# 7. Tensor Manipulation

2019년 4월 5일 금요일 오전 11:44

1. ***인터랙티브 세션과 eval() 메서드***

   앞서 말한바와 같이 Tensorflow에서는 텐서의 값을 살펴보는 간단한 작업조차도 세션을 통해야 지만 가능하다. 이러한 불편을 해소하기 위해 파이썬 콘솔이나 주피터노트북을 사용하는 경우에는 **인터랙티브 세션(Interactive Session)**이라는 것을 제공한다. 인터랙티브 세션을 생성한 후에는 텐서의 `eval()` 메서드를 호출하기만 하면 명시적으로 세션을 지정하지 않더라도 자동으로 세션이 호출되어 텐서의 값이 출력된다.

2. ***Shap / Rank / Axis***

```
t = tf.constant([1,2,3,4])
tf.shape(t).eval()
```

```
array([4], dtype=int32)
```

```
t = tf.constant([[1,2],
                 [3,4]])
tf.shape(t).eval()
```

```
array([2, 2], dtype=int32)
```

```
t = tf.constant([[[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]],
                  [[13, 14, 15, 16], [17, 18, 19, 20], [21, 22, 23, 24]]]])
tf.shape(t).eval()
```

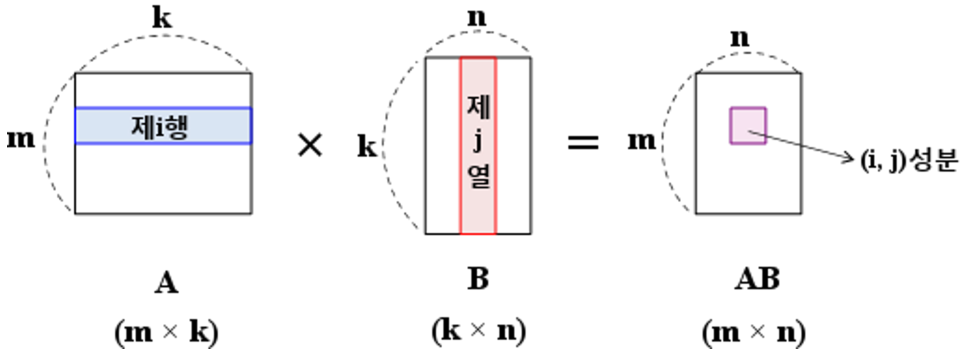
```
array([1, 2, 3, 4], dtype=int32)
```

```
t = tf.constant([[[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]],
                 [[13, 14, 15, 16], [17, 18, 19, 20], [21, 22, 23, 24]]]])
tf.shape(t).eval()
```

```
array([1, 2, 3, 4], dtype=int32)
```

```
[
    [
        [
            [1,2,3,4],
            [5,6,7,8],
            [9,10,11,12]
        ],
        [
            [13,14,15,16],
            [17,18,19,20],
            [21,22,23,24]
        ]
    ]
]
```

### 3. *Matmul / Multiply*



```
matrix1 = tf.constant([[1., 2.], [3., 4.]])
matrix2 = tf.constant([[1.],[2.]])
print("Metrix 1 shape", matrix1.shape)
print("Metrix 2 shape", matrix2.shape)
tf.matmul(matrix1, matrix2).eval()
```

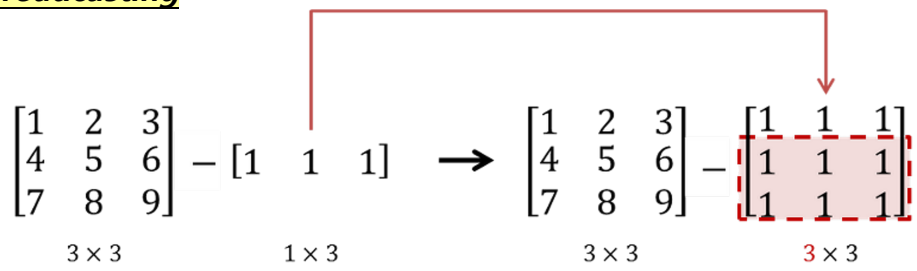
```
Metrix 1 shape (2, 2)
Metrix 2 shape (2, 1)

array([[  5.],
       [ 11.]], dtype=float32)
```

```
(matrix1*matrix2).eval()
```

```
array([[ 1.,  2.],
       [ 6.,  8.]], dtype=float32)
```

4. ***Broadcasting***



```
matrix1 = tf.constant([[1., 2.]])
matrix2 = tf.constant(3.)
(matrix1+matrix2).eval()
```

```
array([[ 4.,  5.]], dtype=float32)
```

```
matrix1 = tf.constant([[1., 2.]])
matrix2 = tf.constant([3., 4.])
(matrix1+matrix2).eval()
```

```
array([[ 4.,  6.]], dtype=float32)
```

```
matrix1 = tf.constant([[1., 2.]])
matrix2 = tf.constant([[3.],[4.]])
(matrix1+matrix2).eval()
```

```
array([[ 4.,  5.],
       [ 5.,  6.]], dtype=float32)
```

5. ***Reduce mean***

```
tf.reduce_mean([1, 2], axis=0).eval()
```

```
1
```

```
x = [[1., 2.],
     [3., 4.]]


tf.reduce_mean(x).eval()
```

```
2.5
```

```
tf.reduce_mean(x, axis=0).eval()
```

```
array([ 2.,  3.], dtype=float32)
```

```
tf.reduce_mean(x, axis=1).eval()
```

```
array([ 1.5,  3.5], dtype=float32)
```

```
tf.reduce_mean(x, axis=-1).eval()
```

```
array([ 1.5,  3.5], dtype=float32)
```

6. ***Reduce sum***

```
x = [[1., 2.],
     [3., 4.]]
```

```
tf.reduce_sum(x).eval()
```

```
10.0
```

```
tf.reduce_sum(x, axis=0).eval()
```

```
array([ 4.,  6.], dtype=float32)
```

```
tf.reduce_sum(x, axis=-1).eval()
```

```
array([ 3.,  7.], dtype=float32)
```

```
tf.reduce_mean(tf.reduce_sum(x, axis=-1)).eval()
```

```
5.0
```

7. ***Argmax***

```
x = [[0, 1, 2],
     [2, 1, 0]]
tf.argmax(x, axis=0).eval()
```

```
array([1, 0, 0])
```

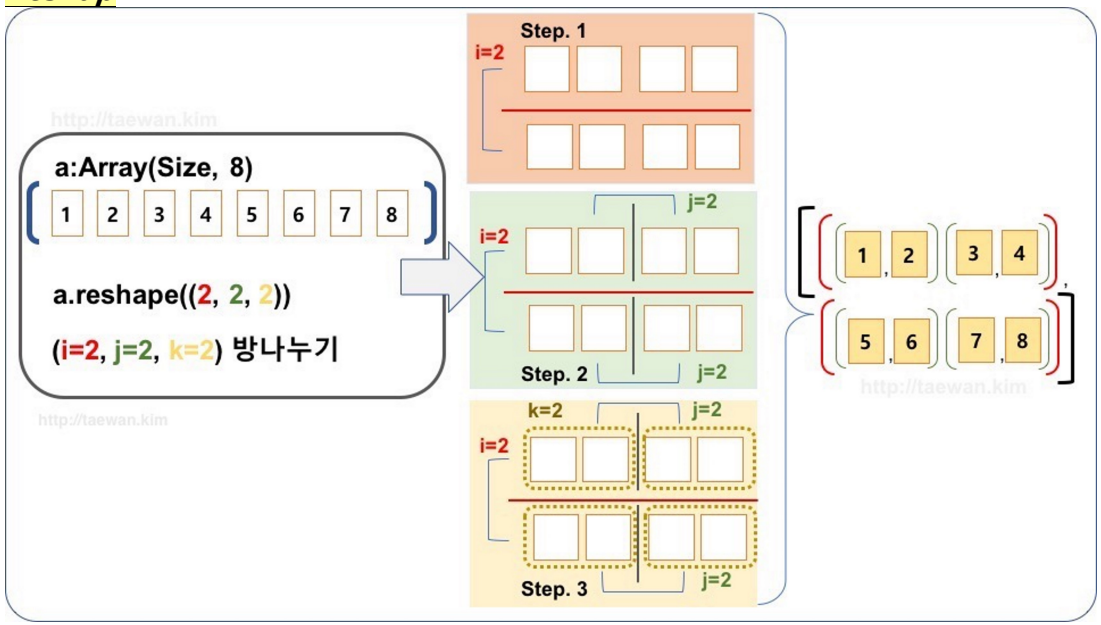
```
tf.argmax(x, axis=1).eval()
```

```
array([2, 0])
```

```
tf.argmax(x, axis=-1).eval()
```

```
array([2, 0])
```

## 8. *Reshap*



```
t = np.array([[[0, 1, 2],
               [3, 4, 5]],

              [[6, 7, 8],
               [9, 10, 11]]])
t.shape
```

```
(2, 2, 3)
```

```
tf.reshape(t, shape=[-1, 3]).eval()
```

```
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [ 6,  7,  8],
       [ 9, 10, 11]])
```

```
tf.reshape(t, shape=[-1, 1, 3]).eval()
```

```
array([[[ 0,  1,  2]],

       [[ 3,  4,  5]],

       [[ 6,  7,  8]],

       [[ 9, 10, 11]]])
```

9. ***Squeeze, expand***

```
tf.squeeze([[0], [1], [2]]).eval()
```

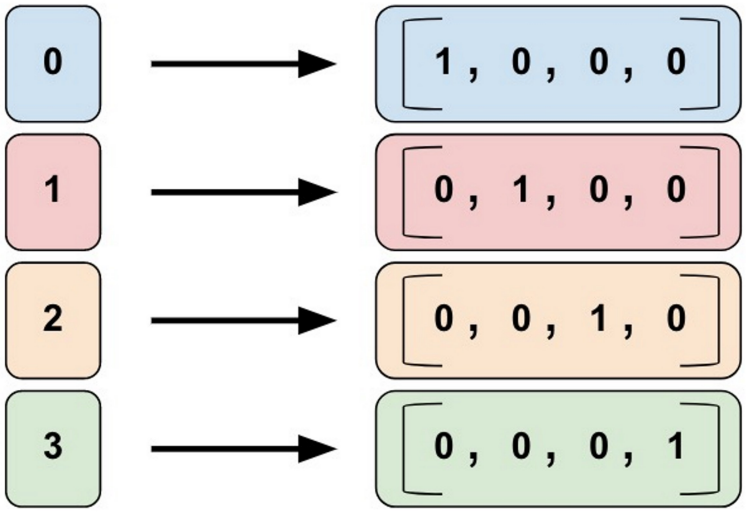
```
array([0, 1, 2], dtype=int32)
```

```
tf.expand_dims([0, 1, 2], 1).eval()
```

```
array([[0],
       [1],
       [2]], dtype=int32)
```

10. ***One hot***

- https://m.blog.naver.com/libido1014/120113775017 : 변수의 종류와 통계




```
tf.one_hot([[0], [1], [2], [0]], depth=3).eval()
```

```
array([[[ 1.,  0.,  0.]],

       [[ 0.,  1.,  0.]],

       [[ 0.,  0.,  1.]],

       [[ 1.,  0.,  0.]]], dtype=float32)
```

```
t = tf.one_hot([[0], [1], [2], [0]], depth=3)
tf.reshape(t, shape=[-1, 3]).eval()
```

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

## *11. Casting*

In [1]:
```
import tensorflow as tf
```

In [2]:
```
sess = tf.Session()
sess.run(tf.global_variables_initializer())
```

In [3]:
```
x = tf.constant([1.9, 2.1], dtype=tf.float32)
print(sess.run(x))
x = tf.cast(x, tf.int32)
print(sess.run(x))
```

```
[ 1.89999998  2.0999999 ]
[1 2]
```

In [4]:
```
x = tf.constant([True, False], dtype=tf.float32)
print(sess.run(x))
x = tf.cast(x, tf.int32)
print(sess.run(x))
```

```
[ 1.  0.]
[1 0]
```

In [5]:
```
x = tf.constant([1.9, 2.1], dtype=tf.float32)
print(sess.run(x))
x = tf.cast(x >= 2.0, tf.float32)
print(sess.run(x))
```

```
[ 1.89999998  2.0999999 ]
[ 0.  1.]
```

## *12. Stack*

```
x = [1, 4]
y = [2, 5]
z = [3, 6]

# Pack along first dim.
tf.stack([x, y, z]).eval()
```

```
array([[1, 4],
       [2, 5],
       [3, 6]], dtype=int32)
```

```
tf.stack([x, y, z], axis=1).eval()
```

```
array([[1, 2, 3],
       [4, 5, 6]], dtype=int32)
```

13. ***One and Zeros like***

```
x = [[0, 1, 2],
     [2, 1, 0]]

tf.ones_like(x).eval()
```

```
array([[1, 1, 1],
       [1, 1, 1]], dtype=int32)
```

```
tf.zeros_like(x).eval()
```

```
array([[0, 0, 0],
       [0, 0, 0]], dtype=int32)
```

14. ***Zip***

```
for x, y in zip([1, 2, 3], [4, 5, 6]):
    print(x, y)
```

```
1 4
2 5
3 6
```

```
for x, y, z in zip([1, 2, 3], [4, 5, 6], [7, 8, 9]):
    print(x, y, z)
```

```
1 4 7
2 5 8
3 6 9
```