# Tensorflow Intro

•••

B03901048 EE3 Ching-Lun Tai

## Outline

## PART I
## Building NN

## Find toolkit? Tensorflow!

TensorFlow™

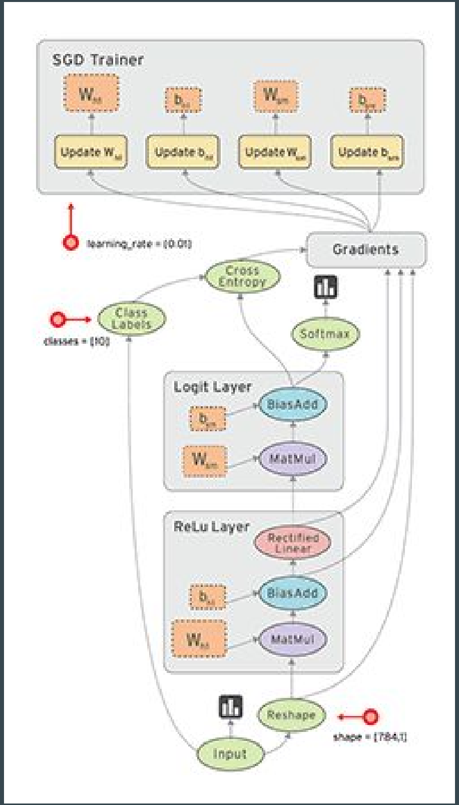
Q: What is Tensorflow?

A: A python-based toolkit for neural networks developed by Google.

Q: Advantages?

A: Open source, visualizable, and many metaframeworks for use!

# Processing structure



-data-flow graph based

(method of visualization will be discussed later)

-node: mathematical operation

-line: data between nodes, represented by tensors.

0-dim: scalars

1-dim: vectors

2↑-dim: matrices

## General Principle

To build a neural network with Tensorflow, you can follow the steps below:

1) Import the modules you need.

2) Define an add_layer function to construct layers.

3) Define the variables and initialize them.

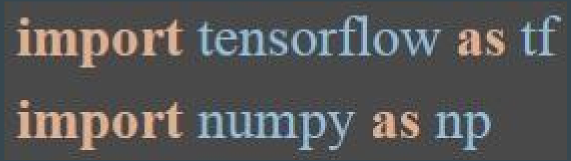
4) Create a session to perform the operation.

5) Build the NN, and send data with placeholders and feed_dict.

6) Train and test the NN, and observe the results with Tensorboard.

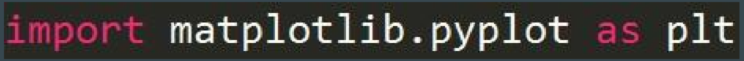
## Importing module

-Basic



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

-result visualization (will be discussed later)



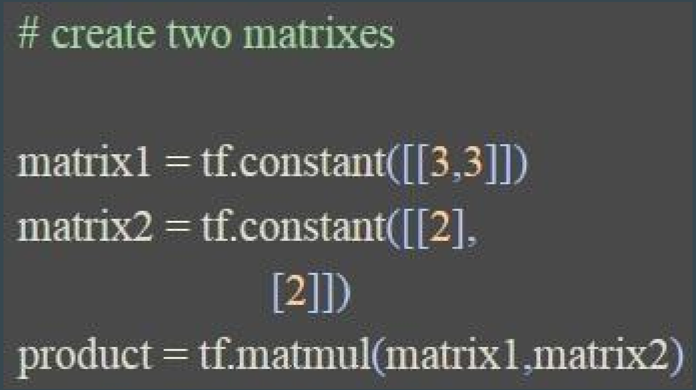
```
import matplotlib.pyplot as plt
```

## Session (1)

To control the operation of our program, we have to create a session.

<u>There are two ways to create a session.</u>

Ex: A simple matrix multiplication

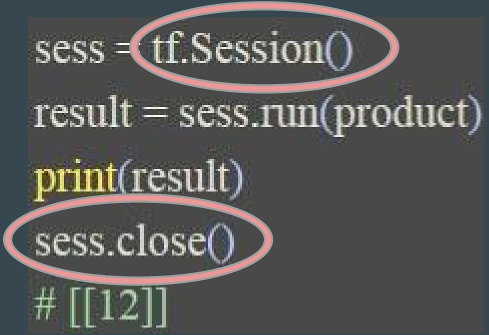


```
# create two matrixes

matrix1 = tf.constant([[3,3]])
matrix2 = tf.constant([[2],
                       [2]])
product = tf.matmul(matrix1,matrix2)
```

# Session (2)

-method 1: create a variable for a session

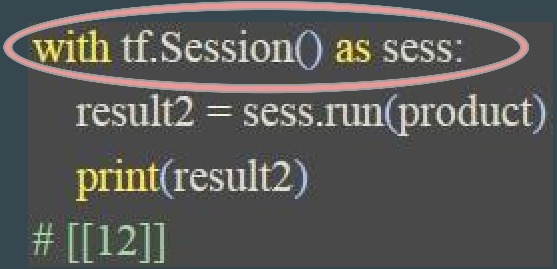


```
sess = tf.Session()
result = sess.run(product)
print(result)
sess.close()
# [[12]]
```

→Remember to capitalize the S!

→Remember to close the session if using this method!

-method 2: use with to create a session



```
with tf.Session() as sess:
    result2 = sess.run(product)
    print(result2)
# [[12]]
```

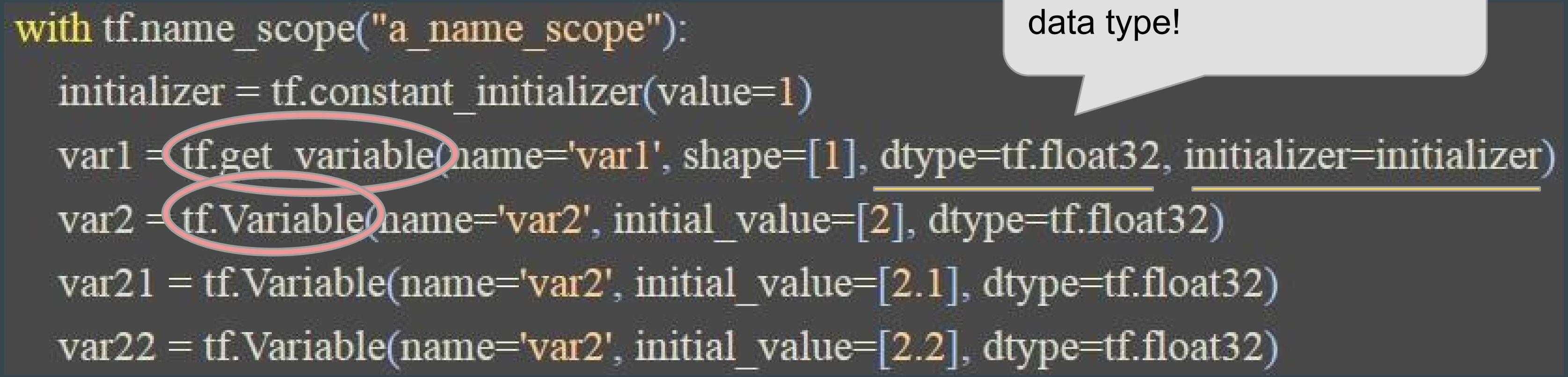
→If you choose to use with, you don’t have to close the session explicitly.

# Variables & Scope (1)

There are two kinds of scopes and two methods of creating variables.

I. tf.name_scope()

Remember to define the data type!

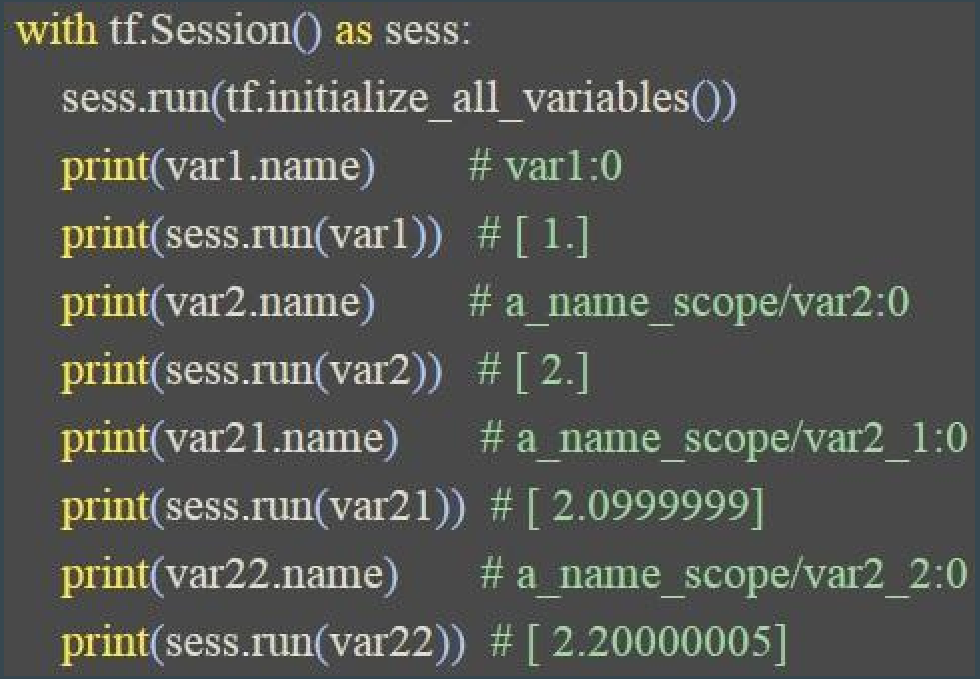


```
with tf.name_scope("a_name_scope"):
    initializer = tf.constant_initializer(value=1)
    var1 = tf.get_variable(name='var1', shape=[1], dtype=tf.float32, initializer=initializer)
    var2 = tf.Variable(name='var2', initial_value=[2], dtype=tf.float32)
    var21 = tf.Variable(name='var2', initial_value=[2.1], dtype=tf.float32)
    var22 = tf.Variable(name='var2', initial_value=[2.2], dtype=tf.float32)
```

Note: Tensorflow adopts the data type float32 only! (neglect few exceptions)

## Variables & Scope (2)

Result of I.



```
with tf.Session() as sess:
    sess.run(tf.initialize_all_variables())
    print(var1.name)        # var1:0
    print(sess.run(var1))   # [ 1.]
    print(var2.name)        # a_name_scope/var2:0
    print(sess.run(var2))   # [ 2.]
    print(var21.name)       # a_name_scope/var2_1:0
    print(sess.run(var21))  # [ 2.0999999]
    print(var22.name)       # a_name_scope/var2_2:0
    print(sess.run(var22))  # [ 2.20000005]
```

tf.get_variable→no effect!!

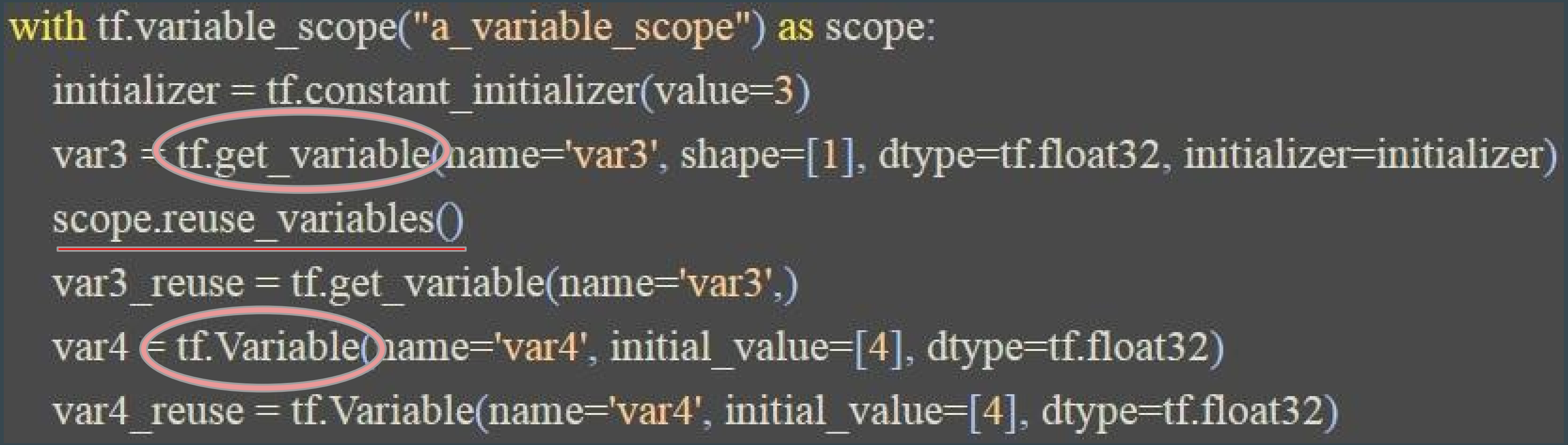
tf.Variable→names will change!!

## Variables & Scope (3)

II. tf.variable_scope()→to reuse the variables

If you want to reuse some variables, remember to add the following line:

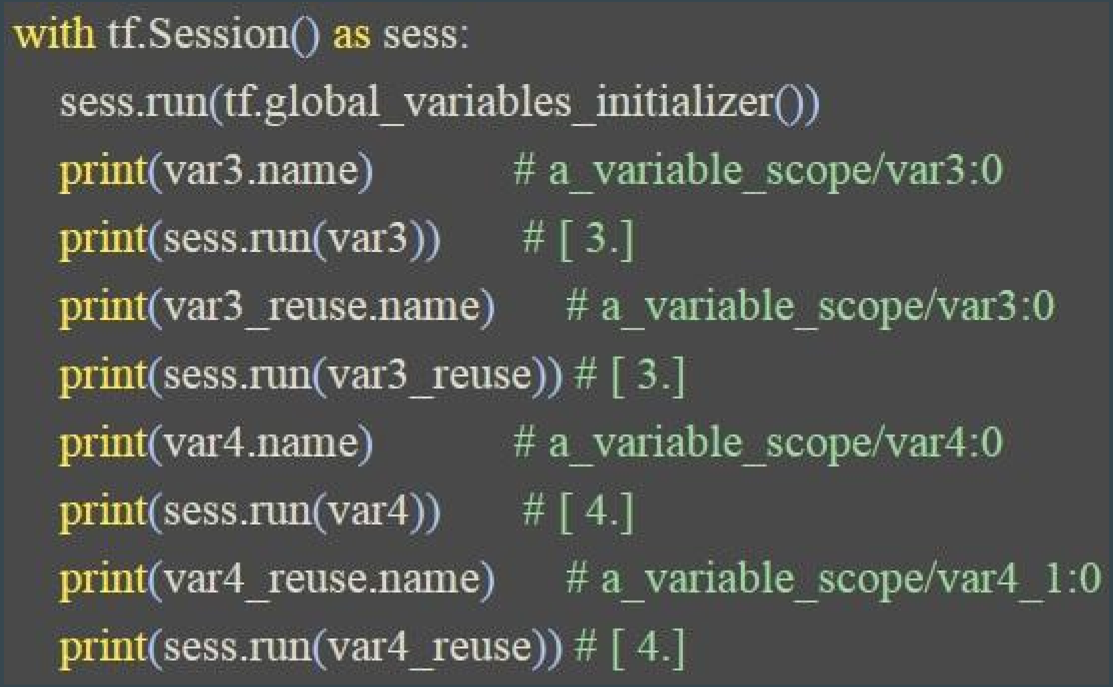
scope.reuse_variables()



```
with tf.variable_scope("a_variable_scope") as scope:
    initializer = tf.constant_initializer(value=3)
    var3 = tf.get_variable(name='var3', shape=[1], dtype=tf.float32, initializer=initializer)
    scope.reuse_variables()
    var3_reuse = tf.get_variable(name='var3',)
    var4 = tf.Variable(name='var4', initial_value=[4], dtype=tf.float32)
    var4_reuse = tf.Variable(name='var4', initial_value=[4], dtype=tf.float32)
```

## Variables & Scope (4)

Result of II.



```
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    print(var3.name)            # a_variable_scope/var3:0
    print(sess.run(var3))       # [ 3.]
    print(var3_reuse.name)      # a_variable_scope/var3:0
    print(sess.run(var3_reuse)) # [ 3.]
    print(var4.name)            # a_variable_scope/var4:0
    print(sess.run(var4))       # [ 4.]
    print(var4_reuse.name)      # a_variable_scope/var4_1:0
    print(sess.run(var4_reuse)) # [ 4.]
```

tf.get_variable→can be reused!!

tf.Variable→names will change!!

# Variables & Scope (5)

1) If you define any variable, remember to initialize with the following lines:

```
init = tf.global_variables_initializer()

init = tf.global_variables_initializer()

init = tf.global_variables_initializer()
```

So important that we repeat 3 times !!

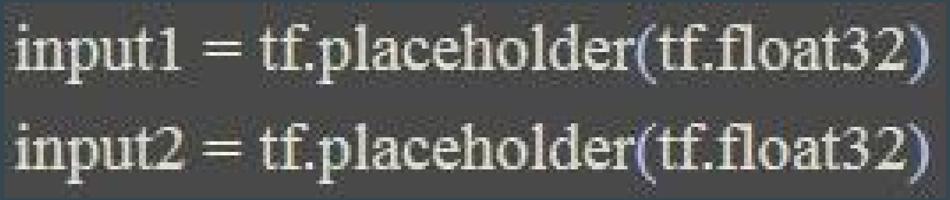
2) Activate the initialization with session using the following line:

```
sess.run(init)
```

## Placeholder (1)

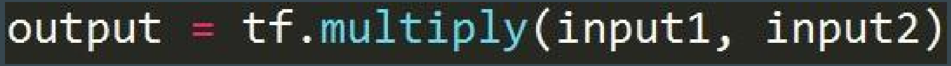
If we want to send the “outside” data into our neural networks, we have to use placeholder as a container.

1) Define the data type of the placeholder



```
input1 = tf.placeholder(tf.float32)
input2 = tf.placeholder(tf.float32)
```

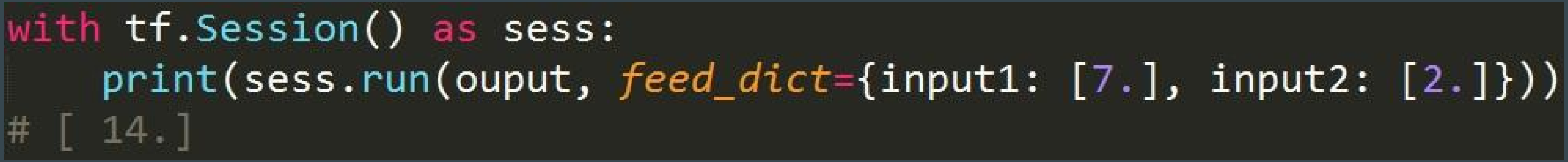
Note: Tensorflow adopts the data type float32 only! (neglect few exceptions)



```
output = tf.multiply(input1, input2)
```

## Placeholder (2)

2) Use session to perform the “sending”, and we employ feed_dict={} to designate the variables that will be sent in.

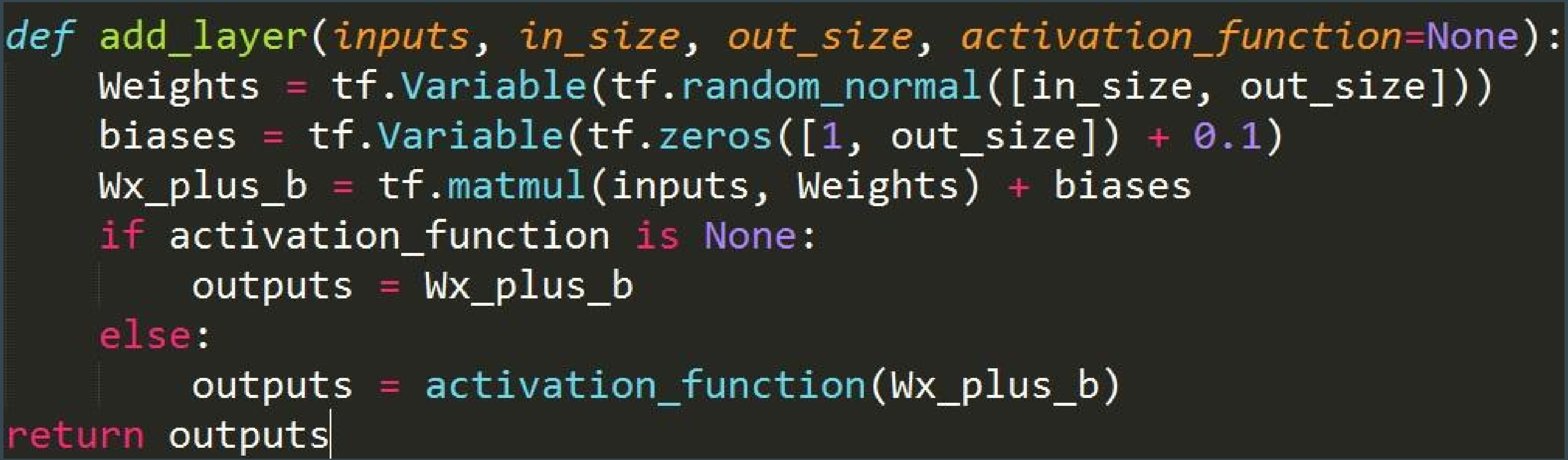


```
with tf.Session() as sess:
    print(sess.run(ouput, feed_dict={input1: [7.], input2: [2.]}))
# [ 14.]
```

Note: It method 2 of creating a session is used, we don’t need to close it explicitly!

## Define add_layer

We can define a function add_layer for further additions of layers in our neural networks.



```
def add_layer(inputs, in_size, out_size, activation_function=None):
    Weights = tf.Variable(tf.random_normal([in_size, out_size]))
    biases = tf.Variable(tf.zeros([1, out_size]) + 0.1)
    Wx_plus_b = tf.matmul(inputs, Weights) + biases
    if activation_function is None:
        outputs = Wx_plus_b
    else:
        outputs = activation_function(Wx_plus_b)
return outputs
```

Note: The value of biases are suggested not to be zero and can be adjusted.

## Example: building a simple neural network

After we introduce the basic use of Tensorflow, you can try building your own neural network!

If you are a newcomer to Tensorflow, you can run the following example found on the Internet:

https://github.com/MorvanZhou/tutorials/blob/master/tensorflowTUT/tensorflow11_build_network.py

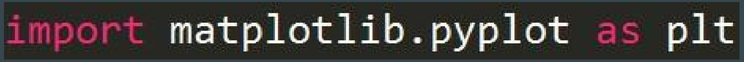
It will definitely help you grasp the concept of building a neural network with Tensorflow!

## PART II
## Visualization

## Result visualization (1)

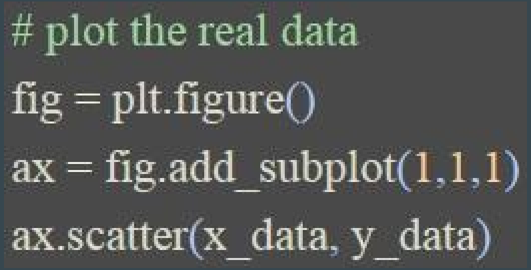
For intuitive visulaization of our results, we can generate some plots.

1) Import the module



```
import matplotlib.pyplot as plt
```

2) plt.figure()→add_subplot()→plt.show()



```
# plot the real data
fig = plt.figure()
ax = fig.add_subplot(1,1,1)
ax.scatter(x_data, y_data)
```

ax.scatter→scattering plot

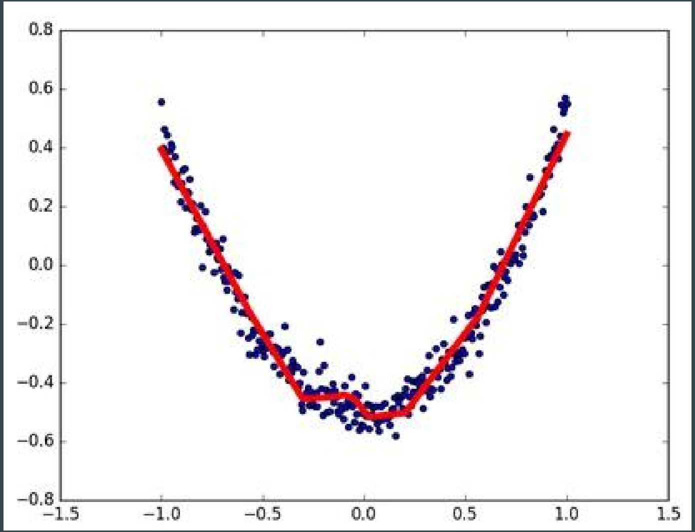
ax.line→line

# Result visualization (2)

3) For continuous display, add the following line:

```
plt.ion()
```

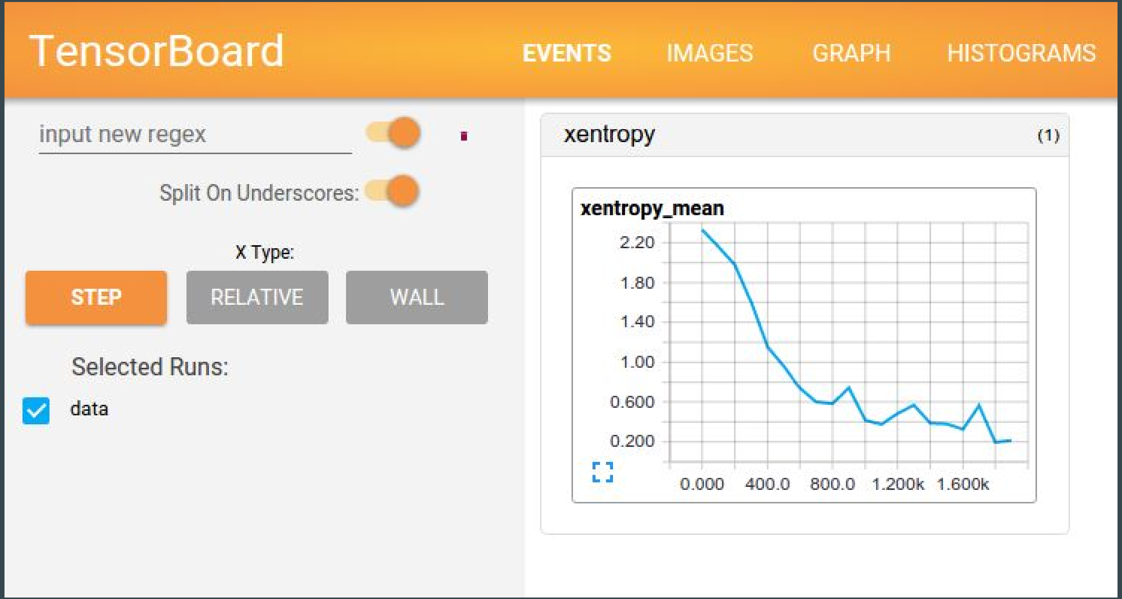
Ex:



## Tensorboard (1)

Tensorflow features Tensorboard, which can help us analyze our neural networks.

Note: It is better to use Google Chrome to browse Tensorboard.

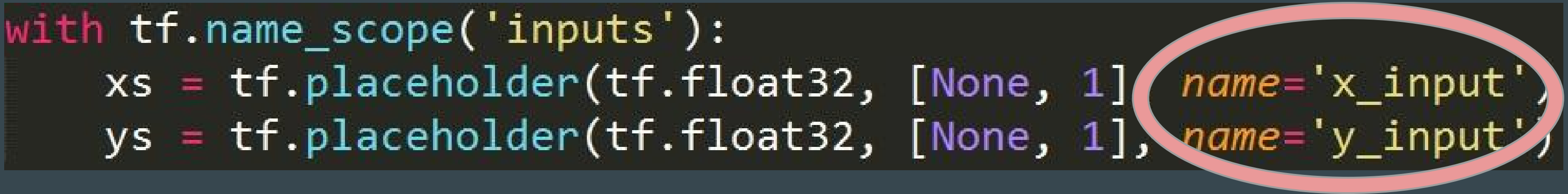


## Tensorboard (2)

Steps:

1) Define the names (inputs, weights ,biases, loss, ...)

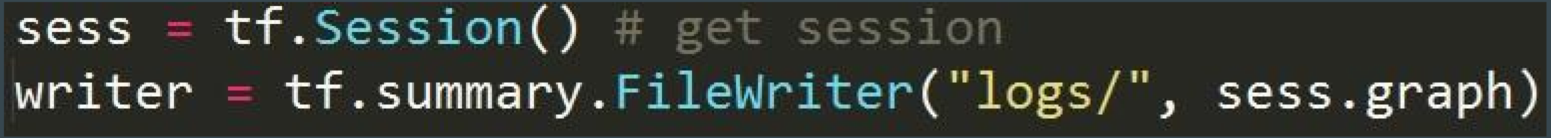
Ex: Inputs



```
with tf.name_scope('inputs'):
    xs = tf.placeholder(tf.float32, [None, 1], name='x_input')
    ys = tf.placeholder(tf.float32, [None, 1], name='y_input')
```

# Tensorboard (3)

2) Create a session and use tf.summary.FileWriter() to restore the graph



```python
sess = tf.Session() # get session
writer = tf.summary.FileWriter("logs/", sess.graph)
```

3) Go back to your terminal and enter the following command line:

```
tensorboard --logdir='logs/'
```

4) Copy the website address displayed on your terminal to the web browser and you can see the Tensorboard!

# Tensorboard (4)

There are two kinds of summary diagrams.

I. tf.scalar_summary()→loss

II. tf.histogram_summary()→others(weights, biases, outputs, ...)

Note:

(1) These two kinds of summary diagrams must be used together, otherwise there may be some bugs!

(2) The diagrams will dynamically change when the program is running!

# PART III
# Appendix

## Activation functions & Optimizers

I. Activation functions

https://www.tensorflow.org/versions/r0.10/api_docs/python/nn/activation_functions_

II. Optimizers

https://www.tensorflow.org/api_guides/python/train

Basic: GradientDescentOptimizer

Advanced: MomentumOptimizer & AdamOptimizer

Alphago (a.k.a. master): RMSPropOptimizer

## Reference

I. Morvan Zhou’s youtube channel & website

https://www.youtube.com/playlist?list=PLXO45tsB95cKI5AIlf5TxxFPzb-0zeVZ8

https://morvanzhou.github.io/tutorials/machine-learning/tensorflow/

II. Tensorflow

https://www.tensorflow.org/

III. CS224d: Tensorflow Tutorial

https://cs224d.stanford.edu/lectures/CS224d-Lecture7.pdf

http://www.planwallpaper.com/static/images/thank-you-clothesline-752x483.jpg