姓名：向建宇

学号：1801210687

**作业任务：设置两个以上不同的学习率值和滑动平均模型的decay值，运行例子程序，观察及显示运行结果。**

1.**设置不同的学习率运行及结果：**

由于在反向传输时，不同的学习率会使得更新的速度有所不同，设置的学习率过大，可能导致参数在极优值的两侧来回移动。设置得太小又会迭代很多次才达到效果，而指数形式得学习率可以使得数据在前期下降得相当快在后面趋于稳定的时候缓慢增长，从而达到快速得极优解的目的。

**首先是常数学习率：**

设置学习率为0.5，并每隔十次迭代打印一次

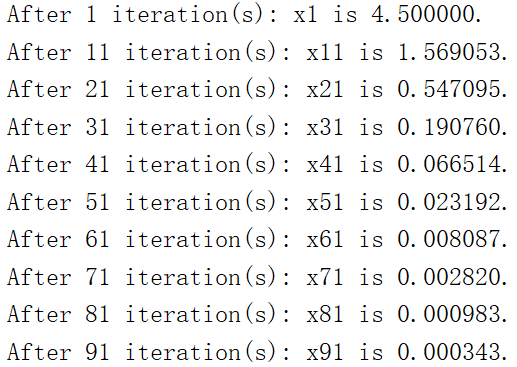
|  |
| --- |
| train\_op1 = tf.train.GradientDescentOptimizer(0.05).minimize(y) **with** tf.Session() **as** sess:  sess.run(tf.global\_variables\_initializer())  **for** i **in** range(100):  sess.run(train\_op1)  **if** i % 10 == 0:  x\_value = sess.run(x)  print(**"After %s iteration(s): x%s is %f."** % (i + 1, i + 1, x\_value)) |

下面是设置指数函数的学习率：

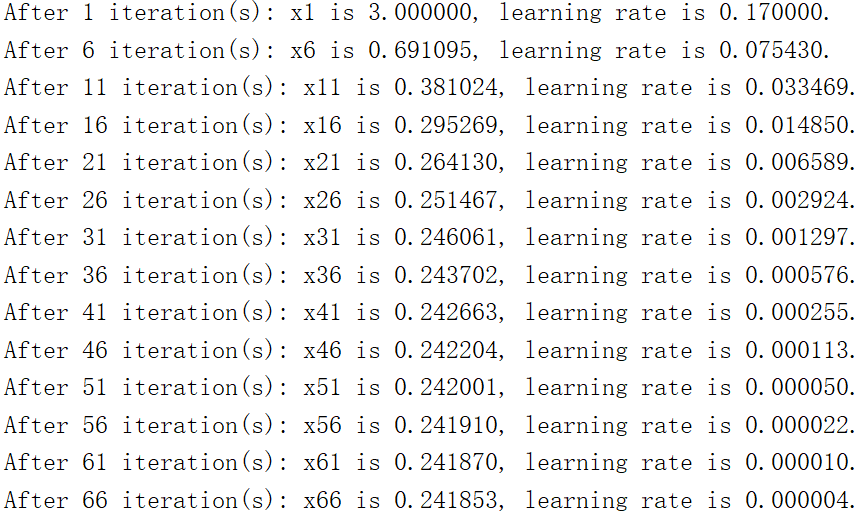
|  |
| --- |
| global\_step = tf.Variable(0) LEARNING\_RATE1 = tf.train.exponential\_decay(0.2, global\_step, 1, 0.85, staircase=**True**) train\_op2 = tf.train.GradientDescentOptimizer(LEARNING\_RATE1).minimize(y, global\_step=global\_step) **with** tf.Session() **as** sess:  sess.run(tf.global\_variables\_initializer())  **for** i **in** range(70):  sess.run(train\_op2)  **if** i % 5 == 0:  LEARNING\_RATE\_value1 = sess.run(LEARNING\_RATE1)  x\_value = sess.run(x)  print(**"After %s iteration(s): x%s is %f, learning rate is %f."** % (  i + 1, i + 1, x\_value, LEARNING\_RATE\_value1)) |

**结果截图：**

常数学习率：



指数学习率：



可以很明确的看见指数函数下降的更快，并且随着迭代次数的增加，趋于稳定

详细代码见附录1

**2.设置不同的滑动平均模型运行及结果：**

移动平均法是用一组最近的实际数据值来预测未来一期或几期内公司产品的需求量、公司产能等的一种常用方法。移动平均法适用于即期预测。当产品需求既不快速增长也不快速下降，且不存在季节性因素时，移动平均法能有效地消除预测中的随机波动，是非常有用的。移动平均法根据预测时使用的各元素的权重不同

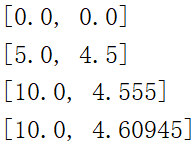
移动平均法是一种简单平滑预测技术，它的基本思想是：根据时间序列资料、逐项推移，依次计算包含一定项数的序时平均值，以反映长期趋势的方法。因此，当时间序列的数值由于受周期变动和随机波动的影响，起伏较大，不易显示出事件的发展趋势时，使用移动平均法可以消除这些因素的影响，显示出事件的发展方向与趋势（即趋势线），然后依趋势线分析预测序列的长期趋势。

主要是用ExponentialMovingAverage（）函数来实现

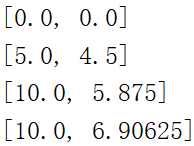
其主要代码为：

|  |
| --- |
| v1 = tf.Variable(0, dtype=tf.float32)  #step变量模拟神经网络中的迭代次数，可以用于动态控制衰减率 step = tf.Variable(0, trainable=**False**)  #定义一个滑动平均的类。初始化时给定了衰减率为0.99和控制衰减率的变量step。 ema = tf.train.ExponentialMovingAverage(0.99, step) maintain\_averages\_op = ema.apply([v1]) **with** tf.Session() **as** sess:  init\_op = tf.global\_variables\_initializer()  sess.run(init\_op)  print(sess.run([v1, ema.average(v1)]))  sess.run(tf.assign(v1, 5))  sess.run(maintain\_averages\_op)  print(sess.run([v1, ema.average(v1)]))  sess.run(tf.assign(step, 10000))  sess.run(tf.assign(v1, 10))  sess.run(maintain\_averages\_op)  print(sess.run([v1, ema.average(v1)]))  sess.run(maintain\_averages\_op)  print(sess.run([v1, ema.average(v1)])) |

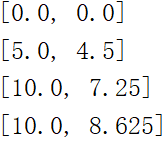
第一个测试使用的参数是0.99



第二个测试使用的参数是0.75



第三个测试使用的参数是0.50



由于代码都一样只是改变了上面代码传入的值，所以不再展示代码

由结果可知：衰减率越高，变化确实越小。详细代码见附录2

附录1：

|  |
| --- |
| **import** tensorflow **as** tf  TRAINING\_STEPS = 10 LEARNING\_RATE = 0.1 x = tf.Variable(tf.constant(5, dtype=tf.float32), name=**"x"**) y = tf.square(x) train\_op = tf.train.GradientDescentOptimizer(LEARNING\_RATE).minimize(y) **with** tf.Session() **as** sess:  sess.run(tf.global\_variables\_initializer())  **for** i **in** range(TRAINING\_STEPS):  sess.run(train\_op)  x\_value = sess.run(x)  print(**"After %s iteration(s): x%s is %f."** % (i + 1, i + 1, x\_value))  train\_op1 = tf.train.GradientDescentOptimizer(0.05).minimize(y) **with** tf.Session() **as** sess:  sess.run(tf.global\_variables\_initializer())  **for** i **in** range(100):  sess.run(train\_op1)  **if** i % 10 == 0:  x\_value = sess.run(x)  print(**"After %s iteration(s): x%s is %f."** % (i + 1, i + 1, x\_value))  *#使用指数学习率* global\_step = tf.Variable(0) LEARNING\_RATE1 = tf.train.exponential\_decay(0.2, global\_step, 1, 0.85, staircase=**True**) train\_op2 = tf.train.GradientDescentOptimizer(LEARNING\_RATE1).minimize(y, global\_step=global\_step) **with** tf.Session() **as** sess:  sess.run(tf.global\_variables\_initializer())  **for** i **in** range(70):  sess.run(train\_op2)  **if** i % 5 == 0:  LEARNING\_RATE\_value1 = sess.run(LEARNING\_RATE1)  x\_value = sess.run(x)  print(**"After %s iteration(s): x%s is %f, learning rate is %f."** % (  i + 1, i + 1, x\_value, LEARNING\_RATE\_value1)) |

附录2：

|  |
| --- |
| **import** tensorflow **as** tf  v1 = tf.Variable(0, dtype=tf.float32) step = tf.Variable(0, trainable=**False**) decey = [0.99,0.75,0.50] **for** num **in** decey:  ema = tf.train.ExponentialMovingAverage(num, step)  maintain\_averages\_op = ema.apply([v1])  **with** tf.Session() **as** sess:  init\_op = tf.global\_variables\_initializer()  sess.run(init\_op)  print(sess.run([v1, ema.average(v1)]))  sess.run(tf.assign(v1, 5))  sess.run(maintain\_averages\_op)  print(sess.run([v1, ema.average(v1)]))  sess.run(tf.assign(step, 10000))  sess.run(tf.assign(v1, 10))  sess.run(maintain\_averages\_op)  print(sess.run([v1, ema.average(v1)]))  sess.run(maintain\_averages\_op)  print(sess.run([v1, ema.average(v1)])) |