# TensorFlow
## - Getting Started -

2019 - 2020

Ando Ki, Ph.D.
adki@future-ds.com

## Table of contents

## TensorFlow programming

- Step 0: Import necessary modules
- Step 1: Build a computational graph
  - ‘tf.Session()’
  - The graph contains followings
    - parameter specifications
    - model architecture
    - optimization process
    - and so on
- Step 2: Initialize a session
  - If there are any variables, use ‘tf.global_variables_initializer()’ and ‘Session.run()’.
- Step 3: Fetch and feed data with ‘Session.run(fetch, feed)’
  - Fetch: list of graph nodes; return the outputs of those nodes
  - Feed: dictionary mapping from graph nodes to concrete values
    - Specifies the value of each graph node given in the dictionary.
  - Followings happens at this step
    - compilation
    - optimization
    - and so on

TensorFlow does not actually run any computation until the session is created and the run function is called.

Python dict, {…}

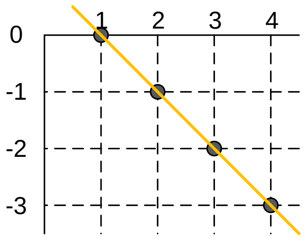
Python list, […]

3

## Getting started with some examples

See: ~/tensorflow-projects/getting

- ‘getting_const.py’
  - Two constants case
- ‘getting_add.py’
  - Add two constants case
- ‘getting_pla.py’
  - Add two variables using placeholders
- ‘getting_var.py’
  - W*x+b
- ‘getting_los.py’
  - W*x+b with loss function
- ‘getting_tra.py’
  - W*x+b with gradient descent optimizer



4

## A simple TensorFlow program: getting_const.py

```
# Importing TensorFlow
import tensorflow as tf

#------------------------------------------------------------------
# Building computational graph
const1 = tf.constant(3.0, dtype=tf.float32, name="const1")
const2 = tf.constant(4.0, name="const2")
print(const1, const2)

#------------------------------------------------------------------
# Running the computational graph
sess = tf.Session()
result = sess.run([const1, const2])
print result
```

Import necessary module

It does not print the value, since it is no t evaluated yet.

Launches the graph

Executes operations

Return value is in numpy ndarray type

```
$ source ~/tensorflow/bin/activate
(tensorflow)$ python getting.py
(<tf.Tensor 'const1:0' shape=() dtype=float32>, <tf.Tensor 'const2:0' shape=() dtype=float32>)
[3.0, 4.0]
$ deactivate
```

5

## A simple TensorFlow program: getting_const.py

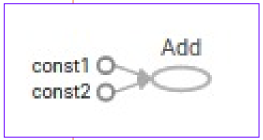
- This example shows how to use constant in TensorFlow
  - Step 1: go to your project directory
    - [user@host] cd $(PROJECT)/codes/tensorflow-project/getting
  - Step 2: see the codes: getting_const.py
  - Step 3: run Python under virtual environment
    - (do not forget to run '$ source ~/tensorflow/bin/activate')
    - [user@host]  python getting_const.py

```
[user@host] cd $(PROJECT)/codes/tensorflow-project/getting
[user@host] source ~/tensorflow/bin/activate
(tensorflow) [user@host] python getting_const.py
(tensorflow) [user@host] deactivate
[user@host]
```

6

## A simple TensorFlow program: getting_add.py (1/2)

```
#------------------------------------------------------------------
# Importing TensorFlow
import tensorflow as tf

#------------------------------------------------------------------
# Building computational graph
const1 = tf.constant(3.0, dtype=tf.float32, name="const1")
const2 = tf.constant(4.0, name="const2") # tf.float32 implicitly
nodeA  = tf.add(const1, const2)

#------------------------------------------------------------------
# Prepare graph
sess = tf.Session()

#------------------------------------------------------------------
# Prepare for tensorboard
tf.summary.FileWriter('/tmp/tensorflow', sess.graph)

#------------------------------------------------------------------
# Running the computational graph
print(sess.run(nodeA))
```



Activate TensorBoard

7

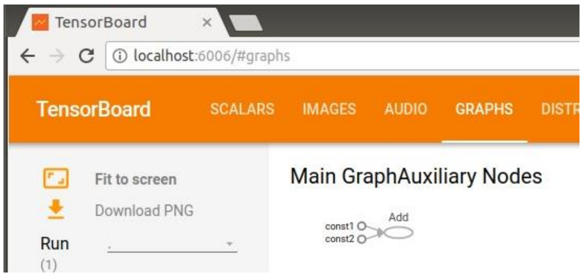
## A simple TensorFlow program: getting_add.py (2/2)

```
[adki@ando-ubuntu] source ~/tensorflow/bin/activate
(tensorflow)$ python getting_add.py
7.0
(tensorflow)$ tensorboard --logdir=/tmp/tensorflow --port=6006
Starting TensorBoard 54 at http://ando-ubuntu:6006
(Press CTRL+C to quit)
```

Run the model

Run program for TensorBoard

- After running 'tensorboard' program.
- Open a web-browser and go to 'http://localhost:6006'



8

## A simple TensorFlow program: getting_add.py

- This example shows how to add two constants in TensorFlow
  - Step 1: go to your project directory
    - [user@host] cd $(PROJECT)/codes/tensorflow-project/getting
  - Step 2: see the codes: getting_add.py
  - Step 3: run Python under virtual environment
    - (do not forget to run '$ source ~/tensorflow/bin/activate')
    - [user@host]  python getting_add.py
  - Step 4: run tensorboard to see graph

```
[user@host] cd $(PROJECT)/codes/tensorflow-project/getting
[user@host] source ~/tensorflow/bin/activate
(tensorflow) [user@host] python getting_add.py
(tensorflow) [user@host] deactivate
[user@host]
```

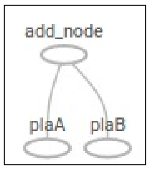
9

## A simple TensorFlow program: getting_pla.py (1/2)

```
#------------------------------------------------------------------
# Importing TensorFlow
import tensorflow as tf
#------------------------------------------------------------------
# Building computational graph
plaA = tf.placeholder(dtype=tf.float32, name="plaA")
plaB = tf.placeholder(dtype=tf.float32, name="plaB")
add_node = tf.add(plaA, plaB, name="add_node")
#------------------------------------------------------------------
# Prepare graph
sess = tf.Session()
#------------------------------------------------------------------
# Prepare for tensorboard
tf.summary.FileWriter('/tmp/tensorflow', sess.graph)
#------------------------------------------------------------------
# Running the computational graph
print(sess.run(add_node, {plaA:3, plaB:4.5})) # single value
print(sess.run(add_node, {plaA:[1,3], plaB:[2,4]})) # vector value
```

Placeholder enables us to feed arbitrary inputs.

placeholder is a promise to provide a value later



Activate TensorBoard

How to feed value
- scalar
- vector

10

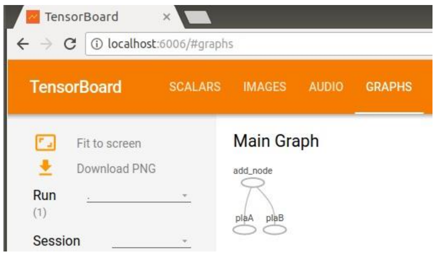
## A simple TensorFlow program: getting_pla.py (2/2)

```
[adki@ando-ubuntu] source ~/tensorflow/bin/activate
(tensorflow)$ python getting_pla.py
7.5
[  3.    7.]
(tensorflow)$ tensorboard --logdir=/tmp/tensorflow
Starting TensorBoard 54 at http://ando-ubuntu:6006
(Press CTRL+C to quit)
```

Run the model

Run program for TensorBoard

- After running ‘tensorboard’ program,
- Open a web-browser and go to ‘http://localhost:6006’



## A simple TensorFlow program: getting_pla.py

- This example shows how to add two variables in TensorFlow
  - Step 1: go to your project directory
    - [user@host] cd $(PROJECT)/codes/tensorflow-project/getting
  - Step 2: see the codes: getting_pla.py
  - Step 3: run Python under virtual environment
    - (do not forget to run ‘$ source ~/tensorflow/bin/activate’)
    - [user@host]  python getting_pla.py
  - Step 4: run tensorboard to see graph

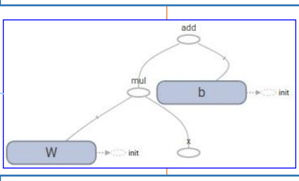
```
[user@host] cd $(PROJECT)/codes/tensorflow-project/getting
[user@host] source ~/tensorflow/bin/activate
(tensorflow) [user@host] python getting_pla.py
(tensorflow) [user@host] deactivate
[user@host]
```

## A simple TensorFlow program: getting_var.py (1/2)

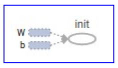
```
import tensorflow as tf

#-----------------------------------------------------------------
W = tf.Variable([.3], dtype=tf.float32, name="W")
b = tf.Variable([-.3], dtype=tf.float32, name="b")
x = tf.placeholder(tf.float32, name="x")
linear_model = W * x + b

sess = tf.Session()
#-----------------------------------------------------------------
# initialize all the variables in a TensorFlow
init = tf.global_variables_initializer()
sess.run(init) # actually initialized at this point
#-----------------------------------------------------------------
tf.summary.FileWriter('/tmp/tensorflow', sess.graph)
#-----------------------------------------------------------------
# Running the computational graph
print(sess.run(linear_model, {x:[1,2,3,4]}))
```

Variable allow us to add trainable parameters to a graph. Variable requires initial value.
Note that 'V' in capital letter since it is class.

placeholder is a promise to provide a value later



'init' is a handle to the sub-graph



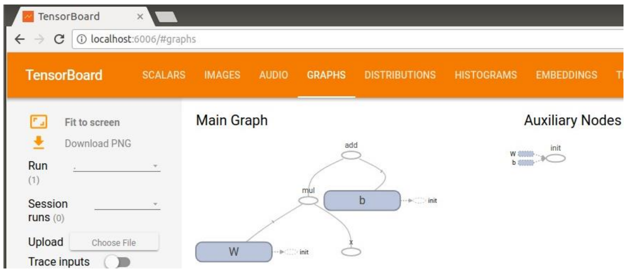
Evaluate 'linear_model' for several values of x simultaneously

## A simple TensorFlow program: getting_var.py (2/2)

```
[adki@ando-ubuntu] source ~/tensorflow/bin/activate
(tensorflow)$ python getting_var.py
[ 0.    0.30000001   0.60000002    0.90000004]
(tensorflow)$ tensorboard --logdir=/tmp/tensorflow
Starting TensorBoard 54 at http://ando-ubuntu:6006
(Press CTRL+C to quit)
```

| W | * | x | + | b | = | lm |
|---|---|---|---|---|---|---|
| 0.3 | | | | -0.3 | | |
| | | 1 | | | | 0 |
| | | 2 | | | | 0.3 |
| | | 3 | | | | 0.6 |
| | | 4 | | | | 0.9 |

- After running 'tensorboard' program.
- Open a web-browser and go to 'http://localhost:6006'



## A simple TensorFlow program: getting_var.py

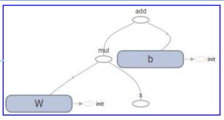
- This example shows how to use variables in TensorFlow
  - Step 1: go to your project directory
    - [user@host] cd $(PROJECT)/codes/tensorflow-project/getting
  - Step 2: see the codes: getting_var.py
  - Step 3: run Python under virtual environment
    - (do not forget to run '$ source ~/tensorflow/bin/activate')
    - [user@host]  python getting_var.py
  - Step 4: run tensorboard to see graph

```
[user@host] cd $(PROJECT)/codes/tensorflow-project/getting
[user@host] source ~/tensorflow/bin/activate
(tensorflow) [user@host] python getting_var.py
(tensorflow) [user@host] deactivate
[user@host]
```

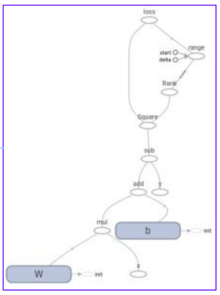
15

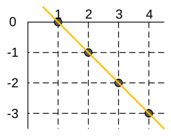
## A simple TensorFlow program: getting_los.py (1/2)

```
import tensorflow as tf
#-----------------------------------------------------------------
W = tf.Variable([.3], dtype=tf.float32, name="W")
b = tf.Variable([-.3], dtype=tf.float32, name="b")
x = tf.placeholder(tf.float32, name="x")
linear_model = W * x + b

#-----------------------------------------------------------------
# y will provide the desired values
y = tf.placeholder(tf.float32, name="y")
squared_deltas = tf.square(linear_model - y)
loss = tf.reduce_sum(squared_deltas, name="loss")
#-----------------------------------------------------------------
sess = tf.Session()
init = tf.global_variables_initializer()
sess.run(init) # actually initialized at this point
tf.summary.FileWriter('/tmp/tensorflow', sess.graph)
#-----------------------------------------------------------------
# Running the computational graph
print(sess.run(loss, {x:[1,2,3,4], y:[0,-1,-2,-3]}))
```



Loss function using square of errors





Provide desire values along with input values

16

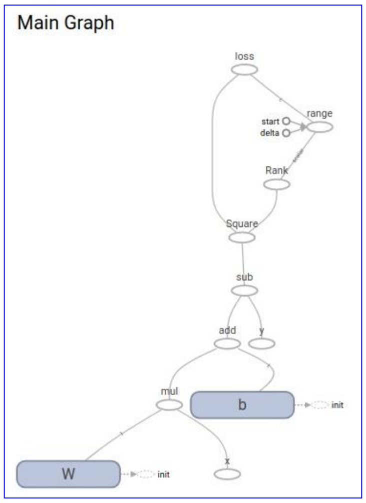
## A simple TensorFlow program: getting_los.py (2/2)

```
[adki@ando-ubuntu] source ~/tensorflow/bin/activate
(tensorflow)$ python getting_los.py
23.66
(tensorflow)$ tensorboard --logdir=/tmp/tensorflow
Starting TensorBoard 54 at http://ando-ubuntu:6006
(Press CTRL+C to quit)
```

- After running ‘tensorboard’ program,
- Open a web-browser and go to ‘http://localhost:6006’

| W | * | x | + | b | = | lm | y | lm-y | square | loss |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.3 | | | | -0.3 | | | | | | 23.66 |
| | | 1 | | | | 0 | 0 | 0 | 0 | |
| | | 2 | | | | 0.3 | -1 | 1.3 | 1.69 | |
| | | 3 | | | | 0.6 | -2 | 2.6 | 6.76 | |
| | | 4 | | | | 0.9 | -3 | 3.9 | 15.21 | |

Perfect value will be W=-1, b=1



## A simple TensorFlow program: getting_los.py

- This example shows how to use loss function in TensorFlow
  - Step 1: go to your project directory
    - [user@host] cd $(PROJECT)/codes/tensorflow-project/getting
  - Step 2: see the codes: getting_los.py
  - Step 3: run Python under virtual environment
    - (do not forget to run ‘$ source ~/tensorflow/bin/activate’)
    - [user@host]  python getting_los.py
  - Step 4: run tensorboard to see graph

```
[user@host] cd $(PROJECT)/codes/tensorflow-project/getting
[user@host] source ~/tensorflow/bin/activate
(tensorflow) [user@host] python getting_los.py
(tensorflow) [user@host] deactivate
[user@host]
```

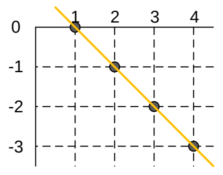
## A simple TensorFlow program: getting_tra.py (1/3)

```
import tensorflow as tf
#-----------------------------------------------------------------
W = tf.Variable([.3], dtype=tf.float32, name="W")
b = tf.Variable([-.3], dtype=tf.float32, name="b")
x = tf.placeholder(tf.float32, name="x")
linear_model = W * x + b

#-----------------------------------------------------------------
y = tf.placeholder(tf.float32, name="y")
squared_deltas = tf.square(linear_model - y)
loss = tf.reduce_sum(squared_deltas, name="loss")
#-----------------------------------------------------------------
# optimizer using gradient descent
optimizer = tf.train.GradientDescentOptimizer(0.01)
train = optimizer.minimize(loss)
#-----------------------------------------------------------------
sess = tf.Session()
init = tf.global_variables_initializer()
sess.run(init) # actually initialized at this point
tf.summary.FileWriter('/tmp/tensorflow', sess.graph)
```

```
#-----------------------------------------------------------------
# Running the computational graph
x_train  = [1,2,3,4]
y_expect = [0,-1,-2,-3]
for i in range(1000):
    sess.run(train, {x:x_train, y:y_expect})
    if i%100==0:
       cW, cb, closs =\
       sess.run([W, b, loss], {x:x_train, y:y_expect})
       print("W: %s b: %s loss: %s"%(cW, cb, closs))

#-----------------------------------------------------------------
# Evaluate current accuracy
cW, cb, closs =\
sess.run([W, b, loss], {x:x_train, y:y_expect})
print("W: %s b: %s loss: %s"%(cW, cb, closs))
```

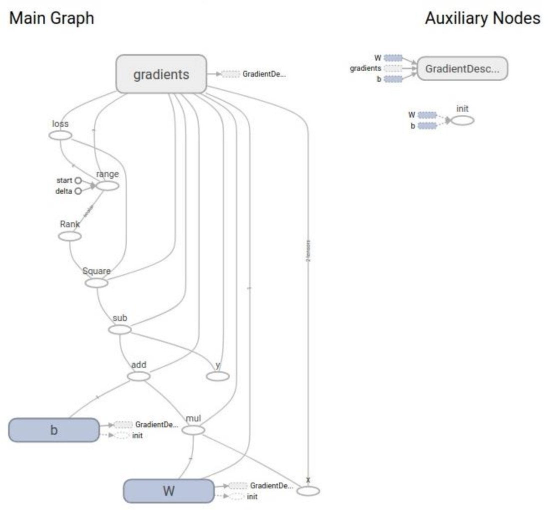
## A simple TensorFlow program: getting_tra.py (2/3)

```
[adki@ando-ubuntu] source ~/tensorflow/bin/activate
(tensorflow)$ python getting_tra.py
W: [-0.21999997] b: [-0.456] loss: 4.01814
W: [-0.84270465] b: [ 0.53753263] loss: 0.14288
W: [-0.95284992] b: [ 0.86137295] loss: 0.0128382
W: [-0.98586655] b: [ 0.95844597] loss: 0.00115355
W: [-0.99576342] b: [ 0.98754394] loss: 0.000103651
W: [-0.99873012] b: [ 0.99626648] loss: 9.3124e-06
W: [-0.99961936] b: [ 0.99888098] loss: 8.36456e-07
W: [-0.99988592] b: [ 0.9996646] loss: 7.51492e-08
W: [-0.99996579] b: [ 0.99989945] loss: 6.75391e-09
W: [-0.99998969] b: [ 0.99996972] loss: 6.12733e-10
W:-0.999997 b:0.999991 loss:5.699974e-11

(tensorflow)$ tensorboard --logdir=/tmp/tensorflow
Starting TensorBoard 54 at http://ando-ubuntu:6006
(Press CTRL+C to quit)
```



Perfect value will be W=-1, b=1
See the loss is very small.

## A simple TensorFlow program: getting_tra.py (3/3)



21

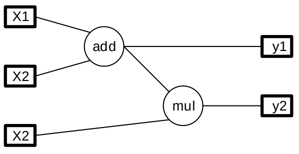
## A simple TensorFlow program: getting_tra.py

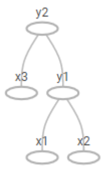
- This example shows how to use train function in TensorFlow
  - Step 1: go to your project directory
    - [user@host] cd $(PROJECT)/codes/tensorflow-project/getting
  - Step 2: see the codes: getting_tra.py
  - Step 3: run Python under virtual environment
    - (do not forget to run '$ source ~/tensorflow/bin/activate')
    - [user@host]  python getting_tra.py
  - Step 4: run tensorboard to see graph

```
[user@host] cd $(PROJECT)/codes/tensorflow-project/getting
[user@host] source ~/tensorflow/bin/activate
(tensorflow) [user@host] python getting_tra.py
(tensorflow) [user@host] deactivate
[user@host]
```

22

## Your project 1



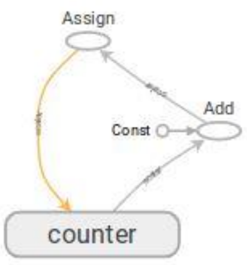


- Make your own TensorFlow program, using ‘tf.placeholder()’, ‘tf.add()’, ‘tf.multiply()’.
- Make sure to call ‘sess.run()’
- Give values of ‘x1/x2/x3’.
- And print result of ‘y1/y2’.

See: ~/tensorflow-projects/getting/proj_placehold.py

## Your project 2: counter

See: ~/tensorflow-projects/getting/proj_counter.py



- Make your own TensorFlow program, using ‘tf.Variable()’, ‘tf.add()’, ‘tf.assign()’.
- Make sure to call ‘sess.run()’

```
# Importing TensorFlow
import tensorflow as tf
#-------------------------------------------------------------------
counter   = tf.Variable(0, name="counter")
new_value = tf.add(counter, tf.constant(1))
update    = tf.assign(counter, new_value)
#-------------------------------------------------------------------
with tf.Session() as sess:
    tf.summary.FileWriter('./log', sess.graph)
    sess.run(tf.global_variables_initializer())
    print(sess.run(counter))
    for _ in range(3):
        sess.run(update) # running 'update' graph
        print(sess.run(counter)) # get value of 'counter'
```

## Tenforflow

- tf.constant()
- print()
- tf.Session.run()

- tf.add()
- tf.placeholder()
- tf.Variable()
- tf.global_variables_initializer()

- tf.square()
- tf.reduce_sum()
- tf.assign()

- tf.train.GradientDescentOptimizer()
- minimize()

- Functions
  - mathematical operators: add, sub, mul, div, abs, mod, neg
  - array: concat, slice, split, constant, rank, shape, shuffle
  - matrix: diag, transpose, matmul, matrix_determinant, matrix_inverse
  - neural net: softmax, sigmoid, ReLU, Convlution2D, MaxPool
  - session: save, restore
  - queuing, synchronization: enqueue, dequeue, MutexAcquire, MutexRelease
  - flow control: merge, switch, enter, leave, NextIteration

㈜퓨쳐디자인시스템
34051 대전광역시 유성구 문지로 193, KAIST 문지캠퍼스, F723호
(042) 864-0211~0212 / contact@future-ds.com / www.future-ds.com

Future Design Systems, Inc.
Faculty Wing F723, KAIST Munji Campus, 193 Munji-ro, Yuseong-gu, Daejeon 34051, Korea
+82-042-864-0211~0212 / contact@future-ds.com / www.future-ds.com

FUTURE
Design Systems