**作业5.6**

1. **实验要求**

试设计一个BP改进算法，能通过动态调整学习率显著提高收敛速度，编程实现该算法，并选择两个标准UCI数据集与标准BP算法进行实验比较。

1. **实验原理**

BP算法在训练结果精度较低的情况时，目标对参数的梯度下降更大，此时需要较大的学习率使得训练的速度更快。而当训练结果精度较高时，此时目标对参数的梯度下降变小，若此时学习率仍然很大，会导致训练很容易跳出全局最优，从而导致收敛速度下降，需要反复震荡才能达到最优。因此有必要使得训练精度较大时学习率减小，当然学习率减小的速度应该变缓，以防止无法收敛到全局最优。

1. **实验过程和代码**

这里我们设定的学习率变化如下：训练精度小于0.7时，学习率为1-训练精度，训练精度大于0.7时，学习率为0.45-0.2\*训练精度。过程如下：

1. 准备数据，设立标记。
2. 利用tensorflow搭建一个隐层，这里由于有八个属性，所以设置输入的w为8，自己设定中间隐层的权值数目为20，输出很显然为1。设定激活函数为sigmoid函数，优化目标为输出预测值与实际值差的平方和。
3. 设定精度计算函数，即1-误判数 /样本总数。
4. 设定学习率根据准确率变化而变化的学习率变化函数。
5. 利用tensorflow会话训练优化目标，代入学习率变化函数，得到预测值。
6. 将预测值和标记值代入到精度计算函数得到准确率，同时输出第一次准确率达到90%时的训练次数。

相关代码如下：

**import** tensorflow **as** tf  
**from** numpy **import** \*  
**import** matplotlib.pyplot **as** plt  
*#数据*data = mat([  
 [1, 1, 1, 1, 1, 1, 0.697, 0.460, 1],  
 [2, 1, 2, 1, 1, 1, 0.774, 0.376, 1],  
 [2, 1, 1, 1, 1, 1, 0.634, 0.264, 1],  
 [1, 1, 2, 1, 1, 1, 0.608, 0.318, 1],  
 [3, 1, 1, 1, 1, 1, 0.556, 0.215, 1],  
 [1, 2, 1, 1, 2, 2, 0.403, 0.237, 1],  
 [2, 2, 1, 2, 2, 2, 0.481, 0.149, 1],  
 [2, 2, 1, 1, 2, 1, 0.437, 0.211, 1],  
 [2, 2, 2, 2, 2, 1, 0.666, 0.091, 0],  
 [1, 3, 3, 1, 3, 2, 0.243, 0.267, 0],  
 [3, 3, 3, 3, 3, 1, 0.245, 0.057, 0],  
 [3, 1, 1, 3, 3, 2, 0.343, 0.099, 0],  
 [1, 2, 1, 2, 1, 1, 0.639, 0.161, 0],  
 [3, 2, 2, 2, 1, 1, 0.657, 0.198, 0],  
 [2, 2, 1, 1, 2, 2, 0.360, 0.370, 0],  
 [3, 1, 1, 3, 3, 1, 0.593, 0.042, 0],  
 [1, 1, 2, 2, 2, 1, 0.719, 0.103, 0]])  
x\_data=data[:,0:8].astype(float32)  
y\_data=data[:,8].astype(float32)  
*#搭建网络*x=tf.placeholder(tf.float32,[None,8])  
y=tf.placeholder(tf.float32,[None,1])  
**def** addlayer(data,insize,outsize,active=None):  
 w=tf.Variable(tf.random\_normal([insize,outsize]))  
 b=tf.Variable(tf.zeros([1,outsize])+0.1)  
 w\_b=tf.matmul(data,w)+b  
 **return** active(w\_b)  
layer=addlayer(x,8,20,tf.nn.sigmoid)  
output=addlayer(layer,20,1,tf.nn.sigmoid)  
loss=tf.reduce\_mean(tf.reduce\_sum(tf.square(y-output),reduction\_indices=[1]))  
init=tf.initialize\_all\_variables()  
*#定义精度***def** accuracy(l1,l2):  
 **for** i **in** range(len(l2)):  
 **if** l2[i] > 0.5:  
 l2[i] = 1  
 **else**:  
 l2[i] = 0  
 s = abs(l1 - l2)  
 **return** 1-sum(s) / 17  
*#学习率根据精度动态设计***def** train(l1,l2):  
 **if**(accuracy(l1,l2)<0.7):  
 rate=1-accuracy(l1,l2)  
 **else**:  
 rate=0.45-0.2\*accuracy(l1,l2)  
 **return** tf.train.GradientDescentOptimizer(rate).minimize(loss)  
*#训练网络***with** tf.Session() **as** sess:  
 sess.run(init)  
 **for** i **in** range(2000):  
 l1 = array(y\_data).flatten()  
 l2 = sess.run(output, feed\_dict={x: x\_data, y: y\_data}).flatten()  
 sess.run(train(l1,l2),feed\_dict={x:x\_data[i%17],y:y\_data[i%17]})  
 *#print(array(y\_data).flatten(),sess.run(output,feed\_dict={x:x\_data,y:y\_data}).flatten())* **if** i%10==0:  
 **print**(accuracy(l1,l2))  
 **if**(accuracy(l1,l2)>0.9):  
 **print**(i)  
 **break**

1. **实验结果**

这里得出第一次得到准确度为0.9的训练次数为434次，而在5.5题中未变的学习率达到同样的准确度则需要823次，由此可见本人设计的学习率动态变化算法是可以改善训练速度的。