Tensor manipulation

Slicing

In [1]:

```
import tensorflow as tf
import numpy as np

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

print(t.ndim, t.shape)
print(t[-1], t[4:-1], t[:2], t[4:])
```

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

In [2]:

```
t2 = np.array([[0., 1., 2.], [3., 4., 5.], [6., 7., 8.], [9., 10., 
11.]])

print(t2)
print(t2.ndim, t2.shape)
print(t2[-1])
print(t2[2:-1])
print(t2[:2])
print(t2[2:])
print(t2[-1, -1])
print(t2[1:-1, 0:1])
print(t2[2:,:2])
```

```
[[  0.   1.   2.]
 [  3.   4.   5.]
 [  6.   7.   8.]
 [  9.  10.  11.]]
2 (4, 3)
[  9.  10.  11.]
[[6. 7. 8.]]
[[0. 1. 2.]
 [3. 4. 5.]]
[[  6.   7.   8.]
 [  9.  10.  11.]]
11.0
[[3.]
 [6.]]
[[  6.   7.]
 [  9.  10.]]
```

Shape, Rank, Axis

In [3]:

```
sess = tf.Session()

t3 = tf.constant([1, 2, 3, 4])
_t3 = t3.eval(session=sess)
print(_t3, _t3.shape, t3)

t4 = tf.constant([[1,2],[3,4]])
_t4 = sess.run(t4)
print(_t4, _t4.shape, t4)
```

```
[1 2 3 4] (4,) Tensor("Const:0", shape=(4,), dtype=int32)
[[1 2]
 [3 4]] (2, 2) Tensor("Const_1:0", shape=(2, 2), dtype=int32)
```

Matmul vs multiply

In [4]:

```
m1 = tf.constant([[1., 2.], [3., 4.]])
m2 = tf.constant([[1.], [2.]])

_m1, _m2 = sess.run([m1, m2])
print(_m1, "matrix 1 shape: ", _m1.shape)
print(_m2, "matrix 2 shape: ", _m2.shape)
mm = tf.matmul(m1, m2)
mm2 = m1 * m2
_mm, _mm2 = sess.run([mm, mm2])
print(_mm)
print(_mm2)
```

```
[[1. 2.]
 [3. 4.]] matrix 1 shape:  (2, 2)
[[1.]
 [2.]] matrix 2 shape:  (2, 1)
[[ 5.]
 [11.]]
[[1. 2.]
 [6. 8.]]
```

Broadcasting

In [5]:

```
# Operations between same shapes
m3 = tf.constant([[3., 1.]])
m4 = tf.constant([[2., 2.]])
mm3 = m3 + m4
_mm3 = sess.run(mm3)
print(_mm3)

# Operations between different shapes
m5 = tf.constant(3.)
mm5 = m3 + m5
_mm5 = sess.run(mm5)
print(_mm5)

m6 = tf.constant([[3.], [4.]])
mm6 = m3 + m6
_mm6 = sess.run(mm6)
print(_mm6)
```

```
[[5. 3.]]
[[6. 4.]]
[[6. 4.]
 [7. 5.]]
```

Reduce mean

In [6]:

```
r1 = tf.reduce_mean([1, 2], axis=0)
print(sess.run(r1)) # 1, not 1.5 because of int32

_r2 = [[ 1., 2.],
      [ 3., 4.]]
print("_r2[0][1] = ", _r2[0][1])
# [0] is index of axis = 0, [1] is index of axis = 1
r2 = tf.reduce_mean(_r2)
print(sess.run(r2))

r3 = tf.reduce_mean(_r2, axis=0)
print(sess.run(r3))

r4 = tf.reduce_mean(_r2, axis=1)
print(sess.run(r4))

r5 = tf.reduce_mean(_r2, axis=-1)
print(sess.run(r5))

_r6 = np.array([[[1., 2.],[3., 4.]],
              [[5., 6.],[7., 8.]]])
print("-- 2x2x2 ------")
print(_r6.shape)

r6 = tf.reduce_mean(_r6, axis=0)
print("---------------")
print(_r6[0,:,:])
print(_r6[1,:,:])
print(sess.run(r6))

r7 = tf.reduce_mean(_r6, axis=1)
print("---------------")
print(_r6[:,0,:])
print(_r6[:,1,:])
print(sess.run(r7))

r8 = tf.reduce_mean(_r6, axis=2)
print("---------------")
print(_r6[:,:,0])
print(_r6[:,:,1])
print(sess.run(r8))
```

```
1
_r2[0][1] =  2.0
2.5
[2. 3.]
[1.5 3.5]
[1.5 3.5]
-- 2x2x2 ------
(2, 2, 2)
---------------
[[1. 2.]
 [3. 4.]]
[[5. 6.]
 [7. 8.]]
[[3. 4.]
 [5. 6.]]
---------------
[[1. 2.]
 [5. 6.]]
[[3. 4.]
 [7. 8.]]
[[2. 3.]
 [6. 7.]]
---------------
[[1. 3.]
 [5. 7.]]
[[2. 4.]
 [6. 8.]]
[[1.5 3.5]
 [5.5 7.5]]
```

reduce_sum

In [7]:

```
_s2 = np.array(_r2)
s1 = tf.reduce_sum(_s2, axis=0)
print("---------------")
print(_s2[0,:])
print(_s2[1,:])
print(sess.run(s1))

s2 = tf.reduce_sum(_s2, axis=1)
print("---------------")
print(_s2[:,0])
print(_s2[:,1])
print(sess.run(s2))

s3 = tf.reduce_mean(tf.reduce_sum(_s2, axis=-1))
print("---------------")
print(sess.run(s3))
```

```
---------------
[1. 2.]
[3. 4.]
[4. 6.]
---------------
[1. 3.]
[2. 4.]
[3. 7.]
---------------
5.0
```

argmax

In [8]:

```
_m2 = np.array([[0, 1, 2],
                [2, 1, 0]])
m1 = tf.argmax(_m2, axis=0)
print("---------------")
print(_m2[0,:])
print(_m2[1,:])
print(sess.run(m1)) # return value is index of series

m2 = tf.argmax(_m2, axis=1)
print("---------------")
print(_m2[:,0])
print(_m2[:,1])
print(_m2[:,2])
print(sess.run(m2))
```

```
---------------
[0 1 2]
[2 1 0]
[1 0 0]
---------------
[0 2]
[1 1]
[2 0]
[2 0]
```

reshape, squeeze, expand

In [9]:

```
print(_m2.shape)

h1 = tf.reshape(_m2, shape=[-1, 2])
print(sess.run(h1))

h2 = tf.reshape(_m2, shape=[-1, 1, 1, 2])
h3 = sess.run(h2)
print(h3)

h4 = tf.squeeze([[0], [1], [2]])
print(sess.run(h4))

h5 = tf.squeeze(h3)
h6 = sess.run(h5)
print(h6)

h7 = tf.expand_dims(h6, 1)
h8 = sess.run(h7)
print(h6.shape, h8.shape, h8)
```

```
(2, 3)
[[0 1]
 [2 2]
 [1 0]]
[[[[0 1]]]


 [[[2 2]]]


 [[[1 0]]]]
[0 1 2]
[[0 1]
 [2 2]
 [1 0]]
(3, 2) (3, 1, 2) [[[0 1]]

 [[2 2]]

 [[1 0]]]
```

onehot

In [10]:

```
_o0 = np.array([[0], [1], [2], [0]])
_o1 = tf.one_hot(_o0, depth=4)
o1 = sess.run(_o1)
print(o1)

_o2 = tf.reshape(o1, shape=[-1, 4])
o2 = sess.run(_o2)
print(o2)

_o3 = tf.argmax(o2, axis=-1)
o3 = sess.run(_o3)
print(o3)
```

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

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

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

 [[1. 0. 0. 0.]]]
[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [1. 0. 0. 0.]]
[0 1 2 0]
```

casting

In [11]:

```
c1 = tf.cast([1.8, 2.2, 3.3, 4.4], tf.int32).eval(session=sess)
print(c1)

c2 = tf.cast([True, False, 1 == 1, 1 == 0], tf.int32).eval(session=
sess)
print(c2)
```

```
[1 2 3 4]
[1 0 1 0]
```

stack

In [12]:
```
_c3 = [1, 2]
_c4 = [3, 4]
_c5 = [5, 6]
c6 = tf.stack([_c3, _c4, _c5]).eval(session=sess)
print(c6)

c7 = tf.stack([_c3, _c4, _c5], axis=1).eval(session=sess)
print(c7)
```

```
[[1 2]
 [3 4]
 [5 6]]
[[1 3 5]
 [2 4 6]]
```

ones and zeros like

In [13]:
```
print(_m2)
z1 = tf.ones_like(_m2).eval(session=sess)
print(z1)

z2 = tf.zeros_like(_m2).eval(session=sess)
print(z2)
```

```
[[0 1 2]
 [2 1 0]]
[[1 1 1]
 [1 1 1]]
[[0 0 0]
 [0 0 0]]
```

zip

In [14]:
```
_z3 = [1, 2, 3]
_z4 = [4, 5, 6]
_z5 = [7, 8, 9]

for x, y in zip(_z3, _z4):
    print(x, y)

for x, y, z in zip(_z3, _z4, _z5):
    print(x, y, z)
```

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