tensorflow接口研读constant_op

constant_op函数使用，分为生成常量，序列以及随机值。随机种子在文末介绍使用方法。

2.1 tf.zeros(shape, dtype=tf.float32, name=None)

功能：生成一个值全为0的tensor。默认为float32类型。
输入：shape：一维的int32型的列表。
例：

```
a=tf.zeros([2,3])

a==>[[0. 0. 0.]
     [0. 0. 0.]]
```

2.2 tf.zeros_like(tensor, dtype=None, name=None, optimize=True)

功能：生成一个值全为0的tensor，其形状与输入tensor相同。
输入：dtype:未指定时返回tesnsor的类型
例：

```
x=tf.constant([1,2,3,4,5,6],shape=[2,3])
a=tf.zeros_like(x)

a==>[[0. 0. 0.]
     [0. 0. 0.]]
```

2.3 tf.ones(shape, dtype=tf.float32, name=None)

功能：生成一个值全为1的tensor。默认为float32类型。
输入：shape：一维的int32型的列表。
例：

```
a=tf.ones([2,3])

a==>[[1. 1. 1.]
     [1. 1. 1.]]
```

2.4 tf.ones_like(tensor, dtype=None, name=None, optimize=True)

功能：生成一个值全为1的tensor，其形状与输入tensor相同。
输入：dtype:未指定时返回tesnsor的类型
例：

```
x=tf.constant([1,2,3,4,5,6],shape=[2,3])
a=tf.ones_like(x)

a==>[[1. 1. 1.]
     [1. 1. 1.]]
```

2.5 tf.fill(dims, value, name=None)

功能：生成一个值全为value的tensor，其形状与dims相同。
输入：dims：一维的int32型的列表，
例：

```
a=tf.fill([2,3],7)

a==>[[7. 7. 7.]
     [7. 7. 7.]]
```

2.6 tf.constant(value, dtype=None, shape=None, name='Const', verify_shape=False)

功能：生成一个常量tensor
输入：value：一个常量，或者一个list;
dtype：数据类型;
shape：生成形状。
例：

```
a=tf.constant([1,2,3,4,5,6],shape=[2,3])
b=tf.constant(2,shape=[2,3])

a==>[[1 2 3]
     [4 5 6]]
b==>[[2 2 2]
     [2 2 2]]
```

2.7 tf.linspace(start, stop, num, name=None)

功能：生成在区间[start,stop]中定长间隔的值。序列值的间隔大小为‘(stop-start)/(num-1)’
输入：start：区间起始值，类型为float32或float64;
stop：区间中止值，类型为float32或float64;
num：生成数据数量。
例：

```
a=tf.linspace(1.,7.,4)

a==>[1. 3. 5. 7.]
```

2.8 tf.range(start, limit=None, delta=1, dtype=None, name='range')

功能：生成一个序列值，从start开始，每次递增delta，直到不超过limit的值结束。
输入：start：起始值;
limit：限制值，不能超过;
delta：步长。
例：

```
a=tf.range(1,10,3)

a==>[1 4 7]
```

2.9 tf.random_normal(shape, mean=0.0, stddev=1.0, dtype=tf.float32, seed=None, name=None)

功能：从正太分布中随机输出。
输入：shape：一维整型tensor，指定tensor的形状;
mean：正太分布的平均值，默认为0;
stddev：正太分布的标准差，默认为1;
seed：随机种子。
例：

```
a=tf.random_normal([2,2],seed=112)

a==>[[-0.72891599 -1.35909426]
     [ 0.06045228  1.12680387]]
```

2.10 tf.truncated_normal(shape, mean=0.0, stddev=1.0, dtype=tf.float32, seed=None, name=None)

功能：从截断正太分布中随机输出。
输入：shape：一维整型tensor，指定tensor的形状;
mean：正太分布的平均值，默认为0;
stddev：正太分布的标准差，默认为1;
seed：随机种子。
例：

```
a=tf.truncated_normal([2,2],seed=112)

a==>[[-0.72891599 -1.35909426]
     [ 0.06045228  1.12680387]]
```

2.11 tf.random_uniform(shape, minval=0, maxval=None, dtype=tf.float32, seed=None, name=None)

功能：从均匀分布中随机输出，默认区域为‘[0,1)’。
输入：shape：一维整型tensor，指定tensor的形状;
minval：均匀分布的最小值，默认为0;
maxval：均匀分布的最大值，如果类型为float32则默认为1;
seed：随机种子。
例：

```
a=tf.random_uniform([2,2],seed=112)

a==>[[0.30445623 0.57834935]
     [0.52905083 0.00853038]]
```

2.12 tf.random_shuffle(value, seed=None, name=None)

功能：将tensor第一个维度的数据重新随机排列。
输入：value：tensor。
seed：随机种子。
例：

```
x=tf.constant([1,2,3,4,5,6],shape=[3,2])
a=tf.random_shuffle(x,seed=11)

a==>[[5 6]
     [3 4]
     [1 2]]
```

2.13 tf.random_crop(value, size, seed=None, name=None)

功能：将tensor按照指定大小进行随机裁剪。

输入：value：tensor；
size：裁剪后的大小，size<=value.shape，如果不想改变大小，应配置为value的shape。
例：

```
x=tf.constant([1,2,3,4,5,6,7,8,9],shape=[3,3])
a=tf.random_crop(x,size=[2,2],seed=11)

a==>[[2 3]
     [5 6]]
```

2.14 tf.multinomial(logits, num_samples, seed=None, name=None)

功能：绘制多项式分布。
输入：logits：shape为[batch_size,num_classes]的2维tensor，每行[i，：]代表每类出现的概率。
num_samples：独立采样数目。
例：

```
x=tf.constant([[1,1,1]],dtype=tf.float32)#表示有3类，出现概率相等。
a = tf.multinomial(x, 10)

a==>[[0 0 2 1 2 1 2 0 0 1]]
```

2.15 tf.random_gamma(shape, alpha, beta=None, dtype=tf.float32, seed=None, name=None)

功能：对每一个给定的gamma分布进行shape尺度的采样。
例如samples = tf.random_gamma([30], [[1.],[3.],[5.]], beta=[[3., 4.]])
即给定了6个gamma分布，每个分布输出30个数据，输出tensor形状为[30,3,2]。
输入：shape：每一个gamma分布进行采样的尺度；
alpha： gamma分布的alpha变量，可以为任意尺度，但需和beta对应；
beta：gamma分布的beta变量。
例：

```
a = tf.random_gamma([2], [0.5, 1.5])#每一片应为[:,0],[:,1]

a==>[[3.03589034e+00 2.56953764e+00]
     [2.959940042e-03 3.52107930e+00]]
```

2.16 tf.set_random_seed(seed)

功能：设置随机数种子。
为确保每次随机数生成数据一致，可以设置随机数种子。随机种子有两种设置方法：
1、op级别的设置，如前文提高的随机输入tf.random_shuffle(value, seed=None, name=None)中，
函数变量seed即设置种子值。
2、graph级别，即tf.set_random_seed(seed)函数，可使整个graph的随机数产生从设置种子中获取。
例：
1、未设置种子

```
a=tf.random_uniform([1])
b=tf.random_normal([1])
print("Session 1")
with tf.Session() as sess1:
    tf.global_variables_initializer().run()
    print sess1.run(a)
    print sess1.run(a)
    print sess1.run(b)
    print sess1.run(b)
print("Session 2")
with tf.Session() as sess2:
    tf.global_variables_initializer().run()
    print sess2.run(a)
    print sess2.run(a)
    print sess2.run(b)
    print sess2.run(b)
```

运行结果为：

```
Session 1
[ 0.71059418]
[ 0.71678996]
[ 0.27808592]
[ 0.77504641]
Session 2
[ 0.45193291]
[ 0.74479854]
[-0.01035937]
[ 0.54787332]
```

因没有设置随机种子，每次运行结果都不一样
2、设置op级别种子

```
a=tf.random_uniform([1],seed=1)
b=tf.random_normal([1])
print("Session 1")
with tf.Session() as sess1:
```

```
    tf.global_variables_initializer().run()
    print sess1.run(a)
    print sess1.run(a)
    print sess1.run(b)
    print sess1.run(b)
print("Session 2")
with tf.Session() as sess2:
    tf.global_variables_initializer().run()
    print sess2.run(a)
    print sess2.run(a)
    print sess2.run(b)
    print sess2.run(b)
```

运行结果为：

```
Session 1
[ 0.23903739]
[ 0.22267115]
[-0.48983803]
[-0.13116723]
Session 2
[ 0.23903739]
[ 0.22267115]
[-1.77008951]
[-0.18568291]
```

变量a设置为种子1，每次运行按照种子进行取数，每个Session都从种子的第一个数开始取值。

3、设置graph级别种子

```
tf.set_random_seed(1)
a=tf.random_uniform([1])
b=tf.random_normal([1])
print("Session 1")
with tf.Session() as sess1:
    tf.global_variables_initializer().run()
    print sess1.run(a)
    print sess1.run(a)
    print sess1.run(b)
    print sess1.run(b)
print("Session 2")
with tf.Session() as sess2:
    tf.global_variables_initializer().run()
    print sess2.run(a)
    print sess2.run(a)
    print sess2.run(b)
    print sess2.run(b)
```

运行结果为：

```
Session 1
[ 0.77878559]
[ 0.0978868]
[-0.4487586]
[-0.82540691]
Session 2
[ 0.77878559]
[ 0.0978868]
[-0.4487586]
[-0.82540691]
```

设置graph级种子后，两次运行结果完全一致。