Tensor Operation

Dong Kook Kim

Acknowledgement

https://hunkim.github.io/ml/

Exercise 01-6.

tf2-01-6-tf_operations.py

Simple 1D array and slicing

```
t = np.array([0., 1., 2., 3., 4., 5., 6.])
```

Image from http://www.frosteye.net/1233

Simple 1D array and slicing

```
t = np.array([0., 1., 2., 3., 4., 5., 6.])
pp.pprint(t)
print(t.ndim) # rank
print(t.shape) # shape
print(t[0], t[1], t[-1])
print(t[2:5], t[4:-1])
print(t[:2], t[3:])
```

```
array([ 0.,  1.,  2.,  3.,  4.,  5.,  6.])
1
(7,)
0.0 1.0 6.0
[ 2.  3.  4.] [ 4.  5.]
[ 0.  1.] [ 3.  4.  5.  6.]
```

Image from http://www.frosteye.net/1233

2D Array

```
t = np.array([[1., 2., 3.], [4., 5., 6.], [7., 8., 9.], [10., 11., 12.]])
pp.pprint(t)
print(t.ndim) # rank
print(t.shape) # shape
```

```
array([[  1.,   2.,   3.],
       [  4.,   5.,   6.],
       [  7.,   8.,   9.],
       [ 10.,  11.,  12.]])
2
(4, 3)
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-08-tensor_manipulation.ipynb

Shape, 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)
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-08-tensor_manipulation.ipynb

Shape, Rank, Axis

```
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]
        ]
    ]
]
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-08-tensor_manipulation.ipynb

Matmul VS 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)
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-08-tensor_manipulation.ipynb

Matmul VS 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)
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-08-tensor_manipulation.ipynb

Broadcasting


```
# Operations between the same shapes
matrix1 = tf.constant([[3., 3.]])
matrix2 = tf.constant([[2., 2.]])
(matrix1 + matrix2).eval()
```

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

https://docs.scipy.org/doc/numpy/user/basics.broadcasting.html

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()
```

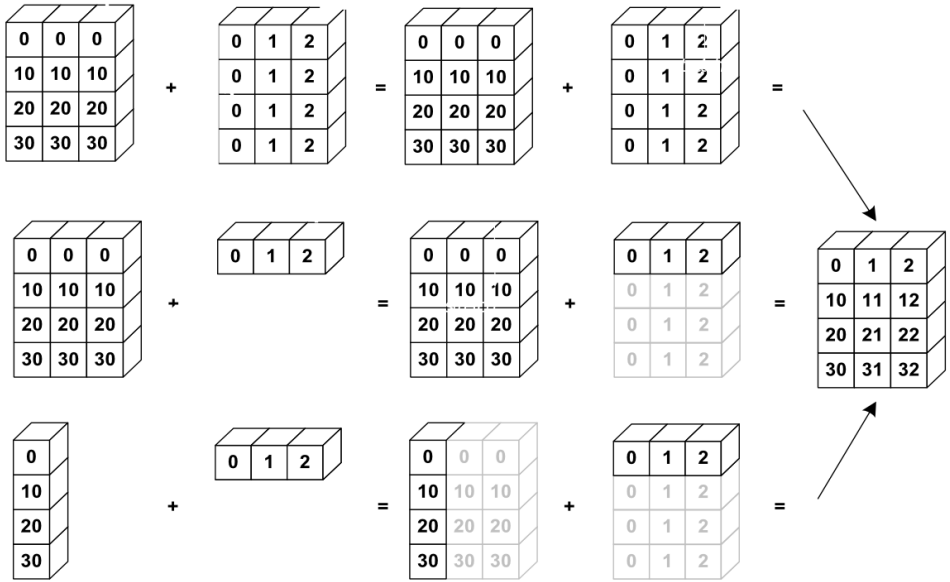
```
array([[ 4.,  6.]], dtype=float32)
```

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

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

https://docs.scipy.org/doc/numpy/user/basics.broadcasting.html

Broadcasting



range

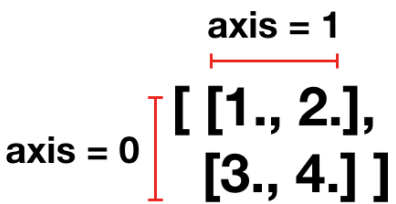
```
start=3
limit=18
steps=3
print(tf.range(start. limit, steps)) # [3, 6, 9, 12, 15]

start=3
limit=1
steps=-0.5
print(tf.range(start, limit, steps)) # [3, 2.5, 2, 1.5]

limit=5
print(tf.range(limit)) # [0, 1, 2, 3, 4]
```

Random Number Generation

- Gaussian, Uniform distributions

```
print(tf.random.normal([3]).numpy())
print(tf.random.uniform([2]))
print(tf.random.uniform([2, 3]))
```

Reduce mean



reduce_mean(data, axis=0) = [2., 3.]
reduce_mean(data, axis=1) = [1.5, 3.5]

```
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)
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-08-tensor_manipulation.ipynb

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
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-08-tensor_manipulation.ipynb

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])
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-08-tensor_manipulation.ipynb

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]]])
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-08-tensor_manipulation.ipynb

Reshape (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)
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-08-tensor_manipulation.ipynb

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)
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-08-tensor_manipulation.ipynb

Casting

```
tf.cast([1.8, 2.2, 3.3, 4.9], tf.int32).eval()
```

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

```
tf.cast([True, False, 1 == 1, 0 == 1], tf.int32).eval()
```

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

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-08-tensor_manipulation.ipynb

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)
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-08-tensor_manipulation.ipynb

Ones 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)
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-08-tensor_manipulation.ipynb

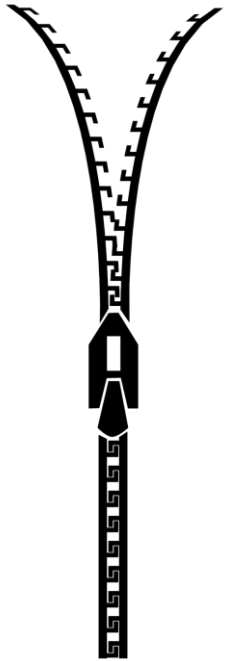
Zip

```
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
```



https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-08-tensor_manipulation.ipynb