

시즌 1 - 딥러닝의 기본 - ML lab 08

노트북: 모두를 위한 머신러닝

만든 날짜: 2019-01-07 오후 2:12

수정한 날짜: 2019-01-07 오후 5:05

작성자: rr

태그: #모두를 위한, ML lab

URL: https://search.naver.com/search.naver?ie=UTF-8&sm=whl_hy&query=%EB%94%94%EB%A9%98%EC%85%98

ML lab 08

= Matmul VS multiply

- matmul

```
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 matmul -> (2, 1)


- matrix multiply를 하지 않고 그냥 multiply를 한 경우

```
(matrix1*matrix2).eval()
array([[ 1.,  2.],
       [ 6.,  8.]], dtype=float32)
```

matmul을 한 경우와 결과가 다름

= Broadcasting

shape이 달라도 연산할 수 있게 만들어줌

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

```
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)
```

잘 모르고 사용하면 생각하지 않은 값이 만들어지니 주의!
필요하다면 같은 shape 만들어서 연산

= Reduce mean

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

결과가 1.5가 나올 것을 기대했는데 1로 나옴
int형으로 썼는지 확인, 평균을 계산할 땐 반드시 float형으로 쓰기

- axis

axis=0 -> ↓ 방향으로 평균

axis=1 -> → 방향으로 평균

axis=-1 -> 가장 안쪽에 있는 것을 평균내어라, 가장 많이 씴!
축이 없으면 모두 평균내어라

= 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
```

= Argmax

가장 큰 값의 위치

```
x = [ [0, 1, 2],  
       [3, 4, 5] ]  
tf.argmax(x, axis=0).eval()    # array([1, 1, 1])  
tf.argmax(x, axis=1).eval()    # array([2, 2])  
tf.argmax(x, axis=-1).eval()   # array([2, 2])
```

= Reshape

```
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]])
```

보통 제일 안 쪽의 값 (3) 은 건들이지 않고 모양만 바꿈

- squeeze

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

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

[[0], [1], [2]] -> [0, 1, 2]

값을 펴준다

- expand_dims

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

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

dimension을 추가하고 싶다

= One hot

```
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)
```

one_hot 하면 rank를 expand하게 됨
reshape 하면 rank 다시 줄일 수 있음

= Casting

type casting

= Stack

축을 이용해서 쌓는 방법을 바꿀 수도 있음

= Ones and Zeros like

가지고 있는 shape와 똑같은 shape 텐서를 만들어줌
ones_like -> 1로 채워진 텐서를 만들어줌
zeros_like -> 0으로 채워진 텐서를 만들어줌

= zip

한 개가 아니라 복수의 텐서를 묶어서 한번에 처리