Python

텐서플로우 맛보기

김선녕(sykim.lecture@gmail.com)

텐서플로우(TensorFlow)란?

- 머신러닝 알고리즘을 구현하고 실행하기 위한 프로그래밍 인터페이스
- 확장이 용이하고 다양한 플래폼을 지원
- 초창기 구글 내부 용도로 개발되었지만 2015년 11월 오픈 소스 라이선스로 릴리즈
- 프런트엔드 인터페이스로 여러 프로그래밍 언어를 지원
  - 파이썬, C++, Java, Node.js, Go등
- 텐서플로우 연산은 데이터 흐름을 표현하는 방향 그래프를 구성하여 수행
  - 방향 그래프(directed graph) : 노드(node)사이를 잇는 에지(edge)에 방향이 있는 그래프
  - 그래프의 노드(Node)는 수치 연산을 나타내고 엣지(edge)는 노드 사이를 이동하는 다차원 데이터 배열(텐서,tensor)를 나타낸다.
- 텐서플로우 설치
  - (base) C:\Users\kswkw>pip install tensorflow
- Hello TensorFlow

```
1 import tensorflow as tf
2 print(tf.__version__)
3 print('Hello, TensorFlow!')
```

```
2.0.0
Hello, TensorFlow!
```

텐서플로우 1.x 방식


```python
# 텐서플로우 1.x 방식의 API 사용 예

# 그래프 생성
g = tf.Graph()
with g.as_default():
    x = tf.compat.v1.placeholder(dtype=tf.float32, shape=(None), name='x')
    w = tf.Variable(2.0, name='weight')
    b = tf.Variable(0.7, name='bias')

    z = w * x + b

    init = tf.compat.v1.global_variables_initializer()

# 세션을 만들고 그래프 g를 전달
with tf.compat.v1.Session(graph=g) as sess :
    ## w와 b를 초기화 한다.
    sess.run(init)
    ## z를 평가한다
    for t in [1.0, 0.6, -1.8] :
        print('x = %4.1f --> z=%4.1f'%(t, sess.run(z, feed_dict={x:t})))
```

```
x =  1.0 --> z= 2.7
x =  0.6 --> z= 1.9
x = -1.8 --> z=-2.9
```

텐서플로우 2.x 방식


```python
# 텐서플로우 2.x 방식의 API 사용 예

w = tf.Variable(2.0, name='weight')
b = tf.Variable(0.7, name='bias')

# z를 평가한다
for x in [1.0, 0.6, -1.8] :
    z = w * x + b
    print('x = %4.1f --> z=%4.1f'%(x, z))
```

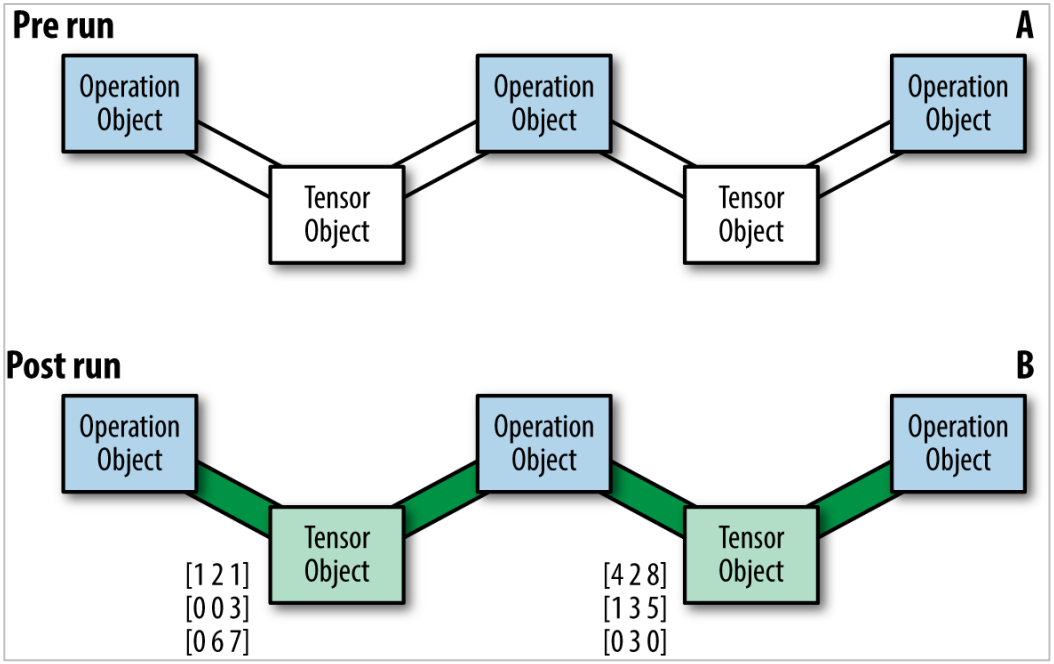
```
x =  1.0 --> z= 2.7
x =  0.6 --> z= 1.9
x = -1.8 --> z=-2.9
```

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

Rank

- Rank 1 : 초기값으로 아이템의 리스트를 전달

```
mystr = tf.Variable(["Hello"], tf.string)
cool_numbers  = tf.Variable([3.14159, 2.71828], tf.float32)
first_primes = tf.Variable([2, 3, 5, 7, 11], tf.int32)
its_very_complicated = tf.Variable([12.3 - 4.85j, 7.5 - 6.23j], tf.complex64)
```

- Higher ranks
  - Rank 2 : 적어도 하나의 행과 적어도 하나의 열로 구성

```
mynat = tf.Variable([[7],[11]], tf.int16)
myxor = tf.Variable([[False, True],[True, False]], tf.bool)
linear_squares = tf.Variable([[4], [9], [16], [25]], tf.int32)
squarish_squares = tf.Variable([ [4, 9], [16, 25] ], tf.int32)
rank_of_squares = tf.rank(squarish_squares)
mynatC = tf.Variable([[7],[11]], tf.int32)
```

  - Higher-rank Tensors : n차원 배열로 구성(예: tensors of rank 4)

```
my_image = tf.zeros([10, 299, 299, 3])  # batch x height x width x color
```

Shape

- 텐서의 Shape은 각 차원에 있는 요소(element)의 수
- 파이썬의 int형의 list/tuple을 통하여 표현되거나, tf.TensorShape로 표현
- Getting a tf.Tensor object's shape
  - zeros = tf.zeros(my_matrix.shape[1])

Rank	Shape	Dimension number	Example
0	[]	0-D	A 0-D tensor. A scalar.
1	[D0]	1-D	A 1-D tensor with shape [5].
2	[D0, D1]	2-D	A 2-D tensor with shape [3, 4].
3	[D0, D1, D2]	3-D	A 3-D tensor with shape [1, 4, 3].
n	[D0, D1, ... Dn-1]	n-D	A tensor with shape [D0, D1, ... Dn-1].

텐서의 랭크와 크기 확인


```
t1 = tf.constant(np.pi)
t2 = tf.constant([1,2,3,4])
t3 = tf.constant([[1,2], [3,4]])

print(np.pi)

# Rank를 구한다
r1 = tf.rank(t1)
r2 = tf.rank(t2)
r3 = tf.rank(t3)

# 크기를 구한다
s1 =t1.get_shape()
s2 =t2.get_shape()
s3 =t3.get_shape()

print('Rank: ', r1.numpy(), r2.numpy(), r3.numpy())
print('크기:', s1, s2, s3)

```

```
3.141592653589793
Rank:  0 1 2
크기: () (4,) (2, 2)
```

Reshape

- Changing the shape of a tf.Tensor
  - 텐서 요소의 수는 shape 크기의 곱
  - tf.reshape : 요소(element)를 고정된 상태로 유지하면서 tf.Tensor의 모양 변경

```
rank_three_tensor = tf.ones([3, 4, 5])
matrix = tf.reshape(rank_three_tensor, [6, 10])  # Reshape existing content into
                                                 # a 6x10 matrix
matrixB = tf.reshape(matrix, [3, -1])  #  Reshape existing content into a 3x20
                                       # matrix. -1 tells reshape to calculate
                                       # the size of this dimension.
matrixAlt = tf.reshape(matrixB, [4, 3, -1])  # Reshape existing content into a
                                             #4x3x5 tensor

# Note that the number of elements of the reshaped Tensors has to match the
# original number of elements. Therefore, the following example generates an
# error because no possible value for the last dimension will match the number
# of elements.
yet_another = tf.reshape(matrixAlt, [13, 2, -1])  # ERROR!
```

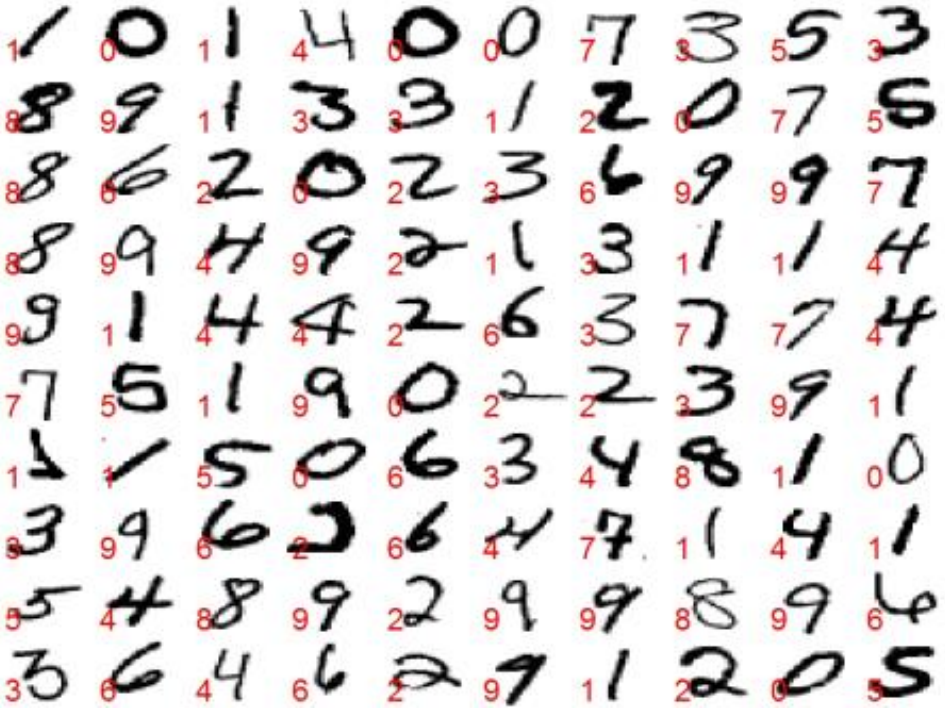
배열구조


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

x_array = np.arange(18).reshape(3, 2, 3)

x2 = tf.reshape(x_array, shape=(-1, 6))

# 각 열의 합을 계산
xsum = tf.reduce_sum(x2, axis=0)

# 각 열의 평균 계산
xmean = tf.reduce_mean(x2, axis=0)

print('입력크기: ', x_array.shape)
print('크기가 변경된 입력:₩n', x2.numpy())
print('열의 합:₩n', xsum.numpy())
print('열의 평균:₩n', xmean.numpy())
```

```
입력크기:  (3, 2, 3)
크기가 변경된 입력:
 [[ 0  1  2  3  4  5]
 [ 6  7  8  9 10 11]
 [12 13 14 15 16 17]]
열의 합:
 [18 21 24 27 30 33]
열의 평균:
 [ 6  7  8  9 10 11]
```

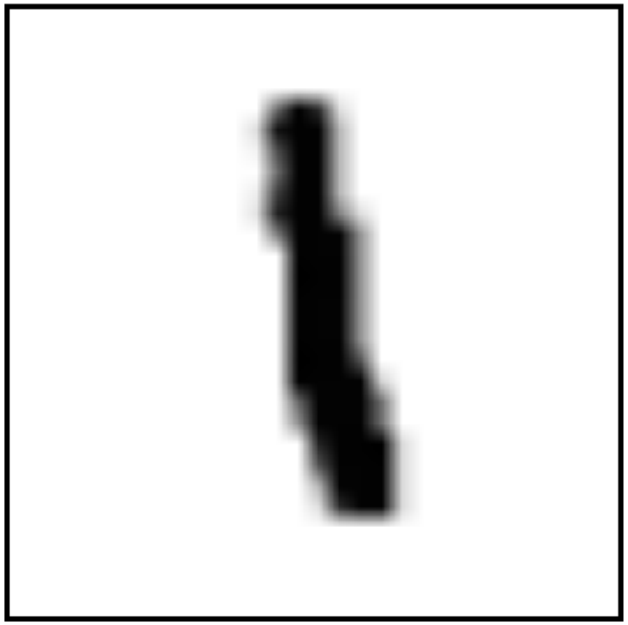
MNIST Data Set (http://yann.lecun.com/exdb/mnist/)

- 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
  - 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

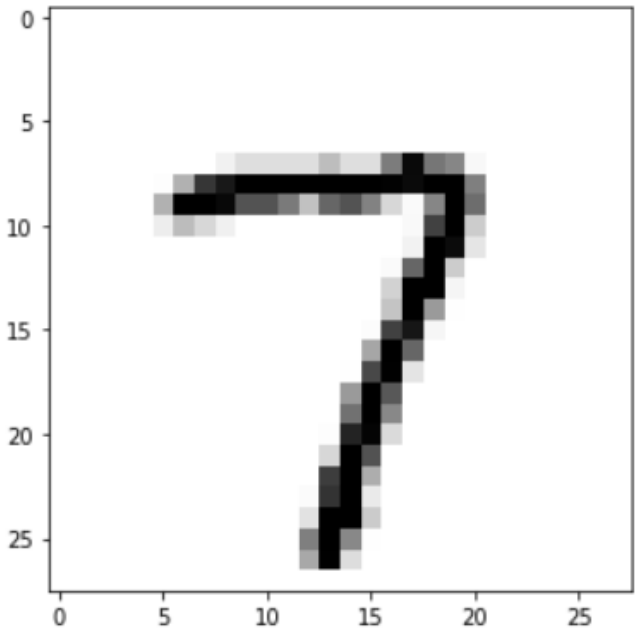

```
import tensorflow as tf
import numpy as np
from tensorflow.examples.tutorials.mnist import input_data
import matplotlib.pyplot as plt

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


```
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


```
1  import tensorflow as tf
2  import numpy as np
3  from tensorflow.examples.tutorials.mnist import input_data
4  import matplotlib.pyplot as plt
5
6  mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
7
8  print(mnist.train.labels[1])
9
10 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


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

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