

**机器学习作业**

**题 目: 基于BP算法的三层前向神经网络**

**学 号: 919106840333**

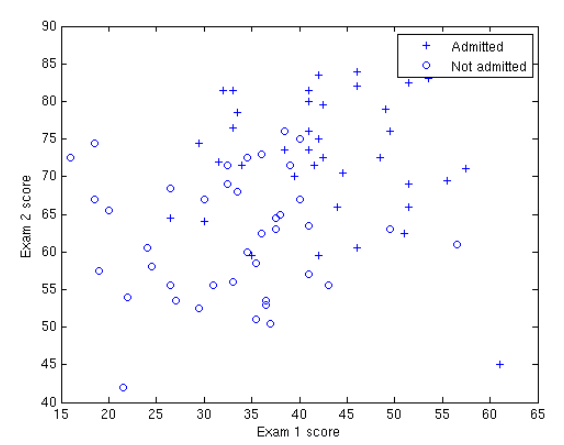
**姓 名: 孙傲歆**

**指导教师: 潘金山**

**2021年12月**

**一、题目要求**

• 给出下列训练数据:



• 使用BP算法实现三层前向神经网络 (自己编码，不要使用Tensorflow/Pytorch等框架)，并对结果进行5倍交叉验证；

• 将其与Logistic回归和Softmax回归进行比较。

**二、算法思路**

**1.** **BP算法的三层前向神经网络:**

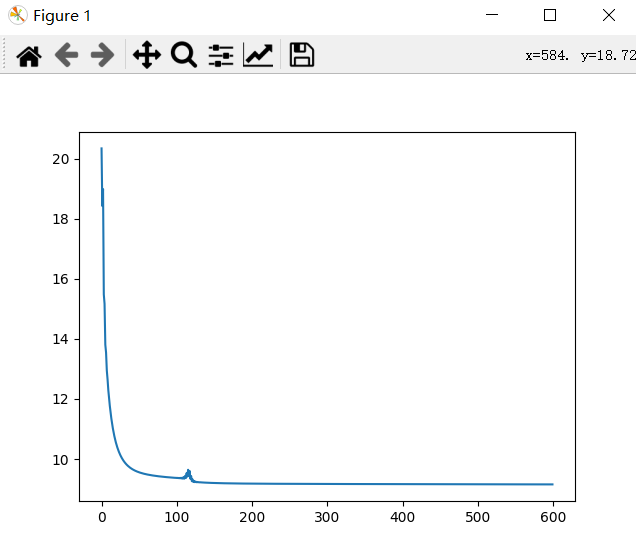
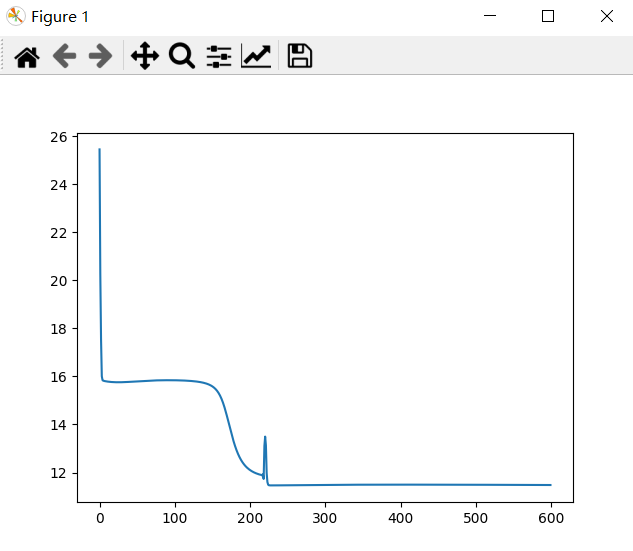
由于三层前向神经网络模型已经不是线性模型，所以我们不能像之前的logistics回归和softmax回归那样给出最终结果直线。但我们仍可以计算每一次的准确率来评估模型。

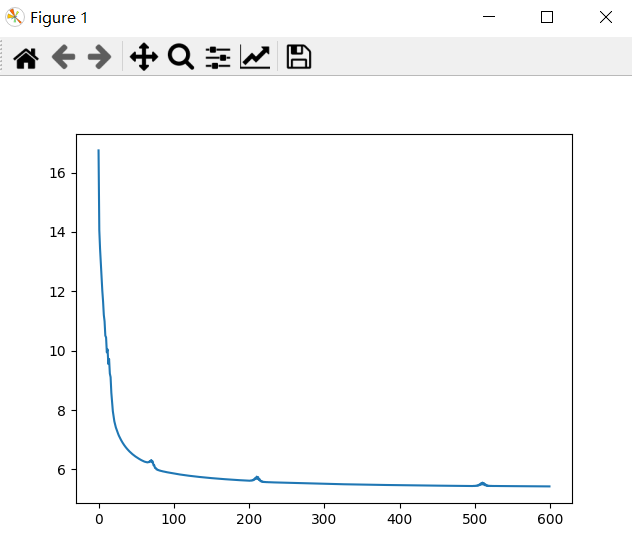
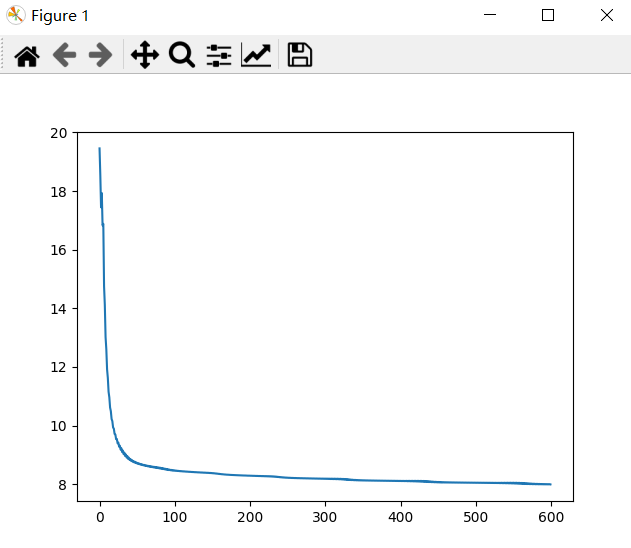
而5倍交叉验证法就是将整个数据集分为不相交的5个部分，然后选取其中的一个部分为测试集，其他部分为训练集，获得测试准确率，然后重复上述步骤，直到每个部分都充当过一次测试集，我们以平均准确率作为模型的最终准确率。本数据集一共有80组数据，也就是说我们每次取其中的16组作为测试集，其余数据作为训练集。

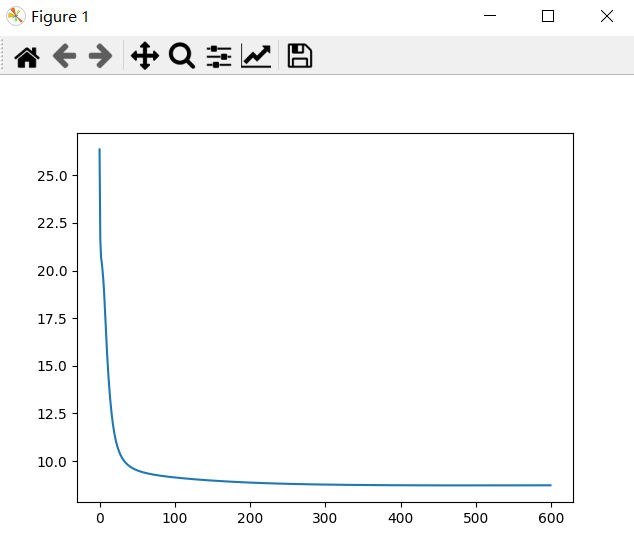
之后，便是代入神经网络的梯度下降公式进行计算，这里我们选择SGD，设置学习率为0.05，步数为600，调用python的numpy库中的random.normal函数，每次从正态分布曲线中随机抽取样本。对于每次迭代，绘制出其损失函数的变化，并打印准确率。

**三、运行结果**

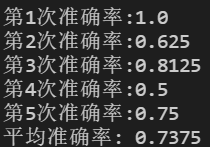
**1.** **三层前向神经网络:**







**图1.1**



**图1.2**

上图为代码运行结果。**图1.1**是五次迭代的损失函数变化图，我们可以看到经过600次梯度下降计算，其损失函数最终都趋于平缓，可以认为结果收敛。**图1.2**是每次迭代的准确率，以及最终的平均准确率。我们可以看到，除了第二次、第四次迭代的准确率较低，其他几次的迭代率都很高。最终平均准确率为0.7375。

**2.与logistics回归和softmax回归的比较:**

从上述结果，我们可以看到，神经网络算法的平均准确率较高，可以达到0.8左右，但是在进行交叉验证时，及个别情况下其准确率十分之低。而且我们可以看到损失函数按照梯度下降时，有时会发生严重的震荡。而logistics回归和softmax回归梯度下降则显得更为平缓。

而且神经网络是非线性模型，无法给出最终数据的分界线。而logistics回归和softmax回归是线性模型，可以给出最终的数据分界线。

**四、可运行代码**

**1.homework\_5.py：**

#神经网络算法

import numpy as np

import matplotlib.pyplot as plt

def sigmods(x):

return 1/(np.exp(-x)+1)

data\_x = np.loadtxt("ex4Data/ex4x.dat")

data\_y = np.loadtxt("ex4Data/ex4y.dat")

mean = data\_x.mean(axis=0)

variance = data\_x.std(axis=0)

data\_x = (data\_x-mean)/variance

data\_y = data\_y.reshape(-1, 1)

temp = np.ones(data\_y.size)

data\_x = np.c\_[data\_x, temp]

y = np.zeros([data\_y.size, 2])

for i in range(data\_y.size):

    if data\_y[i] == 0:

        y[i][0] = 1

    if data\_y[i] == 1:

        y[i][1] = 1

learn\_rate = 0.05

data\_sets = list()

for i in range(data\_y.size):

    if i % 8 == 0 and i != 0:

        data\_sets.append(data\_x[i-8:i, :])

    if i == data\_y.size-1:

        data\_sets.append(data\_x[i-7:i+1, :])

positive\_x = data\_x[0:40, :]

positive\_label = y[0:40, :]

negative\_x = data\_x[40:80, :]

negative\_label = y[40:80, :]

total = 0

for j in range(5):

    x\_ = np.r\_[np.delete(positive\_x, range(j\*8, (j+1)\*8), 0), np.delete(negative\_x, range(j\*8, (j+1)\*8), 0)]

    y\_ = np.r\_[np.delete(positive\_label, range(j\*8, (j+1)\*8), 0), np.delete(negative\_label, range(j\*8, (j+1)\*8), 0)]

    test\_x = np.r\_[data\_sets[j], data\_sets[j+5]]

    test\_label = np.r\_[data\_y[j\*8:(j+1)\*8], data\_y[(j+5)\*8:(j+6)\*8]]

    data\_x = np.mat(x\_)

    temp = np.ones(data\_y.size-16)

    weight\_input = np.mat(np.random.normal(size=(data\_x.shape[1], 7)))

    weight\_hidden = np.mat(np.random.normal(size=(8,2)))

    steps = 600

    loss\_values = list()

    for i in range(steps):

        hidden\_input = data\_x\*weight\_input

        hidden\_out = sigmods(hidden\_input)

        hidden\_out\_ = np.c\_[hidden\_out, temp]

        output\_input = hidden\_out\_\*weight\_hidden

        output = sigmods(output\_input)

        loss = 0.5\*np.sum(np.multiply(output-y\_,  output-y\_))

        loss\_values.append(loss)

        output\_error = np.multiply(np.multiply(output-y\_, output), 1-output)

        dew\_hidden = hidden\_out\_.T\*output\_error

        output\_error\_ = dew\_hidden[7]

        weight\_hidden\_ = np.delete(weight\_hidden, 7, axis=0)

        hidden\_error = np.multiply(np.multiply(output\_error\_\*weight\_hidden\_.T, hidden\_out), 1-hidden\_out)

        dew\_input = data\_x.T\*hidden\_error

        weight\_hidden = weight\_hidden-learn\_rate\*dew\_hidden

        weight\_input = weight\_input-learn\_rate\*dew\_input

    plt.plot(loss\_values)

    plt.show()

    temp = np.ones(test\_label.size)

    hidden\_input=test\_x\*weight\_input

    hidden\_out=sigmods(hidden\_input)

    hidden\_out\_ = np.c\_[hidden\_out, temp]

    output\_input = hidden\_out\_\*weight\_hidden

    output =sigmods(output\_input)

    count = 0

    output = np.array(output)

    for i in range(test\_label.size):

        outs = output[i].ravel()

        outs = outs.tolist()

        if int(test\_label[i]) == outs.index(max(outs)):

            count = count+1

    print("第%d次准确率:%s" %(j+1,count/test\_label.size))

    total = total+count/test\_label.size

print("平均准确率:", total/5)