: in more detail!

```
expanded_vectors = tf.expand_dims(vectors, 0)
expanded_centroides = tf.expand_dims(centroides, 1)
```



# K-means in TensorFlow: in more detail!

- "Squared Euclidian Distance"

$$d^2(\text{vector}, \text{centroide}) = (\text{vector}_x - \text{centroide}_x)^2 + (\text{vector}_y - \text{centroide}_y)^2$$

```
diff=tf.sub(expanded_vectors, expanded_centroides)
sqr= tf.square(diff)
distances = tf.reduce_sum(sqr, 2)
assignments = tf.argmin(distances, 0)
```

*diff –> TensorShape([Dimension(4), Dimension(2000), Dimension(2)])*
*sqr –> TensorShape([Dimension(4), Dimension(2000), Dimension(2)])*
*distance –> TensorShape([Dimension(4), Dimension(2000)])*
*assignments –> TensorShape([Dimension(2000)])*

***(*) Shape broadcasting***

# Tensor operations

| Operación | Descripción |
|---|---|
| **tf.reduce_sum** | Calcula la suma de los elementos a lo largo de una de las dimensiones |
| **tf.reduce_prod** | Calcula el producto de los elementos a lo largo de una de las dimensiones |
| **tf.reduce_min** | Calcula el mínimo de los elementos a lo largo de una dimensión |
| **tf.reduce_max** | Calcula el máximo de los elementos a lo largo de una dimensión |
| **tf.reduce_mean** | Calcula la media de los elementos a lo largo de una dimensión |

# Tensor operations

| Operación | Descripción |
|---|---|
| **tf.argmin** | Retorna el índice del elemento con el valor menor en la dimensión del *tensor* indicada |
| **tf.argmax** | Retorna el índice del elemento con el valor máximo en la dimensión del *tensor* indicada |

# K-means in TensorFlow: in more detail!

- New centroids

```
means = tf.concat(0, [tf.reduce_mean(tf.gather(vectors,
tf.reshape(tf.where( tf.equal(assignments, c)),[1,-1])),
reduction_indices=[1]) for c in xrange(k)])
```

means tensor is equal to the concatenation of k tensors that correspond to the mean value of cluster elements.

# Detailed explanation:

Detalle de cada una de las operaciones TensorFlow que intervienen en el cálculo del valor medio de los puntos pertenecientes a un *cluster*:

- Con tf.equal se obtiene un tensor booleano (Dimension(2000)) que indica (con valor “true” ) las posiciones donde el valor del tensor assignment coincide con el cluster c (uno de los k), del que en aquel momento estamos calculando el valor medio de sus puntos.

- Con tf.where se construye un tensor (Dimension (1) x Dimension(2000)) con la posición donde se encuentran los valores “true” en el tensor booleano recibido como parámetro. Es decir, una lista con las posiciones de estos.

- Con tf.reshape se construye un tensor (Dimension (2000) x Dimension(1) ) con los índices de los puntos en el tensor vectors que pertenecen a este cluster c.

- Con tf.gather se construye un tensor (Dimension (1) x Dimension (2000) x Dimension(2) ) que reúne las coordenadas de los puntos que forman el cluster c.

- Con tf.reduce_mean se construye un tensor (Dimension (1) x Dimension(2) ) que contiene el valor medio de todos los puntos que pertenecen a este cluster c.

# K-means in TensorFlow: in more detail!

```
update_centroides = tf.assign(centroides, means)

init_op = tf.initialize_all_variables()

sess = tf.Session()
sess.run(init_op)

for step in xrange(100):
   _ , centroid_values, assignment_values = sess.run(
  [update_centroides,centroides, assignments])
```

# Hands-on 3

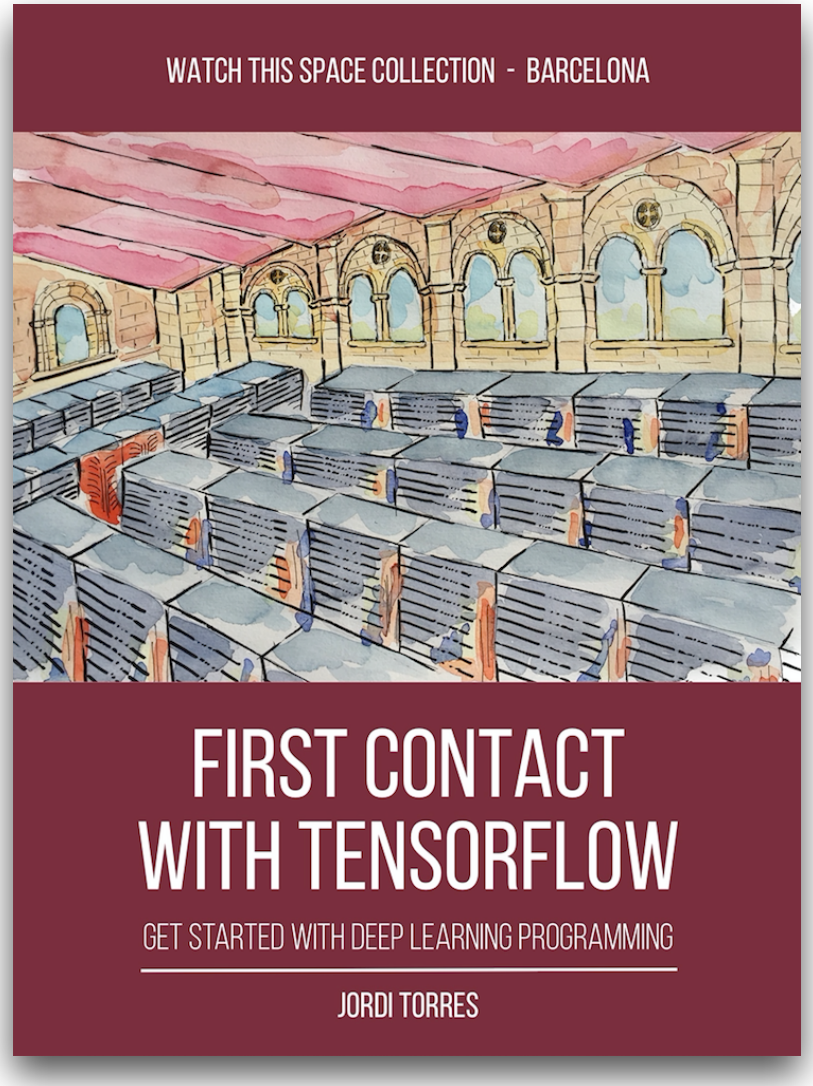
1. Download **`kmeans.py`** from:

   https://github.com/jorditorresBCN/TutorialTensorFlow

2. Try with diff number of clusters and elements

3. Follow teacher's instructions

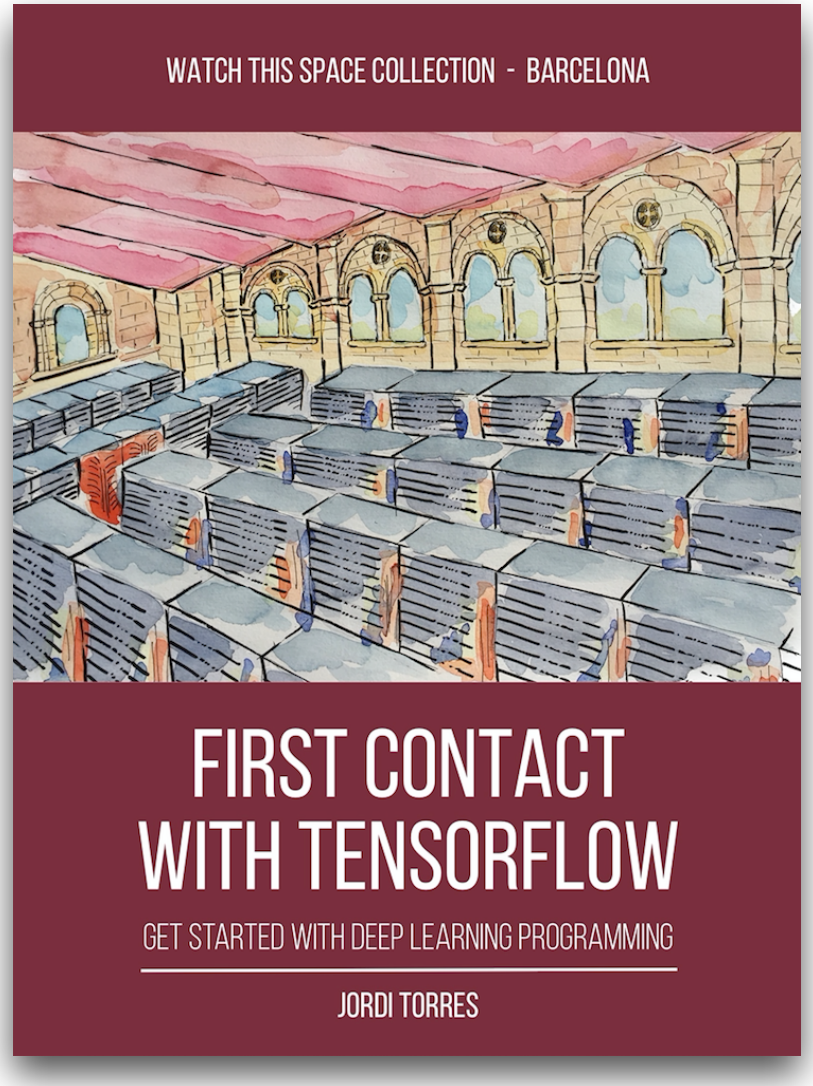
# Bonus: advanced content




## Part2: Advanced

4. Single Layer Neural Network
5. Multi-layer Neural Networs
6. TensorFlow and GPUs
   Closing

# Content:



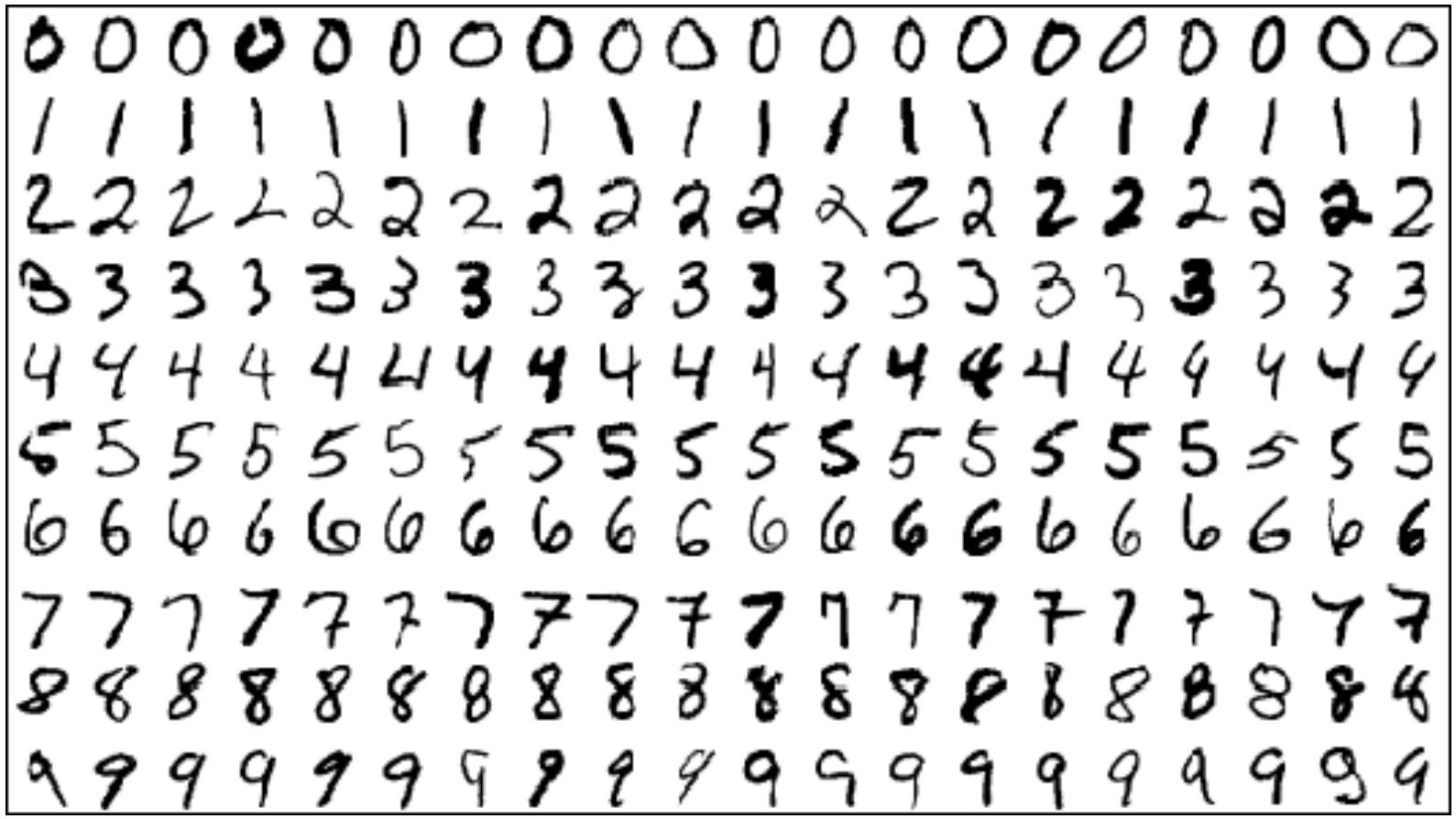

Introduction

1. TensorFlow Basics
2. Linear Regression
3. Clustering
4. Single Layer Neural Network
5. Multi-layer Neural Networs
6. TensorFlow and GPUs

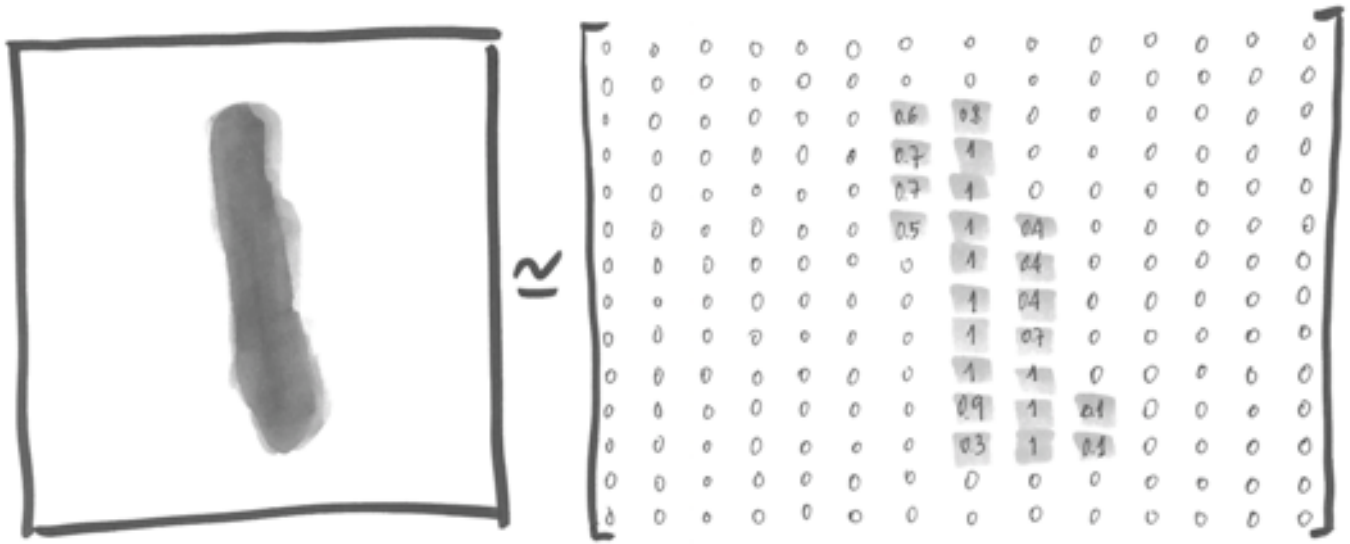
Closing

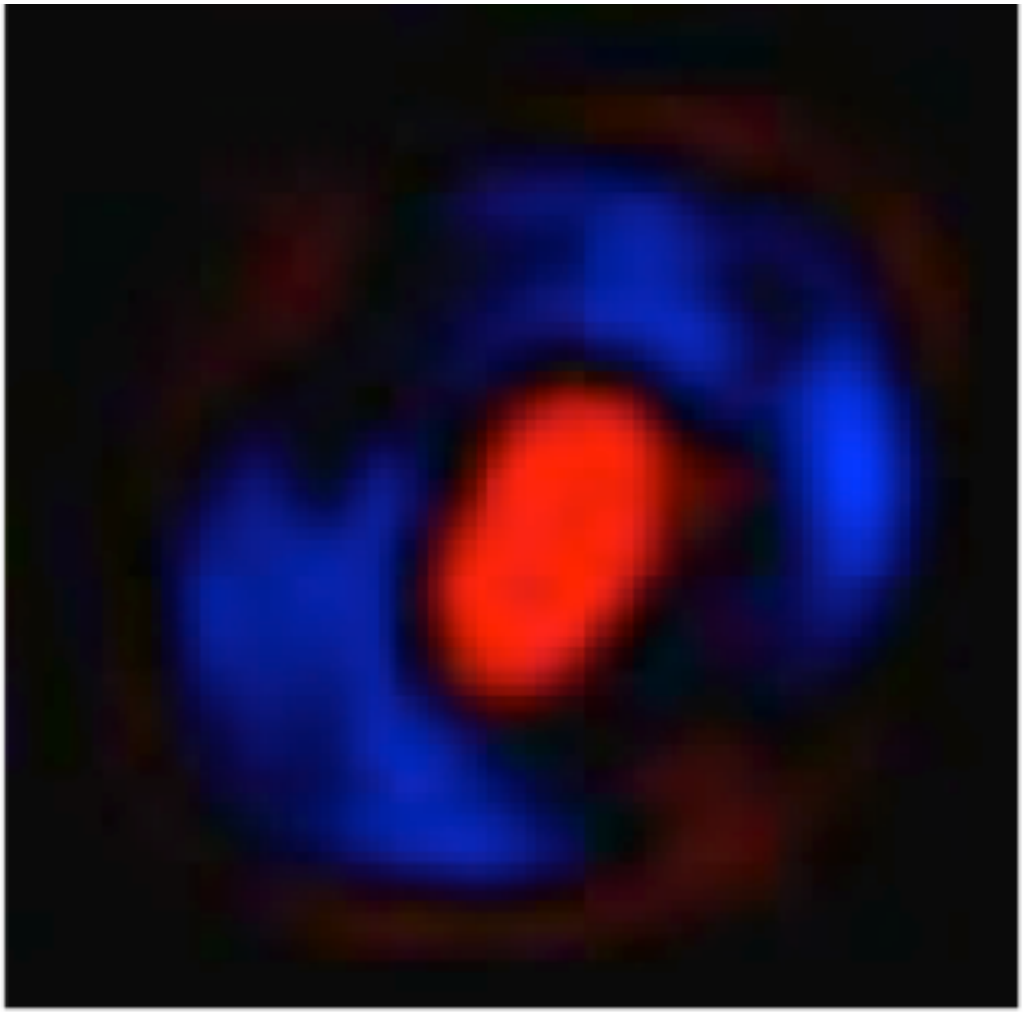

# ONLY AN OVERVIEW (next course)

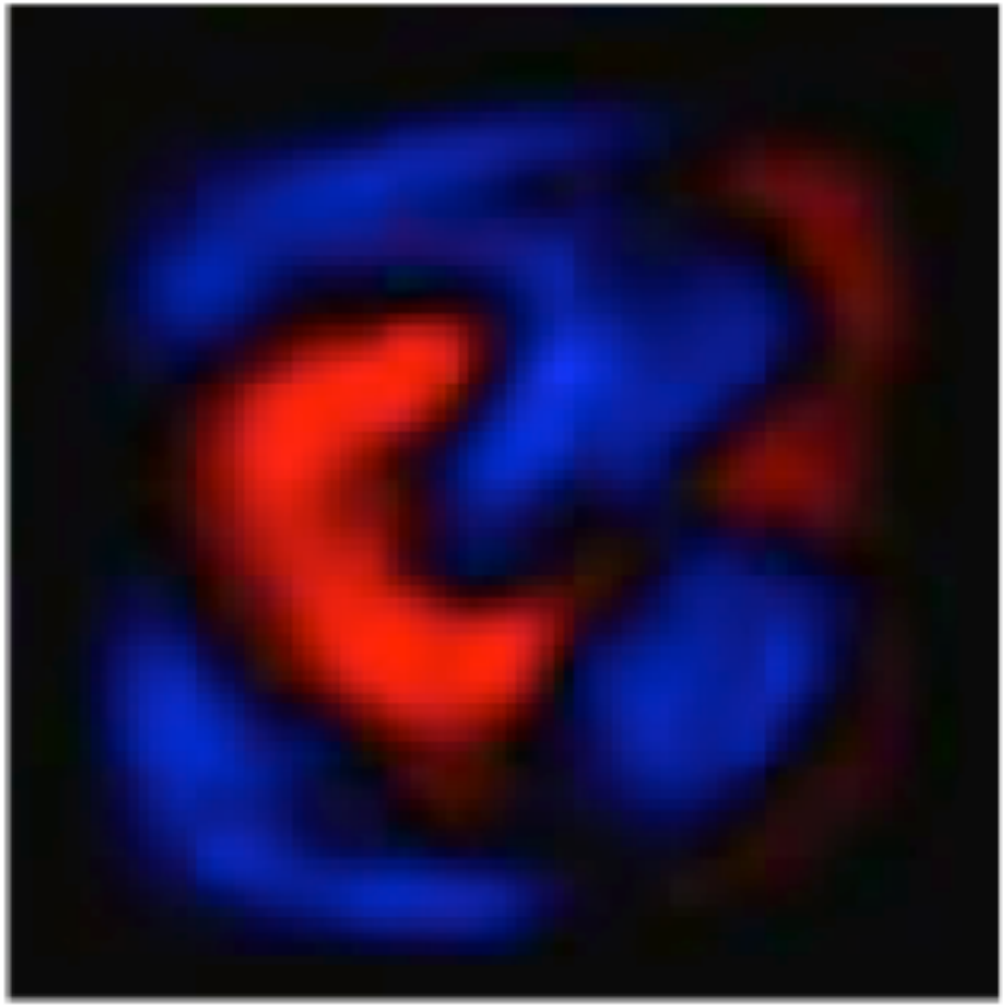
# “Hello World” = MNIST

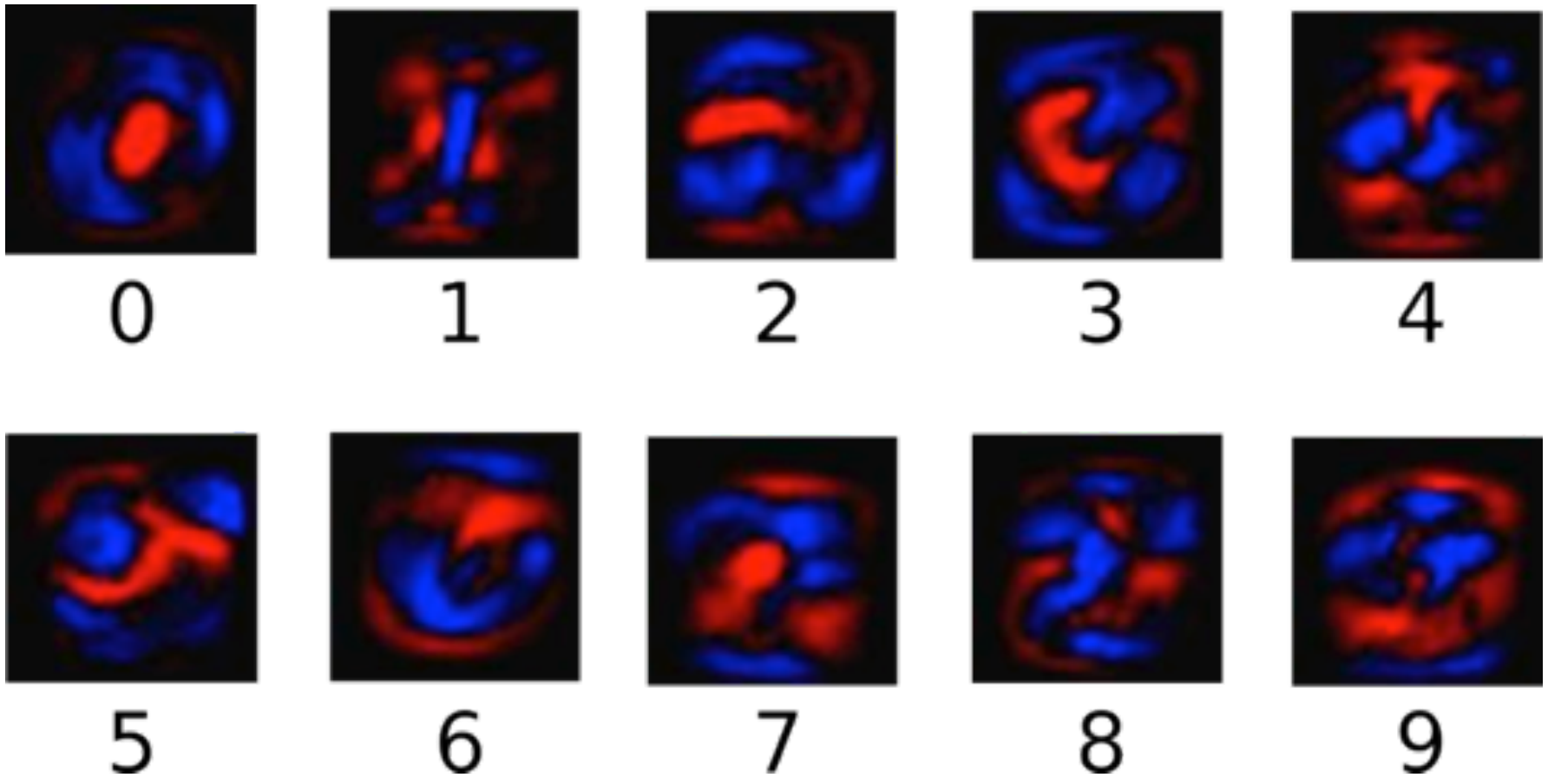


# “Hello World” = MNIST

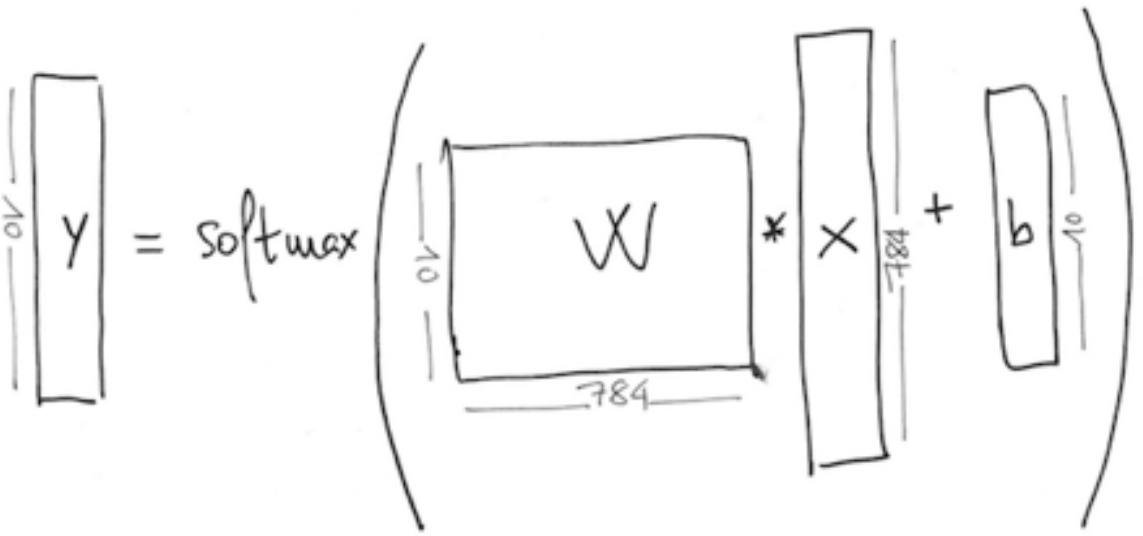







0
1
2
3
4
5
6
7
8
9

$$\text{Softmax}(x)_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)} = \frac{e^{x_i}}{\sum_j e^{x_j}}$$



# Single Layer Neural Network

```
import input_data
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)

import tensorflow as tf
x = tf.placeholder("float", [None, 784])
W = tf.Variable(tf.zeros([784,10]))
b = tf.Variable(tf.zeros([10]))

matm=tf.matmul(x,W)
y = tf.nn.softmax(tf.matmul(x,W) + b)
y_ = tf.placeholder("float", [None,10])

cross_entropy = -tf.reduce_sum(y_*tf.log(y))
train_step = tf.train.GradientDescentOptimizer(0.01).minimize(cross_entropy)

sess = tf.Session()
sess.run(tf.initialize_all_variables())

for i in range(1000):
 batch_xs, batch_ys = mnist.train.next_batch(100)
 sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})
 correct_prediction = tf.equal(tf.argmax(y,1), tf.argmax(y_,1))
 accuracy = tf.reduce_mean(tf.cast(correct_prediction, "float"))
 print sess.run(accuracy, feed_dict={x: mnist.test.images, y_: mnist.test.labels})
```

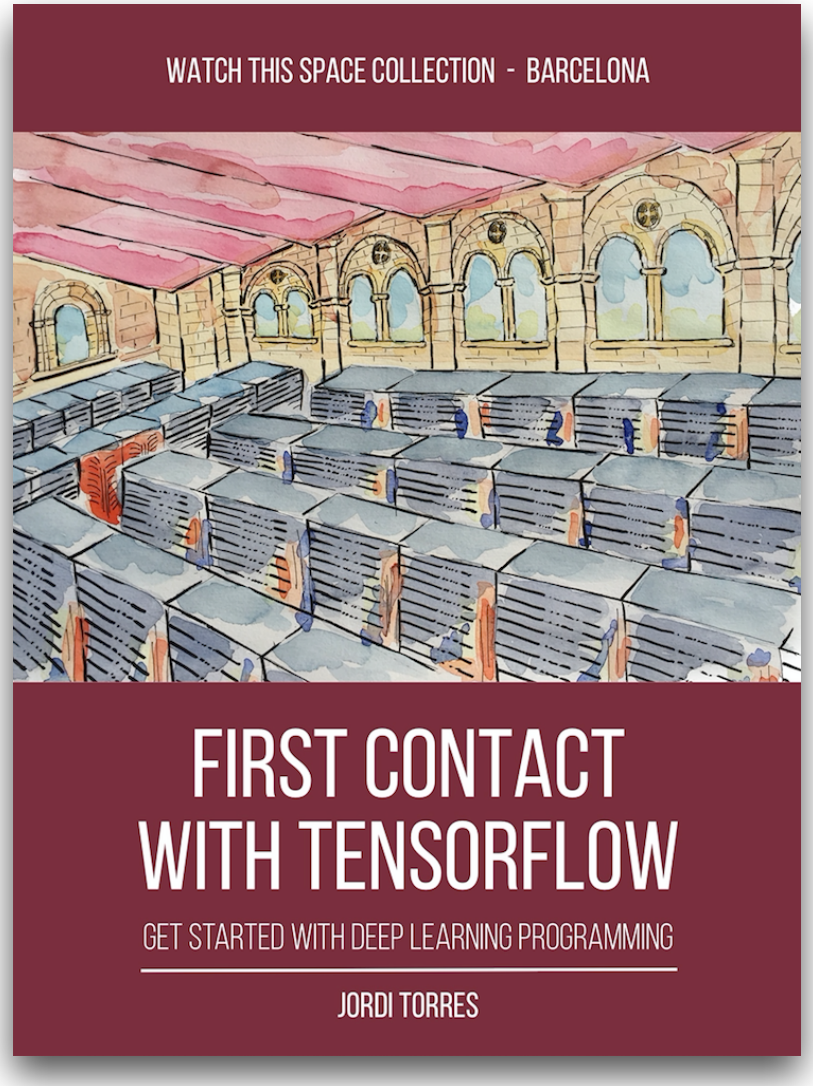
# Hands-on 4

1. Download **`redneuronalsimple.py`** from:

   https://github.com/jorditorresBCN/TutorialTensorFlow

2. Follow teacher's instructions

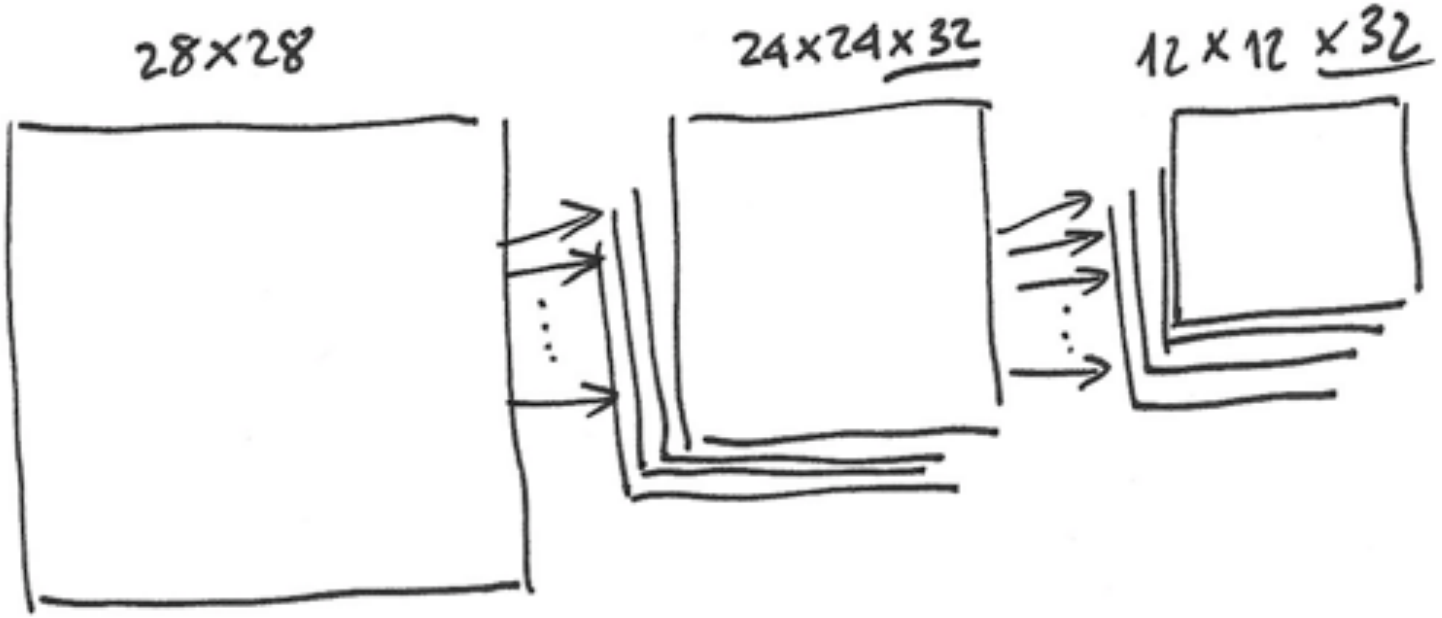
# Content:




# ONLY AN OVERVIEW (next course)

# Convolutional Neural Networks

- Convolution
- Pooling



```
import input_data
mnist = input_data.read_data_sets('MNIST_data', one_hot=True)
import tensorflow as tf


x = tf.placeholder("float", shape=[None, 784])
y_ = tf.placeholder("float", shape=[None, 10])

x_image = tf.reshape(x, [-1,28,28,1])
print "x_image="
print x_image

def weight_variable(shape):
 initial = tf.truncated_normal(shape, stddev=0.1)
 return tf.Variable(initial)


def bias_variable(shape):
 initial = tf.constant(0.1, shape=shape)
 return tf.Variable(initial)



def conv2d(x, W):
 return tf.nn.conv2d(x, W, strides=[1, 1, 1, 1],
padding='SAME')


def max_pool_2x2(x):
  return tf.nn.max_pool(x, ksize=[1, 2, 2, 1],
                        strides=[1, 2, 2, 1], padding='SAME')

W_conv1 = weight_variable([5, 5, 1, 32])
b_conv1 = bias_variable([32])

h_conv1 = tf.nn.relu(conv2d(x_image, W_conv1) + b_conv1)
h_pool1 = max_pool_2x2(h_conv1)

W_conv2 = weight_variable([5, 5, 32, 64])
b_conv2 = bias_variable([64])

h_conv2 = tf.nn.relu(conv2d(h_pool1, W_conv2) + b_conv2)
h_pool2 = max_pool_2x2(h_conv2)
```

```
W_fc1 = weight_variable([7 * 7 * 64, 1024])
b_fc1 = bias_variable([1024])

h_pool2_flat = tf.reshape(h_pool2, [-1, 7*7*64])
h_fc1 = tf.nn.relu(tf.matmul(h_pool2_flat, W_fc1) + b_fc1)

keep_prob = tf.placeholder("float")
h_fc1_drop = tf.nn.dropout(h_fc1, keep_prob)

W_fc2 = weight_variable([1024, 10])
b_fc2 = bias_variable([10])

y_conv=tf.nn.softmax(tf.matmul(h_fc1_drop, W_fc2) + b_fc2)

cross_entropy = -tf.reduce_sum(y_*tf.log(y_conv))
train_step =
tf.train.AdamOptimizer(1e-4).minimize(cross_entropy)
correct_prediction = tf.equal(tf.argmax(y_conv,1),
tf.argmax(y_,1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, "float"))


sess = tf.Session()

sess.run(tf.initialize_all_variables())

for i in range(200):
  batch = mnist.train.next_batch(50)
  if i%10 == 0:
    train_accuracy = sess.run( accuracy, feed_dict={
        x:batch[0], y_: batch[1], keep_prob: 1.0})
    print("step %d, training accuracy %g"%(i, train_accuracy))
  sess.run(train_step,feed_dict={x: batch[0], y_: batch[1],
keep_prob: 0.5})

print("test accuracy %g"% sess.run(accuracy, feed_dict={
    x: mnist.test.images, y_: mnist.test.labels, keep_prob:
1.0}))
```

91% —> 99,2%

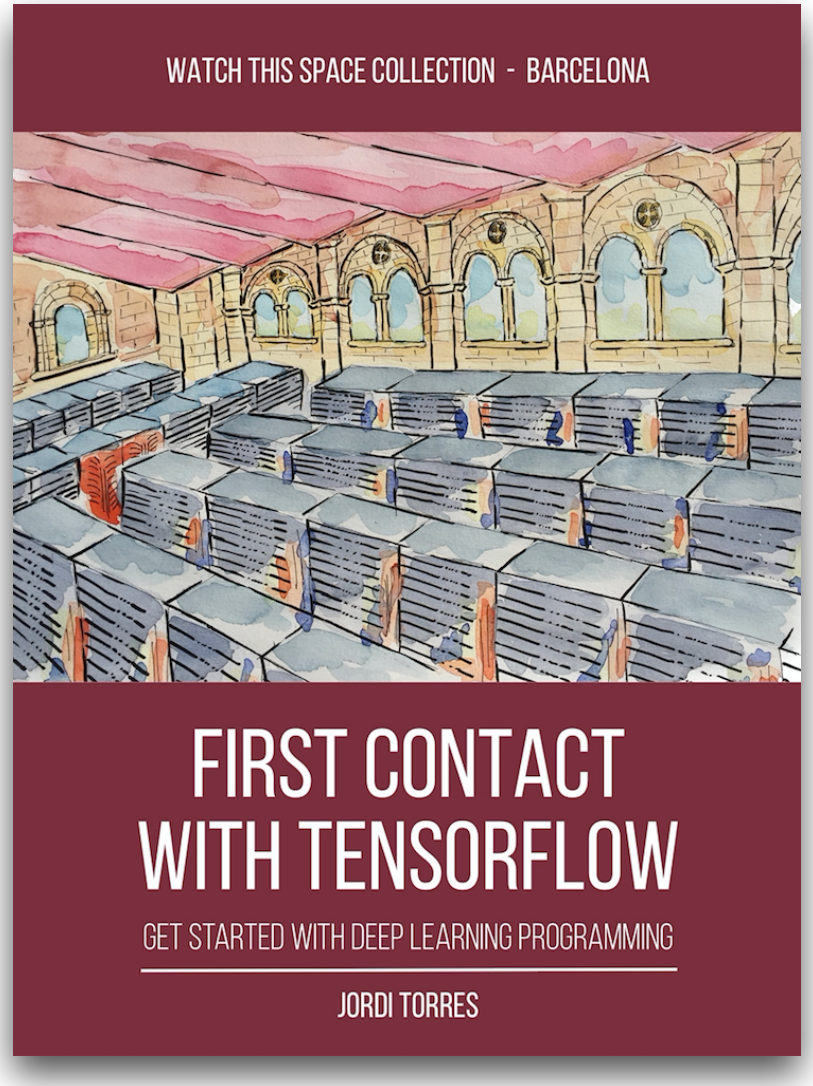
# Hands-on 5

1. Download **`CNN.py`** from:

   https://github.com/jorditorresBCN/TutorialTensorFlow

2. Follow teacher's instructions

# Content:




Introduction

1. TensorFlow Basics
2. Linear Regression
3. Clustering
4. Single Layer Neural Network
5. Multi-layer Neural Networs
6. TensorFlow and GPUs

Closing


# ONLY AN OVERVIEW (next course)

```
import numpy as np
import tensorflow as tf
import datetime

#Processing Units logs
log_device_placement = True

#num of multiplications to perform
n = 10

#Create random large matrix
A = np.random.rand(1e4, 1e4).astype('float32')
B = np.random.rand(1e4, 1e4).astype('float32')

# Creates a graph to store results
c1 = []
c2 = []

def matpow(M, n):
    if n < 1: #Abstract cases where n < 1
        return M
    else:
        return tf.matmul(M, matpow(M, n-1))

with tf.device('/gpu:0'):
    a = tf.constant(A)
    b = tf.constant(B)
    #compute A^n and B^n and store results in c1
    c1.append(matpow(a, n))
    c1.append(matpow(b, n))

with tf.device('/cpu:0'):
  sum = tf.add_n(c1) #Addition of all elements in c1, i.e. A^n + B^n

t1_1 = datetime.datetime.now()
with tf.Session(config=tf.ConfigProto(log_device_placement=log_device_placement)) as sess:
    # Runs the op.
    sess.run(sum)
t2_1 = datetime.datetime.now()


#GPU:0 computes A^n
with tf.device('/gpu:0'):
    #compute A^n and store result in c2
    a = tf.constant(A)
    c2.append(matpow(a, n))

#GPU:1 computes B^n
with tf.device('/gpu:1'):
    #compute B^n and store result in c2
    b = tf.constant(B)
    c2.append(matpow(b, n))

with tf.device('/cpu:0'):
  sum = tf.add_n(c2) #Addition of all elements in c2, i.e. A^n + B^n

t1_2 = datetime.datetime.now()
with tf.Session(config=tf.ConfigProto(log_device_placement=log_device_placement)) as sess:
    # Runs the op.
    sess.run(sum)
t2_2 = datetime.datetime.now()


print "Single GPU computation time: " + str(t2_1-t1_1)
print "Multi GPU computation time: " + str(t2_2-t1_2)
```

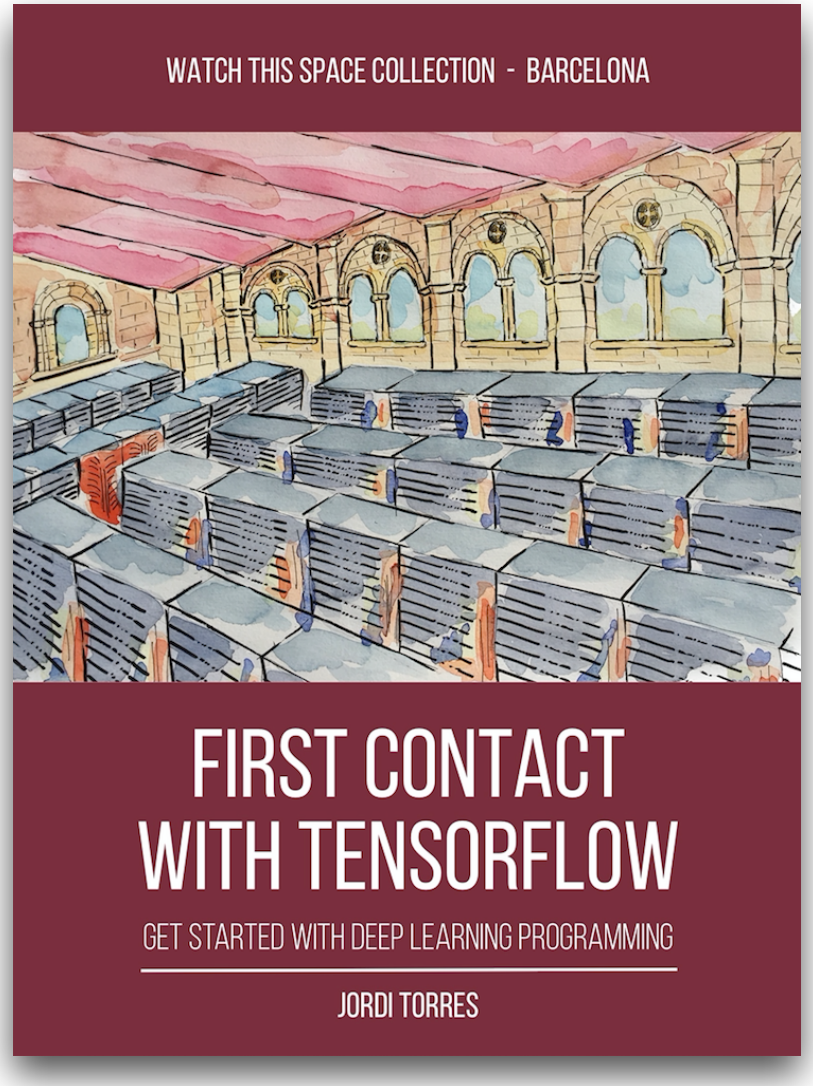
# Hands-on 6

1. Download **`GPUs.py`** from:

   https://github.com/jorditorresBCN/TutorialTensorFlow

2. Follow teacher's instructions

# Content:

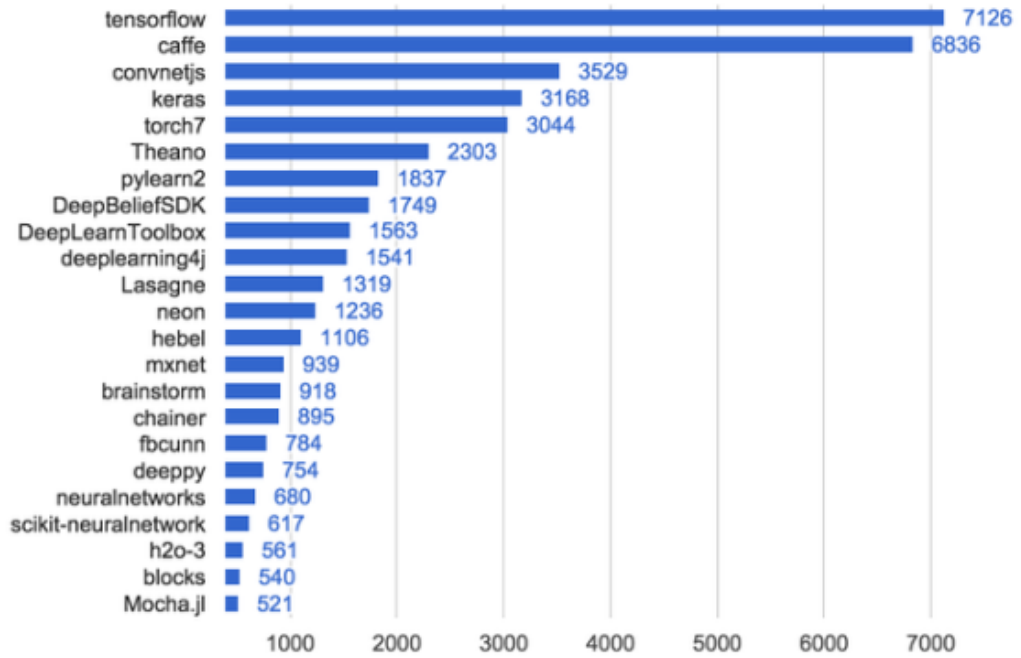


**Kyle McDonald**
@kcimc

Siguiendo

2010-2014: a new deep learning toolkit is released every 47 days. 2015: every 22 days. tensorflow & caffe top github

Ver traducción




RETWEETS 151 ME GUSTA 157

0:05 - 11 nov. 2015

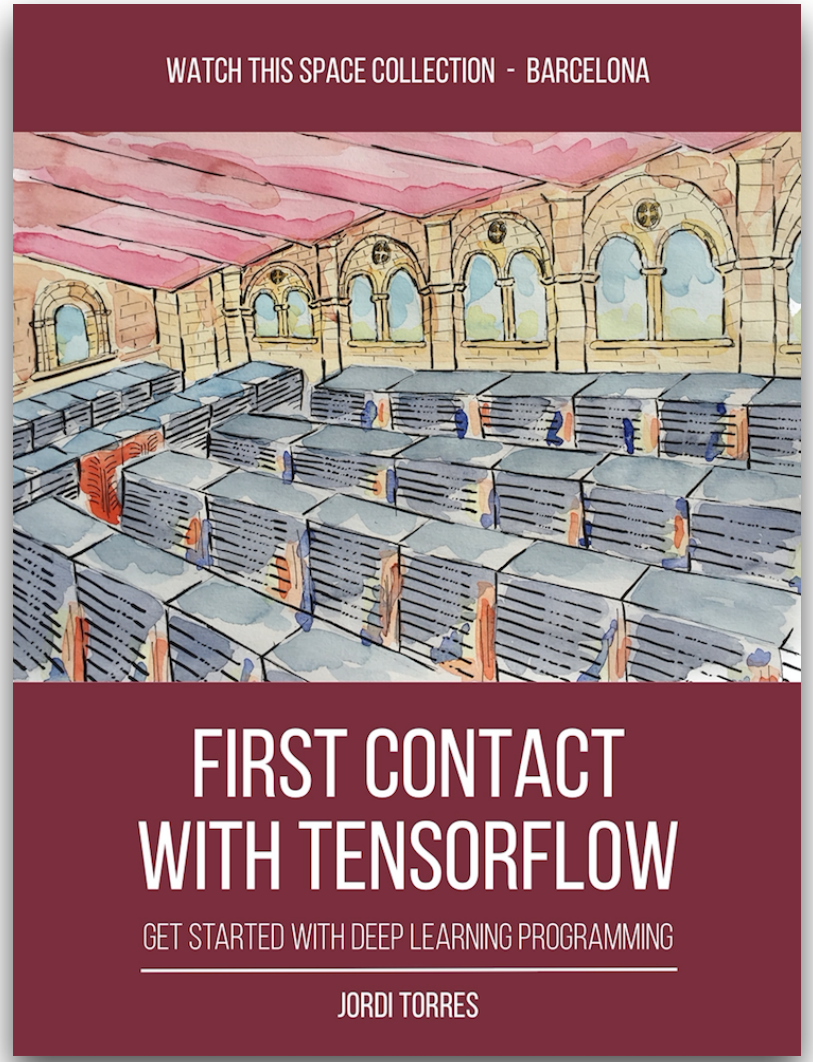
THE DIGITAL POST BETA

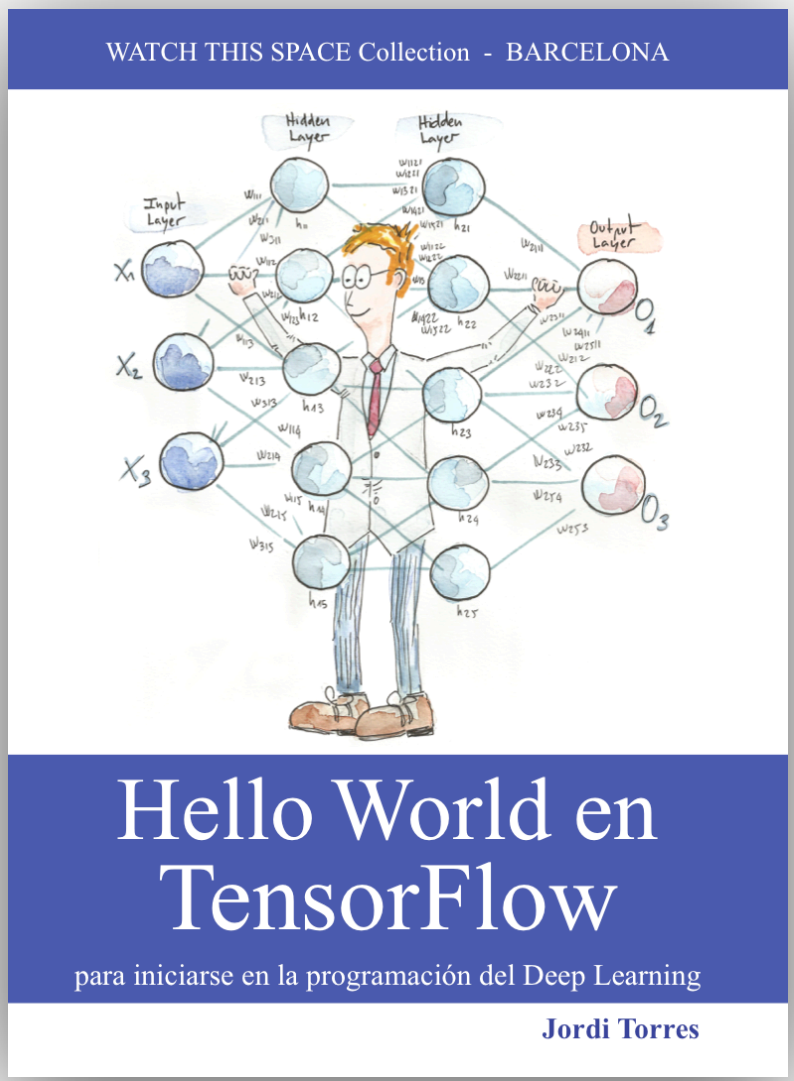
ABOUT US
CONTACT
PITCH US
BOOKMARKS

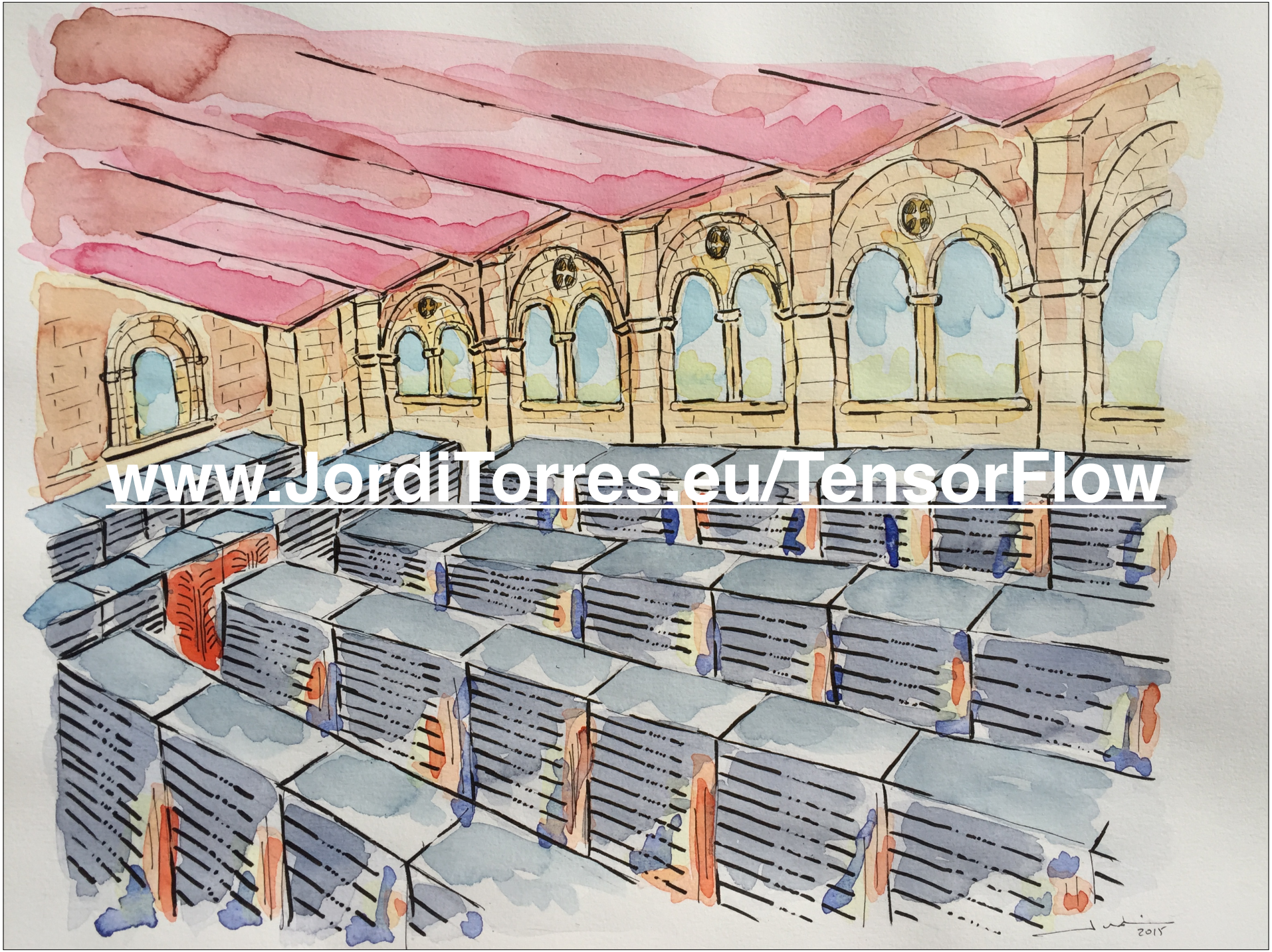

Cognitive Computing is going to transform and improve our lives. But it also presents challenges that we need to be conscious of in order to make the best use of this technology.

*Co-authored by: Mateo Valero and Jordi Torres*

# More information:




www.JordiTorres.eu/TensorFlow