机器学习课题报告一

计算机1601

车兴竹

1611640130

1. **实验内容**

基于鸢尾花数据集（Iris），使用Python实现对数几率回归模型

1. **问题描述**

用python编程，将数据集分为测试集和训练集，实现对数几率回归模型，用训练集对模型进行训练，用测试集验证模型的效能。

以不同比例：50%、70%、90%作训练集，检验模型在测试集上的分类正确率。

1. **数据集描述**

Iris数据集共有150个数据项，包含3个类，每个类有50个实例，每一类代表一种鸢尾花：Iris-setosa、Iris-versicolor，Iris-virginica。

每个实例有四个特征值：

Sepal.Length（花萼长度），单位cm

Sepal.Width （花萼宽度），单位cm

Petal.Length（花瓣长度），单位cm

Petal.Width （花瓣宽度），单位cm

本次课题实验中只挑选了Iris-setosa、Iris-versicolor两种鸢尾花数据，共100个数据。

1. **实验结果图**

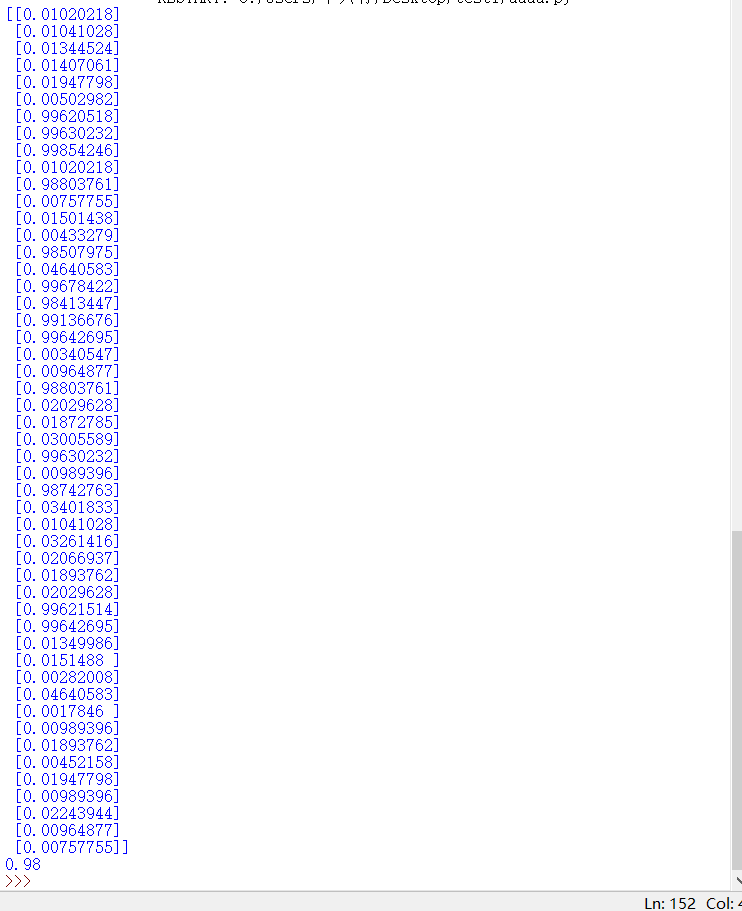
步长：alpha=0.001

迭代次数：maxCycle=1000

输出：y的测试值：y\_test 正确率：correct

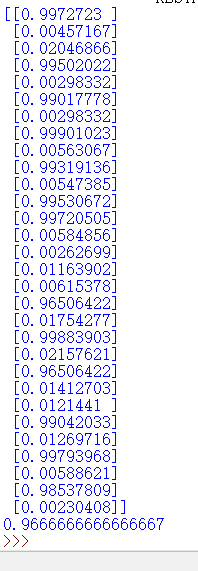
50%：

正确率 0.98



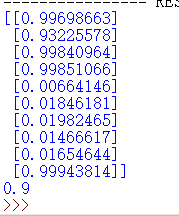
70%：

正确率：0.96



90%：

正确率：0.9



1. **实验结果分析**

更改迭代次数或者步长，正确率并没有改变，说明程序应该还是存在问题。

以不同的比例划分训练集和测试集会造成不同的结果，说明训练集的数据量对结果有很大影响，合理的划分训练集和测试集的比例，能够提高结果的准确性。

1. **源代码**

import numpy as np

import random

from numpy import genfromtxt,zeros

data=genfromtxt('iris.data.txt',delimiter=',',usecols=(0,1,2,3))

target=genfromtxt('iris.data.txt',delimiter=',',usecols=(4),dtype=str)

t=zeros(len(target))

t[target=='Iris-setosa']=0

t[target=='Iris-versicolor']=1

#将数据集按比例随机分为训练集和测试集

def trainTestSplit(trainingSet,trainingLabels,test\_size):

totalNum=int(len(trainingSet))

trainIndex=list(range(totalNum))#存放训练集的下标

testIndex=[] #测试集的下标

x\_test=[] #测试集输入

y\_test=[] #测试集输出

x\_train=[] #训练集输入

y\_train=[] #训练集输出

trainNum = int(totalNum \* test\_size) #划分训练集的样本数

for i in range(trainNum):

randomIndex = int(random.uniform(0,len(trainIndex)))

x\_test.append(trainingSet[randomIndex])

y\_test.append(trainingLabels[randomIndex])

del(trainIndex[randomIndex]) #删除已经放入测试集的下标

for i in range(totalNum-trainNum):

x\_train.append(trainingSet[trainIndex[i]])

y\_train.append(trainingLabels[trainIndex[i]])

return x\_train,y\_train,x\_test,y\_test

#比例

all\_data=trainTestSplit(data,t,0.3)

x\_train=all\_data[0]

y\_train=all\_data[1]

x\_test=all\_data[2]

y\_test=all\_data[3]

#logistic函数

def logistic(x):

return 1.0/(1+np.exp(-x))

#模型

def model(X\_train,w):

return logistic(X\_train\*w)

###梯度下降，求w

def gradAscent(dataIn,Target):

X=np.mat(dataIn)

Y=np.mat(Target).transpose()

m,n=np.shape(X) #矩阵的行列

alpha=0.001 #步长

maxCycle=1000 #迭代次数

weights=np.ones((n,1)) #n\*1的数组

for k in range(maxCycle):

y\_hat=model(X,weights)

#print('y\_hat: ',y\_hat)

error=(y\_hat-Y)

weights=weights-alpha\*X.transpose()\*error

#print(y\_hat)

return weights

W=gradAscent(x\_train,y\_train)

#print (W)

y\_test\_hat=model(x\_test,W)

print (y\_test\_hat)

#正确率

def correct(y\_Test\_hat,y\_Test):

y\_Test\_hat[y\_Test\_hat<=0.5]=0

y\_Test\_hat[y\_Test\_hat>=0.5]=1

n=len(y\_Test\_hat)

y=y\_Test\_hat.reshape((-1,n))

#print (y)

result\_error=y-y\_Test

#print (result\_error)

wrong\_num=result\_error[result\_error!=0]

error\_sum=len(wrong\_num)

correct=(n-error\_sum)/n

print (correct)

return correct

correct(y\_test\_hat,y\_test)