How to build & run your first deep learning network in TensorFlow

Jordi Torres & Maurici Yagües

Barcelona, July 2016

Part of this seminar:

- Documentation
  - Course Contents and First Steps
- Instructors

Jordi Torres

Maurici Yagües

What you will learn in this course

Basic concepts of TensorFlow

- Day 1:
  - How to build a basic TensorFlow graphs and how to train models
  - Case study: Linear Regression in TensorFlow
- Day 2:
  - Basic data structures in TensorFlow
  - Case study: Clustering in TensorFlow

BEGINNER LEVEL

Neural Networks basics in TensorFlow

- Day 3:
  - Single Layer Neural Network in TensorFlow
  - TensorBoard

INTERMEDIATE LEVEL

Deep Learning in TensorFlow

- Day 4:
  - Convolutional Neural Networks
  - TensorFlow High Level APIs: SLIM
- Day 5:
  - Recurrent Neural Networks in TensorFlow

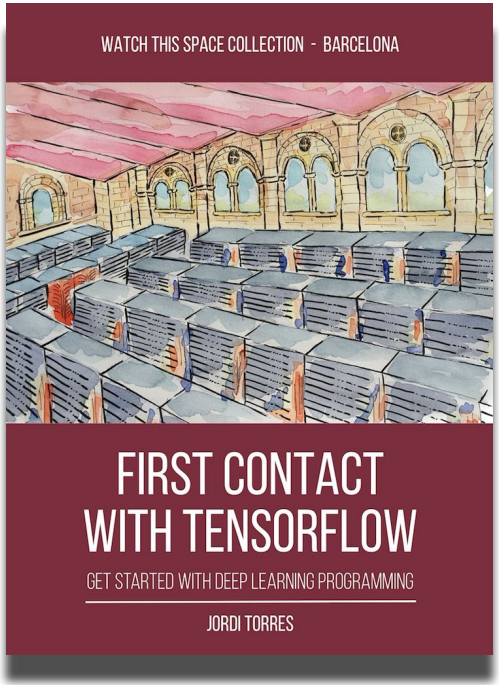
ADVANCED LEVEL

Attendance background?

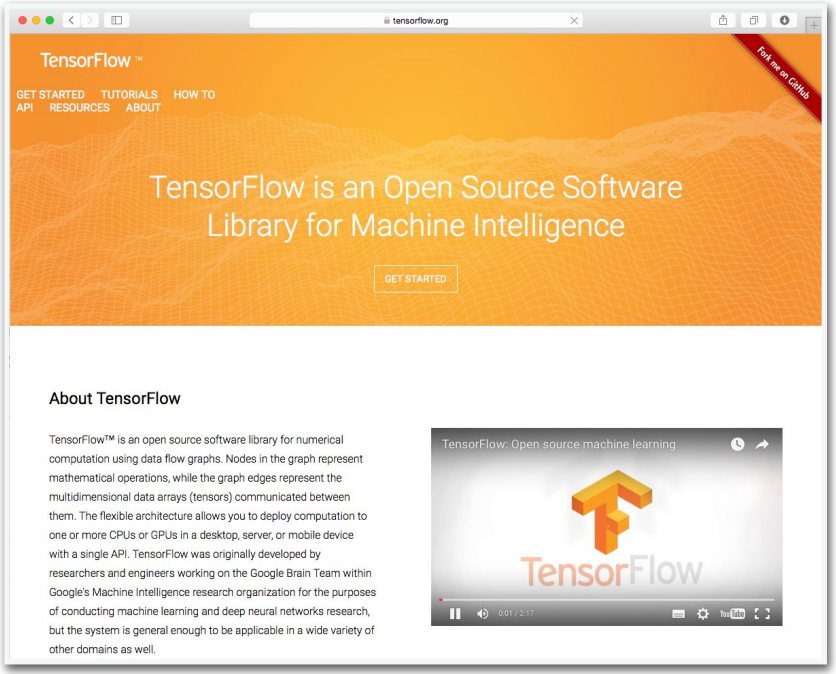
- The course has a **practical nature**, and therefore we reduced the theoretical part as much as possible, **assuming that the attendance has some basic understanding about Machine Learning**
- We assume that the student has some basic knowledge about **Python**. If not, a **Python Quick Start hands-on** that will help to start with this language can be found here

Documentation - Text Book

http://TensorFlowBook.DeepLearning.Barcelona



https://www.TensorFlow.org/



Setup - install all the things!

1. Clone or download this repo:
   https://github.com/jorditorresBCN/FirstContactWithTensorFlow

2. Follow the installation instructions in that repo.
   - You can run the workshop exercises in a Docker container, or
   - You can use a virtual environment

Learn by doing!

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

- Benjamin Franklin

DAY 1

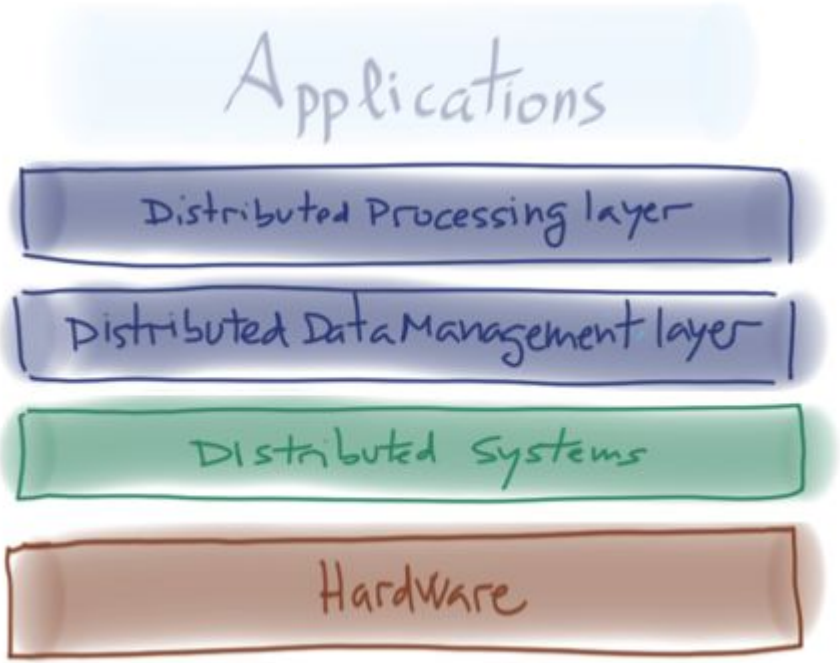
What is TensorFlow

How to build basic TensorFlow graphs and how to train models

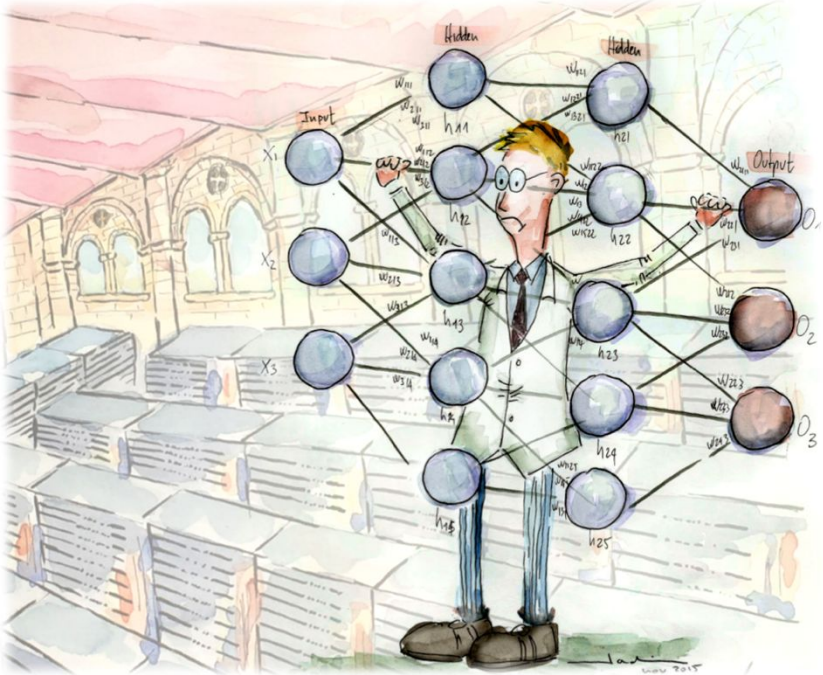
Case study: Linear Regression in TensorFlow

We can now store and perform computation on large data sets

Today systems software stack:



But what we really want is **understand** this data



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

This can be achieved with **machine learning** algorithms

"Field of study that gives computers the ability to learn without being explicitly programmed"

- 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

Figures Source: Julia Ferraioli, Amy Unruh, Eli Bixby - Diving into Machine Learning through TensorFlow - PyCon 2016

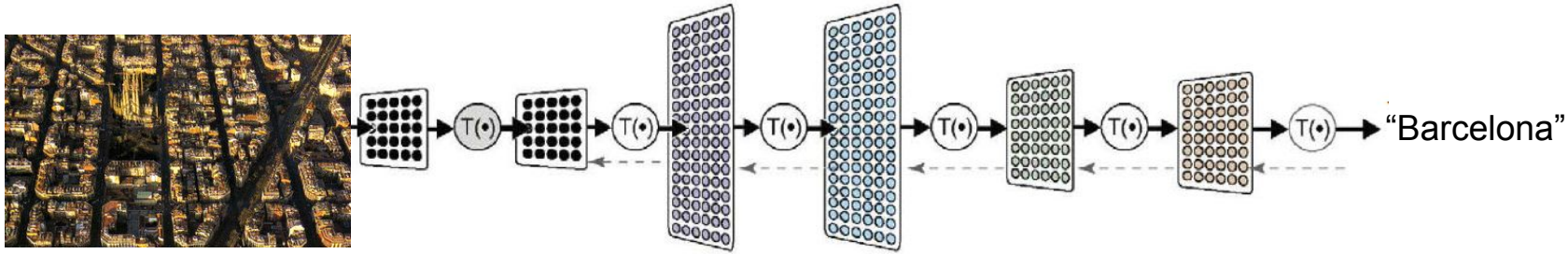
How can we get started with Machine Learning?

- Many ways:
  1. Use a Cloud-based API (Vision, Speech, etc.)
  2. Run our own pretrained model
  3. Use an existing model architecture, and retrain it or fine tune on our dataset
  4. Develop our own machine learning models for new problems
  5. A combination of previous options!

- **TensorFlow will help you, specially for Deep Learning problems!**

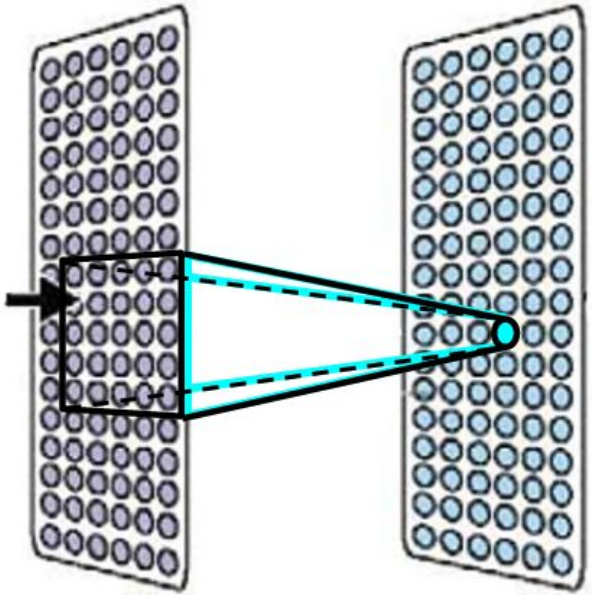
What is Deep Learning?

- A powerful class of machine learning model
- Modern “reincarnation” of artificial neural networks



Source: Source: Large-Scale Deep Learning With TensorFlow - Jeff Dean - Google Brain Team - 2016

How it works?

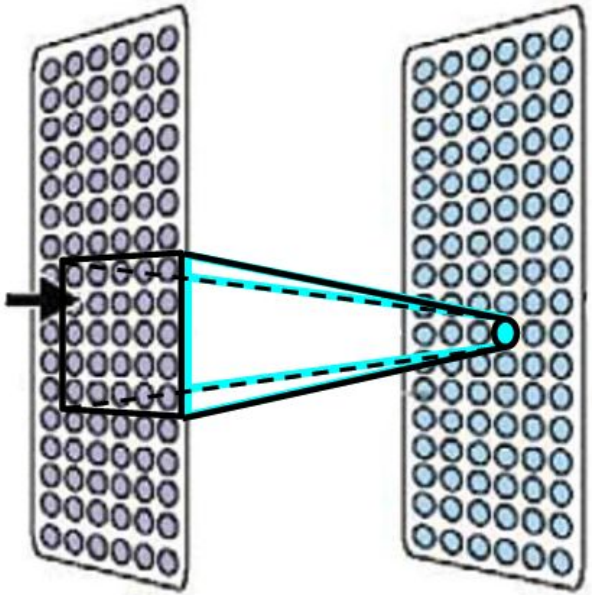


Commonalities with real brains:

- Each neuron is connected to a small subset of other neurons.
- Based on what it sees, it decides what it wants to say.
- Neurons learn to cooperate to accomplish the task.

Source: Source: Large-Scale Deep Learning With TensorFlow - Jeff Dean - Google Brain Team - 2016

How it works?

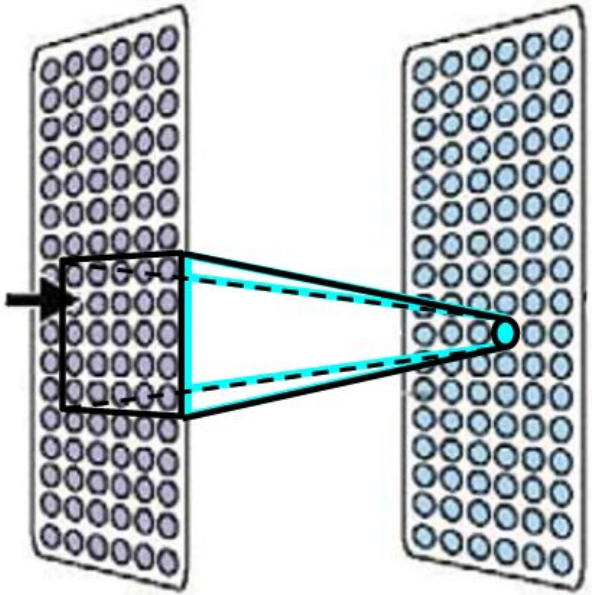


Each neuron implements a relatively simple mathematical function.

$$y = g(\vec{w} \cdot \vec{x} + b)$$

Source: Source: Large-Scale Deep Learning With TensorFlow - Jeff Dean - Google Brain Team - 2016

How it works?



Each neuron implements a relatively simple mathematical function.

$$y = g(\vec{w} \cdot \vec{x} + b)$$

But the composition of 10^6 - 10^9 such functions is surprisingly powerful.

Source: Source: Large-Scale Deep Learning With TensorFlow - Jeff Dean - Google Brain Team - 2016

Why Deep Learning?

- Conventional Machine Learning techniques were limited in their ability to process natural data in their raw form
    - e.g. the hard part is identifying the features in the raw input data, for example SIFT or SURF in images. Deep learning removes that manual step. You still have to make choices about the internal layout of the networks before you start training, but the automatic feature discovery makes life a lot easier.
    - In other ways, too, neural networks are more general than most other machine-learning techniques. Deep Learning 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. In this course we will introduce **Convolutional Neural Nets** and **Recurrent Neural Networks in this course**

Why Deep learning is taking off now?

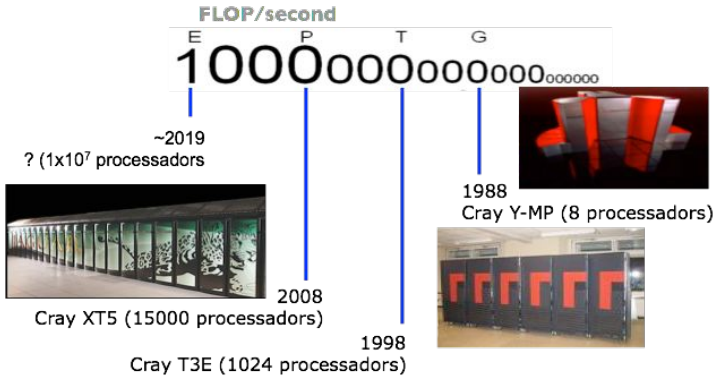
- It is know that many of the ideas used in Deep Learning have been around for decades
- One of the key drivers of its current progress is clearly the **huge deluge of data available today**. Thanks to the advent of Big Data these models can be “trained” by exposing them to large data sets that were previously unavailable

Source: Cray Data Analytics Infographics

Why Deep learning is 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

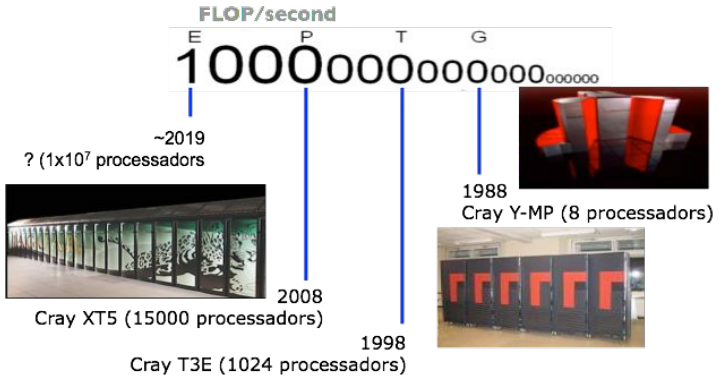
Source: Cray Data Analytics Infographics



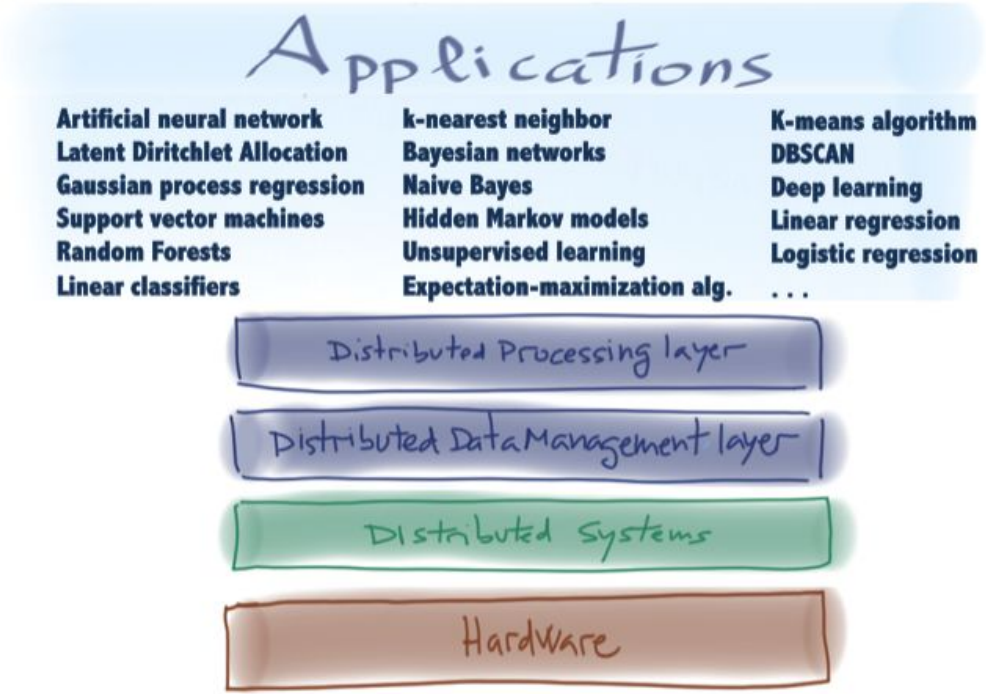
Why Deep learning is taking off now?

- 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 and will work in any new hardware technology that that may appear)

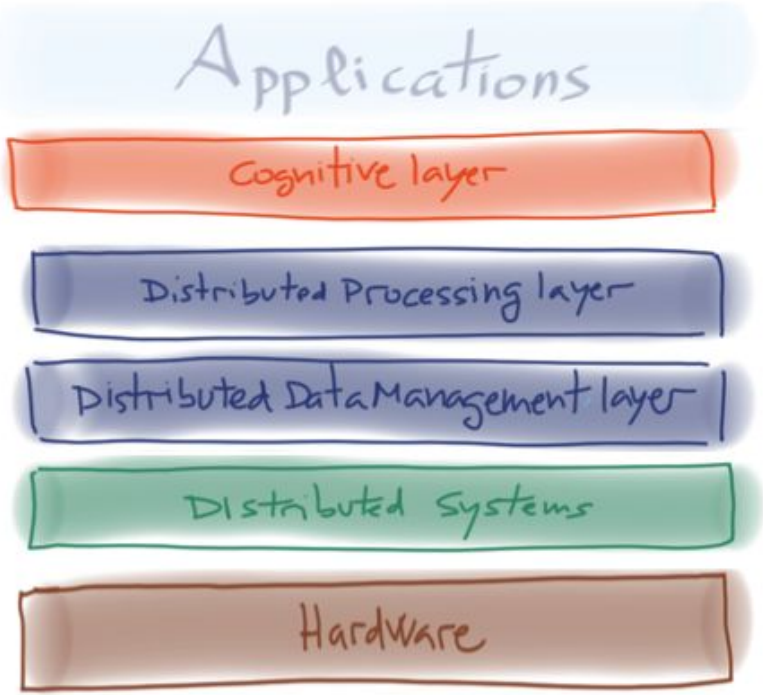
Source: Cray Data Analytics Infographics



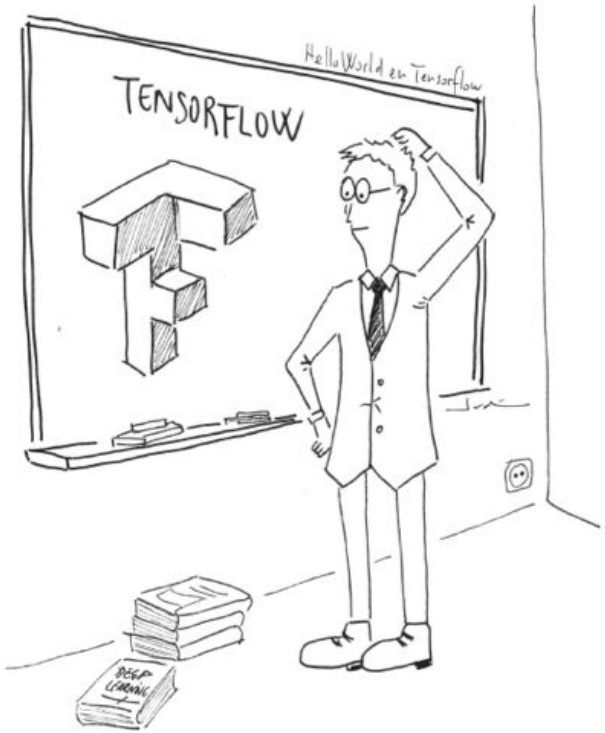
Today system software stacks



Future: Advanced Analytics facilities will be provided by the “system” software stack



What’s TensorFlow?



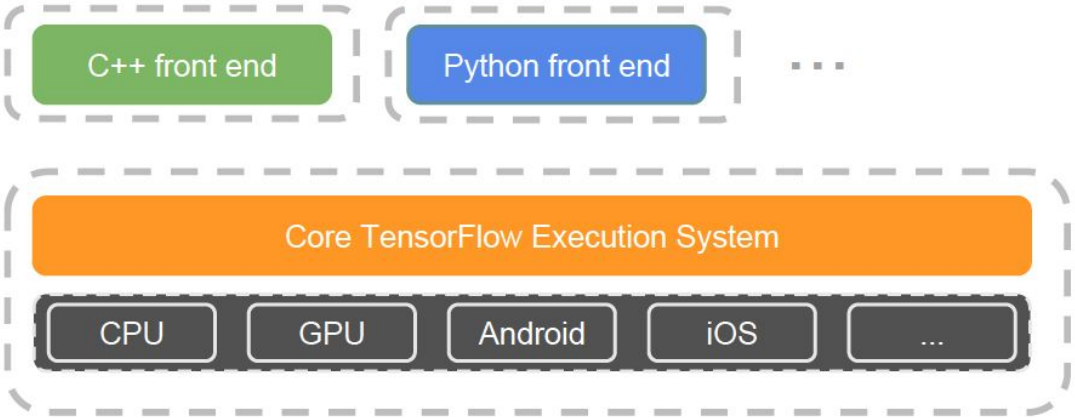
About TensorFlow

- **Open software (Apache 2.0 license) for general Machine Learning. Great for Deep Learning in particular**
- 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 to deploy computation to CPUs or GPUs in a desktop, server, or mobile device with a single API

TensorFlow software stack

Core in C++ : Very low overhead

Different front ends for specifying/driving the computation: Python and C++

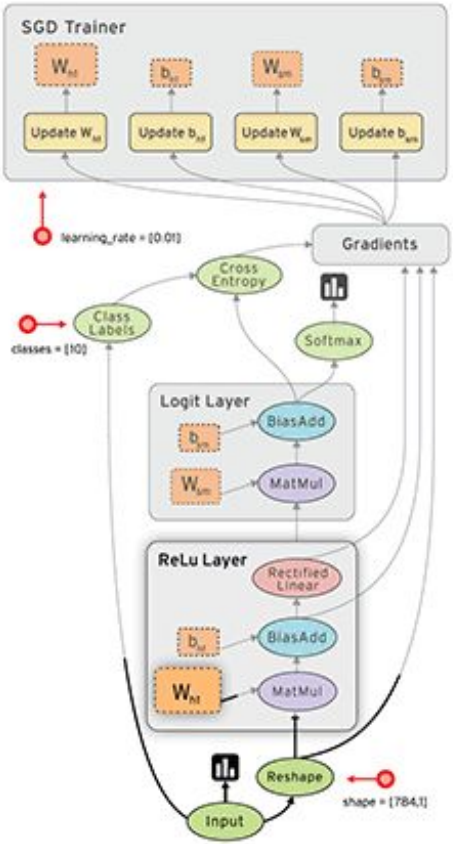


TensorFlow basics

- Operates over TENSORS (n-dimensional data arrays)
- Data flow computation framework
  - Automatic differentiation
  - Support for threads, queues and asynchronous computation
  - Train on CPU, GPUs, (an coming “soon” TPUS)
- Use a Flow Graph that 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
- Nodes are assigned to computational devices and execute asynchronously and in parallel once all the tensors on their incoming edges becomes available

Core TensorFlow concepts

- **Graph**: A TensorFlow computation, represented as a dataflow graph
- **Operation**: a graph node that performs computation on tensors
- **Tensor**: a handle to one of the outputs of an operation



Tensor: Core TensorFlow data structures

- **Constants**
- **Variables:** a modificable tensor that lives in TensorFlow's graph of interacting operations
- **Placeholders**: “symbolic variables”, must be fed with data on execution
- **Session:** encapsulates the environment in which operation objects are executed, and Tensor objects are evaluated

Math operations for manipulating tensors

Operation	Description
tf.add	sum
tf.sub	subtraction
tf.mul	multiplication
tf.div	division
tf.mod	module
tf.abs	return the absolute value
tf.neg	return negative value
tf.matmul(a, b)	multiplies matrix a by matrix b

Math operations for manipulating tensors (cont.)

Operation	Description
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

Operations & kernels

- An operation has a name and represents an abstract computation (e.g., “matrix multiply”, “add”, etc.)
- 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

Operation groups	Operations Examples
Maths	Add, Sub, Mul, Dic, Exp, Log, Greater than, Less than, Equal
Array	Concat, Slice, Constant, Rank, Shape
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

Creating and running your first TensorFlow graph

```
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 at https://github.com/jorditorresBCN/TutorialTensorFlow

Case study: Linear regression

- Linear regression
  - is a statistical technique used to measure the relationship between variables
  - It is an approach for modeling the relationship between a scalar dependent variable ‘y’ and one or more explanatory variables (or independent variables) denoted ‘x’.”
  - A simple linear regression can be:

$$y = W * x + b$$

- Case study: $y = 0.1 * x + 0.3$

Case study: Linear regression

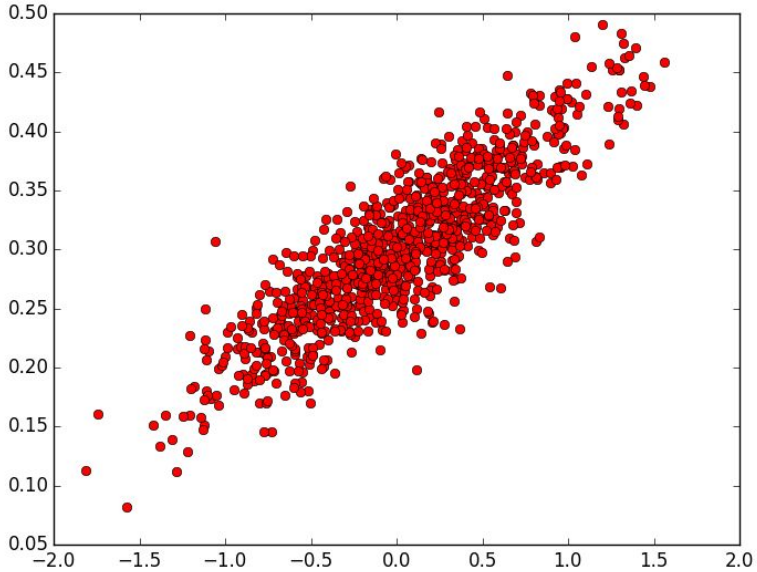
- For this case study we suggest to use a simple Python program that creates data in two-dimensional space

```
import numpy as np

num_puntos = 100
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 at https://github.com/jorditorresBCN/TutorialTensorFlow

Case study: Linear regression

```
import matplotlib.pyplot as plt

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



Case study: Linear regression

We will ask TensorFlow to look for the line that best fits these points!

(we know that **W** should be close to 0.1 and **b** to 0.3, but TensorFlow does not)

- Using Python API for Machine Learning:

```
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
```

Case study: Linear regression

- We will define a cost function: e.g. mean squared error:

```
loss = tf.reduce_mean(tf.square(y - y_data))


optimizer = tf.train.GradientDescentOptimizer(0.5)
train = optimizer.minimize(loss)
```

- TensorFlow automatically adds ops to calculate symbolic gradient of variables w.r.t. loss function and apply these gradients with an optimization algorithm

Case study: Linear regression

- TensorFlow launches the graph and run the training ops in a loop

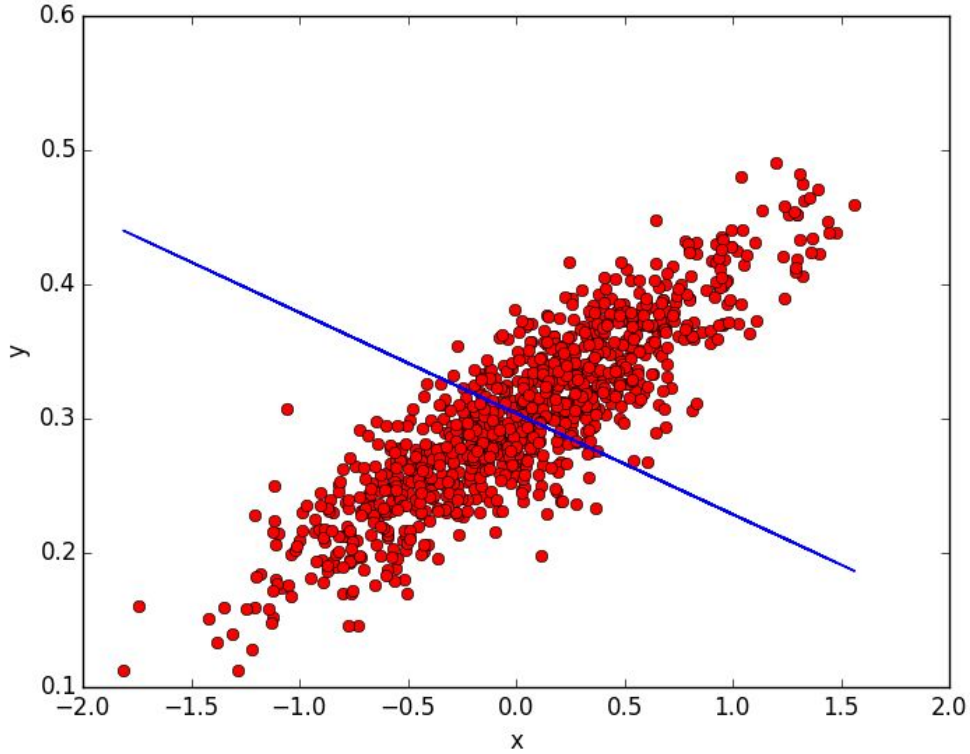
```
init = tf.initialize_all_variables()

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

for step in xrange(8):
    sess.run(train)
```

Case study: Linear regression

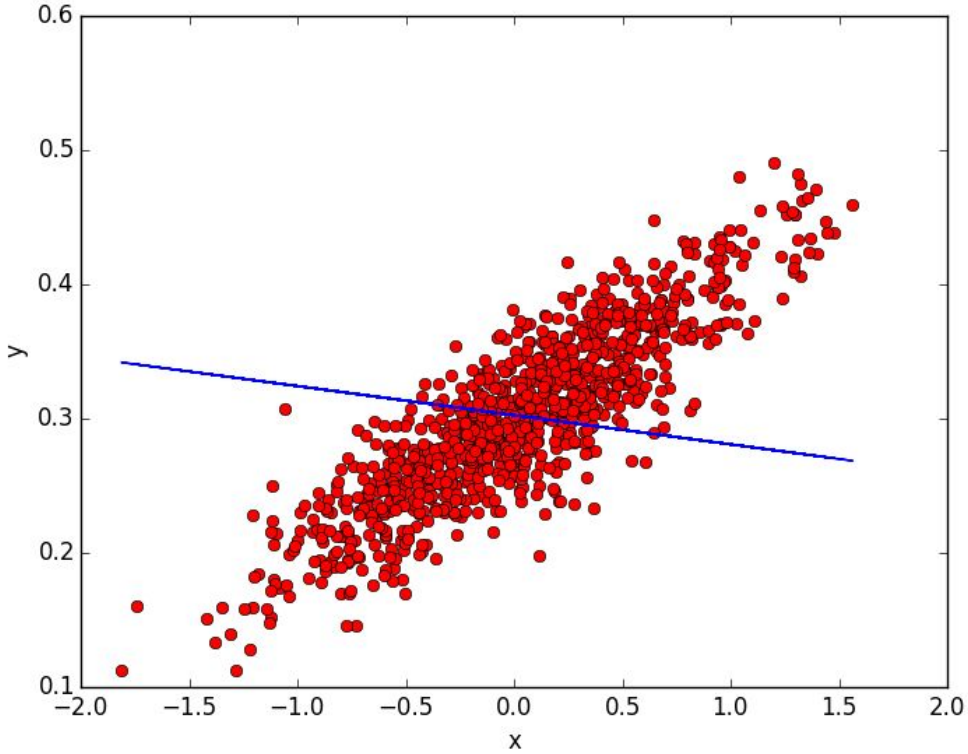
- Summary:

```
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(8):
     sess.run(train)
```

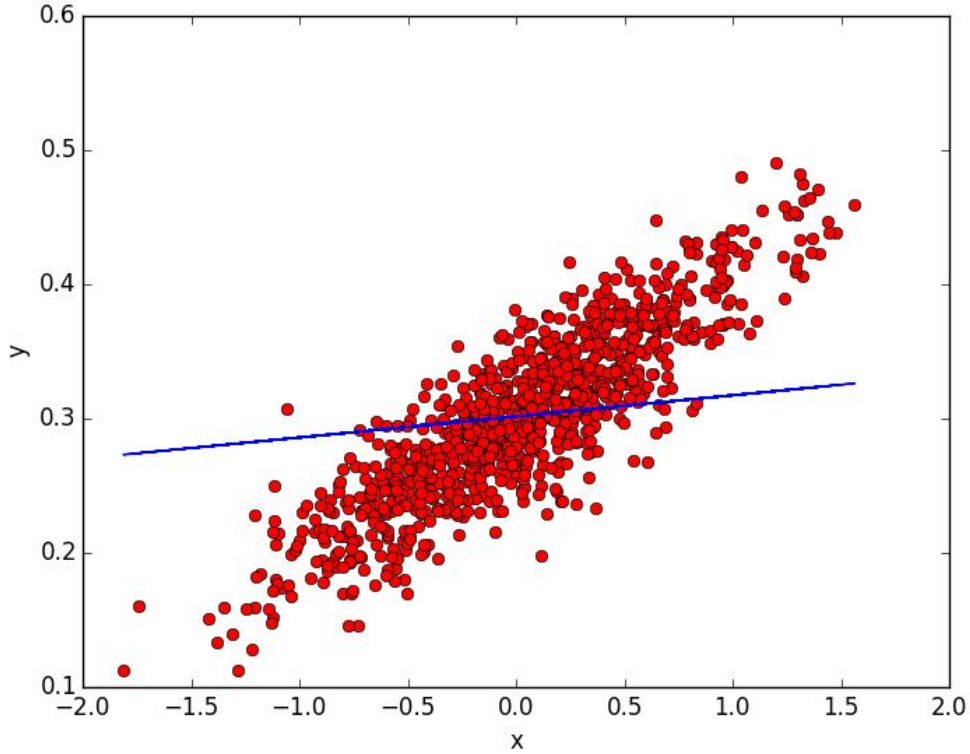
Case study: Linear regression



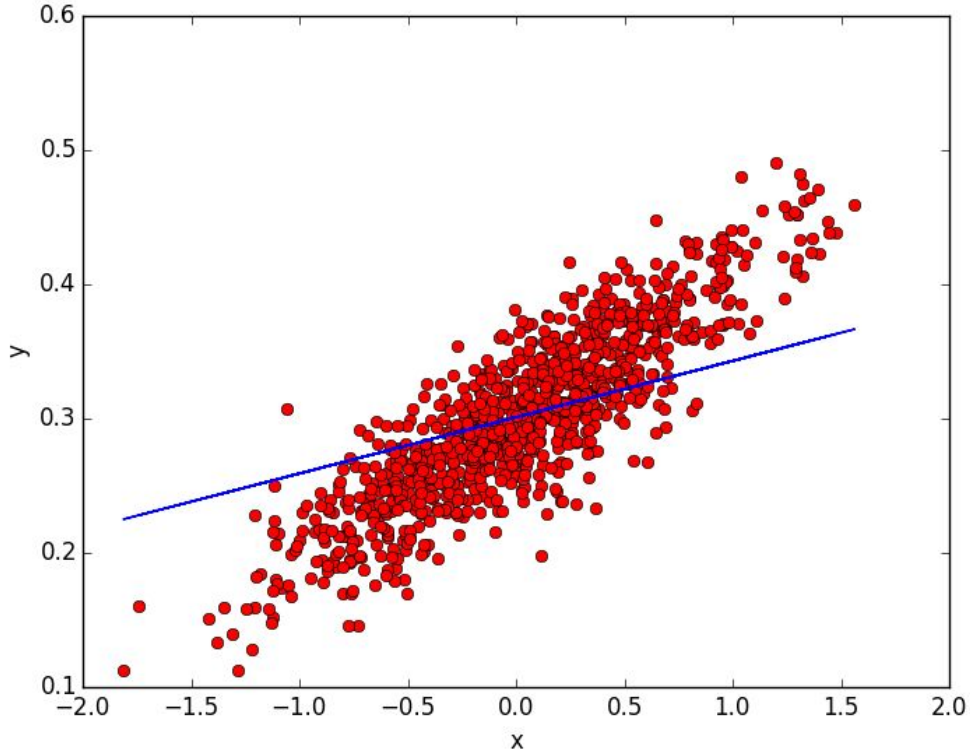
Case study: Linear regression



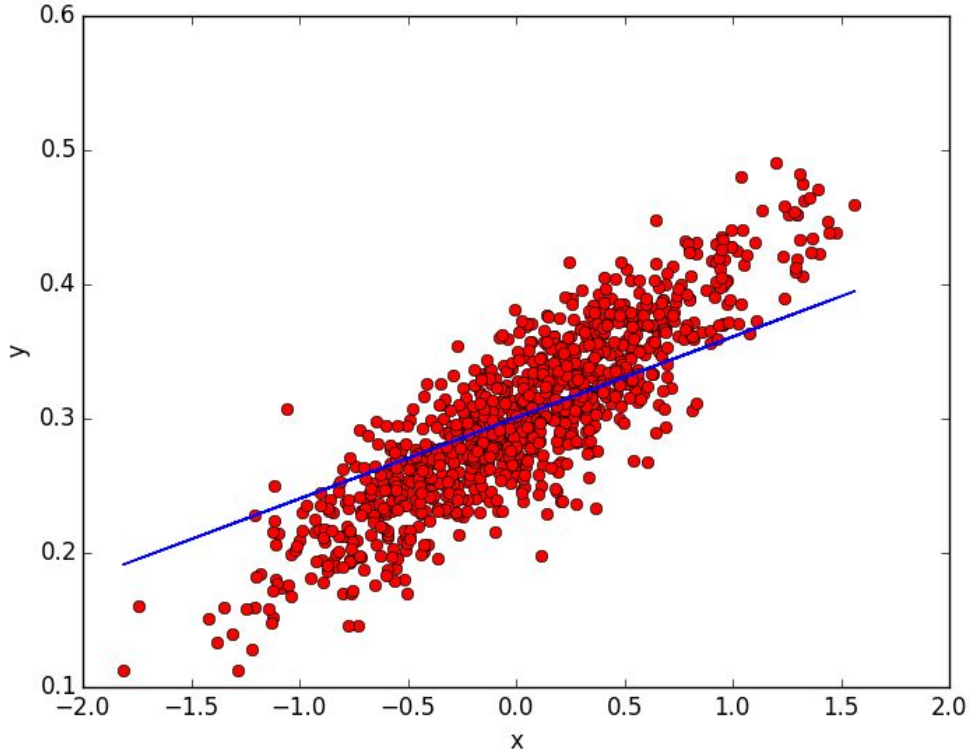
Case study: Linear regression



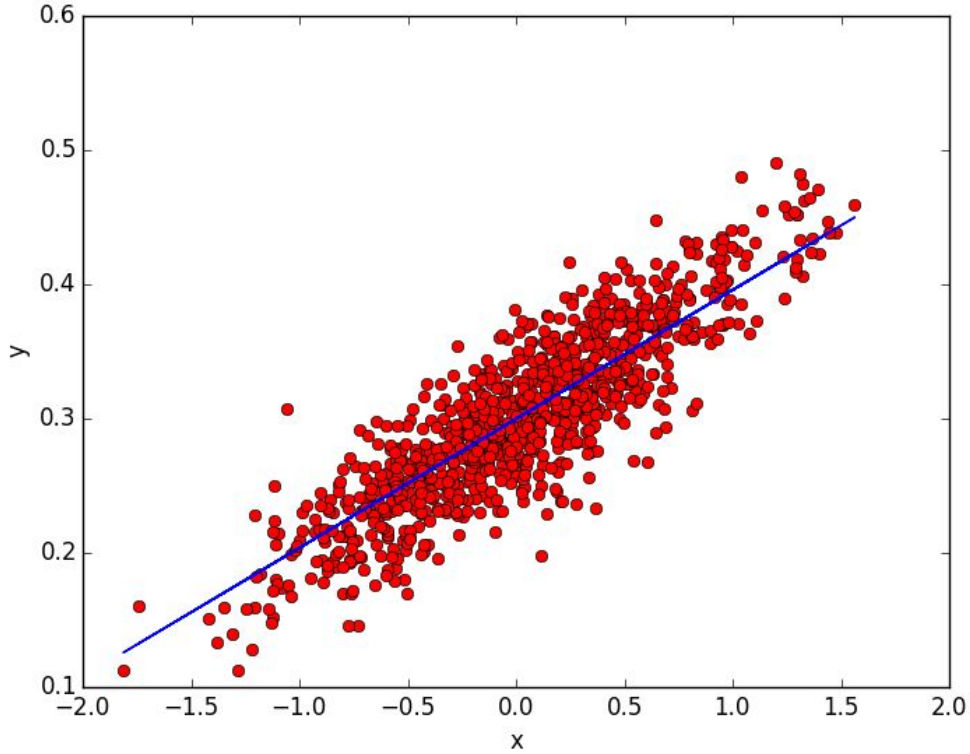
Case study: Linear regression



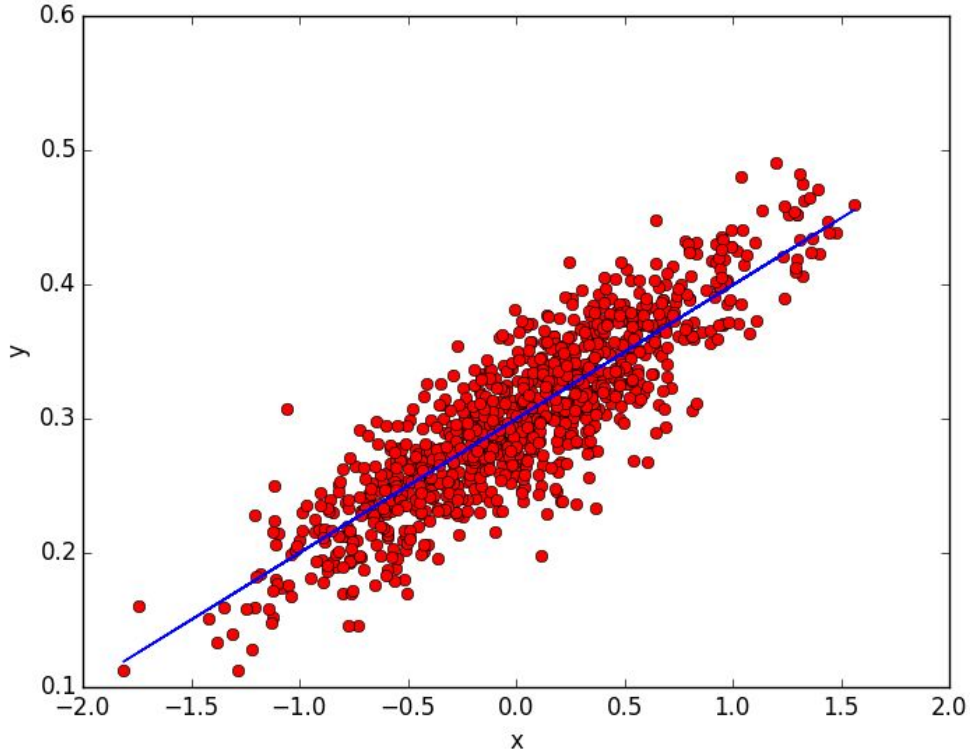
Case study: Linear regression



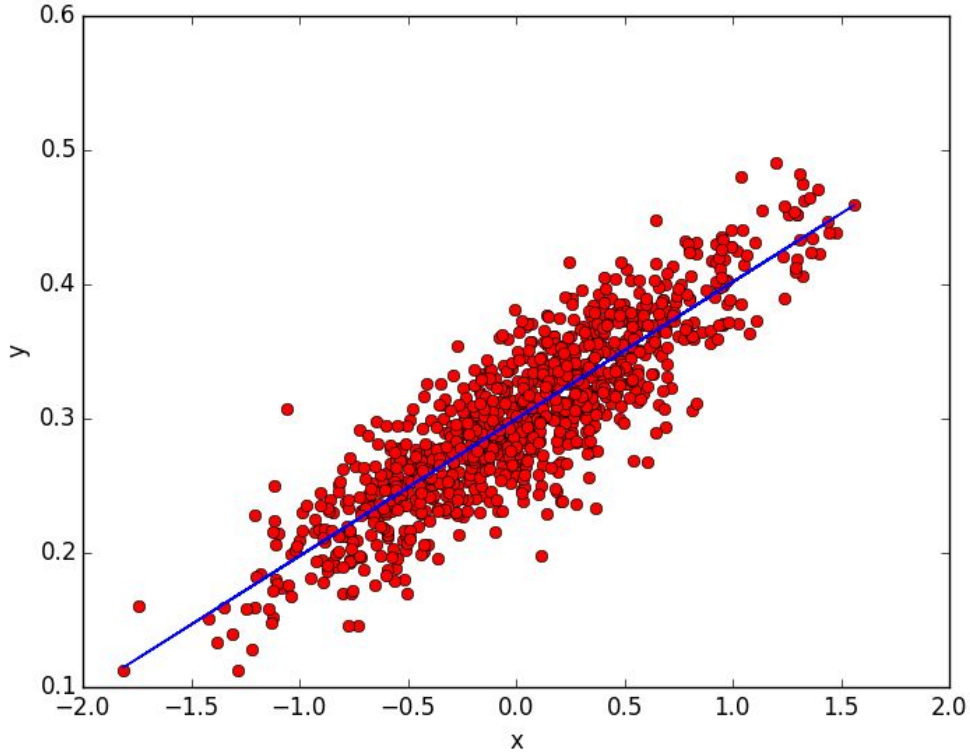
Case study: Linear regression



Case study: Linear regression



Case study: Linear regression



Case study: Linear regression

```
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))
```

Homework

- Reproduce the results (download the code from github) assuming $y = 0.3 * x + 0.9$.

Next session:

- We will review the basic data structure *tensor* from a code example in TensorFlow that implements a clustering algorithm *K-means.*