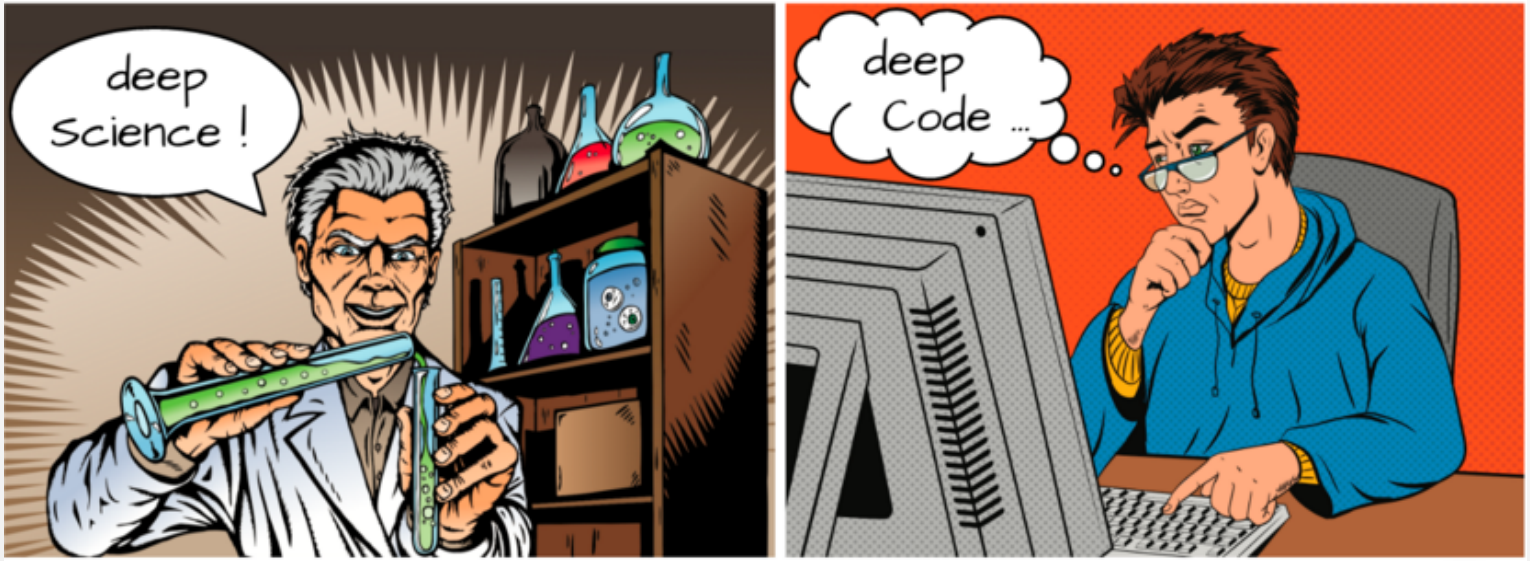
Tensorflow Tutorial

In this tutorial we will learn:

- XOR Example
- Examples using MNIST



XOR

x_1	x_2	x_1 **XOR** x_2
0	0	0
0	1	1
1	0	1
1	1	0

Functions

$$\hat{y} = \sigma(w^T max\{0, W^T x + c\} + b)$$

x is the input matrix with size 4 x 2.
W is the weight matrix of 2 x 2.
c is the bias matrix of 4 x 2.
b is the bias vector of 4 x 1
w is the weight matrix of 2 x 1.
y is output matrix of 4 x 1.

Placeholders

- placeholder – a value that we'll input when we ask TensorFlow to run a computation.

- X = tf.placeholder(tf.float32, shape=[4,2], name = 'X')
- Y = tf.placeholder(tf.float32, shape=[4,1], name = 'Y')

Variables

- A TensorFlow variable is the best way to represent shared, persistent state manipulated by your program.
- We will fill the weight matrices with random values distibuted normally.

- W = tf.Variable(tf.truncated_normal([2,2]), name = "W")
- w = tf.Variable(tf.truncated_normal([2,1]), name = "w")

Variables

- We will fill the bias matrices with zeroes.

- c = tf.Variable(tf.zeros([4,2]), name = "c")
- b = tf.Variable(tf.zeros([4,1]), name = "b")

Name Scopes

- Name scopes behave similarly to functions.
- We will define the hidden layer h = relu(W.X + c). ReLU is the activation function with relu(x) = max{0,x}

- with tf.name_scope("hidden_layer") as scope:
- h = tf.nn.relu(tf.add(tf.matmul(X, W),c))

Name Scopes

- y = sigmoid(h.w + b)

- with tf.name_scope("output") as scope:
- y_estimated = tf.sigmoid(tf.add(tf.matmul(h,w),b))

Loss Function

- Loss function will be the mean of the squared difference.

- with tf.name_scope("loss") as scope:
- loss = tf.reduce_mean(tf.squared_difference(y_estimated, Y))

Training Function

- Make the function of training here.
- We are using gradient descent with a step size of 0.01.

- with tf.name_scope("train") as scope:
- 	train_step = tf.train.GradientDescentOptimizer(0.01).minimize(loss)

Initialize the variables

- Graph creation is done. Let's start initializing the variables.

- INPUT_XOR = [[0,0],[0,1],[1,0],[1,1]]
- OUTPUT_XOR = [[0],[1],[1],[0]]

Initialize the variables

- init = tf.global_variables_initializer()
- Whatever global variables that you have initialized will be returned here.

- sess = tf.Session()
- A session allows to execute graphs or part of graphs. It allocates resources for that and holds the actual values of intermediate results and variables.

Initialize the variables

- **writer = tf.summary.FileWriter("./logs/xor_logs", sess.graph)**
- Store your logs

- **sess.run(init)**
- Run your sessions by passing initial global variables.

- **t_start = time.clock()**
- Start the clock.

WE ARE IN THE FOR LOOP NOW

- Start the for loop.
- for epoch in range(100001):
  - sess.run(train_step, feed_dict={X: INPUT_XOR, Y: OUTPUT_XOR})

- We have passed into the sess.run() function's feed_dict parameter to provide the input examples for this step of training.

WE ARE IN THE FOR LOOP NOW

- We will print out the parameters at every 10000th step.
- if epoch % 10000 == 0:
  - for element in sess.run(y_estimated, feed_dict={X: INPUT_XOR, Y: OUTPUT_XOR}):
    - print('	',element)
- This will print out values of output at every 10000th step.

WE ARE IN THE FOR LOOP NOW

- Let's print W!
- **for element in sess.run(W):**
  - **print(' ',element)**
- Let's print c!
- **for element in sess.run(c):**
  - **print(' ',element)**
- Let's print loss!
- **print('   loss: ', sess.run(loss, feed_dict={X: INPUT_XOR, Y: OUTPUT_XOR}))**

It's done.

- End your for loops.
- End your clocks.
- You are done. Run your program now.
- Let's have a small break and we will see demos on how to use them for real datasets.