1. 如果遇到OpKernal for known op….，在代码首部添加
   1. import os
   2. os.environ[‘TF\_CPP\_MIN\_LOG\_LEVEL’] = 3
2. tensorflow代码中已经把sub()函数改成subtract()
3. initialize\_all\_variables()函数已经更新为global\_variables\_initializer()
4. 变量初始化
   1. tf.Variable(tf.random\_normal(shape,stddev = 0.35, name=’’))
5. 如果想输出TensorFlow中tensor的值

print(tensor.eval(session=tf.Session()))

or

print(sess.run(tensor))

当然也可以直接输出两个tensor的加减乘除的结果，如

print(sess.run(tensor1+tensor2))

1. tf.fill([row\_dim,col\_dim],42)与tf.constant(42, [row\_dim,col\_dim])等价
2. 我们可以基于其它tensor的shape初始化变量

zeros\_similar = tf.zeros\_like(tensor)

ones\_similar = tf.ones\_like(tensor)

在使用tf.global\_variables\_initializer()初始化上述变量时，需按顺序，先初始化tensor，后初始化zeros\_similar和ones\_similar

1. 随机分布tensor：

randunif\_tsr = tf.random\_uniform([row\_dim, col\_dim], minval=0, maxval=1)

注意：取值范围是minval<=x<maxval

生成一个正态分布：

randnorm\_tsr = tf.random\_normal([row\_dim, col\_dim], mean=0.0, stddev=1.0)

1. 随机条目

shuffled\_output = tf.random\_shuffle(input\_tensor)

cropped\_output = tf.random\_crop(input\_tensor, crop\_size)

注：crop\_size是一个arrays格式[]

1. 特征值和特征向量

tf.self\_adjoint\_eig(tensor)

输出：第一行是特征值，剩下的向量中是随后的向量。数学中，称之为举证的特征分解。

1. 四则运算

print(sess.run(tf.div(3, 4)))

0

tf.div(a, b)输出与输入类型相同

print(sess.run(tf.truediv(3, 4)))

0.75

print(sess.run(tf.floordiv(3.0, 4.0)))

0.0

取余

print(sess.run(tf.mod(22.0, 5.0)))

2.0

向量积

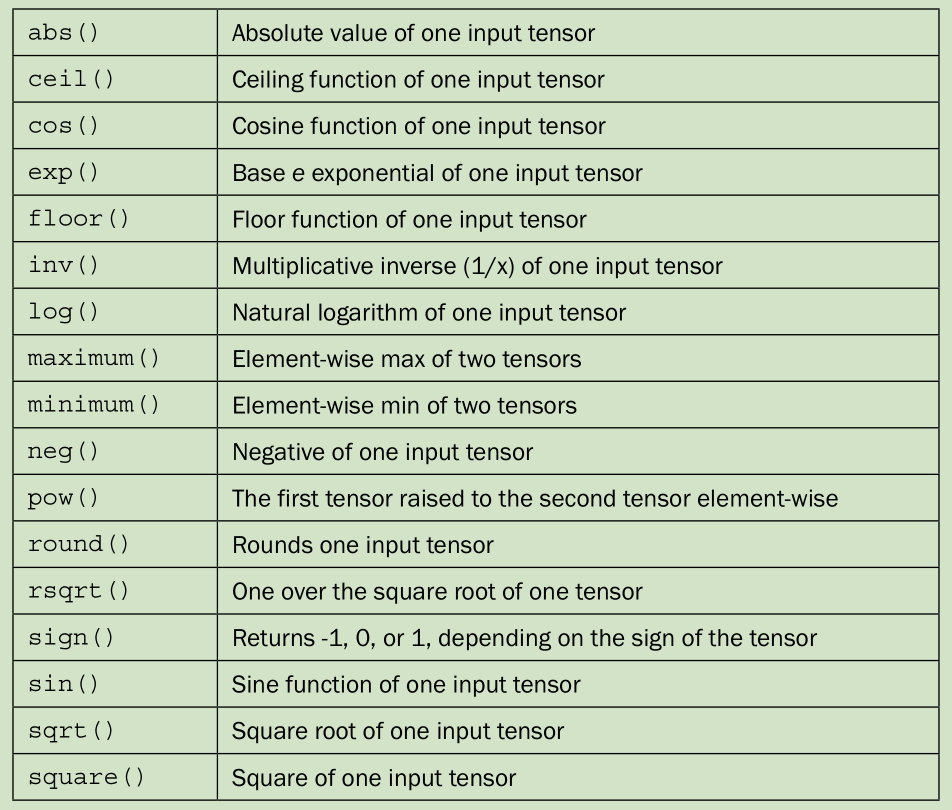
print(sess.run(tf.cross([1., 0., 0.], [0., 1., 0.])))

[0. 0. 1.0]

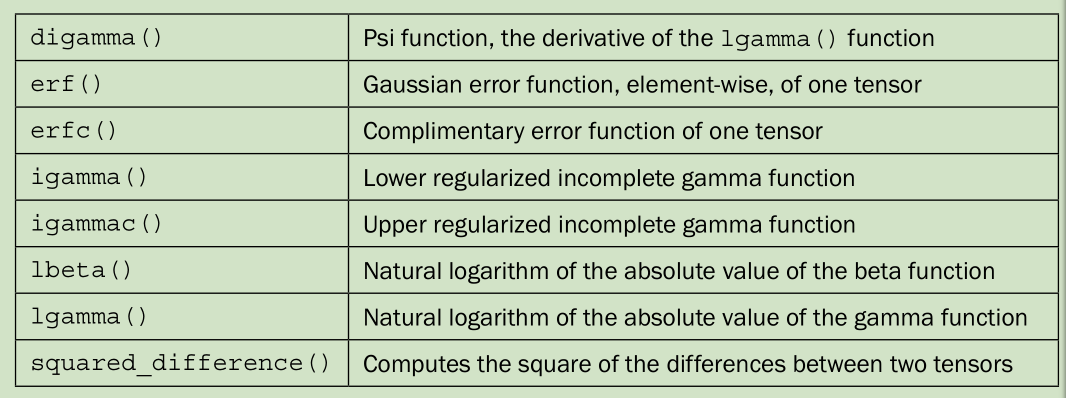
注：现在只能计算3-D的向量

tf.mul  tf.sub   tf.neg 已经废弃

分别可用tf.multiply  tf.subtract  tf.negative替代



一些特别的数学函数



1. 激励函数

* 修正的线性单元（rectified linear unit, relu）

print(sess.run(tf.nn.relu([-3., 3., 10.])))

[0. 3. 10.]

当我们希望修改线性增长部分时，我们可以给max(0, x)函数从筑一个min()函数。如min(max(0, x), 6)。这可以避免数值消失或这数值爆炸，而且计算速度加快。

print(sess.run(tf.nn.relu6([-3., 3., 10.])))

[0. 3. 6.]

* sigmoid

print(sess.run(tf.nn.sigmoid([-1., 0., 1.])))

[0.26894143 0.5 0.7310586]

* hyper tangent

tanh = (exp(x)-exp(-x))/(exp(x)+exp(-x))

print(sess.run(tf.nn.tanh([-1., 0., 1.])))

[-0.76159418 0. 0.76159418]

* softsign

softsign = x/(abs(x) + 1)

print(sess.run(tf.nn.softsign([-1., 0., 1.])))

[-0.5 0. 0.5]

* softplus

softplus = log(exp(x) + 1)

print(sess.run(tf.nn.softplus([-1., 0., 1.])))

[0.31326166 0.69314718 1.31326163]

* ELU(exponential linear unit)指数线性单元

elu = (exp(x) + 1) if x < 0 else x

print(sess.run(tf.nn.elu([-1., 0., -1.])))

[-0.63212055 0. 1. ]

1. 变量的优化算法

* 标准的梯度下降法

tf.train.GradientDescentOptimizer()函数有时容易陷入局部最优

* tf.train.MomentumOptimizer()函数会把前段时间的梯度下降值考虑在内
* tf.train.AdagradOptimizer()函数会在变量变化较小时采用较大的步伐，变化较大时采用较小的步伐。但是容易出现梯度快速变零
* 为了避免梯度快速变零，限制使用的步数。tf.train.AdadeltaOptimizer()函数
* 更多梯度下降算法，参考：

<https://www.tensorflow.org/api_docs/python/train/optimizers>