1 滑动平均模型

使模型在测试数据上更健壮，就是对某些变量如weights和bias进行滑动平均，使这些值在表现上更能和所有的数据产生关系。

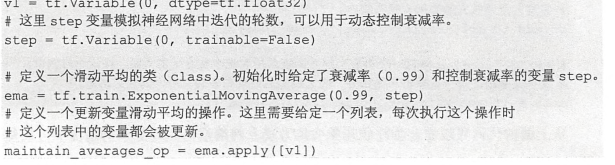
需要提供一个衰减率，用于控制模型更新的速度。

Shadow\_variable = decay\* shadow\_variable+(1-decay)\*variable

Decay一般会设置成接近１的数。Num\_updates 参数动态设置decay的大小

Min(decay, 1+num\_updates/10+num\_updates)

衰减率和控制衰减率



2 计算某层输出结果

Layer1 = tf.nn.relu(tf.matmul(input\_tensor,weights1)+biases1)

Layer1 = tf.nn.relu(tf.matmul(input\_tensor,avg\_class.average(weights1))+avg\_class.average(biases1))

3 学习率的设置

控制参数更新的速度，可以认为是步长。Tf.train.exponential\_decay函数实现了指数衰减学习率。

Decayed\_learning\_rete = learning\_rate \* decay\_rate^(global\_step/decay\_steps)

如果staircase为True global\_step/decay\_steps 被转换成整数

每训练一次，学习率就减小一次。

# 设置指数衰减的学习率  
learning\_rate = tf.train.exponential\_decay(LEARNING\_RATE\_BASE,  
 global\_step,mnist.train.num\_examples/BATCH\_SIZE,  
 LEARNING\_RATE\_DECAY)  
# 使用tf.train.GradientDescentOptimizer 优化算法来优化损失函数  
# 梯度下降求的是什么，weight，最终目的是什么， 假设现在weighs的值确定，那么就需要固定weight然后变换某一个变量来达到优化的目的  
# 这里通过学习率的变化来使误差最小。  
train\_step=tf.train.GradientDescentOptimizer(learning\_rate).minimize(loss,global\_step=global\_step)

下面的方法的参数是learning\_rate,其传入的参数是minimize中的global\_step = global\_step的值

认知上的差异: 在做python实现神经网络Bp的时候，显式的更新权重去减小误差，但是TF这里就完成了对内部所有变量的训练以及更新，  
Add operations to minimize `loss` by updating `var\_list  
有一个var\_list 去被训练更新。

**for** k **in** tf.trainable\_variables():  
 print(k.name,k.eval())

有四个variable，这里分别是两个weight和偏置

train\_op=tf.group(train\_step,variables\_averages\_op)

Train\_step 用来反向偏置的优化，优化之后算出新的权重和偏置

Variables\_averages\_op 将滑动平均函数作用于某些参数，让这些参数变为滑动平均值

Run(accuracy,) 训练得到正确率，这个过程会调用一系列的Tf op操作

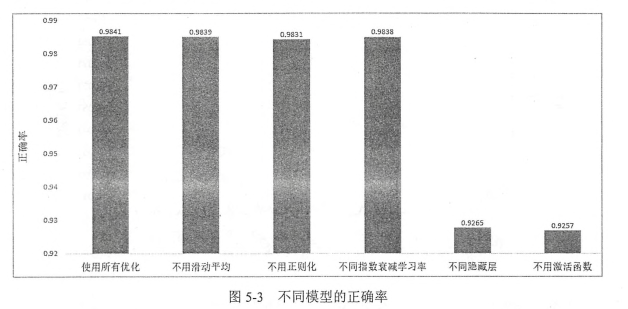
sess.run(train\_op,feed\_dict={x:xs,y\_:ys})

Train\_op 中的train\_step 中包含了一系列操作，从头开始有前向传播获取结果，有BP传播优化loss，而loss会实现前向传播获取值的操作，再调用反向传播优化算法就会将前向和后向结合起来。但是这里面的variables\_averages\_op 这个Op比较奇怪，只是调用了apply方法将所有变量赋予滑动平均，但是感觉调用的顺序有点不对。

train\_op=tf.group(variables\_averages\_op,train\_step)

这样的话感觉上更加舒服。

不同模型的正确率



调整神经网络的结构对最终的正确率有非常大的影响。没有隐藏层或者隐藏层没有激活函数时，模型的正确率只有92.6%,这个数据远小于使用了隐藏层和激活函数时的正确率，这说明神经网络的结构对最终模型的效果有本质的影响。

总的来说，通过MNIST数据集有效的验证了激活函数，隐藏层可以给模型的效果带来质的飞跃。由于MNIST问题本身相对简单，滑动平均模型、指数衰减的学习率和正则化损失对最终正确率的提升效果不明显。但通过进一步分析实验的结果，可以得出这些优化方法确实可以解决第4章中提到的神经网络优化过程中的问题，当需要解决的问题和使用到的神经网络模型更加复杂时，这些优化方法将更有可能对训练效果产生更大的影响。