Python

텐서플로우 맛보기

텐서플로우 설치

MNIST DATA SET 구조

텐서보드(TensorBoard)

텐서플로우(TensorFlow)

- 텐서플로우(TensorFlow™)는 데이터 플로우 그래프(Data flow graph)를 사용하여 수치 연산을 하는 오픈소스 소프트웨어 라이브러리.
- 그래프의 노드(Node)는 수치 연산을 나타내고 엣지(edge)는 노드 사이를 이동하는 다차원 데이터 배열(텐서,tensor)를 나타냅니다.
- 텐서플로우의 동작
  - 연산은 graph로 표현
  - graph는 Session내에서 실행
  - 데이터는 tensor로 표현

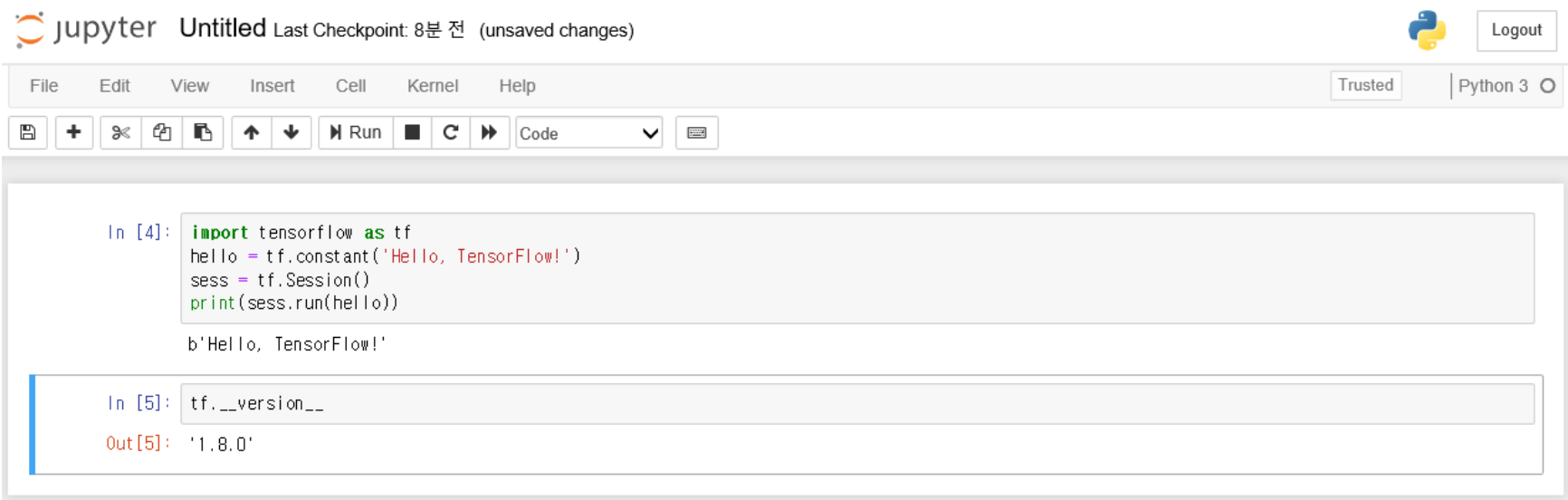
Installing TensorFlow on Windows

텐서플로우 설치

- https://www.tensorflow.org/install/install_windows
- An open source machine learning framework for everyone
- TensorFlow™ is an open source software library for high performance numerical computation.
- requirements
  - 64-bit, x96 desktops or laptops
  - Windows 7 or later
- TensorFlow to install
  - TensorFlow with CPU support only
  - TensorFlow with GPU support : NVIDIA® GPU
  - Installing with native pip : Python 3.6.x 64-bit from python.org
- Installing with native pip(included by default with the Python binary installers)
  - **pip3 install --upgrade tensorflow**
  - pip3 install --upgrade tensorflow-gpu**(gpu가 있는 경우에만 설치)**

Hello, TensorFlow!

텐서플로우 설치



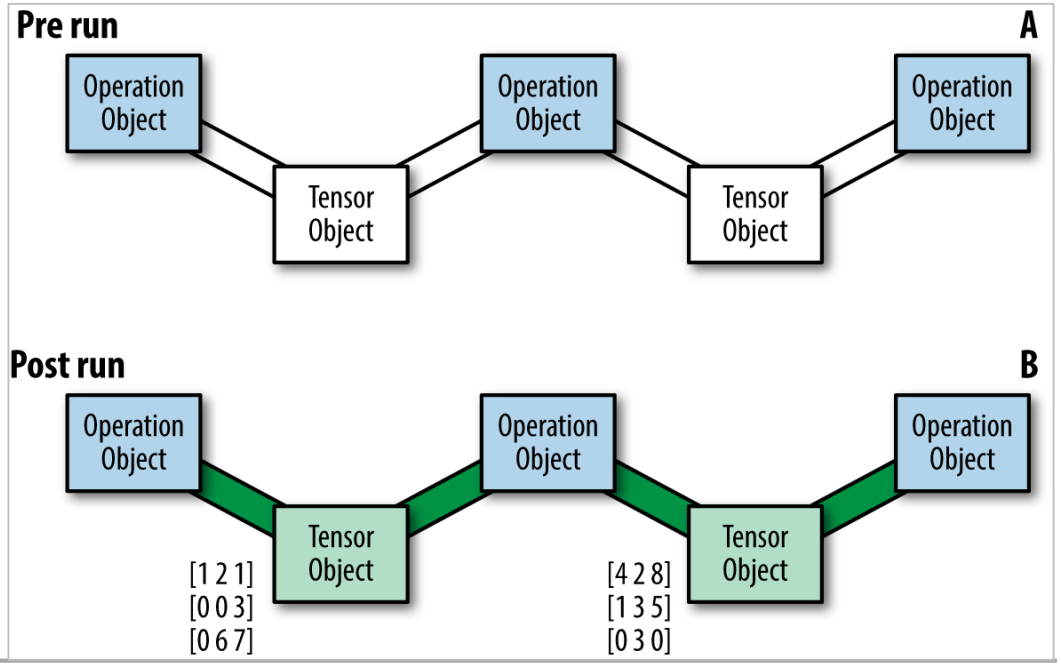
- b'String' '**b**' indicates *Bytes literals*.
  http://stackoverflow.com/questions/6269765/
- Session 객체 : 외부의 텐서플로 연산 메커니즘에 대한 인터페이스 역할
- Tensorflow 설치후 아래의 오류발생시 처리
  - Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2
    - import os
    - os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'

연산그래프(Computation Graph)

- 텐서(Tensor) : *n*차원 배열(*n*-dimensional arrays)을 가리키는 수학용어

1 × 1	scalar	rank 0	3	shape[]
1 × n	vector	rank 1	[1.,2.,3.]	shape[3]
n × n	matrix	rank 2	[[1.,2.,3.],[4.,5.,6.]]	shape[2×3]
n × n × n	tensor	rank 3	[[[1.,2.,3.]],[[4.,5.,6.]]]	Shape[2×1×3]

- 텐서플로우의 텐서 : 다차원 배열, 벡터, 행렬, 스칼라 구분하지 않고 그래프에 전달되는 모든 단위 데이터
- 텐서플로우는 연산 그래프 구조를 통해 노드에서 노드로 이동(Flow).
  - 세션실행전 : 연산 과정을 그래프 형태로 생성(표현)
  - 세션실행후 : 그래프에 데이터가 입력되고 계산될 때 연산 수행



Ref : Learning tensorflow, oreilly

Computational Graph

텐서플로우 설치

```
import tensorflow as tf
```

In [3]:
```
# a rank 0 tensor; this is a scalar with shape []
[1. ,2., 3.] # a rank 1 tensor; this is a vector with shape [3]
[[1., 2., 3.], [4., 5., 6.]] # a rank 2 tensor; a matrix with shape [2, 3]
[[[1., 2., 3.]], [[7., 8., 9.]]] # a rank 3 tensor with shape [2, 1, 3]
```

Out[3]:
```
[[[1.0, 2.0, 3.0]], [[7.0, 8.0, 9.0]]]
```

In [4]:
```
node1 = tf.constant(3.0, tf.float32)
node2 = tf.constant(4.0) # also tf.float32 implicitly
node3 = tf.add(node1, node2)  # node3 = node1 + node2
```

In [5]:
```
print("node1:", node1, "node2:", node2)
print("node3: ", node3)
```

```
node1: Tensor("Const_1:0", shape=(), dtype=float32) node2: Tensor("Const_2:0", shape=
(), dtype=float32)
node3:  Tensor("Add:0", shape=(), dtype=float32)
```

In [6]:
```
sess = tf.Session()
print("sess.run(node1, node2): ", sess.run([node1, node2]))
print("sess.run(node3): ", sess.run(node3))
```

```
sess.run(node1, node2):  [3.0, 4.0]
sess.run(node3):  7.0
```

https://github.com/hunkim/DeepLearningZeroToAll

```
1  import tensorflow as tf
2  import numpy as np
3
4  sess = tf.InteractiveSession()
5
6  s = np.array([[[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]],
7                 [[13, 14, 15, 16], [17, 18, 19, 20], [21, 22, 23, 24]]]])
8  print(s)
9
10 t = tf.constant([[[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]],
11                  [[13, 14, 15, 16], [17, 18, 19, 20], [21, 22, 23, 24]]]])
12 tf.shape(t).eval()
```

```
[[[[ 1  2  3  4]
   [ 5  6  7  8]
   [ 9 10 11 12]]

  [[13 14 15 16]
   [17 18 19 20]
   [21 22 23 24]]]]

array([1, 2, 3, 4])
```

Computational Graph

텐서플로우 설치

In [7]:
```
a = tf.placeholder(tf.float32)
b = tf.placeholder(tf.float32)
adder_node = a + b  # + provides a shortcut for tf.add(a, b)

print(sess.run(adder_node, feed_dict={a: 3, b: 4.5}))
print(sess.run(adder_node, feed_dict={a: [1,3], b: [2, 4]}))
```

```
7.5
[3. 7.]
```

In [8]:
```
add_and_triple = adder_node * 3.
print(sess.run(add_and_triple, feed_dict={a: 3, b:4.5}))
```

```
22.5
```

https://github.com/hunkim/DeepLearningZeroToAll

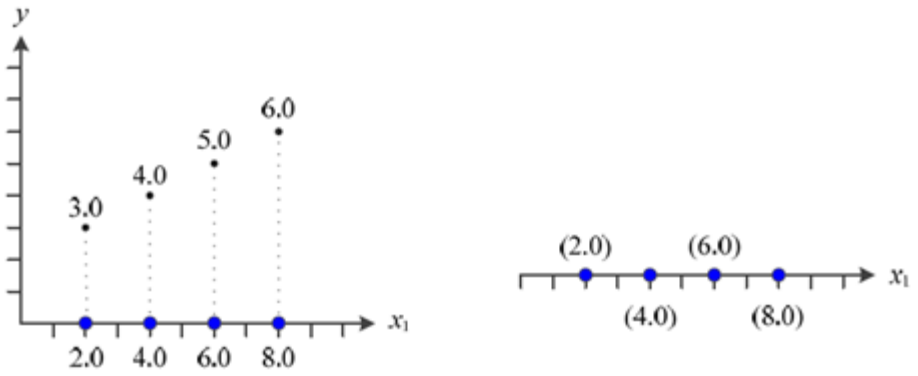
텐서플로우 설치

MNIST DATA SET 구조

텐서보드(TensorBoard)

1차원과 2차원 공간

- 1차원 특징 공간



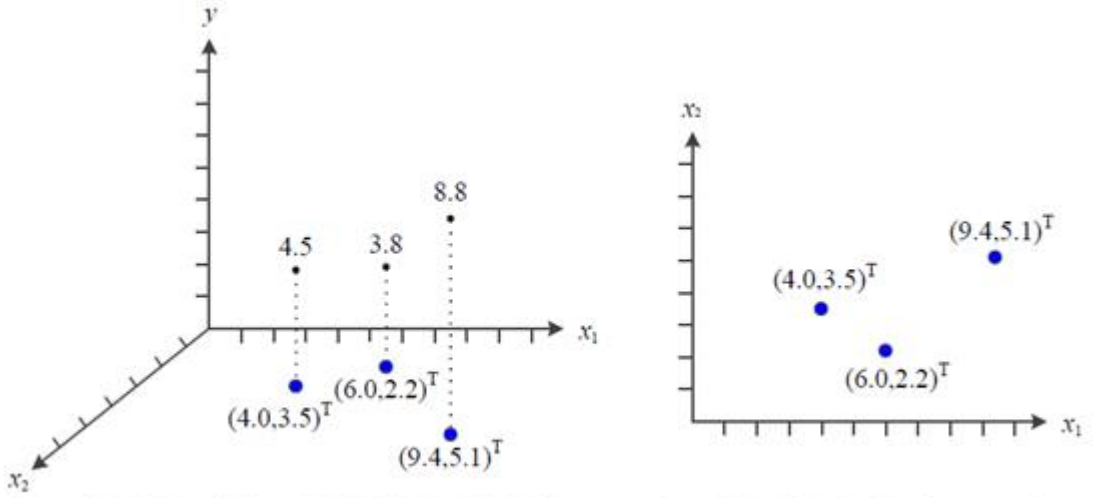
(a) 1차원 특징 공간(왼쪽: 특징과 목푯값을 축으로 표시, 오른쪽: 특징만 축으로 표시)

- 2차원 특징 공간
  - 특징 벡터 표기

    $\mathbf{x}=(x_1,x_2)^T$
  - 예시

    $\mathbf{x}$=(몸무게,키)^T, y=장타율

    $\mathbf{x}$=(체온,두통)^T, y=감기 여부

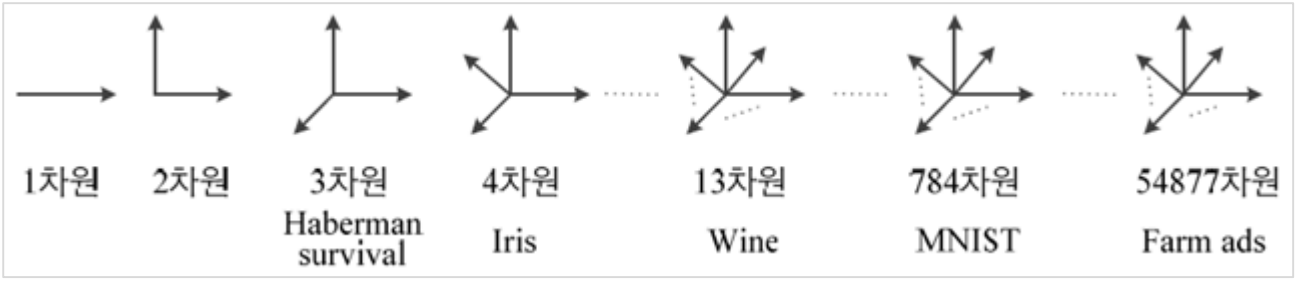


(b) 2차원 특징 공간(왼쪽: 특징 벡터와 목푯값을 축으로 표시, 오른쪽: 특징 벡터만 축으로 표시)

ref : Machine Learning - 한빛아카데미

다차원 특징 공간

MNIST DATA SET 구조

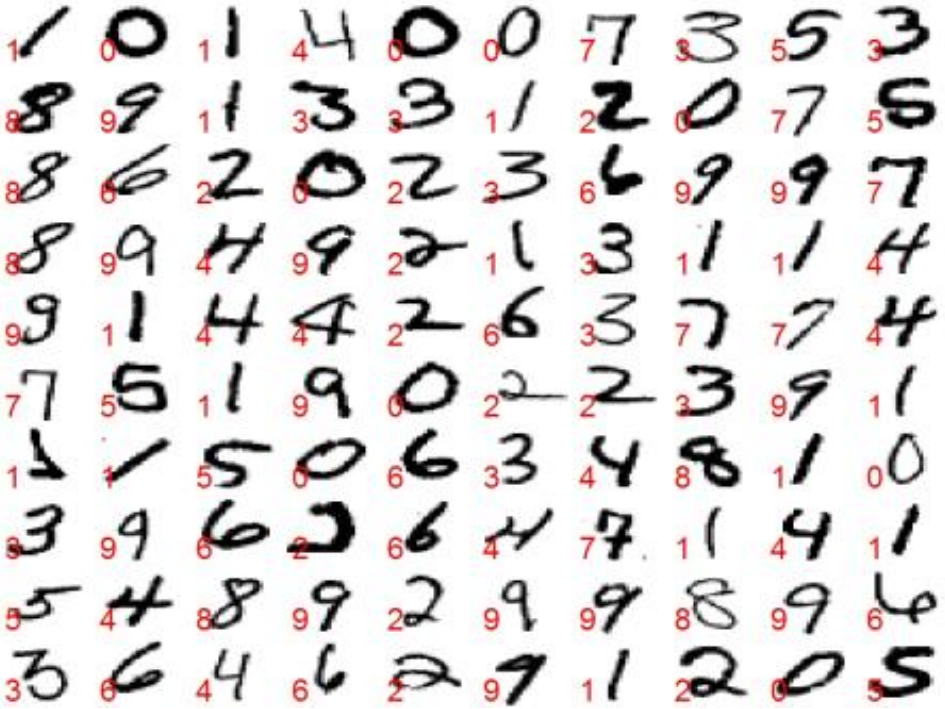


- Haberman Survival Data Set(https://archive.ics.uci.edu/ml/datasets/)
  - 유방암 수술을 받은 환자의 생존을 조사한 사례 데이터
  - Attribute : 3
    - Age of patient at time of operation (numerical)
    - Patient's year of operation (year - 1900, numerical)
    - Number of positive axillary nodes detected (numerical)
- Iris Data Set
  - Attribute : 4
    - sepal length in cm, sepal width in cm
    - petal length in cm, petal width in cm
- Wine Data Set (https://archive.ics.uci.edu/ml/datasets/)
  - 화학 분석을 사용하여 와인의 기원(origin of wine)을 결정
  - Atribute : 13
    - Alcohol, Malic acid , Ash, Alcalinity of ash, Magnesium, Total phenols, Flavanoids , Nonflavanoid phenols, Proanthocyanins, Color intensity, Hue, OD280/OD315 of diluted wines, Proline
- MNIST Data Set
  - Attribute : 784
    - 784 numbers
- Farm Ads Data set(https://archive.ics.uci.edu/ml/datasets/)
  - 다양한 농장 동물 관련 주제를 다루는 12 개의 웹 사이트에있는 텍스트 광고에서 수집된 데이터
  - Attribute : 54,877
    - Text words 54877

ref : Machine Learning - 한빛아카데미

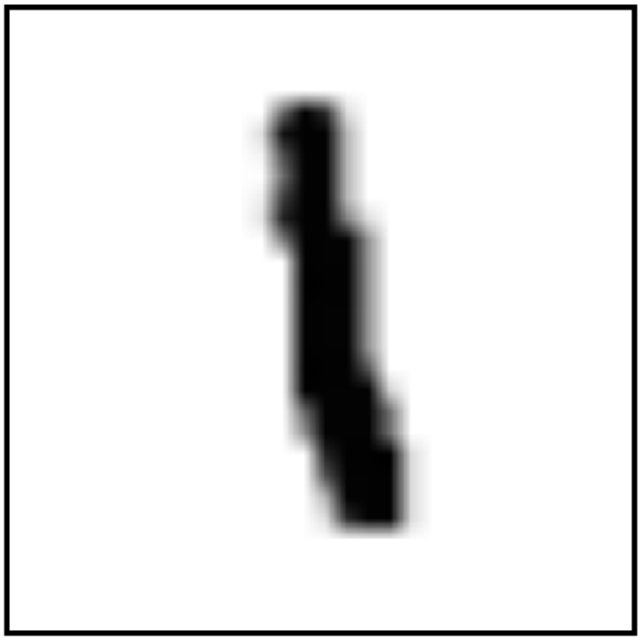
MNIST Data Set (http://yann.lecun.com/exdb/mnist/)

- MNIST : Modified National Institute of Standards and Technology database
- 미국표준국(NIST)에서 수집한 필기 숫자(handwritten digits) 데이터
- 훈련데이터(training set) 60,000, 테스트데이터(test set) 10,000
  - train-images-idx3-ubyte.gz: training set images (9,912,422 bytes)
  - train-labels-idx1-ubyte.gz: training set labels (28,881 bytes)
  - t10k-images-idx3-ubyte.gz: test set images (1,648,877 bytes)
  - t10k-labels-idx1-ubyte.gz: test set labels (4,542 bytes)



MNIST 데이터 셋

- 각 이미지는 28pixels by 28 pixles
  - 28×28 = 784 numbers
  - 28×28 = 784 차원 벡터
  - 벡터의 각 구성 요소는 0과 1 사이의 값으로 픽셀의 강도를 표시



≈

$$\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .6 & .8 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .5 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .7 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .9 & 1 & .1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .3 & 1 & .1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

MNIST 데이터 셋 구조 출력해 보기


```
import tensorflow as tf
import numpy as np
from tensorflow.examples.tutorials.mnist import input_data
import matplotlib.pyplot as plt

mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)

print("훈련 이미지 :",  mnist.train.images.shape)
print("훈련 라벨:",  mnist.train.labels.shape)
print("테스트 이미지 : ", mnist.test.images.shape)
print("테스트 라벨 : ", mnist.test.labels.shape)
print("검증 이미지 : ", mnist.validation.images.shape)
print("검증 라벨 : ", mnist.validation.labels.shape)
print('₩n')

mnist_idx = 100

print('[label]')
print('one-hot vector label = ', mnist.train.labels[mnist_idx])
print('number label = ', np.argmax(mnist.train.labels[mnist_idx]))
print('₩n')

print('[image]')

for index, pixel in enumerate(mnist.train.images[mnist_idx]):
    if index % 28 == 0:
        print('₩n')
    else:
        print("%10f" % pixel, end="")
print('₩n')
```

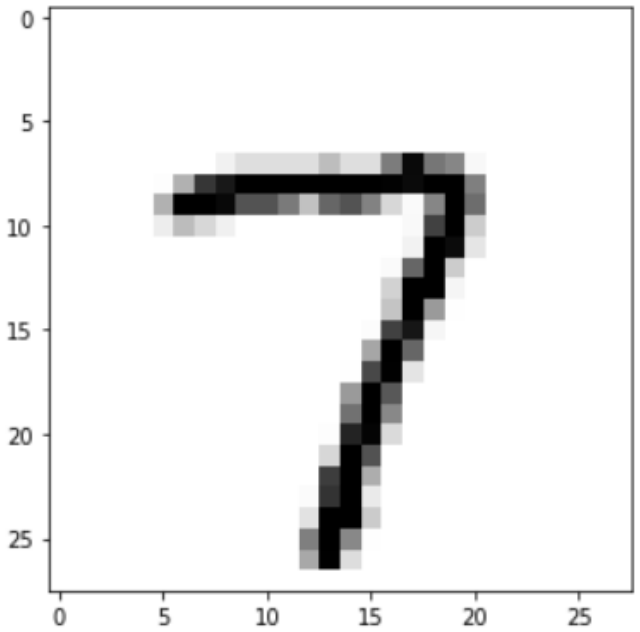
MNIST 데이터 셋 구조 출력해 보기

MNIST DATA SET 구조

```
훈련 이미지 : (55000, 784)
훈련 라벨: (55000, 10)
테스트 이미지 :  (10000, 784)
테스트 라벨 :  (10000, 10)
검증 이미지 :  (5000, 784)
검증 라벨 :  (5000, 10)


[label]
one-hot vector label =  [0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
number label =  7


[image]


  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.00000
0  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.0000
00  0.000000  0.000000  0.000000  0.000000  0.000000
```

MNIST 데이터 셋 구조 출력해 보기


```
1  plt.figure(figsize=(5, 5))
2  image = np.reshape(mnist.train.images[mnist_idx], [28, 28])
3  plt.imshow(image, cmap='Greys')
4  plt.show()
```



MNIST 데이터 셋 구조 출력해 보기

MNIST DATA SET 구조

```python
import tensorflow as tf
import numpy as np
from tensorflow.examples.tutorials.mnist import input_data
import matplotlib.pyplot as plt

mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)

print(mnist.train.labels[1])

print(mnist.train.images[1])
```

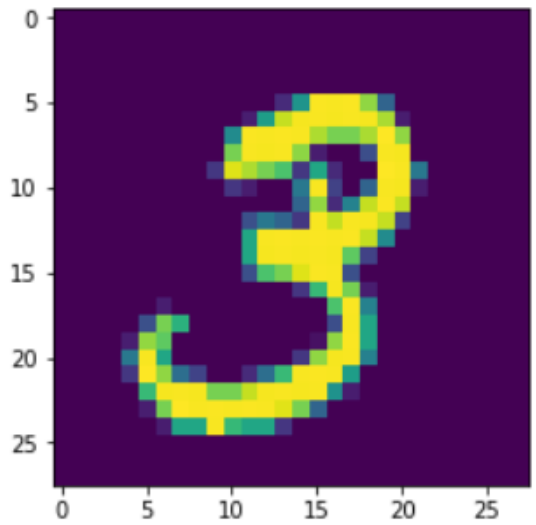
```
Extracting MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
[0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
[0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
```

MNIST 데이터 셋 구조 출력해 보기

MNIST DATA SET 구조

```
1  arr = np.array(mnist.train.images[1])
2  arr.shape = (28,28)
3  plt.imshow(arr)
```

```
<matplotlib.image.AxesImage at 0x2bc14b4b400>
```



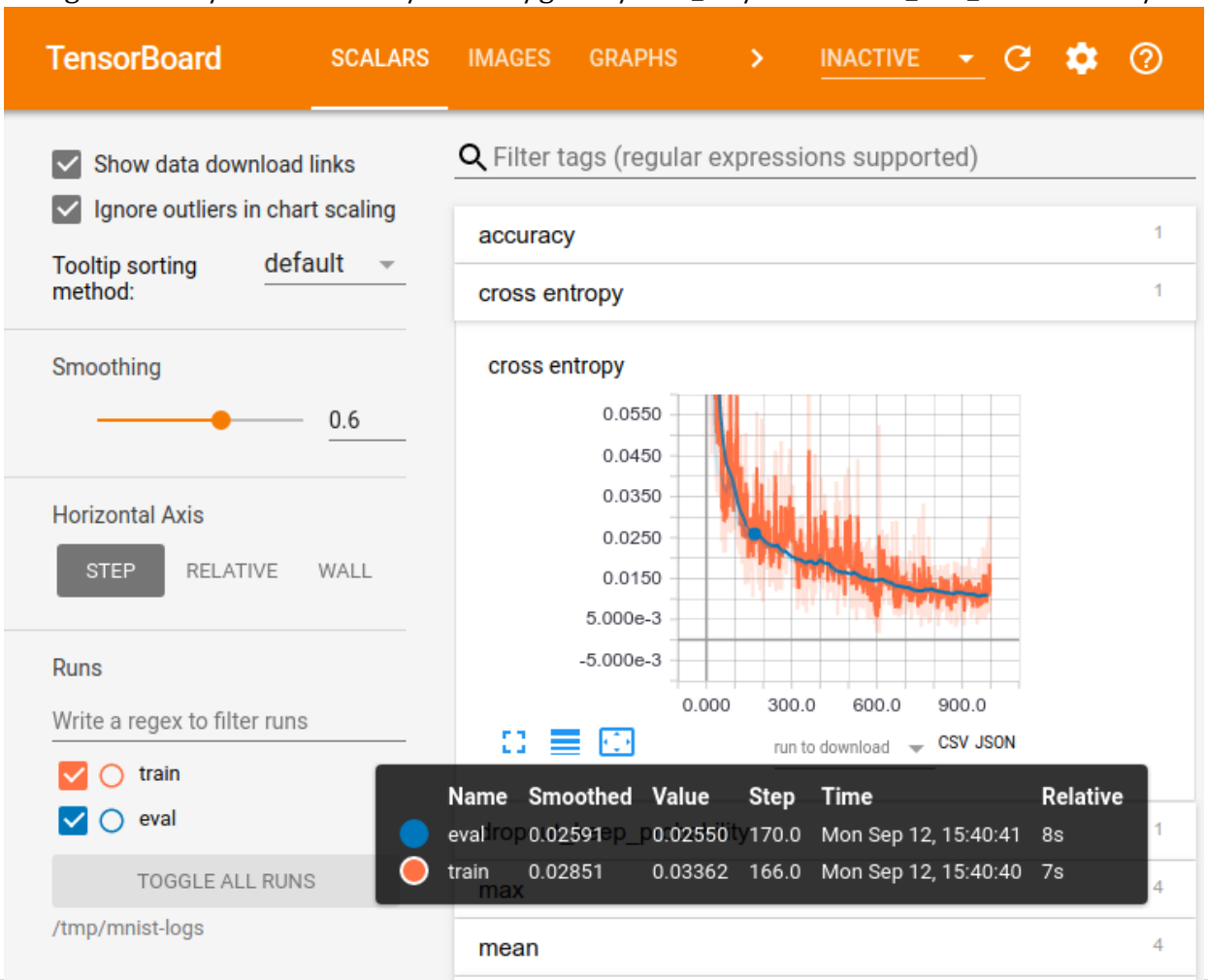
텐서플로우 설치

MNIST DATA SET 구조

텐서보드(TensorBoard)

TensorBoard

- 복잡하고 혼란스러운 거대한 심층 신경망을 쉽게 이해하고, 디버그 및 최적화를 쉽게 만들기 위한 시각화 도구 세트
  - https://tensorflowkorea.gitbooks.io/tensorflow-kr/content/g3doc/how_tos/summaries_and_tensorboard/ (한글문서)



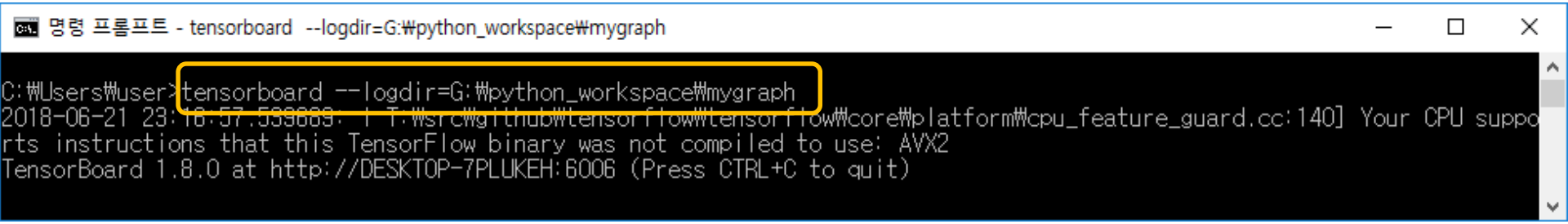
TensorBoard 사용하기

- python_workspace 디렉토리 – mygraph 폴더확인
- 텐서보드 실행
  - command.com 창에서 아래 명령어 실행
  - tensorboard --logdir=G:\python_workspace\mygraph

```
1  import tensorflow as tf
2
3  a = tf.constant(5,name='input_a')
4  b = tf.constant(7,name='input_b')
5
6  c = tf.multiply(a,b,name='mul_c')
7  d = tf.add(a,b,name='add_d')
8  e = tf.add(c,d,name='add_e')
9
10 sess = tf.Session()
11 print(sess.run(e))
```

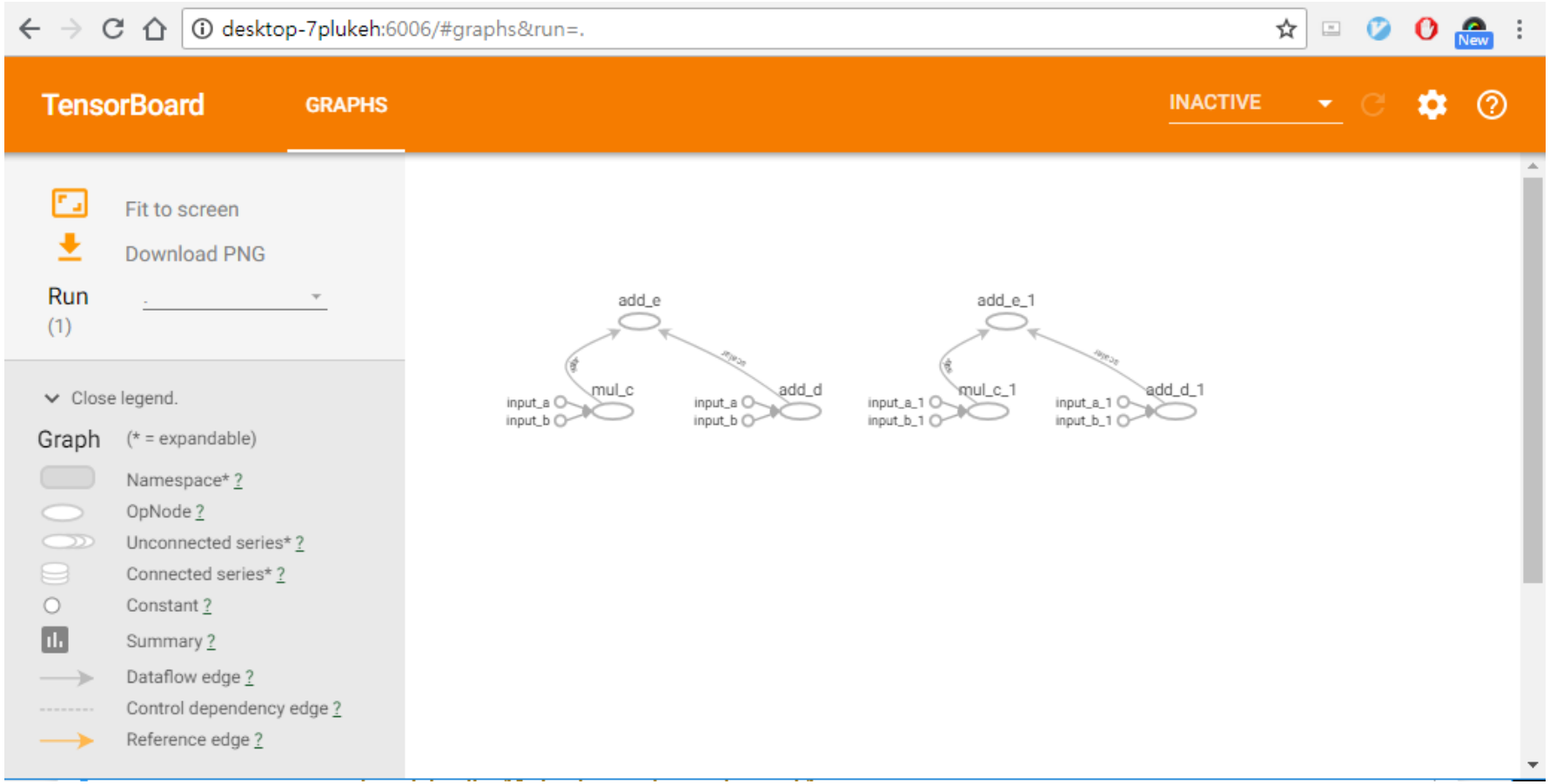
```
47
```

```
1  wirter = tf.summary.FileWriter('./mygraph',sess.graph)
```



시각화

텐서보드(TensorBoard)



노드 기호표

텐서보드(TensorBoard)

Symbol	Meaning
	High-level 노드는 name scope를 나타냅니다. high-level 노드를 펼치기 위해 더블 클릭 하세요.
	서로 연결되지 않은 숫자가 메겨진 노드의 시퀀스.
	서로 연결된 숫자가 메겨진 노드의 시퀀스.
	각각의 연산 노드.
	상수.
	요약노트.
	간선은 연산 사이의 데이터 흐름을 보여줍니다.
	간선은 연산 사이의 컨트롤 종속을 보여줍니다.
	레퍼런스 간선은 나가는 연산 노드가 들어오는 tensor를 변형할 수 있다는 것을 보여줍니다.

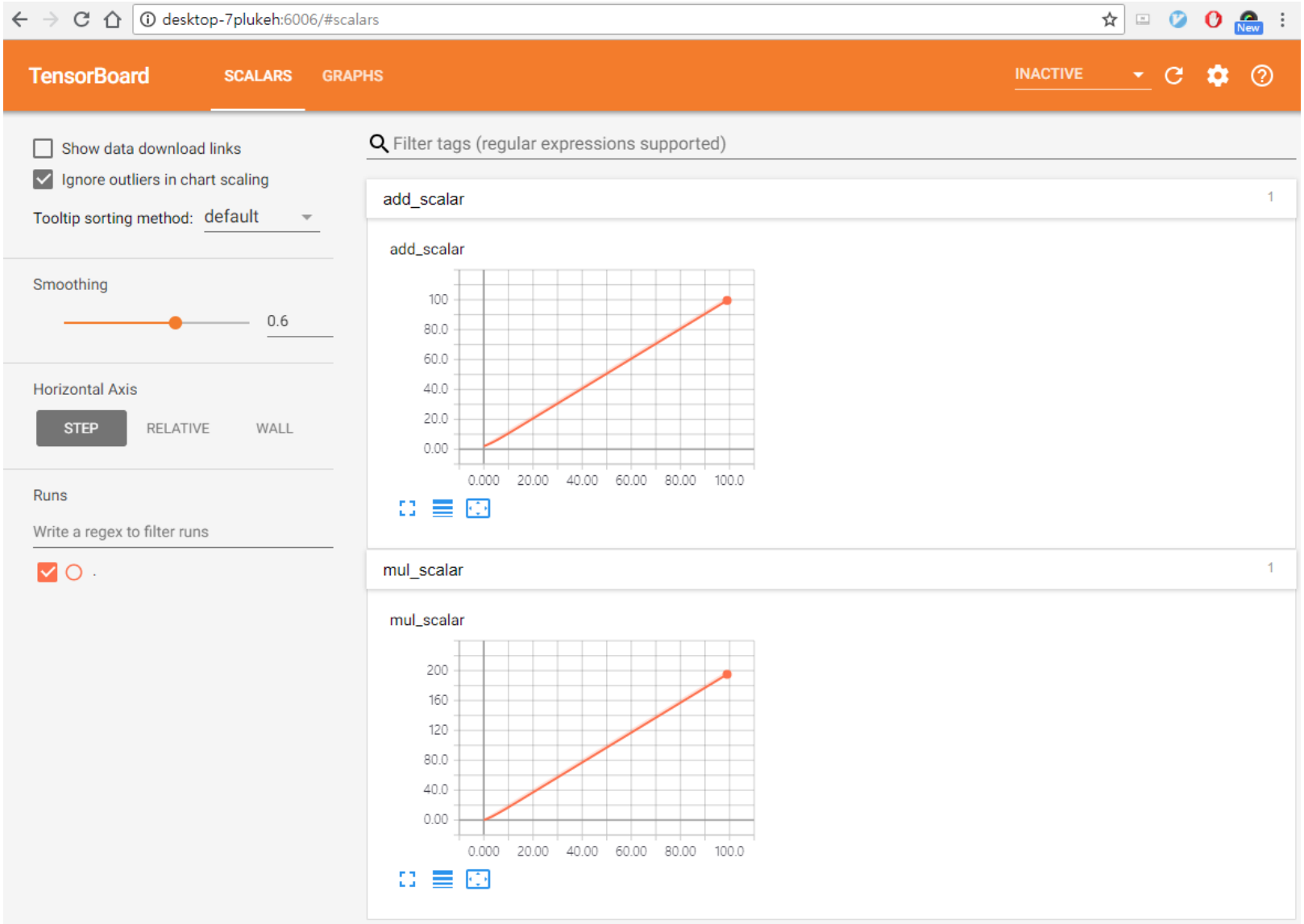
TensorBoard

텐서보드(TensorBoard)

```python
import tensorflow as tf

X = tf.placeholder(tf.float32)
Y = tf.placeholder(tf.float32)

add = tf.add(X, Y)
mul = tf.multiply(X, Y)

# step 1: node 선택
add_hist = tf.summary.scalar('add_scalar', add)
mul_hist = tf.summary.scalar('mul_scalar', mul)

# step 2: summary 통합, 두 개의 코드 모두 동작.
merged = tf.summary.merge_all()
# merged = tf.summary.merge([add_hist, mul_hist])

with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())

    # step 3: writer 생성
    writer = tf.summary.FileWriter('./mygraph', sess.graph)

    for step in range(100):
        # step 4: 노드 추가
        summary = sess.run(merged, feed_dict={X: step * 1.0, Y: 2.0})
        writer.add_summary(summary, step)
```

TensorBoard

텐서보드(TensorBoard)



TensorBoard

텐서보드(TensorBoard)