Numpy Basics

Saehwa Kim

Information and Communications Engineering
Hankuk University of Foreign Studies

Contents

- Creating Numpy Arrays, Shape, Rank (Dimension), Reshaping, and Flattening
- Creating Various Arrays and Data Types
- Array Indexing
- Array Math and Broadcasting
- Array Dot Operation (Matrix Multiplication)
- Array Dimension Extension
- Array Concatenation

Creating Numpy Arrays, Shape, Rank (Dimension), Reshaping, and Flattening

```
>>> import numpy as np
>>> a = np.array([[1,2], [3, 4], [5, 6]])
>>> a
array([[1, 2],
       [3, 4],
       [5, 6]])
>>> a.shape
(3, 2)
>>> a.ndim # == len(a.shape) # rank # dimension
2
>>> a[1, 1]
4
>>> a[-1, 0]
5
```

Can be also:
(-1, 3) or
(2, 3)

```
>>> a = a.reshape((2, -1)) # np.reshape(a, (2, -1)) # Without assignment, a does not change.
>>> a
array([[1, 2, 3],
       [4, 5, 6]])
```

```
>>> a = a.flatten()
>>> a
array([1, 2, 3, 4, 5, 6])
>>> a.shape
(6,)
```

Creating Various Arrays and Data Types

```
>>> import numpy as np
>>> a = np.zeros((2, 2))
>>> a
array([[0., 0.],
       [0., 0.]])
>>> a = np.ones((1, 2))
>>> a
array([[1., 1.]])
>>> a = np.eye(2)
>>> a
array([[1., 0.],
       [0., 1.]])
>>> a = np.random.random((2, 2))
>>> a
array([[0.62500022, 0.74466799],
       [0.51267045, 0.30867129]])
```

```
>>> a = np.array([2, 3])
>>> a.dtype
dtype('int64')
>>> a = np.array([2.3, 3.4])
>>> a.dtype
dtype('float64')
>>> a = np.array([2.3, 3.4], dtype=np.float16)
>>> a.dtype
dtype('float16')
```


Array Indexing

```
>>> import numpy as np
>>> a = np.reshape(np.array(range(1, 13)),(-1, 4))
>>> a
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
>>> a[:2, 1:3]
array([[ 2,  3],
       [ 6,  7]])
>>> a[1, 1:3]
array([6, 7])
>>> a[1:2, 1:3]
array([[6, 7]])
```

```
>>> a[:, [1, 3]] # == a[:, (1, 3)]
array([[ 2,  4],
       [ 6,  8],
       [10, 12]])

>>> a %2 == 1
array([[ True, False,  True, False],
       [ True, False,  True, False],
       [ True, False,  True, False]])
>>> a[a%2==1]
array([ 1,  3,  5,  7,  9, 11])
```


Array Math and Broadcasting

```
>>> import numpy as np
>>> a = np.array([[1, 2], [3, 4]])
>>> a
array([[1, 2],
       [3, 4]])
>>> b = a + 4  # Broadcasting
>>> b
array([[5, 6],
       [7, 8]])
>>> a + b
array([[ 6,  8],
       [10, 12]])
>>> np.add(a, b)
array([[ 6,  8],
       [10, 12]])
>>> a - b # == np.subtract(a, b)
array([[-4, -4],
       [-4, -4]])
```

```
>>> a*b # == np.multiply(a, b)
array([[ 5, 12],
       [21, 32]])
>>> a/b # == np.divide(a, b)
array([[0.2       , 0.33333333],
       [0.42857143, 0.5       ]])
>>> np.sqrt(a)
array([[1.        , 1.41421356],
       [1.73205081, 2.        ]])
>>> a + np.array([10, 20])
array([[11, 22],
       [13, 24]])
>>> a + np.array([10])
array([[11, 12],
       [13, 14]])
>>> a + np.array([[10], [20]])
array([[11, 12],
       [23, 24]])
```


Array Dot Operation (Matrix Multiplication)

```
>>> import numpy as np
>>> x = np.reshape(
  np.array(range(1, 7)),(2, 3))
>>> x
array([[1, 2, 3],
       [4, 5, 6]])
>>> y = x.T # tansposed matrix
>>> y
array([[1, 4],
       [2, 5],
       [3, 6]])
>>> z = x.dot(y) # == np.dot(x, y)
>>> z
array([[14, 32],
       [32, 77]])
```

```
>>> y[:, 0:1]
array([[1],
       [2],
       [3]])
>>> x.dot(y[:, 0:1])
array([[14],
       [32]])
```

```
>>> y[:, 0]
array([1, 2, 3])

>>> x.dot(y[:, 0])
array([14, 32])
```

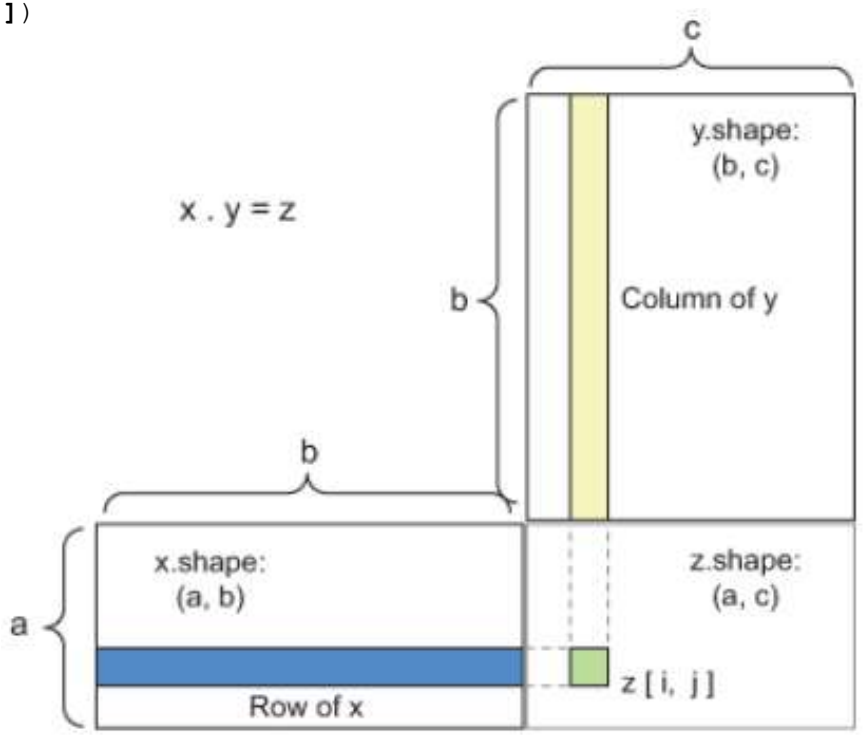

그림 출처:
https://livebook.manning.com/book/deep-learning-with-python/chapter-2/1

Dimension Extension

```
>>> import numpy as np
>>> a = np.array([[1,2], [3, 4], [5, 6]])
>>> a
array([[1, 2],
       [3, 4],
       [5, 6]])
>>> b1 = a.reshape((1, *a.shape))
>>> b1_ = a.reshape((1, 3, 2))
>>> b1__ = np.reshape(a, (1, 3, 2))
>>> b1___ = a[np.newaxis, …]
>>> b1____ = a[np.newaxis, :]
>>> b2 = np.expand_dims(a, axis = 0)
>>> b3 = np.array([a])
>>> b1 # the same for b1_, b1__, b2, b3
array([[[1, 2],
        [3, 4],
        [5, 6]]])
>>> b1.shape
(1, 3, 2)
```

```
>>> c = np.expand_dims(a, axis = -1)
>>> c_ = a.reshape((*a.shape, 1))

>>> c # the same for c_
array([[[1],
        [2]],

       [[3],
        [4]],

       [[5],
        [6]]])
>>> c.shape
(3, 2, 1)
```


Array Concatenation

```
>>> import numpy as np
>>> a = np.reshape( np.array(range(1, 7)),(-1, 3))
>>> a
array([[1, 2, 3],
       [4, 5, 6]])
>>> b = np.ones(a.shape)* 200
>>> b
array([[200., 200., 200.],
       [200., 200., 200.]])

>>> np.concatenate((a, b), axis=0)
array([[  1.,   2.,   3.],
       [  4.,   5.,   6.],
       [200., 200., 200.],
       [200., 200., 200.]])

# the same as the followings
>>> np.r_[a, b] # r: row
>>> np.vstack((a, b)) # v: vertical
```

```
>>> np.concatenate((a, b), axis=1)
array([[  1.,   2.,   3., 200., 200., 200.],
       [  4.,   5.,   6., 200., 200., 200.]])

# the same as the followings
>>> np.c_[a, b] # c: column
>>> np.hstack((a, b)) # h: horizontal
```