# Tensorflow如何检查变量是否存在？

tf.Varaible v.s. tf.get\_variable

tf.Variable函数会返回一个variable，如果给出的name已经存在，会自动修改name，生成个新的。

kernel = tf.Variable(tf.truncated\_normal(kernel\_shape,

dtype=tf.float32,stddev=0.1))

tf.get\_variable函数拥有一个变量检查机制，会检测已经存在的变量是否设置为共享变量，如果已经存在的变量没有设置为共享变量，TensorFlow 运行到第二个拥有相同名字的变量的时候，就会报错。

w = tf.get\_variable('weights', [x.get\_shape()[1], out\_dim],

tf.float32, initializer=tf.random\_normal\_initializer(

stddev=np.sqrt(1.0/out\_dim)))

tf.get\_variable一般和tf.variable\_scope配合使用，用于在同一个的变量域中共享同一个变量，或在不同变量域中使用同名变量。

# Tensorflow 计算符重载：

Tf.add(a,b)和a+b在计算精度上没有区别。类似还有tf.assign和=

tf.nn.bias\_add 是 tf.add 的一个特例

# Tensorflow 变量集合：

* tf.add\_to\_collection：把变量放入一个集合，把很多变量变成一个列表
* tf.get\_collection：从一个结合中取出全部变量，是一个列表
* tf.add\_n：把一个列表的东西都依次加起来

Eg:

(1) if w not in tf.get\_collection(WEIGHT\_DECAY\_KEY):

tf.add\_to\_collection(WEIGHT\_DECAY\_KEY, w)

(2) costs = [tf.nn.l2\_loss(var) for var in tf.get\_collection(utils.WEIGHT\_DECAY\_KEY)]

l2\_loss = tf.multiply(self.\_hp.weight\_decay, tf.add\_n(costs))

total\_loss = self.\_loss\_list[i] + l2\_loss

# Tensorflow 中op之间的依赖：

有两种方式，tf.group和tf.control\_dependencies

(1) control\_dependencies保证在执行完ema\_op后执行代码块中的op。

**import** tensorflow **as** tf

w = tf.Variable(1.0)

ema = tf.train.ExponentialMovingAverage(0.9)

update = tf.assign\_add(w, 1.0)

ema\_op = ema.apply([update])

**with** tf.control\_dependencies([ema\_op]):

ema\_val = ema.average(update)# 输出0，代码块中没有op，而是tensor。导致control\_dependency失效

# ema\_val = tf.identity(ema.average(update)) #解决办法, 一个identity搞定,the op adds send/recv nodes to the graph

**with** tf.Session() **as** sess:

tf.global\_variables\_initializer().run()

**for** i **in** range(3):

**print**(sess.run([ema\_val]))

1. tf\_group打包一个操作集合，c = tf.group(a,b)，当c执行完，操作集合中所有操作执行完，没有返回。

相当于：

with tf.control\_dependencies([a, b]):

c = tf.no\_op()

# Tesnsorflow学习率退火：

tf.train.exponential\_decay， 默认staircase为False，为指数退火，True为指定迭代间隔退火。

指数退火：

Tensoflow的graph概念：

定义：graph是一系列可以作为整体执行的操作。既可以运行单个默认依赖图表，也可以利用多个图表。

Sess.run() 运行默认图表中作为参数传入的操作相对应的完整自己

tf.GraphKeys.UPDATE\_OPS。是update\_ops的集合（包括批处理的移动平均和方差操作），这些操作不是用户创建的图表的父图表，在一个独立图表中。

# Sess.run和tf.eval的区别：

* t.eval()  tf.get\_default\_session().run(t).
* 每次使用 eval 和 run时，都会执行整个计算图，为了获取计算的结果，将它分配给tf.Variable，然后获取。
* 最主要的区别： 可以使用sess.run()在同一步获取多个tensor中的值，

# Tensorflow载入数据：

例子见code。

取样本的时候，一个Reader先从文件名队列中取出文件名，读出数据，Decoder解析后进入样本队列。

