1 NN for XOR

2 Tensorboard for XOR NN

Lecture 08

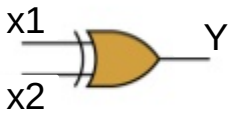
1

NN for XOR

XOR data set

Boolean Expression	Logic Diagram Symbol	Truth Table

$Y = x1 \oplus x2$



x1	x2	Y
0	0	0
0	1	1
1	0	1
1	1	0

```
x_data = np.array([[0, 0], [0, 1], [1, 0], [1, 1]], dtype=np.float32)
y_data = np.array([[0],    [1],    [1],    [0]], dtype=np.float32)
```

ex08_1.ipynb

XOR with logistic regression?

```
import tensorflow as tf
import numpy as np
x_data = np.array([[0, 0], [0, 1], [1, 0], [1, 1]], dtype=np.float32)
y_data = np.array([[0], [1], [1], [0]], dtype=np.float32)

X = tf.placeholder(tf.float32, [None, 2])
Y = tf.placeholder(tf.float32, [None, 1])
W = tf.Variable(tf.random_normal([2, 1]), name='weight')
b = tf.Variable(tf.random_normal([1]), name='bias')

# Hypothesis using sigmoid
hypothesis = tf.sigmoid(tf.matmul(X, W) + b)

# cost/loss function
cost = -tf.reduce_mean(Y * tf.log(hypothesis) + (1 - Y) * tf.log(1 - hypothesis))
train = tf.train.GradientDescentOptimizer(learning_rate=0.1).minimize(cost)

# Accuracy computation
# True if hypothesis>0.5 else False
predicted = tf.cast(hypothesis > 0.5, dtype=tf.float32)
accuracy = tf.reduce_mean(tf.cast(tf.equal(predicted, Y), dtype=tf.float32))

# Launch graph
with tf.Session() as sess:
    # Initialize TensorFlow variables
    sess.run(tf.global_variables_initializer())

    for step in range(10001):
        sess.run(train, feed_dict={X: x_data, Y: y_data})
        if step % 100 == 0:
            print("step: ",step, "\tcost : ",sess.run(cost, feed_dict={X: x_data, Y: y_data}))
            print("W: \n", sess.run(W), "\nb: \n", sess.run(b) ,"\n", "="*50)
    # Accuracy report
    h, c, a = sess.run([hypothesis, predicted, accuracy], feed_dict={X: x_data, Y: y_data})
    print("Hypothesis \n ", h.reshape(-1,4), "\ny_data \n ", y_data.reshape(-1,4),  "\nCorrect \n ", c.reshape(-1,4), "\n Accuracy \n ", a)
```

True , False

True → 1.
False → 0.

ex08_1.ipynb Notes

XOR with logistic regression?

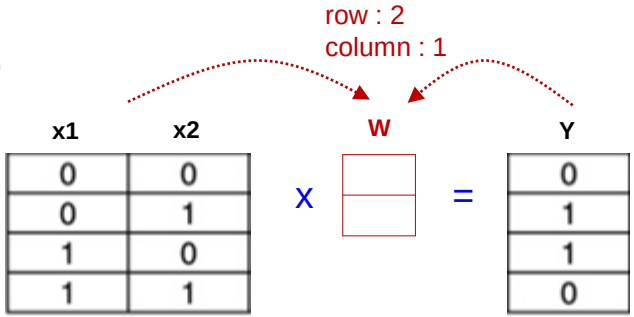
```
step:  0          cost :  0.85884523
W:
 [[0.27288756]
 [2.2499688 ]]
b:
 [-0.87927616]
 ==================================================
step:  100        cost :  0.74584824
W:
 [[0.18489242]
 [1.2742372 ]]
b:
 [-0.8568866]
 ==================================================
step:  200        cost :  0.7148558
W:
 [[0.20223512]
 [0.79047865]]
b:
 [-0.59119153]
.
.
.
step:  9900       cost :  0.6931472
W:
 [[1.3336123e-07]
 [1.3270730e-07]]
b:
 [-1.7855265e-07]
 ==================================================
step:  10000      cost :  0.6931472
W:
 [[1.3336123e-07]
 [1.3270730e-07]]
b:
 [-1.7855265e-07]
 ==========================
Hypothesis
  [[0.5 0.5 0.5 0.5]]
y_data
  [[0. 1. 1. 0.]]
Correct
  [[0. 0. 0. 0.]]
 Accuracy
  0.5
```

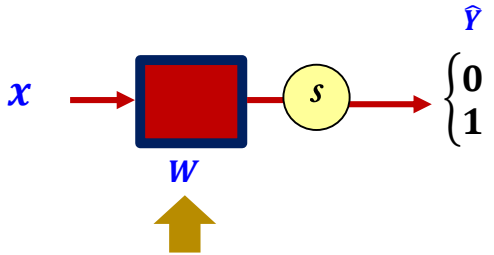
But
it doesn't work!

ex08_1.ipynb Notes

Neural Net (Single Layer)

```
import tensorflow as tf
import numpy as np
x_data = np.array([[0, 0], [0, 1], [1, 0], [1, 1]], dtype=np.float32)
y_data = np.array([[0], [1], [1], [0]], dtype=np.float32)

X = tf.placeholder(tf.float32, [None, 2])
Y = tf.placeholder(tf.float32, [None, 1])
W = tf.Variable(tf.random_normal([2, 1]), name='weight')
b = tf.Variable(tf.random_normal([1]), name='bias')

# Hypothesis using sigmoid
hypothesis = tf.sigmoid(tf.matmul(X, W) + b)
```

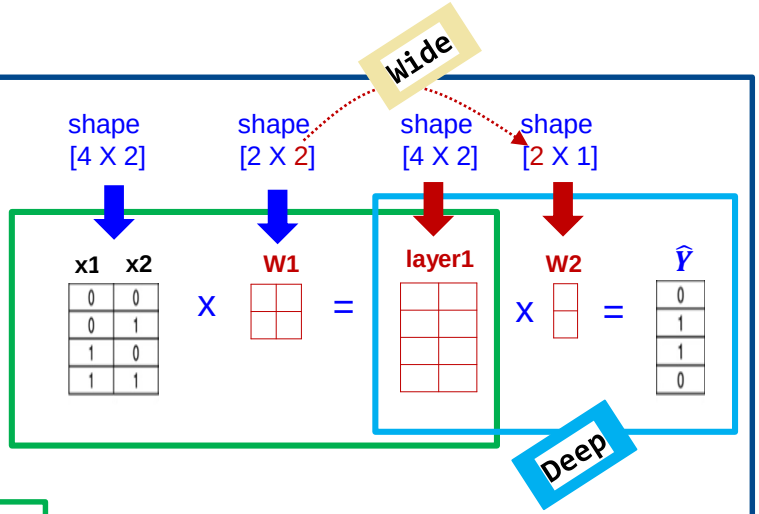


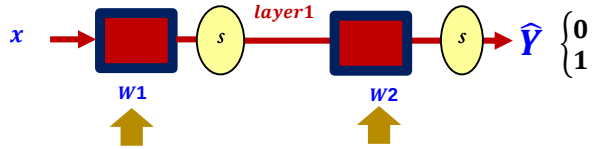


Neural Net (Multi Layer)

```
import tensorflow as tf
import numpy as np
x_data = np.array([[0, 0], [0, 1], [1, 0], [1, 1]],
dtype=np.float32)
y_data = np.array([[0], [1], [1], [0]], dtype=np.float32)

X = tf.placeholder(tf.float32, [None, 2])
Y = tf.placeholder(tf.float32, [None, 1])

W1 = tf.Variable(tf.random_normal([2, 2]), name='weight1')
b1 = tf.Variable(tf.random_normal([2]), name='bias1')
layer1 = tf.sigmoid(tf.matmul(X, W1) + b1)

W2 = tf.Variable(tf.random_normal([2, 1]), name='weight2')
b2 = tf.Variable(tf.random_normal([1]), name='bias2')

hypothesis = tf.sigmoid(tf.matmul(layer1, W2) + b2)
```

$\hat{Y}$





ex08_2.ipynb

NN for XOR (Multi Layer)

```
import tensorflow as tf
import numpy as np
x_data = np.array([[0, 0], [0, 1], [1, 0], [1, 1]], dtype=np.float32)
y_data = np.array([[0], [1], [1], [0]], dtype=np.float32)
X = tf.placeholder(tf.float32)
Y = tf.placeholder(tf.float32)

W1 = tf.Variable(tf.random_normal([2, 2]), name='weight1')
b1 = tf.Variable(tf.random_normal([2]), name='bias1')
layer1 = tf.sigmoid(tf.matmul(X, W1) + b1)

W2 = tf.Variable(tf.random_normal([2, 1]), name='weight2')
b2 = tf.Variable(tf.random_normal([1]), name='bias2')
hypothesis = tf.sigmoid(tf.matmul(layer1, W2) + b2)

# cost/loss function
cost = -tf.reduce_mean(Y * tf.log(hypothesis) + (1 - Y) * tf.log(1 - hypothesis))
train = tf.train.GradientDescentOptimizer(learning_rate=0.1).minimize(cost)
# Accuracy computation
# True if hypothesis>0.5 else False
predicted = tf.cast(hypothesis > 0.5, dtype=tf.float32)
accuracy = tf.reduce_mean(tf.cast(tf.equal(predicted, Y), dtype=tf.float32))
# Launch graph
with tf.Session() as sess:
    # Initialize TensorFlow variables
    sess.run(tf.global_variables_initializer())
    for step in range(10001):
        sess.run(train, feed_dict={X: x_data, Y: y_data})
        if step % 100 == 0:
            print("step : ", step, "cost :", sess.run(cost, feed_dict={X: x_data, Y: y_data}))
            print("W1 : \n", sess.run(W1), "\nb1: \n", sess.run(b1))
            print("Layer 1: \n", sess.run(layer1, feed_dict={X: x_data, Y: y_data}))
            print("W2 : \n", sess.run(W2), "\nb2: \n", sess.run(b2))
            print("hypothesis : \n", sess.run(hypothesis, feed_dict={X:x_data}),"\n" , "="*50)
    # Accuracy report
    p, a = sess.run([predicted, accuracy],feed_dict={X: x_data, Y: y_data})
    print("● Correct: \n", p, "\n● Accuracy:  \n", a)
```

ex08_2.ipynb Notes

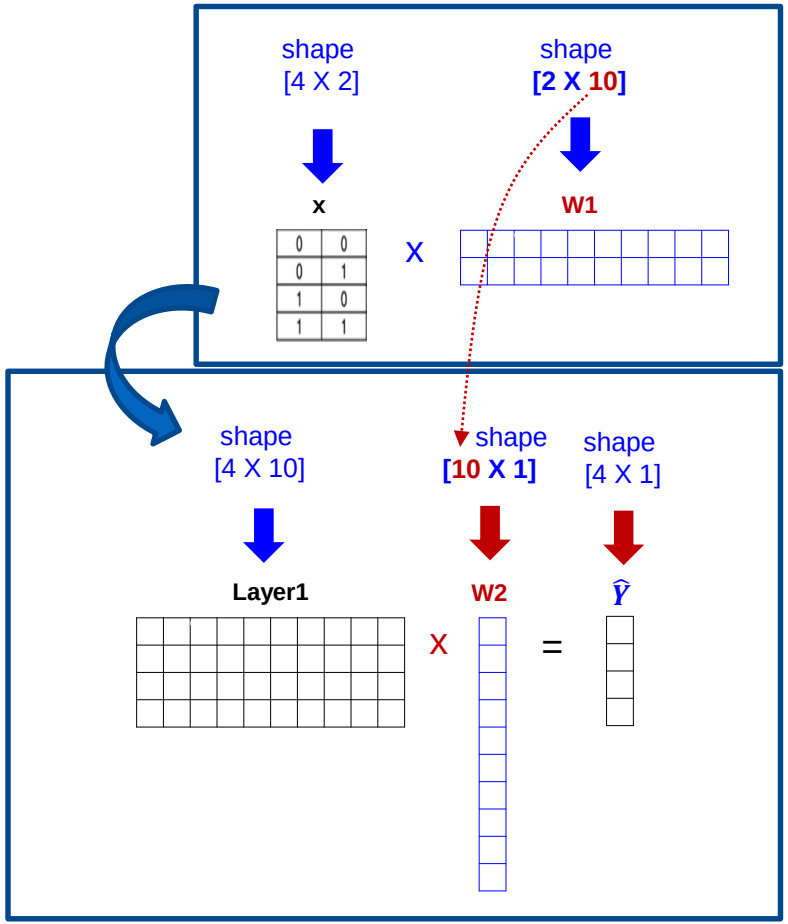
```
step :  0 cost : 0.76641256
W1 :
 [[-0.11243332 -0.48388383]
 [ 0.5747957  -0.46335787]]
b1:
 [ 0.48714975 -0.46306232]
Layer 1:
 [[0.6194348  0.38625962]
 [0.74306214 0.28365153]
 [0.59259814 0.27949938]
 [0.72101706 0.19618613]]
W2 :
 [[0.39562288]
 [0.32466972]]
b2:
 [0.41144657]
hypothesis :
 [[0.6860929 ]
 [0.68944204]
 [0.67626005]
 [0.6814391 ]]
 ================================================
step :  100 cost : 0.6936093
W1 :
 [[-0.11757189 -0.4828046 ]
 [ 0.5614868  -0.4625066 ]]
b1:
 [ 0.4612578  -0.48502386]
Layer 1:
 [[0.61331254 0.38106653]
 [0.73550683 0.27938175]
 [0.5850856  0.27531356]
 [0.71201134 0.19304648]]
W2 :
 [[0.06586391]
 [0.16061953]]
b2:
 [-0.07541786]
hypothesis :
 [[0.50654566]
 [0.5044748 ]
 [0.50183475]
 [0.50062126]]
 ================================================
```

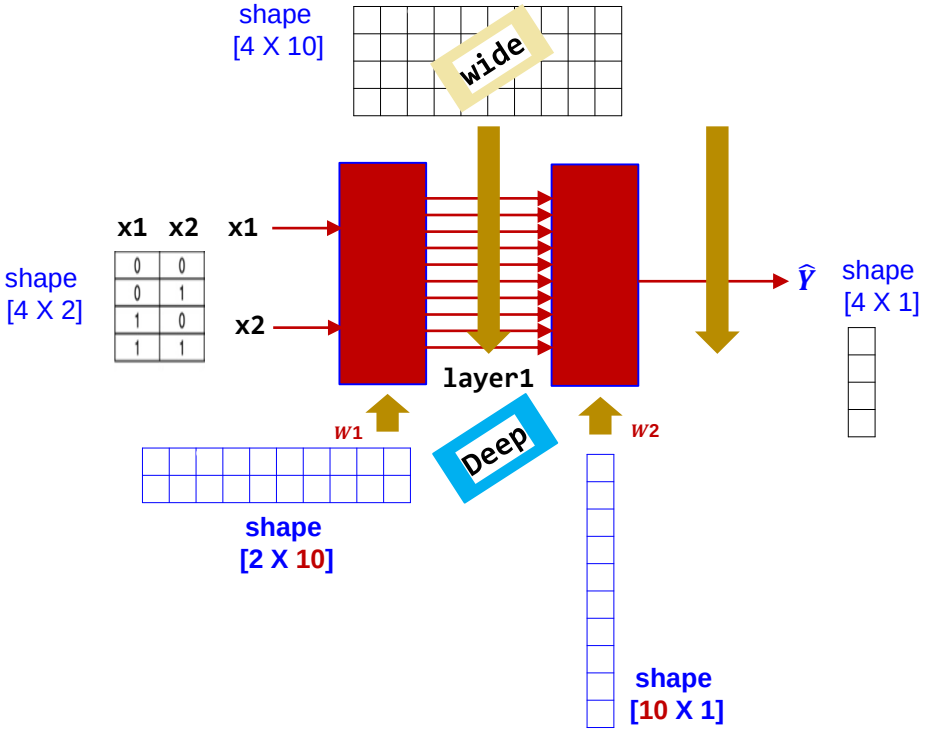
```
step :  200 cost : 0.6935026
W1 :
 [[-0.11463588 -0.47466004]
 [ 0.5607306  -0.45420668]]
b1:
 [ 0.4608377 -0.4835354]
Layer 1:
 [[0.6132129  0.38141763]
 [0.73527795 0.28135666]
 [0.5856962  0.27723965]
 [0.712372   0.19585545]]
W2 :
 [[0.05790563]
 [0.13454905]]
b2:
 [-0.07894189]
hypothesis :
 [[0.5019715 ]
 [0.50037277]
 [0.4980689 ]
 [0.49716514]]
.
.

step :  9900 cost : 0.041418742
W1 :
 [[ 3.9609594 -6.2198763]
 [ 3.967846  -6.2877   ]]
b1:
 [-6.1204443  2.2771738]
Layer 1:
 [[2.19266117e-03 9.06968892e-01]
 [1.04088664e-01 1.78012215e-02]
 [1.03448220e-01 1.90266892e-02]
 [8.59163702e-01 3.60559716e-05]]
W2 :
 [[-8.281344]
 [-8.41226 ]]
b2:
 [4.095111]
hypothesis :
 [[0.02784924]
 [0.95620143]
 [0.9559913 ]
 [0.04652059]]
 ================================================
```

```
step :  10000 cost : 0.039866842
W1 :
 [[ 3.9923384 -6.23685  ]
 [ 3.9991689 -6.303531 ]]
b1:
 [-6.1706285  2.2928936]
Layer 1:
 [[2.08556326e-03 9.08286810e-01]
 [1.02342851e-01 1.77992862e-02]
 [1.01717055e-01 1.90033056e-02]
 [8.60671580e-01 3.54452095e-05]]
W2 :
 [[-8.357225]
 [-8.470572]]
b2:
 [4.128153]
hypothesis :
 [[0.02704105]
 [0.9577944 ]
 [0.957593  ]
 [0.0445754 ]]
================================================
• Correct:
 [[0.]
 [1.]
 [1.]
 [0.]]
• Accuracy:
 1.0
```

Neural Net (Multi Layer)

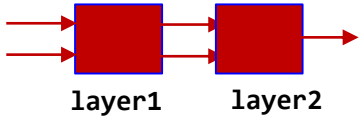
ex08_2.ipynb Notes

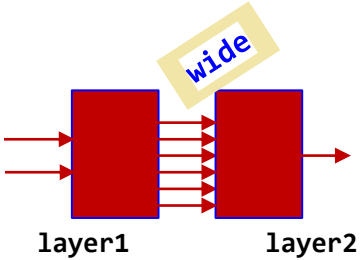




Wide NN for XOR

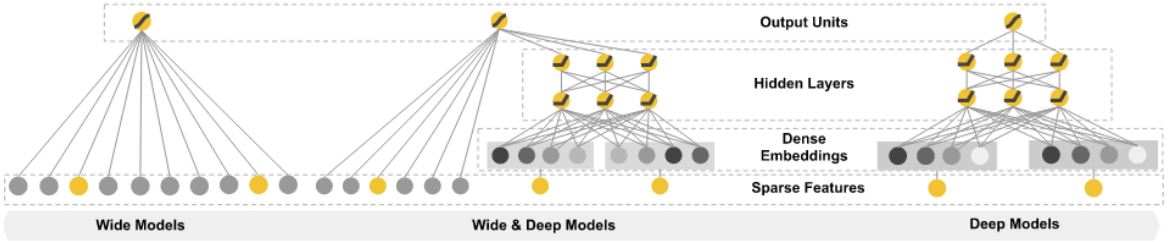
ex08_2(wide).ipynb





원하는 출력 형태로 정의

```
W1 = tf.Variable(tf.random_normal([2, 10]), name='weight1')
b1 = tf.Variable(tf.random_normal([10]), name='bias1')
layer1 = tf.sigmoid(tf.matmul(X, W1) + b1)

W2 = tf.Variable(tf.random_normal([10, 1]), name='weight2')
b2 = tf.Variable(tf.random_normal([1]), name='bias2')
hypothesis = tf.sigmoid(tf.matmul(layer1, W2) + b2)
```



참고 : print(tf.shape(W1).eval(), tf.shape(W2).eval()) *# W1, W2 shape 출력*

Better learning

[2,10], [10,1]

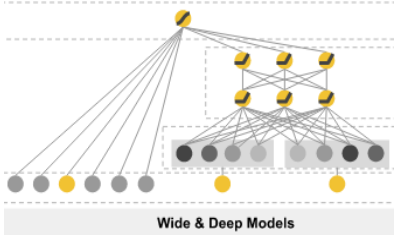
Hypothesis:
[[0.00358802]
[0.99366933]
[0.99204296]
[0.0095663]]
Correct:
[[0.]
[1.]
[1.]
[0.]]
Accuracy: 1.0

[2,2], [2,1]

Hypothesis:
[[0.01338218]
[0.98166394]
[0.98809403]
[0.01135799]]
Correct:
[[0.]
[1.]
[1.]
[0.]]
Accuracy: 1.0

Deep NN for XOR

ex08_2(deep).ipynb




```
W1 = tf.Variable(tf.random_normal([2, 10]), name='weight1')
b1 = tf.Variable(tf.random_normal([10]), name='bias1')
layer1 = tf.sigmoid(tf.matmul(X, W1) + b1)

W2 = tf.Variable(tf.random_normal([10, 10]), name='weight2')
b2 = tf.Variable(tf.random_normal([10]), name='bias2')
layer2 = tf.sigmoid(tf.matmul(layer1, W2) + b2)

W3 = tf.Variable(tf.random_normal([10, 10]), name='weight3')
b3 = tf.Variable(tf.random_normal([10]), name='bias3')
layer3 = tf.sigmoid(tf.matmul(layer2, W3) + b3)

W4 = tf.Variable(tf.random_normal([10, 1]), name='weight4')
b4 = tf.Variable(tf.random_normal([1]), name='bias4')
hypothesis = tf.sigmoid(tf.matmul(layer3, W4) + b4)
```

Better learning

4 layers

```
Hypothesis:
[[ 7.80e-04]
 [ 9.99e-01]
 [ 9.98e-01]
 [ 1.55e-03]]
Correct:
[[ 0.]
 [ 1.]
 [ 1.]
 [ 0.]]
Accuracy:  1.0
```

2 layers

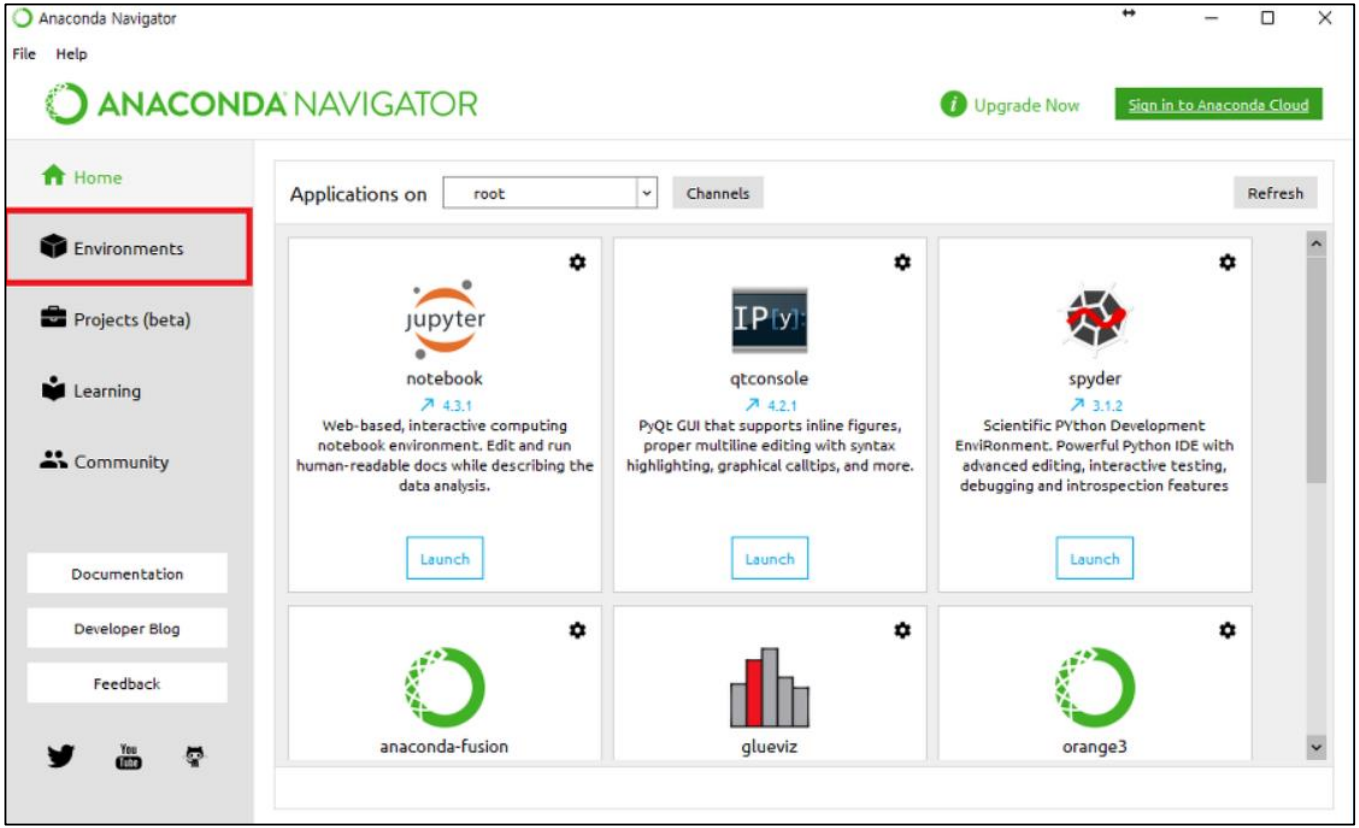
```
Hypothesis:
[[ 0.01338218]
 [ 0.98166394]
 [ 0.98809403]
 [ 0.01135799]]
Correct:
[[ 0.]
 [ 1.]
 [ 1.]
 [ 0.]]
Accuracy:  1.0
```

2

Tensorboard for XOR NN

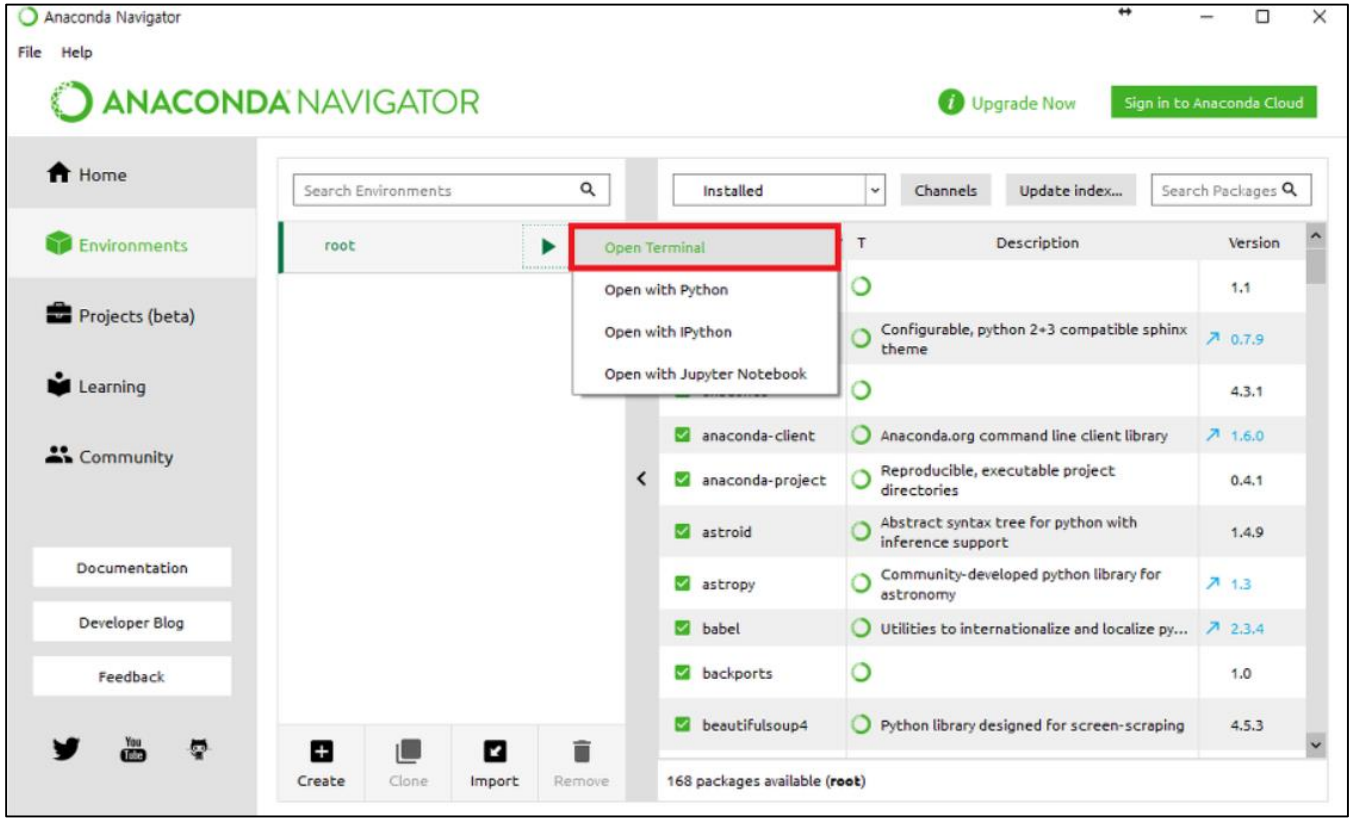
TensorBoard

왼쪽 사이드바에서 Environments 클릭



TensorBoard

root의 오른쪽 화살표 클릭 → Open Terminal 선택



TensorBoard

```
C:₩Users₩Park>conda create -n virtual_envs1 python=3.5
Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 4.4.10
  latest version: 4.5.5

Please update conda by running

    $ conda update -n base conda



## Package Plan ##

  environment location: C:₩Users₩Park₩Anaconda3₩envs₩virtu

  added / updated specs:
    - python=3.5
```

conda create -n virtual_envs1 python=3.5

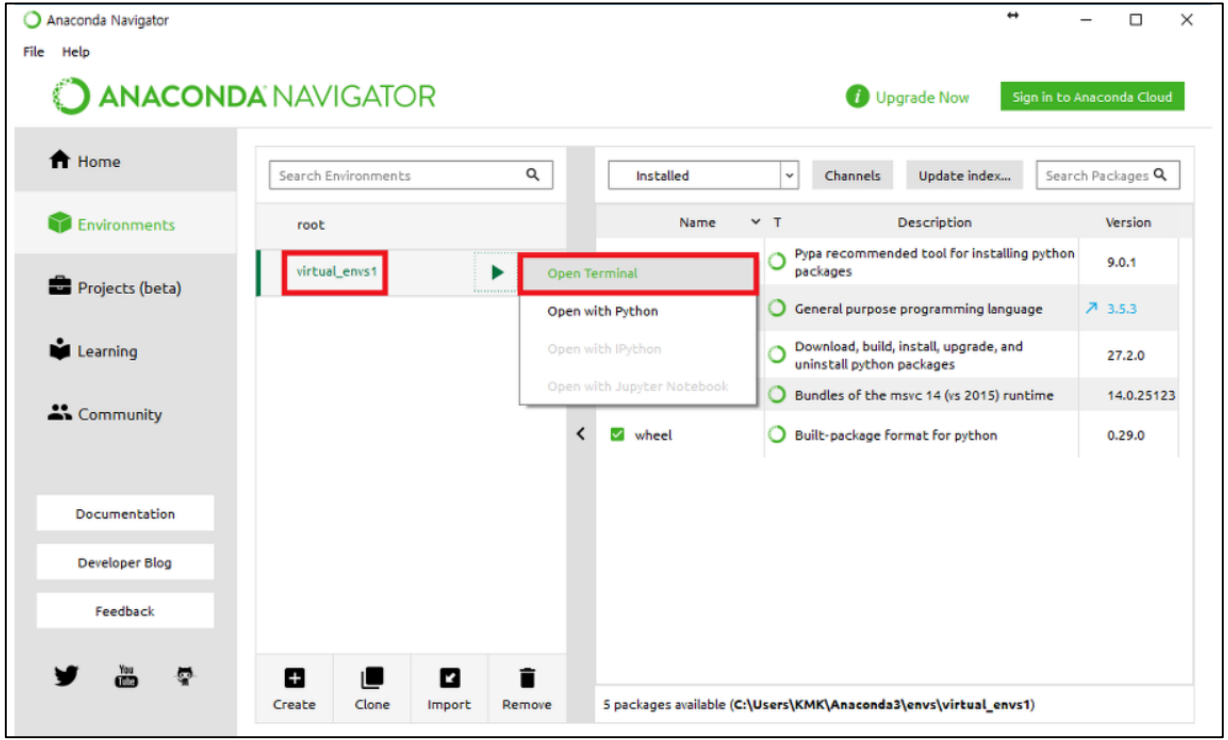
virtual_envs1부분은 원하는 이름으로
python은 무조건 3.5버전으로!!(17년 4월 17일 기준)
윈도우의 경우 tensorflow는
무조건 3.5버전만 지원하기 때문...

```
Proceed ([y]/n)? y

Preparing transaction: done
Verifying transaction: -
SafetyError: The package for setuptools located at C:₩Users₩Park
appears to be corrupted. The path 'Scripts/easy_install.exe'
has a sha256 mismatch.
  reported sha256: 993203a406e04936a07829b1f482fd27d739b640482e2
  actual sha256: e7d7af2a32ccb3fd29664403f39cebd32b2120dee5f5202

SafetyError: The package for wheel located at C:₩Users₩Park₩Anac
appears to be corrupted. The path 'Scripts/wheel.exe'
has a sha256 mismatch.
  reported sha256: 993203a406e04936a07829b1f482fd27d739b640482e2
  actual sha256: fb88d82081d11c5878fe8d644d3d65b472754c9c0abbee0



done
Executing transaction: done
#
# To activate this environment, use:
# > activate virtual_envs1
#
# To deactivate an active environment, use:
# >
#
# * for power-users using bash, you must source
#

C:₩Users₩Park>_
```

프롬프트가 나오면… 설치 완료된 것!!
(python 3.5버전을 가진 가상 환경을 구축함)

TensorBoard

cmd창을 끄고, 다시 Navigator로 이동
vitual_envs1 클릭 → 화살표 클릭 → Open Terminal 클릭



공식 홈페이지
https://www.tensorflow.org/install/install_windows 참조

TensorBoard

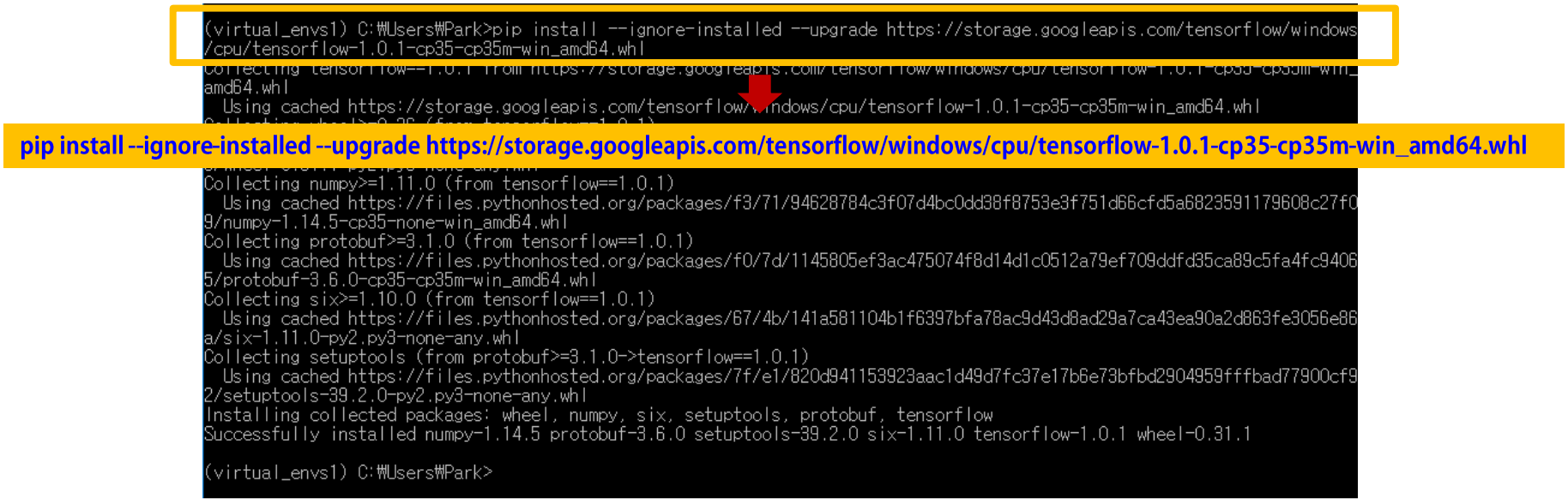
아래의 버전 중에서 자신의 환경에 적합한 하나를 선택해서 입력

CPU 버전

pip install --ignore-installed --upgrade https://storage.googleapis.com/tensorflow/windows/cpu/tensorflow-1.0.1-cp35-cp35m-win_amd64.whl

GPU 버전

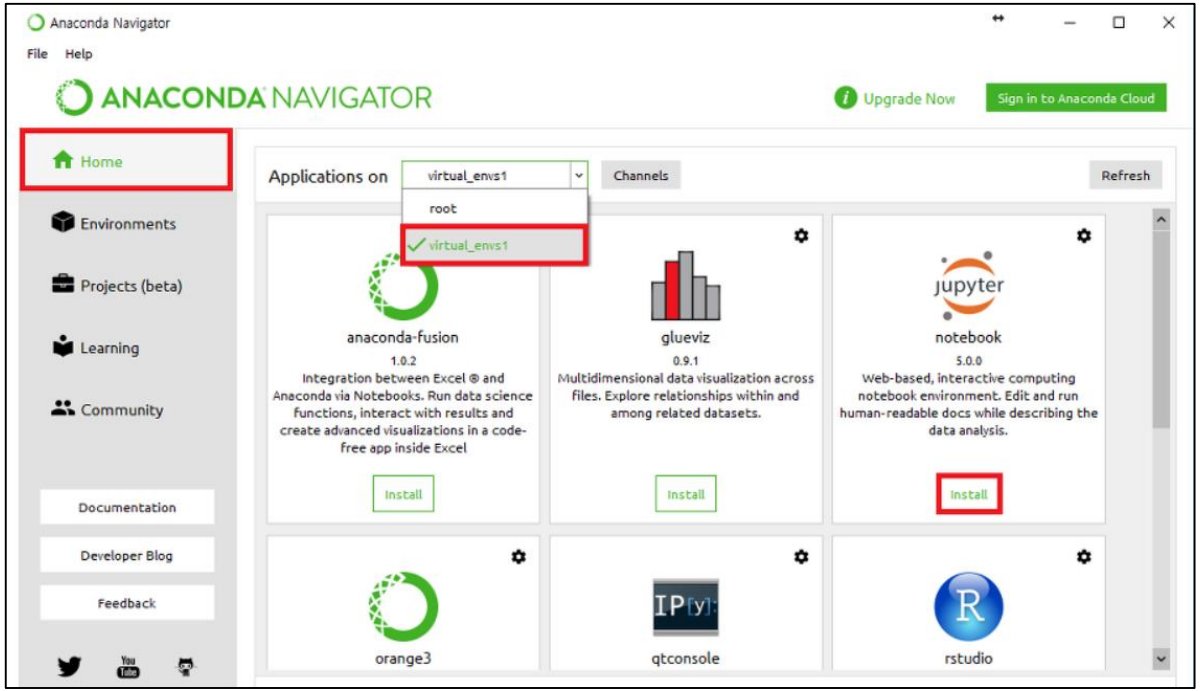
pip install --ignore-installed --upgrade https://storage.googleapis.com/tensorflow/windows/gpu/tensorflow_gpu-1.0.1-cp35-cp35m-win_amd64.whl



TensorBoard

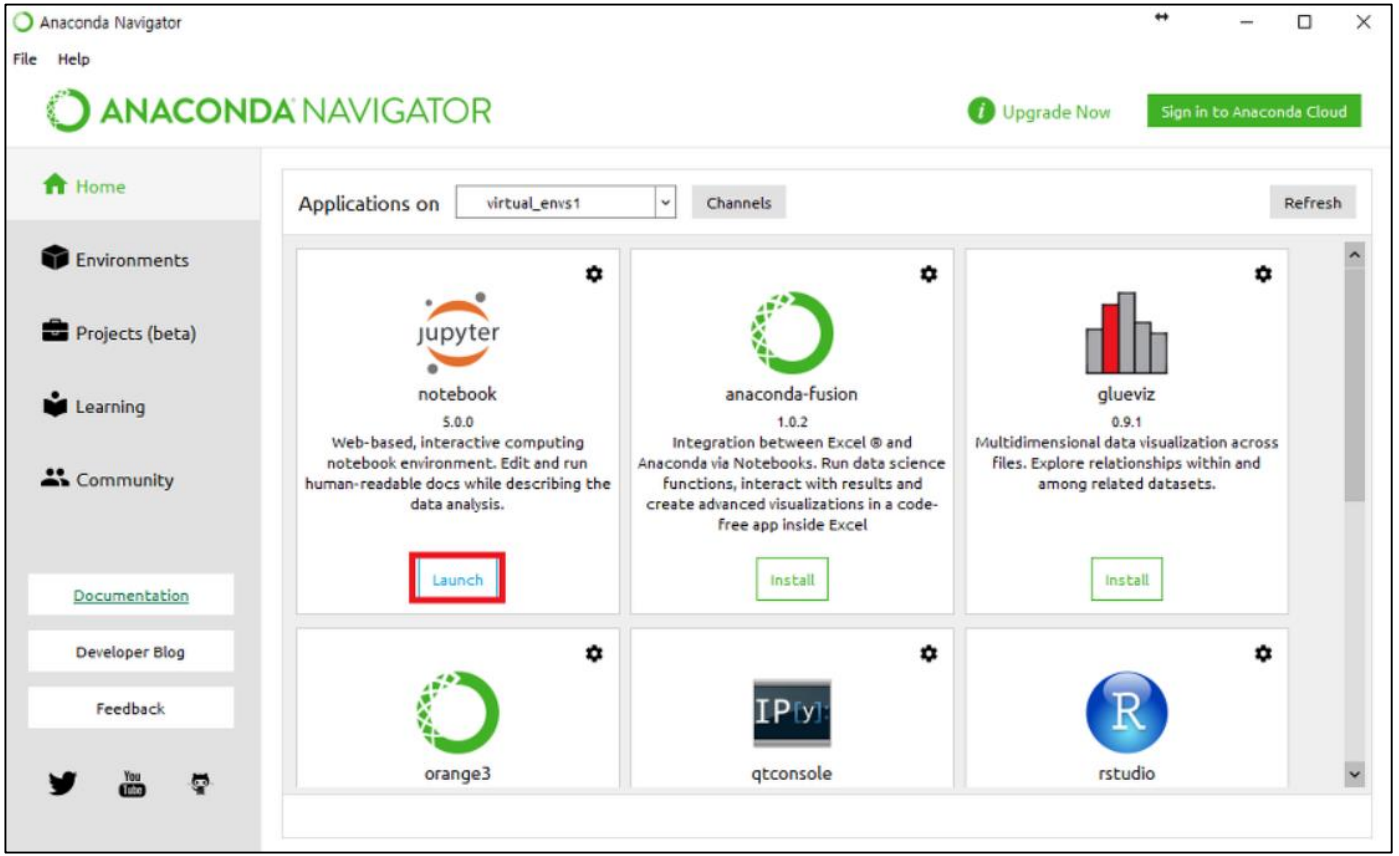
- 설치 완료 후 cmd창 끄고 Navigator로 이동

- 왼쪽 사이드바의 Home 클릭
  → Applications on [virtual_envs1] 클릭
  → 아래 프로그램들에서 jupyter install 클릭



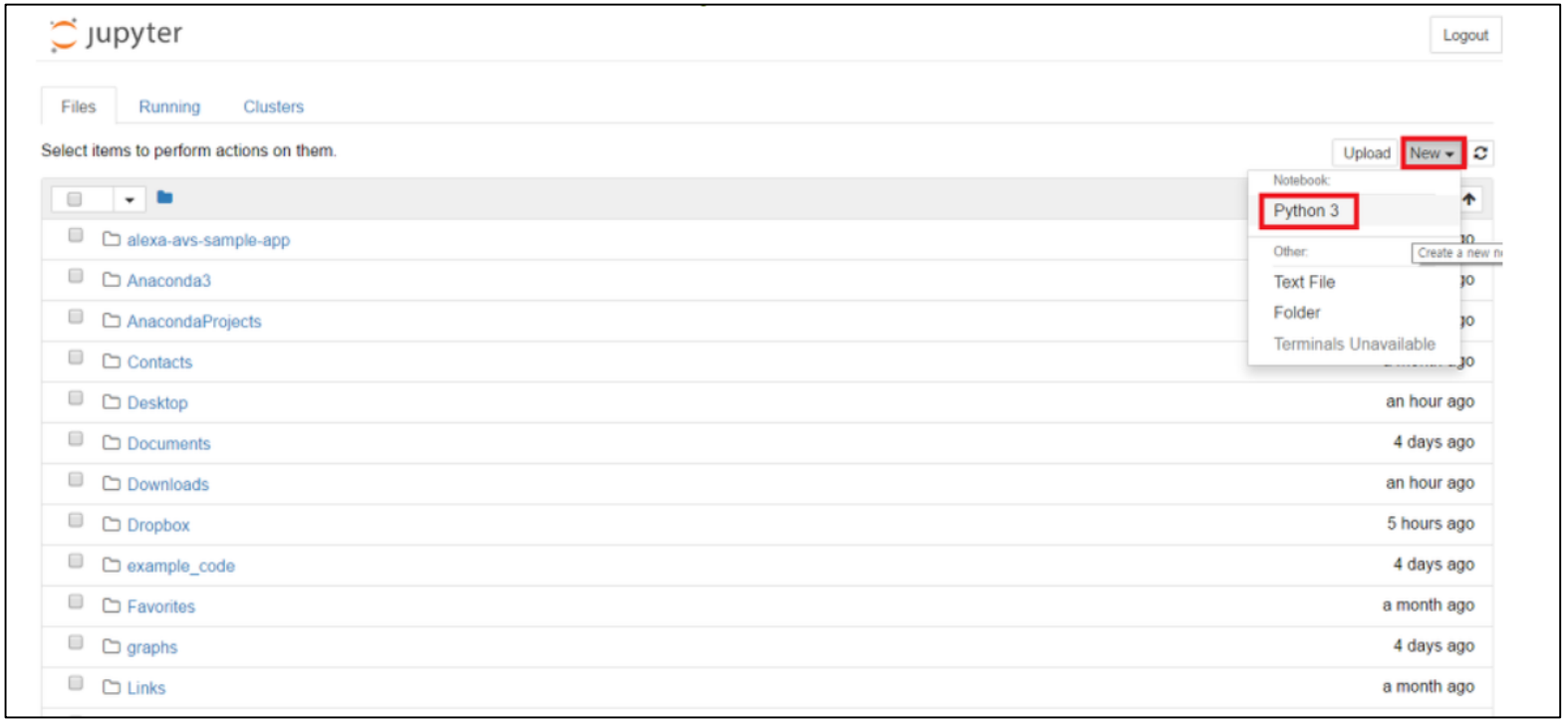
TensorBoard

install 후 Launch 클릭



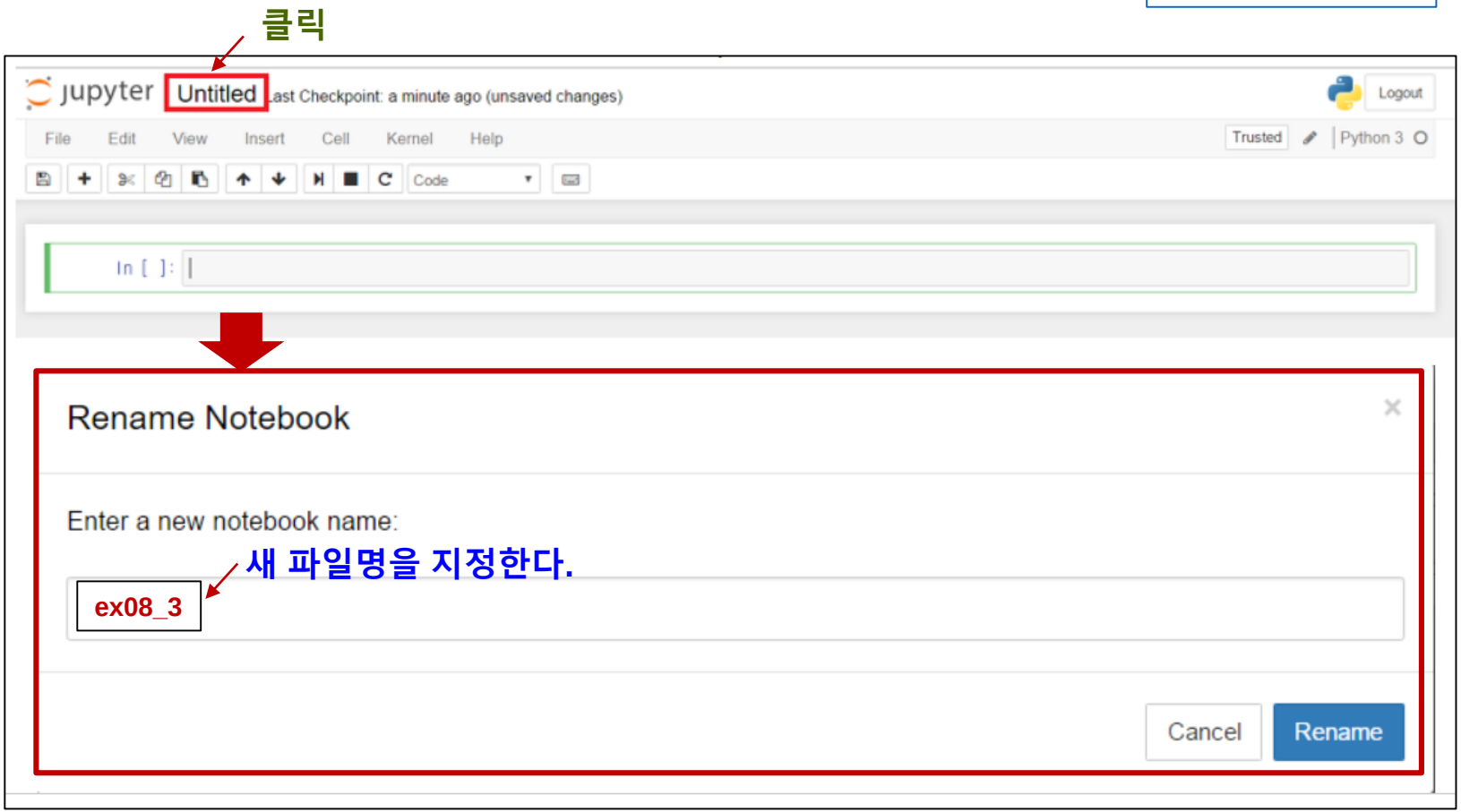
TensorBoard

jupyter가 켜진 후
오른쪽 상단에서 New 클릭,
Python 3 클릭



TensorBoard

ex08_3.ipynb



TensorBoard

ex08_3.ipynb

코드 입력

```
import tensorflow as tf

a = tf.constant(5, name='a')
b = tf.constant(7, name='b')

absum = tf.add(a,b, name='absum')
a_absum_sum = tf.add(a,absum, name='a_absum_sum')

with tf.Session() as sess:
        sess.run(tf.global_variables_initializer())
        print(sess.run(absum))
        print(sess.run(a_absum_sum))
        writer = tf.summary.FileWriter('./graphs/ex08_3')
        writer.add_graph(sess.graph)
```

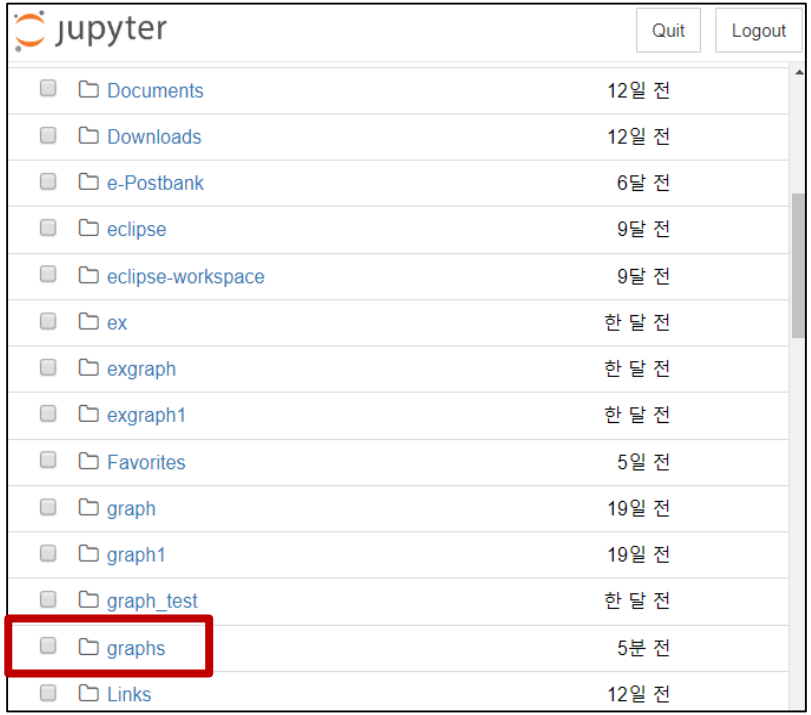
그래프를 저장할 디렉토리명 (임의의 이름으로 설정)

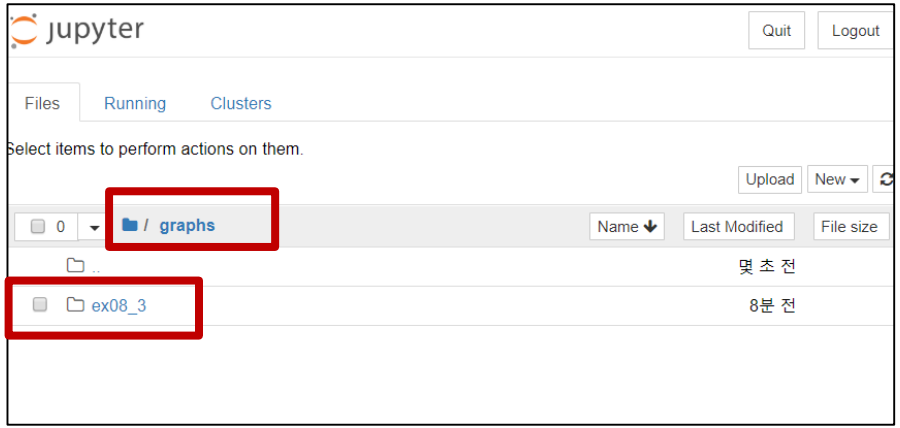
```
12
17
```

TensorBoard

ex08_3.ipynb Notes

- jupyter home으로 다시 이동, graphs/ex08_3 디렉토리가 생성되었는지 확인
  (graphs/ex08_3 디렉토리가 생성 안 되었다면 실행이 안된 것 → 다시 점검 필요)

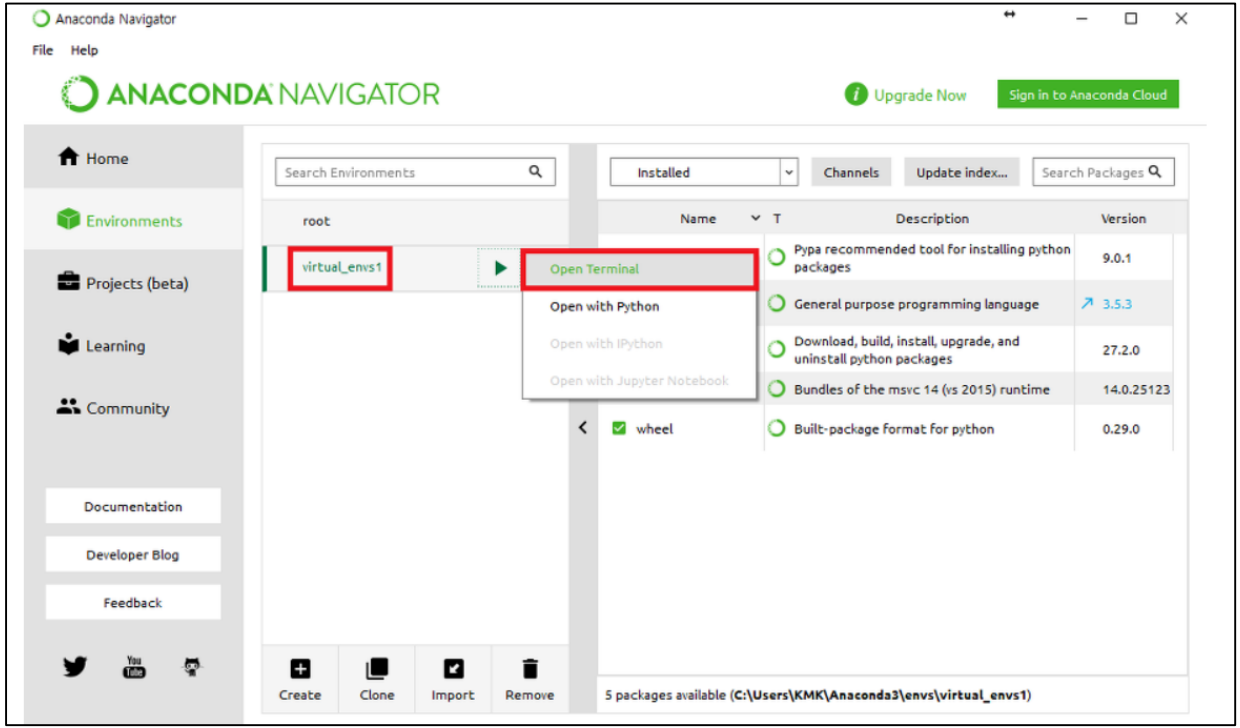




TensorBoard

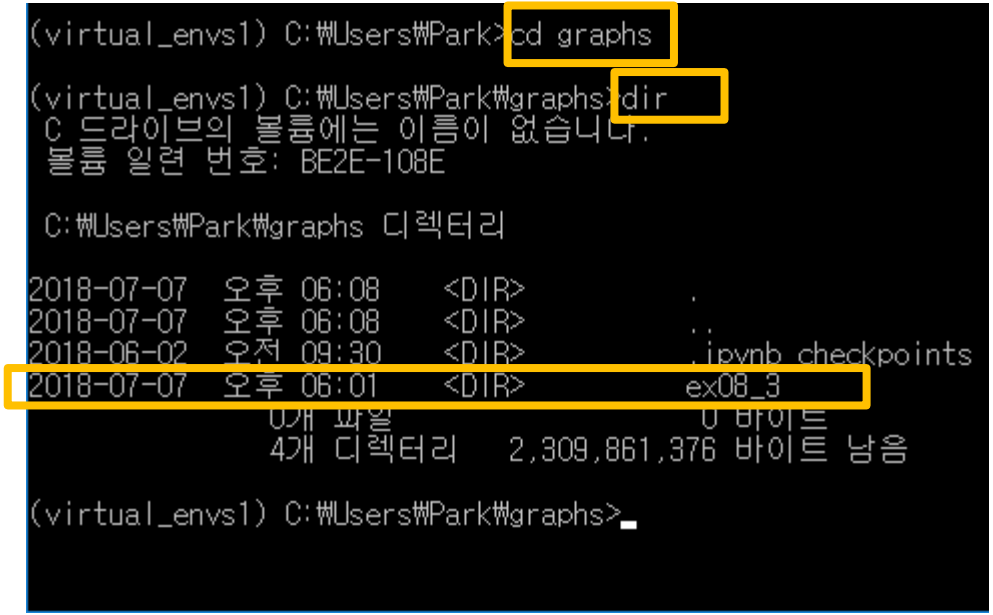
ex08_3.ipynb Notes

- 다시 Navigator로 이동
- Environments의 virtual_envs1 클릭
  → 화살표 클릭
  → Open Terminal 클릭



TensorBoard

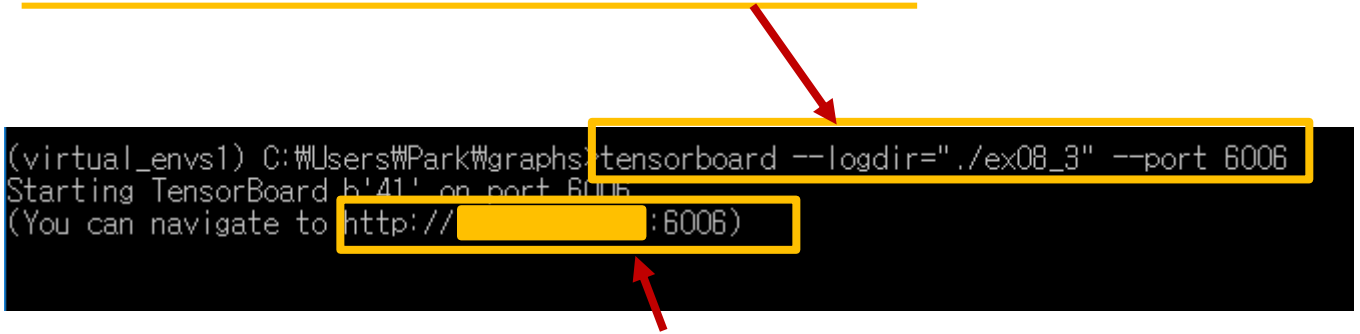
graphs/ex08_3 디렉토리가 생성되었는지 다시 확인



TensorBoard

graphs 디렉토리에서

tensorboard --logdir="./ex08_3" --port 6006 입력



드래그하여 copy (ctrl + c)

TensorBoard

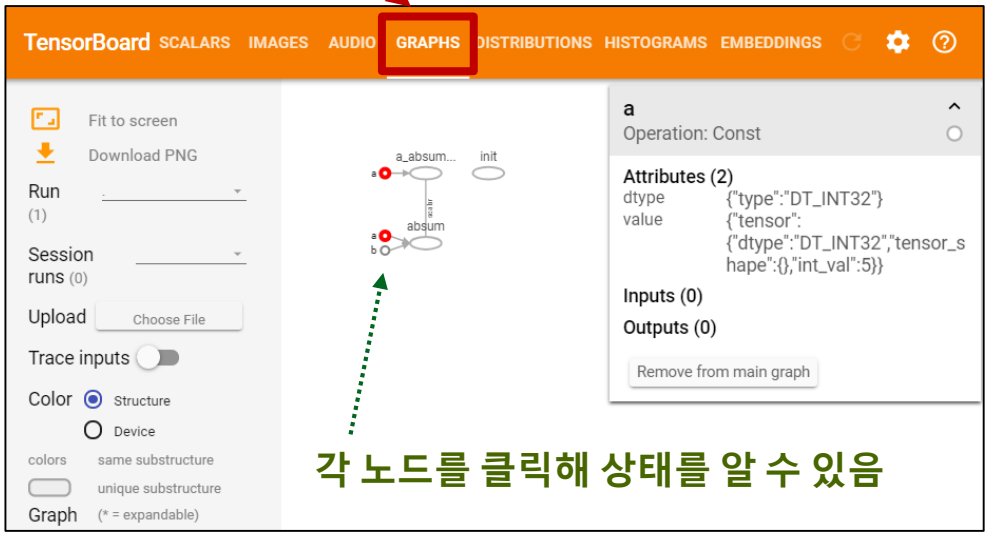
ex08_3.ipynb Notes

웹 브라우저 실행(크롬 또는 익스플로러 낮은 버전은 지원 안 됨.)

- copy한 주소를 브라우저 주소 입력 창에 paste (ctrl + v) → http://***.***.**.*:6006 주소 입력한다는 뜻

```
import tensorflow as tf

a = tf.constant(5, name='a')
b = tf.constant(7, name='b')

absum = tf.add(a,b, name='absum')
a_absum_sum = tf.add(a,absum, name='a_absum_sum')

with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    print(sess.run(absum))
    print(sess.run(a_absum_sum))
    writer = tf.summary.FileWriter('./graphs/ex08_3')
    writer.add_graph(sess.graph)
```

GRAPHS 클릭

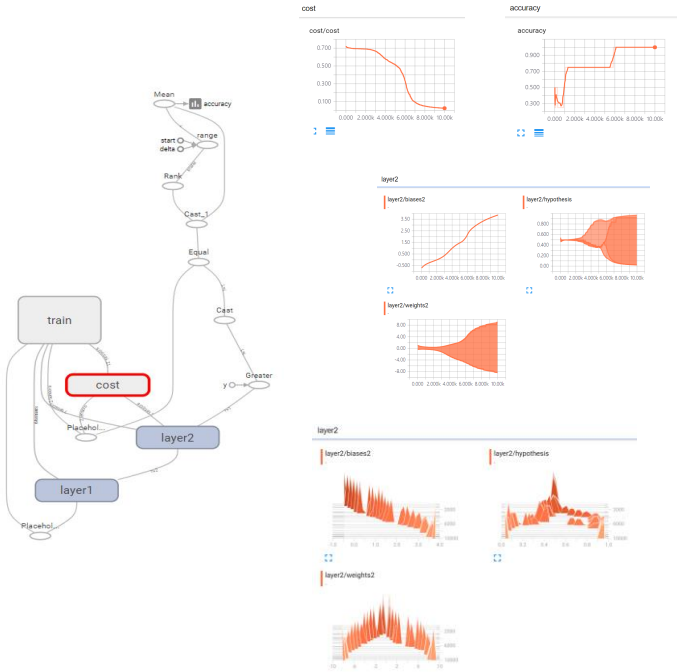


각 노드를 클릭해 상태를 알 수 있음

Old fashion: print, print, print

9400 0.0151413 [array([[6.21692038, 6.05913448],
[-6.33773184, -5.75189114]], dtype=float32), array([[9.93581772],
[-9.43034935]], dtype=float32)]
9500 0.014909 [array([[6.22498751, 6.07049847],
[-6.34637976, -5.76352596]], dtype=float32), array([[9.96414757],
[-9.45942593]], dtype=float32)]
9600 0.0146836 [array([[6.23292685, 6.08166742],
[-6.35489035, -5.77496052]], dtype=float32), array([[9.99207973],
[-9.48807526]], dtype=float32)]
9700 0.0144647 [array([[6.24074268, 6.09264851],
[-6.36326933, -5.78619957]], dtype=float32), array([[10.01962471],
[-9.51631165]], dtype=float32)]
9800 0.0142521 [array([[6.24843407, 6.10344648],
[-6.37151814, -5.79724932]], dtype=float32), array([[10.04679298],
[-9.54414845]], dtype=float32)]
9900 0.0140456 [array([[6.25601053, 6.11406422],
[-6.3796401 , -5.80811596]], dtype=float32), array([[10.07359505],
[-9.57159519]], dtype=float32)]
10000 0.0138448 [array([[6.26347113, 6.12451124],
[-6.38764334, -5.81880617]], dtype=float32), array([[10.10004139],
[-9.59866238]], dtype=float32)]

New way : TensorBoard



5 steps of using TensorBoard

1. 어떤 tensor를 그래프로 기록할지 결정하는 단계

```
w2_hist = tf.summary.histogram("weights2", W2)
cost_summ = tf.summary.scalar("cost", cost)
```

2. 1단계의 summary 정보를 merge 하는 단계

```
summary = tf.summary.merge_all()
```

3. Writer 에 디렉토리를 생성하고, graph를 추가하는 단계

```
# Create summary writer
writer = tf.summary.FileWriter('./graphs/xor_tb')
writer.add_graph(sess.graph)
```

4. 실행 단계 : 2단계의 summary merge 와 add_summary 를 실행

```
s, _ = sess.run([summary, optimizer], feed_dict=feed_dict)
writer.add_summary(s, global_step=global_step)
```

5. TensorBoard를 Launch 하는 단계

```
Ex)  tensorboard --logdir="./graphs/xor_tb" --port 6006
```

ex08_2(tb1).ipynb (1/4)

```
import tensorflow as tf
import numpy as np
x_data = np.array([[0, 0], [0, 1], [1, 0], [1, 1]], dtype=np.float32)
y_data = np.array([[0], [1], [1], [0]], dtype=np.float32)
X = tf.placeholder(tf.float32)
Y = tf.placeholder(tf.float32)

with tf.name_scope("layer1"):
    W1 = tf.Variable(tf.random_normal([2, 2]), name='weight1')
    b1 = tf.Variable(tf.random_normal([2]), name='bias1')
    layer1 = tf.sigmoid(tf.matmul(X, W1) + b1)

    w1_hist = tf.summary.histogram("weights1", W1) # 1단계 : W1 히스토그램 그래프 결정
    b1_hist = tf.summary.histogram("biases1", b1)  # 1단계 : b1 히스토그램 그래프 결정
    layer1_hist = tf.summary.histogram("layer1", layer1)  # 1단계 : layer1 히스토그램 그래프 결정

with tf.name_scope("layer2"):
    W2 = tf.Variable(tf.random_normal([2, 1]), name='weight2')
    b2 = tf.Variable(tf.random_normal([1]), name='bias2')
    hypothesis = tf.sigmoid(tf.matmul(layer1, W2) + b2)

    w2_hist = tf.summary.histogram("weights2", W2)  # 1단계 : W2 히스토그램 그래프 결정
    b2_hist = tf.summary.histogram("biases2", b2)   # 1단계 : b2 히스토그램 그래프 결정
    hypothesis_hist = tf.summary.histogram("hypothesis", hypothesis)  # 1단계 : hypothesis 히스토그램 그래프 결정
```

ex08_2(tb1).ipynb (2/4)

```
# cost/loss function
with tf.name_scope("cost"):
    cost = -tf.reduce_mean(Y * tf.log(hypothesis) + (1 - Y) * tf.log(1 - hypothesis))

    cost_summ = tf.summary.scalar("cost", cost)  # 1단계 : cost 스칼라 그래프 결정

with tf.name_scope("train"):
    train = tf.train.GradientDescentOptimizer(learning_rate=0.1).minimize(cost)

# Accuracy computation
# True if hypothesis>0.5 else False
predicted = tf.cast(hypothesis > 0.5, dtype=tf.float32)
accuracy = tf.reduce_mean(tf.cast(tf.equal(predicted, Y), dtype=tf.float32))

accuracy_sum = tf.summary.scalar("accuracy", accuracy) # 1단계 : accuracy 스칼라 그래프 결정
```

ex08_2(tb1).ipynb (3/4)

```
# Launch graph
with tf.Session() as sess:
    # Initialize TensorFlow variables
    merged_summary = tf.summary.merge_all()  #  2단계 : 1단계의 summary 정보를 merge 하는 단계

    writer = tf.summary.FileWriter("./graphs/ex08_2_tb")   #  3단계 : 디렉토리를 생성
    writer.add_graph(sess.graph)   #  3단계 : graph를 추가하는 단계

    sess.run(tf.global_variables_initializer())

    for step in range(10001):
        # 4단계 : summary merge 실행 단계
        summary, _ = sess.run([merged_summary, train], feed_dict={X: x_data, Y: y_data})

        # 4단계 : add_summary 실행 단계
        writer.add_summary(summary, global_step=step)

        if step % 100 == 0:
            print("step : ", step, "cost :", sess.run(cost, feed_dict={X: x_data, Y: y_data}))
            print("W1 : \n", sess.run(W1), "\nb1: \n", sess.run(b1))
            print("Layer 1: \n", sess.run(layer1, feed_dict={X: x_data, Y: y_data}))
            print("W2 : \n", sess.run(W2), "\nb2: \n", sess.run(b2))
            print("hypothesis : \n", sess.run(hypothesis, feed_dict={X:x_data}),"\n" , "="*50)

    # Accuracy report
p, a = sess.run([predicted, accuracy],feed_dict={X: x_data, Y: y_data})
print("● Correct: \n", p, "\n● Accuracy:  \n", a)
```

ex08_2(tb1).ipynb (4/4)

5 `tensorboard --logdir="./ex08_2_tb" --port 6006`

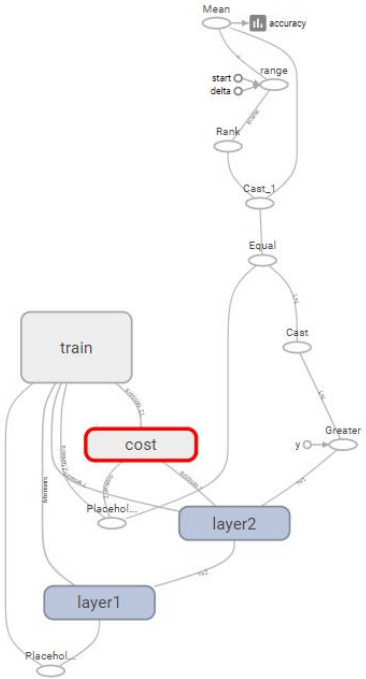
드래그하여 copy (ctrl + c)

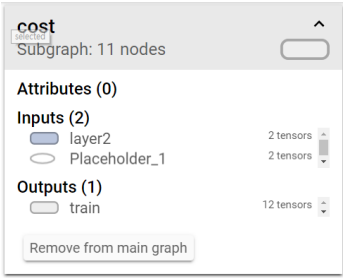
New way!

ex08_2(tb1).ipynb Notes

New way!

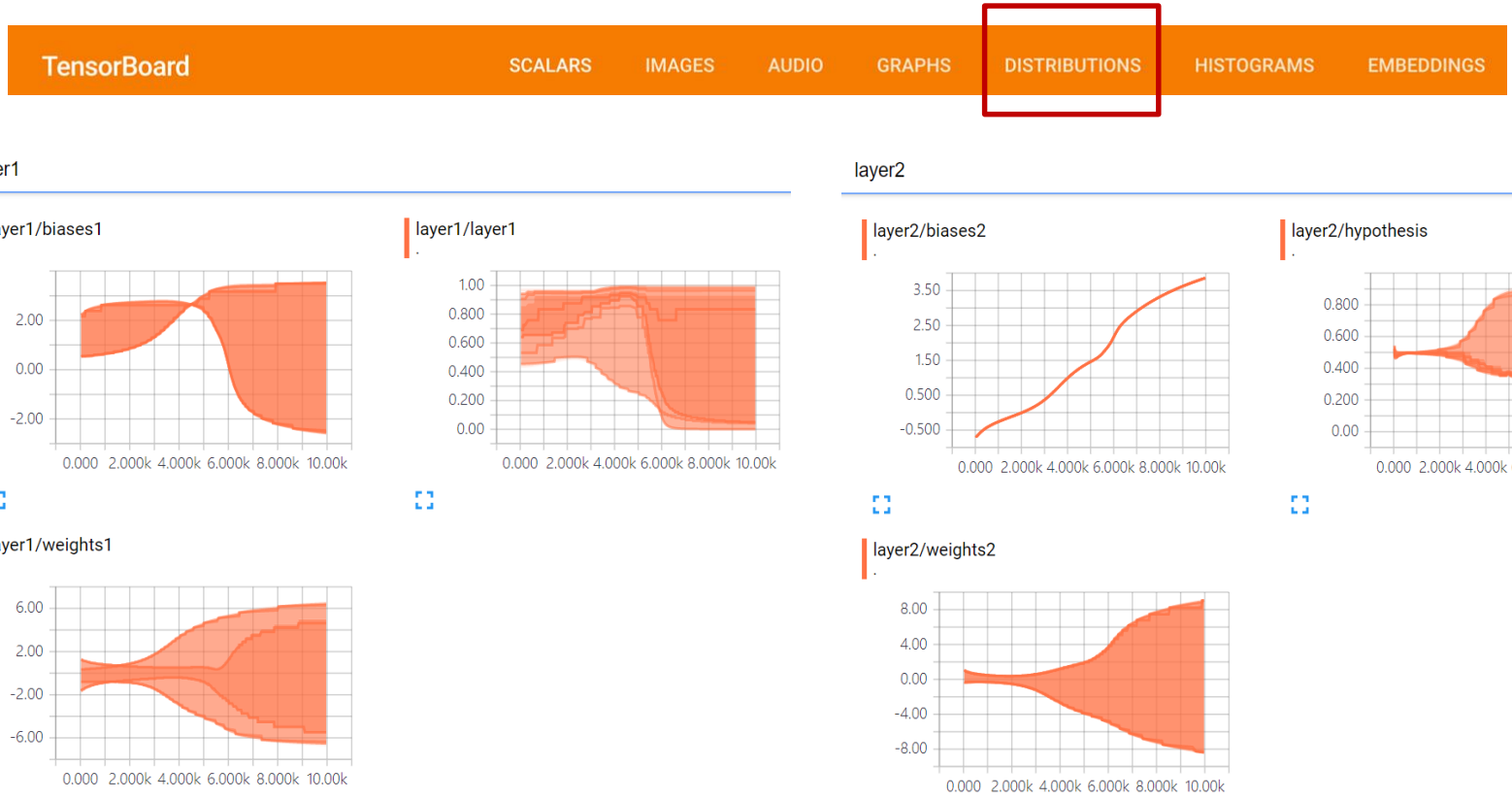
ex08_2(tb1).ipynb Notes





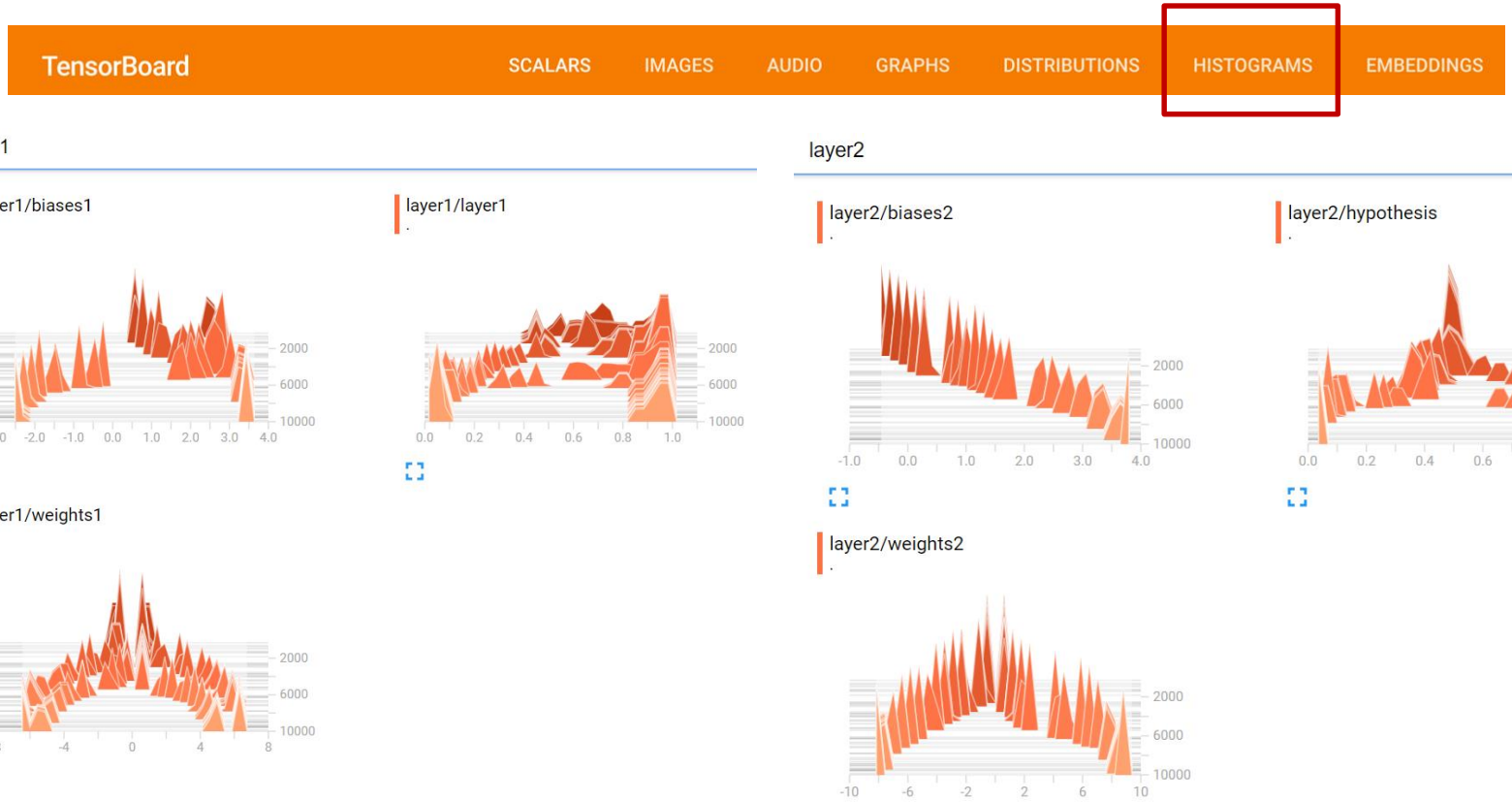
New way!

ex08_2(tb1).ipynb Notes



New way!

ex08_2(tb1).ipynb Notes



ex08_2(tb2).ipynb (1/4)

```
import tensorflow as tf
import numpy as np
x_data = np.array([[0, 0], [0, 1], [1, 0], [1, 1]], dtype=np.float32)
y_data = np.array([[0], [1], [1], [0]], dtype=np.float32)
X = tf.placeholder(tf.float32)
Y = tf.placeholder(tf.float32)

with tf.name_scope("layer1"):
    W1 = tf.Variable(tf.random_normal([2, 2]), name='weight1')
    b1 = tf.Variable(tf.random_normal([2]), name='bias1')
    layer1 = tf.sigmoid(tf.matmul(X, W1) + b1)

    w1_hist = tf.summary.histogram("weights1", W1) # 1단계 : W1 히스토그램 그래프 결정
    b1_hist = tf.summary.histogram("biases1", b1)  # 1단계 : b1 히스토그램 그래프 결정
    layer1_hist = tf.summary.histogram("layer1", layer1)  # 1단계 : layer1 히스토그램 그래프 결정

with tf.name_scope("layer2"):
    W2 = tf.Variable(tf.random_normal([2, 1]), name='weight2')
    b2 = tf.Variable(tf.random_normal([1]), name='bias2')
    hypothesis = tf.sigmoid(tf.matmul(layer1, W2) + b2)

    w2_hist = tf.summary.histogram("weights2", W2)  # 1단계 : W2 히스토그램 그래프 결정
    b2_hist = tf.summary.histogram("biases2", b2)   # 1단계 : b2 히스토그램 그래프 결정
    hypothesis_hist = tf.summary.histogram("hypothesis", hypothesis)  # 1단계 : hypothesis 히스토그램 그래프 결정
```

ex08_2(tb2).ipynb (2/4)

```
# cost/loss function
with tf.name_scope("cost"):
    cost = -tf.reduce_mean(Y * tf.log(hypothesis) + (1 - Y) * tf.log(1 - hypothesis))
```

1

```
    cost_summ = tf.summary.scalar("cost", cost)  # 1단계 : cost 스칼라 그래프 결정
```

```
with tf.name_scope("train"):
        train_a = tf.train.GradientDescentOptimizer(learning_rate=0.1).minimize(cost)
        train_b = tf.train.GradientDescentOptimizer(learning_rate=0.01).minimize(cost)

# Accuracy computation
# True if hypothesis>0.5 else False
predicted = tf.cast(hypothesis > 0.5, dtype=tf.float32)
accuracy = tf.reduce_mean(tf.cast(tf.equal(predicted, Y), dtype=tf.float32))
```

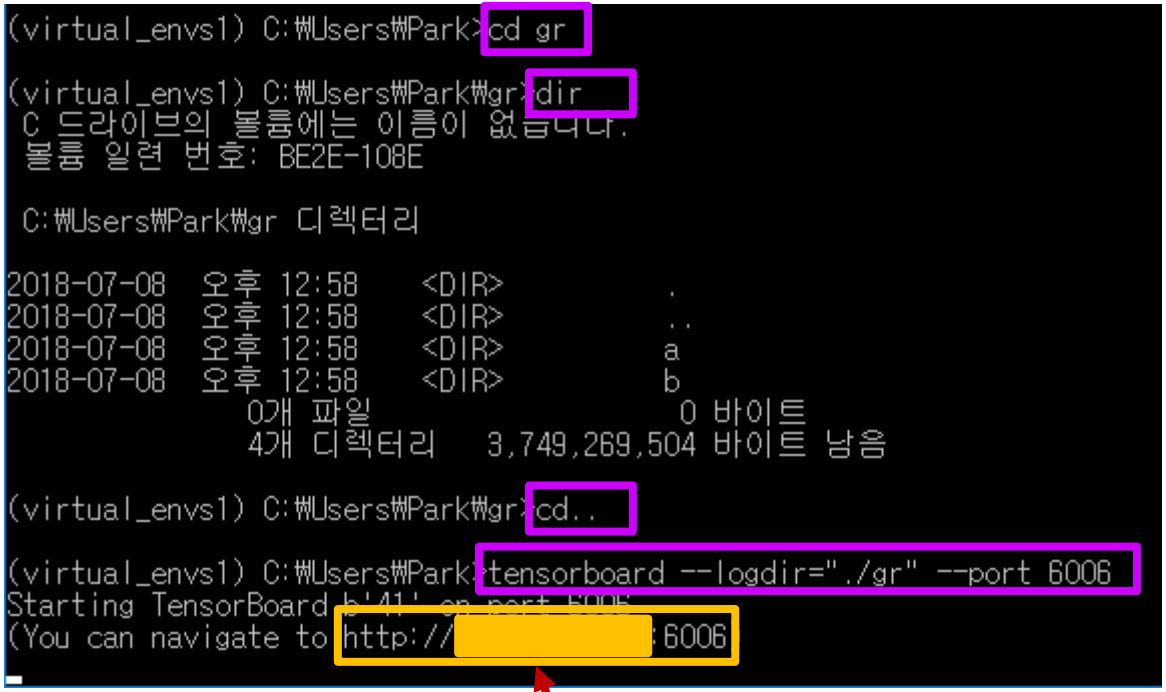
1

```
accuracy_sum = tf.summary.scalar("accuracy", accuracy) # 1단계 : accuracy 스칼라 그래프 결정
```

```
# Launch graph
with tf.Session() as sess:
    # Initialize TensorFlow variables
    merged_summary_a = tf.summary.merge_all()  #  2단계 : 1단계의 summary 정보를 merge 하는 단계
    merged_summary_b = tf.summary.merge_all()  #  2단계 : 1단계의 summary 정보를 merge 하는 단계
    writer_a = tf.summary.FileWriter("./gr/a")   #  3단계 : 디렉토리를 생성
    writer_b = tf.summary.FileWriter("./gr/b")   #  3단계 : 디렉토리를 생성
    writer_a.add_graph(sess.graph)   #  3단계 : graph를 추가하는 단계
    writer_b.add_graph(sess.graph)   #  3단계 : graph를 추가하는 단계

    sess.run(tf.global_variables_initializer())
    for step in range(10001):
        summary_a, _ = sess.run([merged_summary_a, train_a], feed_dict={X: x_data, Y: y_data})  # 4단계 : summary merge 실행 단계
        writer_a.add_summary(summary_a, global_step=step) # 4단계 : add_summary 실행 단계
        if step % 100 == 0:
            print("<< learning_rate=0.1 >> step : ", step, "\tcost :", sess.run(cost, feed_dict={X: x_data, Y: y_data}))
            print("W1 : \n", sess.run(W1), "\nb1: \n", sess.run(b1))
            print("Layer 1: \n", sess.run(layer1, feed_dict={X: x_data, Y: y_data}))
            print("W2 : \n", sess.run(W2), "\nb2: \n", sess.run(b2))
            print("hypothesis : \n", sess.run(hypothesis, feed_dict={X:x_data}),"\n" , "="*70)
    p, a = sess.run([predicted, accuracy],feed_dict={X: x_data, Y: y_data})
    print("● Correct: \n", p, "\n● Accuracy:  \n", a)

    print("★"*35)
    for step in range(10001):
        summary_b, _ = sess.run([merged_summary_b, train_b], feed_dict={X: x_data, Y: y_data})  # 4단계 : summary merge 실행 단계
        writer_b.add_summary(summary_b, global_step=step) # 4단계 : add_summary 실행 단계
        if step % 100 == 0:
            print("<< learning_rate=0.01 >> step : ", step, "\tcost :", sess.run(cost, feed_dict={X: x_data, Y: y_data}))
            print("W1 : \n", sess.run(W1), "\nb1: \n", sess.run(b1))
            print("Layer 1: \n", sess.run(layer1, feed_dict={X: x_data, Y: y_data}))
            print("W2 : \n", sess.run(W2), "\nb2: \n", sess.run(b2))
            print("hypothesis : \n", sess.run(hypothesis, feed_dict={X:x_data}),"\n" , "="*70)
    p, a = sess.run([predicted, accuracy],feed_dict={X: x_data, Y: y_data})
    print("● Correct: \n", p, "\n● Accuracy:  \n", a)
```

ex08_2(tb2).ipynb (4/4)

5 tensorboard --logdir="./gr" --port 6006



드래그하여 copy (ctrl + c)

ex08_2.ipynb Notes

```
<< learning_rate=0.1 >> step :  0    cost : 0.69446325
W1 :
 [[-0.25134176  0.6209769 ]
 [-1.5642004   0.45754623]]
b1:
 [-0.01432582  1.1404738 ]
Layer 1:
 [[0.49641857 0.7577666 ]
 [0.17100431 0.8317415 ]
 [0.43397105 0.85339123]
 [0.13825399 0.9019425 ]]
W2 :
 [[-1.107915 ]
 [-0.7879375]]
b2:
 [1.040976]
hypothesis :
 [[0.4735033 ]
 [0.5488827 ]
 [0.47196794]
 [0.5441666 ]]
 ==================================================
<< learning_rate=0.1 >> step :  100  cost : 0.6900866
W1 :
 [[-0.44198576  0.5848986 ]
 [-1.5781376   0.42583543]]
b1:
 [-0.05923467  1.1456093 ]
Layer 1:
 [[0.48519567 0.758708  ]
 [0.16282295 0.82798946]
 [0.3772539  0.8494774 ]
 [0.11111935 0.8962601 ]]
W2 :
 [[-1.1186256]
 [-0.8045731]]
b2:
 [0.99852866]
hypothesis :
 [[0.46141198]
 [0.53748274]
 [0.47328946]
 [0.53820556]]
.
.
```

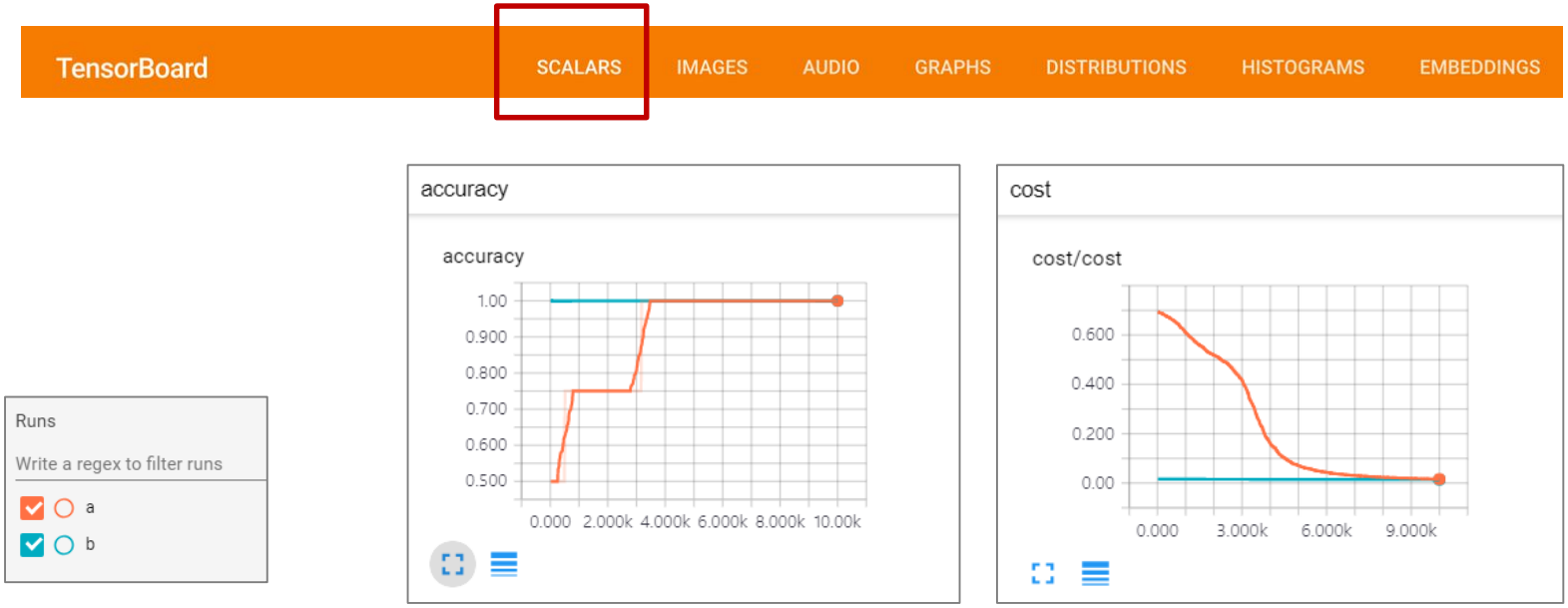
```
<< learning_rate=0.1 >> step :  10000 cost : 0.015991995
W1 :
 [[-6.636684   4.6767774]
 [-6.6102934  4.6733856]]
b1:
 [ 2.6041815 -7.2471156]
Layer 1:
 [[9.3113023e-01 7.1171887e-04]
 [1.7878572e-02 7.0848368e-02]
 [1.7421037e-02 7.1071967e-02]
 [2.3871640e-05 8.9119893e-01]]
W2 :
 [[ -9.980243]
 [-10.138073]]
b2:
 [4.935925]
hypothesis :
 [[0.01256499]
 [0.9826936 ]
 [0.9827328 ]
 [0.01631287]]
 ==================================================
• Correct:
 [[0.]
 [1.]
 [1.]
 [0.]]
• Accuracy:
 1.0

★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★
<< learning_rate=0.01 >> step :  0    cost : 0.015991706
W1 :
 [[-6.636691   4.6767883]
 [-6.610301   4.673397 ]]
b1:
 [ 2.6041868 -7.2471333]
Layer 1:
 [[9.3113053e-01 7.1170629e-04]
 [1.7878538e-02 7.0847958e-02]
 [1.7421005e-02 7.1071528e-02]
 [2.3871411e-05 8.9119953e-01]]
```

```
W2 :
 [[ -9.98027  ]
 [-10.1381035]]
b2:
 [4.9359393]
hypothesis :
 [[0.01256481]
 [0.982694  ]
 [0.982733  ]
 [0.01631257]]
 ==========================================
<< learning_rate=0.01 >> step : 10000 cost : 0.013763509
W1 :
 [[-6.7051926  4.779451 ]
 [-6.680719   4.7762365]]
b1:
 [ 2.6513317 -7.4070253]
Layer 1:
 [[9.3409300e-01 6.0660538e-04]
 [1.7474433e-02 6.7183003e-02]
 [1.7059177e-02 6.7384720e-02]
 [2.1778184e-05 8.9554363e-01]]
W2 :
 [[-10.239922]
 [-10.418846]]
b2:
 [5.0685163]
hypothesis :
 [[0.0109561 ]
 [0.9850739 ]
 [0.9851055 ]
 [0.01389494]]
 ==========================================
• Correct:
 [[0.]
 [1.]
 [1.]
 [0.]]
• Accuracy:
 1.0
```

Multiple runs

`learning_rate=0.1` VS `learning_rate=0.01`

ex08_2(tb2).ipynb Notes

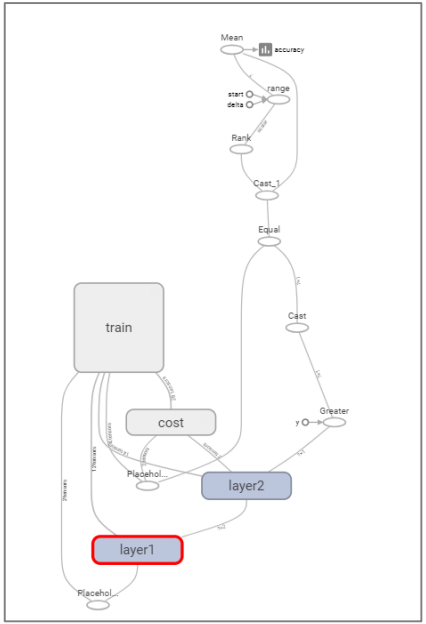


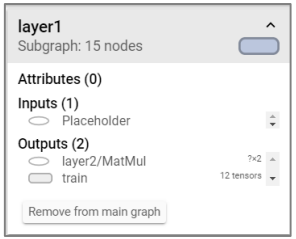
Multiple runs

`learning_rate=0.1` VS `learning_rate=0.01`

ex08_2(tb2).ipynb Notes

TensorBoard SCALARS IMAGES AUDIO GRAPHS DISTRIBUTIONS HISTOGRAMS EMBEDDINGS

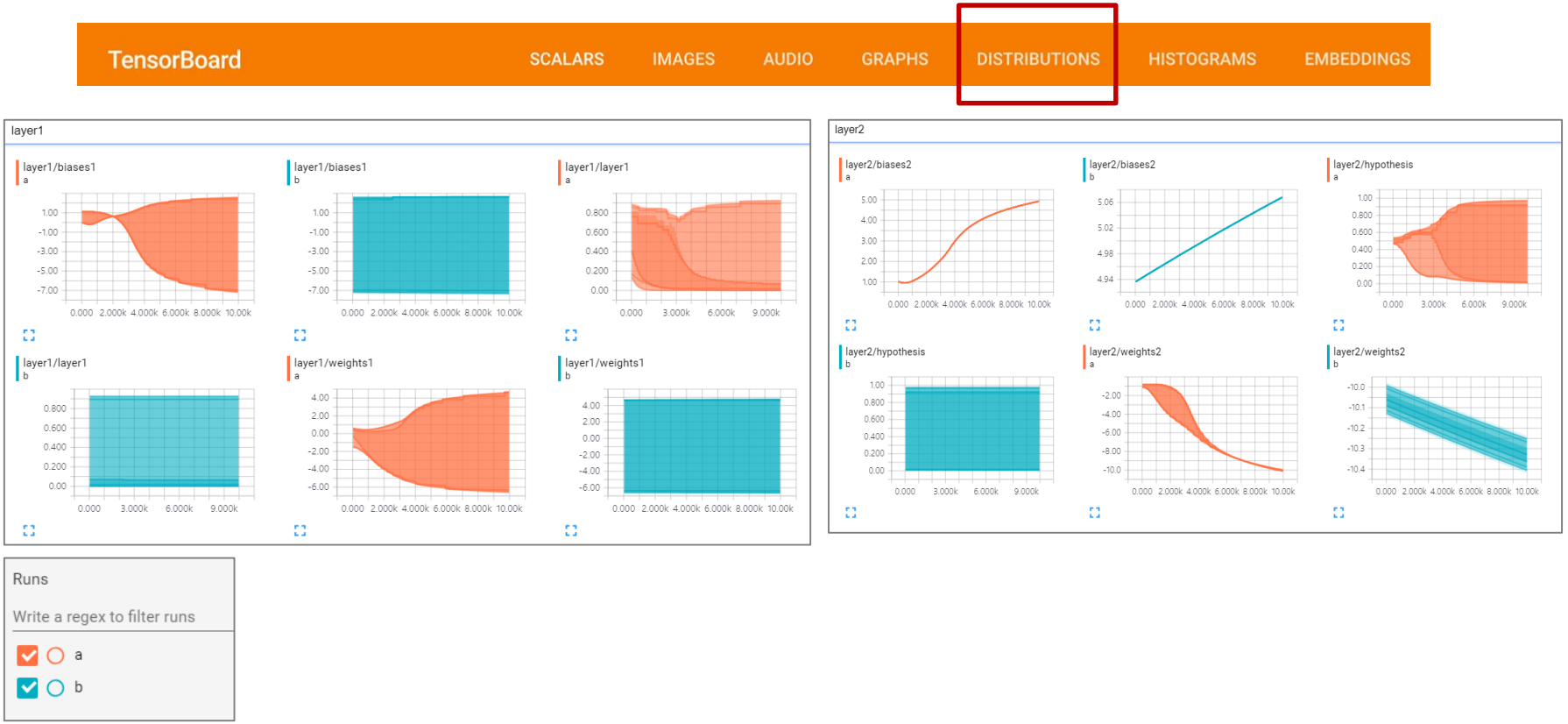




Multiple runs

`learning_rate=0.1` VS `learning_rate=0.01`

ex08_2(tb2).ipynb Notes



Multiple runs

`learning_rate=0.1` VS `learning_rate=0.01`

ex08_2(tb2).ipynb Notes