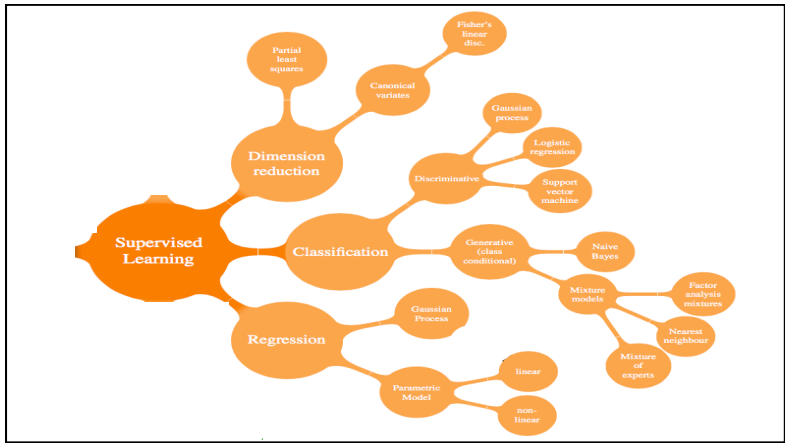
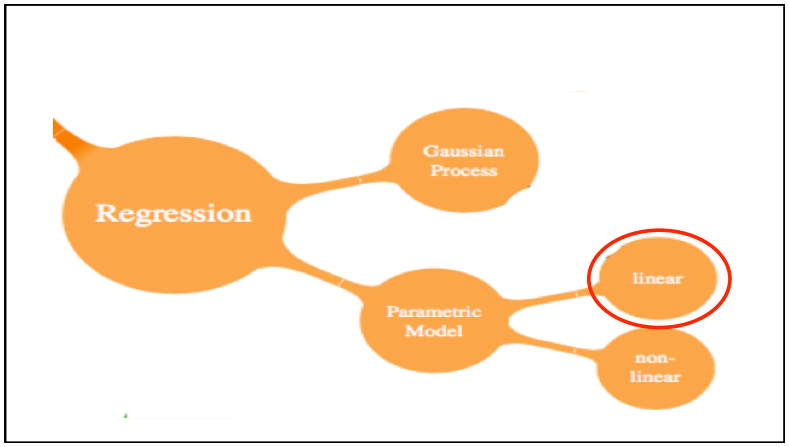




Linear Regression

The data set

Credit officers decide on credit lines:

$$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_N, y_N)$$

$y_n \in \mathbb{R}$ is the credit line for customer $\mathbf{x}_n$.

Linear regression tries to replicate that.

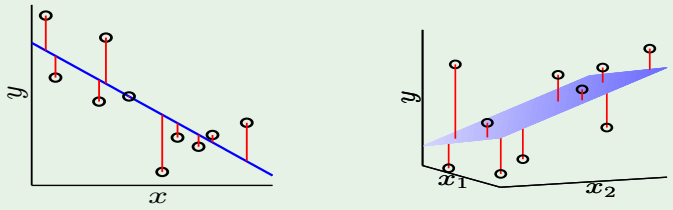
How to measure the error

How well does $h(\mathbf{x}) = \mathbf{w}^{\mathsf{T}}\mathbf{x}$ approximate $f(\mathbf{x})$?

In linear regression, we use squared error $(h(\mathbf{x}) - f(\mathbf{x}))^2$

in-sample error: $E_{\text{in}}(h) = \frac{1}{N}\sum_{n=1}^{N}(h(\mathbf{x}_n) - y_n)^2$

Illustration of linear regression



The expression for E_{in}

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{w}^{\mathsf{T}}\mathbf{x}_n - y_n)^2 = \frac{1}{N}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2$$

where $\mathbf{X} = \begin{bmatrix} —\mathbf{x}_1^{\mathsf{T}}— \\ —\mathbf{x}_2^{\mathsf{T}}— \\ \vdots \\ —\mathbf{x}_N^{\mathsf{T}}— \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$

Minimizing E_{in}

$$E_{\text{in}}(\mathbf{w}) = \tfrac{1}{N}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2$$

$$\nabla E_{\text{in}}(\mathbf{w}) = \tfrac{2}{N}\mathbf{X}^{\mathsf{T}}(\mathbf{X}\mathbf{w} - \mathbf{y}) = \mathbf{0}$$

$$\mathbf{X}^{\mathsf{T}}\mathbf{X}\mathbf{w} = \mathbf{X}^{\mathsf{T}}\mathbf{y}$$

$$\mathbf{w} = \mathbf{X}^{\dagger}\mathbf{y} \quad \text{where} \quad \mathbf{X}^{\dagger} = (\mathbf{X}^{\mathsf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathsf{T}}$$

$\mathbf{X}^{\dagger}$ is the '**pseudo-inverse**' of $\mathbf{X}$

The pseudo-inverse

$$\mathrm{X}^\dagger = (\mathrm{X}^\mathrm{T}\mathrm{X})^{-1}\mathrm{X}^\mathrm{T}$$

$$\underbrace{\left(\underbrace{\begin{bmatrix}\quad\end{bmatrix}}_{d+1\ \times\ d+1}\right)^{-1}}_{d+1\ \times\ N}\underbrace{\begin{bmatrix}\quad\end{bmatrix}}_{d+1\ \times\ N}$$

The linear regression algorithm

1: Construct the matrix X and the vector $\mathbf{y}$ from the data set $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)$ as follows

$$\underbrace{\mathrm{X} = \begin{bmatrix} —\mathbf{x}_1^\mathrm{T}— \\ —\mathbf{x}_2^\mathrm{T}— \\ \vdots \\ —\mathbf{x}_N^\mathrm{T}— \end{bmatrix}}_{\text{input data matrix}}, \qquad \underbrace{\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}}_{\text{target vector}}.$$

2: Compute the pseudo-inverse $\mathrm{X}^\dagger = (\mathrm{X}^\mathrm{T}\mathrm{X})^{-1}\mathrm{X}^\mathrm{T}$.

3: Return $\mathbf{w} = \mathrm{X}^\dagger\mathbf{y}$.

Regression

Regression Model	Pros	Cons
Linear Regression	Works on any size of dataset, gives informations about relevance of features	The Linear Regression Assumptions
Polynomial Regression	Works on any size of dataset, works very well on non linear problems	Need to choose the right polynomial degree for a good bias/variance tradeoff
SVR	Easily adaptable, works very well on non linear problems, not biased by outliers	Compulsory to apply feature scaling, not well known, more difficult to understand
Decision Tree Regression	Interpretability, no need for feature scaling, works on both linear / nonlinear problems	Poor results on too small datasets, overfitting can easily occur
Random Forest Regression	Powerful and accurate, good performance on many problems, including non linear	No interpretability, overfitting can easily occur, need to choose the number of trees

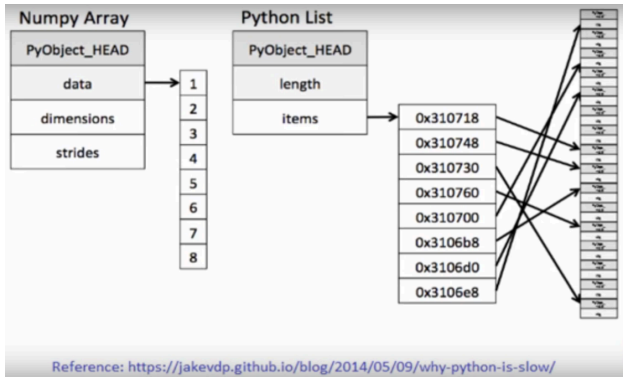
Machine Learning A-Z — © SuperDataScience

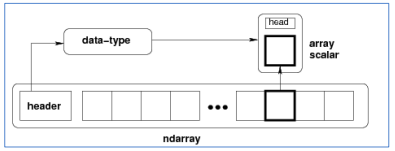
Deep Learning A-Z™: Online Course in Artificial Intelligence

Kirill Eremenko

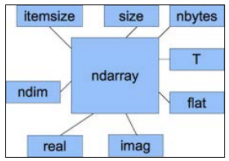
Numpy

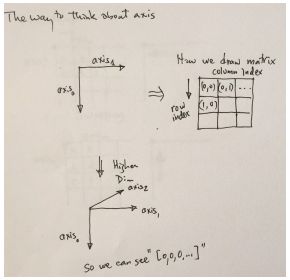
(stack, concatenate)

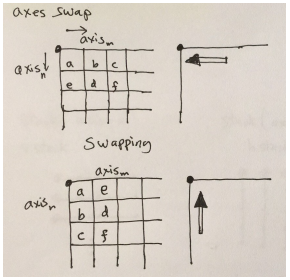


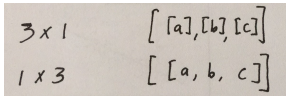


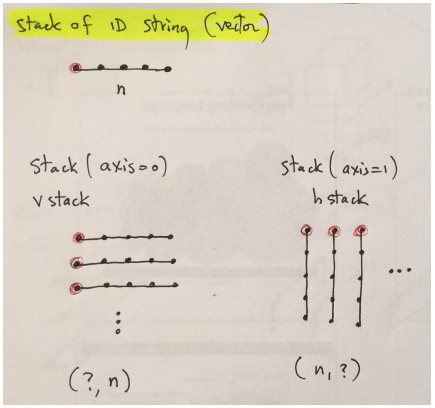
attributes of ndarray:

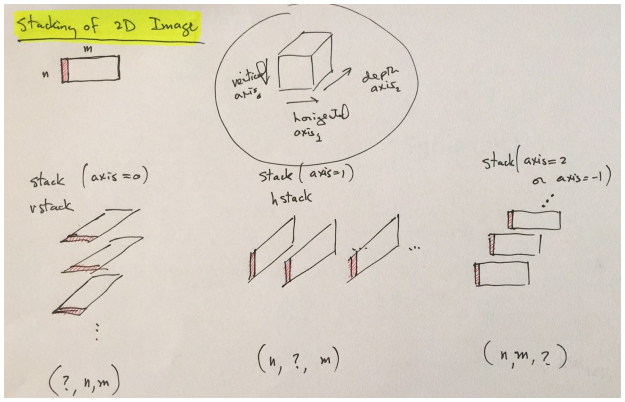












TensorFlow

(Which platform)

Popular Python Libraries

Numpy/Scipy
(numerical and scientific computing)

Pandas
(different data format)

Matplotlib/Pandas/Seaborn/Plotly
(visualization: base, dataframe, stat, interactive)

SciKit-Learn
(basic machine learning algorithms)

PySpark
(big data)

Installing Jupyter Notebook:

First, run the following commands to install iPython, an incredibly useful interactive Python kernel, and the backbone of the Jupyter Notebook. We highly recommend installing both the Python 2 and Python 3 kernels below, as it will give you more options moving forward (i.e., run all of the following!):

```
# Python 2.7
$ sudo python2 -m pip install ipykernel
$ sudo python2 -m ipykernel install
# Python 3
$ sudo python3 -m pip install jupyterhub notebook ipykernel
$ sudo python3 -m ipykernel install
```

After that, two simple commands should get you going. First install the `build-essential` dependency:

```
$ sudo apt-get install build-essential
```

Then use `pip` to install the Jupyter Notebook (`pip3` if you are using Python 3):

```
# For Python 2.7
$ sudo pip install jupyter
# For Python 3
$ sudo pip3 install jupyter
```

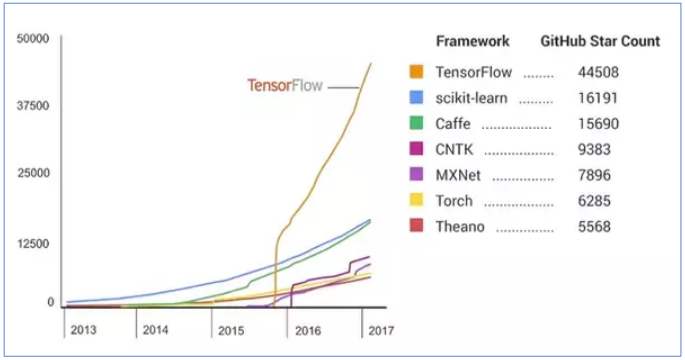
Official installation instructions are available at the Jupyter website:

http://jupyter.readthedocs.io/en/latest/install.html

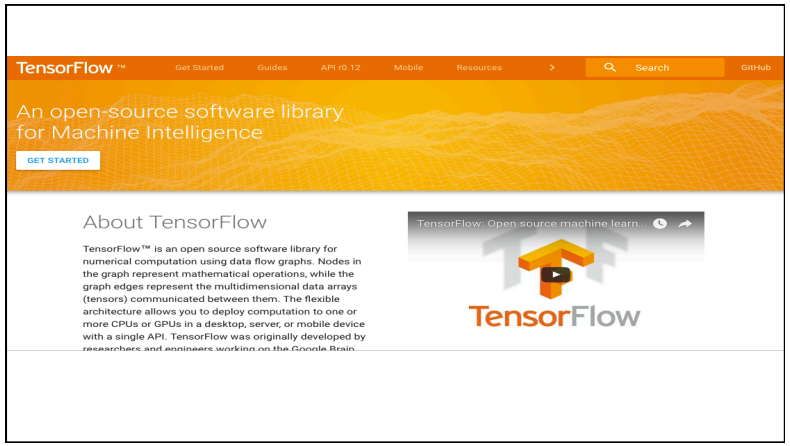
Caffe	Berkeley	Py, C++	L/M/W
DL4j	Skymind	Java	L/M/W Mo
CNTK	Microsoft	Py, C++	L/M
PaddlePaddle	Baidu	Py, C++	L/M
Theano	Montreal	Py	L/M/W
Torch	FB/GL/TW	Lua, C	L/M/W Mo
MxNet	D(ist)MLC	Py, C++, Matlib, R	L/M/W Mo
Tensorflow	**Google**	**Py, C++,**	**L/M/W/Mo**

Table 9-1. Open source Deep Learning libraries (not an exhaustive list)

Library	API	Platforms	Started by	Year
Caffe	Python, C++, Matlab	Linux, macOS, Windows	Y. Jia, UC Berkeley (BVLC)	2013
Deeplearning4j	Java, Scala, Clojure	Linux, macOS, Windows, Android	A. Gibson, J.Patterson	2014
H2O	Python, R	Linux, macOS, Windows	H2O.ai	2014
MXNet	Python, C++, others	Linux, macOS, Windows, iOS, Android	DMLC	2015
TensorFlow	Python, C++	Linux, macOS, Windows, iOS, Android	Google	2015
Theano	Python	Linux, macOS, iOS	University of Montreal	2010
Torch	C++, Lua	Linux, macOS, iOS, Android	R. Collobert, K. Kavukcuoglu, C. Farabet	2002



TensorFlow
(Installation)



About TensorFlow

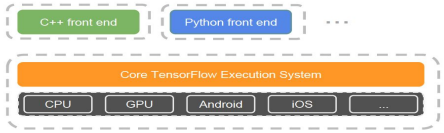
- **Open software (Apache 2.0 license) for general Machine Learning. Great for Deep Learning in particular**
- TensorFlow was originally developed by the Google Brain Team within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks 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

23

TensorFlow software stack

Core in C++ : Very low overhead

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



24

Overview

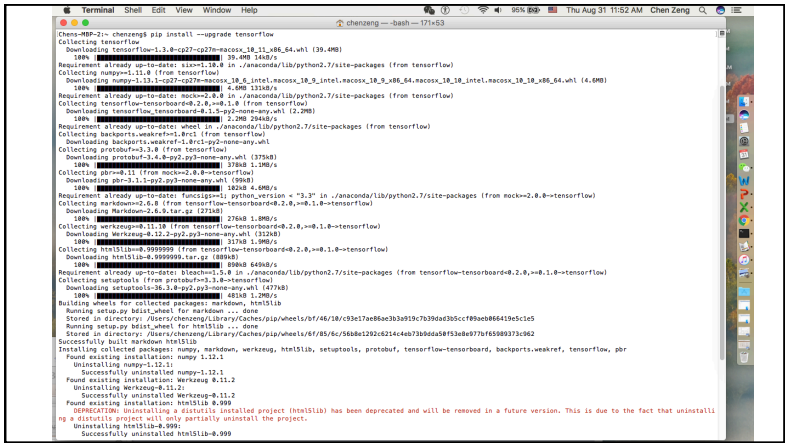
We support different ways to install TensorFlow:

- Pip install: Install TensorFlow on your machine, possibly upgrading previously installed Python packages. May impact existing Python programs on your machine.
- Virtualenv install: Install TensorFlow in its own directory, not impacting any existing Python programs on your machine.
- Anaconda install: Install TensorFlow in its own environment for those running the Anaconda Python distribution. Does not impact existing Python programs on your machine.
- Docker install: Run TensorFlow in a Docker container isolated from all other programs on your machine.
- Installing from sources: Install TensorFlow by building a pip wheel that you then install using pip.

If you are familiar with Pip, Virtualenv, Anaconda, or Docker, please feel free to adapt the instructions to your particular needs. The names of the pip and Docker images are listed in the corresponding installation sections.

If you encounter installation errors, see common problems for some solutions.

```
Chens-MBP-2:~ chenzeng$ pip install tensorflow
Collecting tensorflow
  Downloading tensorflow-0.12.1-cp27-cp27m-macosx_10_11_x86_64.whl (38.8MB)
    100% |████████████████████████████████| 38.8MB 15kB/s
Collecting mock>=2.0.0 (from tensorflow)
  Downloading mock-2.0.0-py2.py3-none-any.whl (56kB)
    100% |████████████████████████████████| 61kB 4.4MB/s
Requirement already satisfied (use --upgrade to upgrade): six>=1.10.0 in ./anaconda/lib/python2.7/site-package
s (from tensorflow)
Collecting numpy>=1.11.0 (from tensorflow)
  Downloading numpy-1.12.0-cp27-cp27m-macosx_10_6_intel.macosx_10_9_intel.macosx_10_9_x86_64.macosx_10_10_inte
l.macosx_10_10_x86_64.whl (4.4MB)
    100% |████████████████████████████████| 4.4MB 138kB/s
Collecting protobuf>=3.1.0 (from tensorflow)
  Downloading protobuf-3.1.0.post1-py2.py3-none-any.whl (347kB)
    100% |████████████████████████████████| 348kB 1.6MB/s
Requirement already satisfied (use --upgrade to upgrade): wheel in ./anaconda/lib/python2.7/site-packages (fro
m tensorflow)
Collecting funcsigs>=1; python_version < "3.3" (from mock>=2.0.0->tensorflow)
  Downloading funcsigs-1.0.2-py2.py3-none-any.whl
Collecting pbr>=0.11 (from mock>=2.0.0->tensorflow)
  Downloading pbr-1.10.0-py2.py3-none-any.whl (96kB)
    100% |████████████████████████████████| 102kB 5.1MB/s
Requirement already satisfied (use --upgrade to upgrade): setuptools in ./anaconda/lib/python2.7/site-packages
/setuptools-18.5-py2.7.egg (from protobuf>=3.1.0->tensorflow)
Installing collected packages: funcsigs, pbr, mock, numpy, protobuf, tensorflow
  Found existing installation: funcsigs 0.4
    Uninstalling funcsigs-0.4:
      Successfully uninstalled funcsigs-0.4
  Found existing installation: numpy 1.10.1
    DEPRECATION: Uninstalling a distutils installed project (numpy) has been deprecated and will be removed in
 a future version. This is due to the fact that uninstalling a distutils project will only partially uninstall
 the project.
    Uninstalling numpy-1.10.1:
      Successfully uninstalled numpy-1.10.1
Successfully installed funcsigs-1.0.2 mock-2.0.0 numpy-1.12.0 pbr-1.10.0 protobuf-3.1.0.post1 tensorflow-0.12.
1
You are using pip version 8.1.2, however version 9.0.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
Chens-MBP-2:~ chenzeng$
```




```
Found existing installation: numpy 1.12.1
  Uninstalling numpy-1.12.1:
    Successfully uninstalled numpy-1.12.1
Found existing installation: tensorflow 0.12.1
  Uninstalling tensorflow-0.12.1:
    Successfully uninstalled tensorflow-0.12.1
```

Where

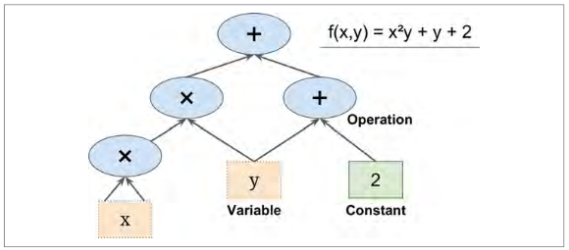
Original: /Users/chenzeng/anaconda/lib/python2.7/site-packages/tensorflow

What

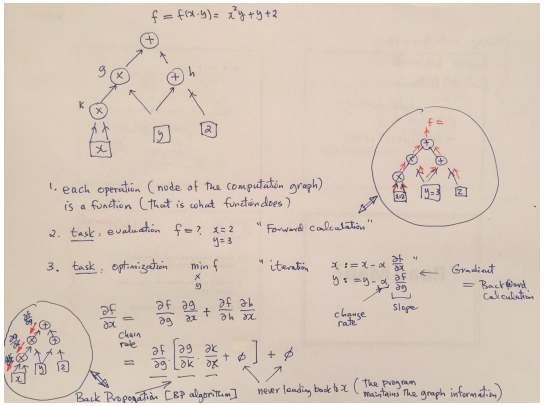
- __init__.py
- __init__.pyc
- contrib
- core
- examples
- include
- models
- python
- tensorboard
- tools

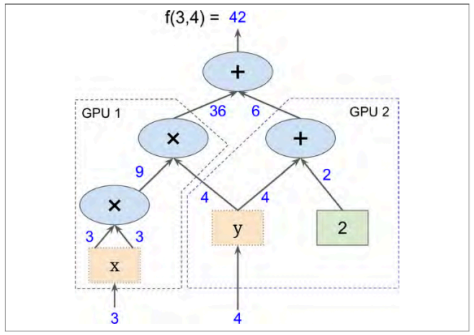
TensorFlow
(Way of Thinking)

Computational Graph



Hands-On Machine Learning with Scikit-Learn & TensorFlow





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

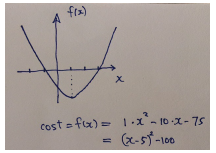
TensorFlow
(Case study)

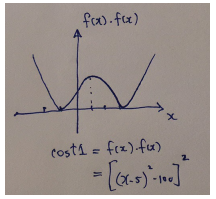
Example 1:
(A) Optimization
(B) Solve equation as optimization

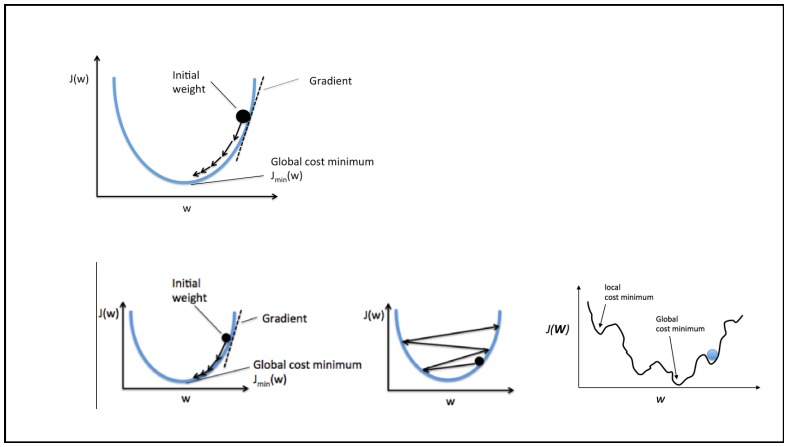
Tensor: Core TensorFlow data structures

- **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


```
In [1]: # solving a quadratic equation as optimization: min f^2 (so looking for
        # f=0 as the solution)

        import numpy as np
        import tensorflow as tf

        coes = np.array([[1], [-10], [-75]]) #coes.shape=(3,1)

        w = tf.Variable([7],dtype=tf.float32)
        x = tf.placeholder(tf.float32, (3,1))

        func = x[0][0]*w**2 + x[1][0]*w + x[2][0]  #(w-5)**2
        cost = tf.multiply(func, func)

        train = tf.train.GradientDescentOptimizer(0.001).minimize(cost) #0.001=learning rate

        init = tf.global_variables_initializer()
        sess = tf.Session()
        sess.run(init)
        print(sess.run(w))

        for i in range(1000):
            sess.run(train, feed_dict={x:coes})
        print(sess.run(w))

        [ 7.]
        [ 15.]
```

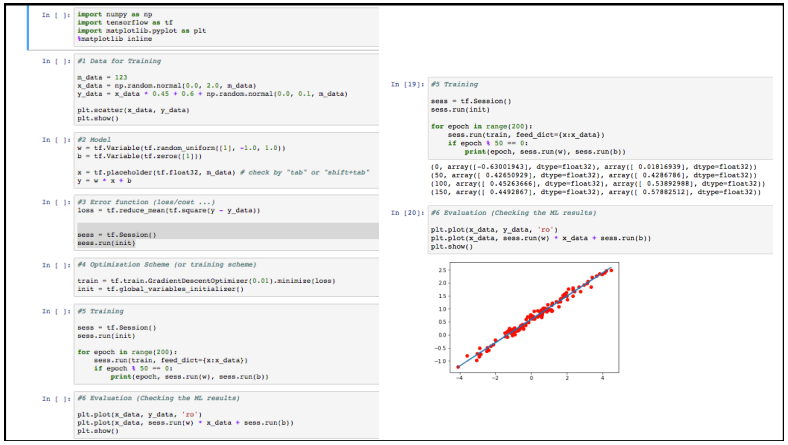






TensorFlow
(Case study)

Example 2:
(A) Linear Regression



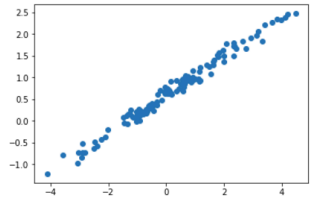
```
In [ ]: import numpy as np
        import tensorflow as tf
        import matplotlib.pyplot as plt
        %matplotlib inline

In [ ]: #1 Data for Training

        m_data = 123
        x_data = np.random.normal(0.0, 2.0, m_data)
        y_data = x_data * 0.45 + 0.6 + np.random.normal(0.0, 0.1, m_data)

        plt.scatter(x_data, y_data)
        plt.show()

In [ ]: #2 Model
        w = tf.Variable(tf.random_uniform([1], -1.0, 1.0))
        b = tf.Variable(tf.zeros([1]))

        x = tf.placeholder(tf.float32, m_data) # check by "tab" or "shift+tab"
        y = w * x + b

In [ ]: #3 Error function (loss/cost ...)
        loss = tf.reduce_mean(tf.square(y - y_data))

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

In [ ]: #4 Optimization Scheme (or training scheme)

        train = tf.train.GradientDescentOptimizer(0.01).minimize(loss)
        init = tf.global_variables_initializer()

In [ ]: #5 Training

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

        for epoch in range(200):
            sess.run(train, feed_dict={x:x_data})
            if epoch % 50 == 0:
                print(epoch, sess.run(w), sess.run(b))

In [ ]: #6 Evaluation (Checking the ML results)

        plt.plot(x_data, y_data, 'ro')
        plt.plot(x_data, sess.run(w) * x_data + sess.run(b))
        plt.show()
```

```
In [19]: #5 Training

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

         for epoch in range(200):
             sess.run(train, feed_dict={x:x_data})
             if epoch % 50 == 0:
                 print(epoch, sess.run(w), sess.run(b))

(0, array([-0.63001943], dtype=float32), array([ 0.01816939], dtype=float32))
(50, array([ 0.42650929], dtype=float32), array([ 0.4286786], dtype=float32))
(100, array([ 0.45263666], dtype=float32), array([ 0.53892988], dtype=float32))
(150, array([ 0.4492867], dtype=float32), array([ 0.57882512], dtype=float32))

In [20]: #6 Evaluation (Checking the ML results)

         plt.plot(x_data, y_data, 'ro')
         plt.plot(x_data, sess.run(w) * x_data + sess.run(b))
         plt.show()
```



```
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
%matplotlib inline
```

```
#1 Data for Training

m_data = 123
x_data = np.random.normal(0.0, 2.0, m_data)
y_data = x_data * 0.45 + 0.6 + np.random.normal(0.0, 0.1, m_data)

plt.scatter(x_data, y_data)
plt.show()
```




```
#2 Model
w = tf.Variable(tf.random_uniform([1], -1.0, 1.0))
b = tf.Variable(tf.zeros([1]))

x = tf.placeholder(tf.float32, m_data) # check by "tab" or "shift+tab"
y = w * x + b
```

```
#3 Error function (loss/cost ...)
loss = tf.reduce_mean(tf.square(y - y_data))

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

```
#4 Optimization Scheme (or training scheme)

train = tf.train.GradientDescentOptimizer(0.01).minimize(loss)
init = tf.global_variables_initializer()
```

```
#5 Training

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

for epoch in range(200):
    sess.run(train, feed_dict={x:x_data})
    if epoch % 50 == 0:
        print(epoch, sess.run(w), sess.run(b))
```

```
(0, array([-0.63001943], dtype=float32), array([ 0.01816939], dtype=float32))
(50, array([ 0.42650929], dtype=float32), array([ 0.4286786], dtype=float32))
(100, array([ 0.45263666], dtype=float32), array([ 0.53892988], dtype=float32))
(150, array([ 0.4492867], dtype=float32), array([ 0.57882512], dtype=float32))
```

```
#6 Evaluation (Checking the ML results)

plt.plot(x_data, y_data, 'ro')
plt.plot(x_data, sess.run(w) * x_data + sess.run(b))
plt.show()
```