课题报告

1. 报告题目：

使用Python实现对数几率回归模型

1. 问题描述：

安装pycharm，下载鸢尾花数据集iris，读懂数据集的相关说明，了解数据格式。

编程实现对数几率回归模型，并对 Iris 数据集进行分类以验证模型的效能：

(1). 将数据集的 50%作为训练集，50%作为测试集，检验模型在测试集上的分类正确率 (2). 将数据集的 70%作为训练集，30%作为测试集，检验模型在测试集上的分类正确率 (3). 将数据集的 90%作为训练集，10%作为测试集，检验模型在测试集上的分类正确率

1. 数据集描述：

鸢尾花数据集包含有Setosa，Verscicour和Virginica三个种类，这三个类别标签就是我们想要预测的量（此次实验选择其中的两种鸢尾花进行实验）。其中每个种类包含有50个数据，每个数据包含4种变量（或者称为特征），这四种变量可以看作是要分析的自变量，分别是花瓣长度，花瓣宽度，花萼长度，花萼宽度（选取其中的两个特征做坐标图）。

iris以鸢尾花的特征作为数据来源，常用在分类操作中。该数据集由3种不同类型的鸢尾花的50个样本数据构成。其中的一个种类与另外两个种类是线性可分离的，后两个种类是非线性可分离的。

该数据集包含了5个属性：

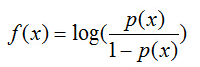
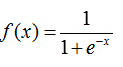
& Sepal.Length（花萼长度），单位是cm;

& Sepal.Width（花萼宽度），单位是cm;

& Petal.Length（花瓣长度），单位是cm;

& Petal.Width（花瓣宽度），单位是cm;

& 种类：Iris Setosa（山鸢尾）、Iris Versicolour（杂色鸢尾），以及Iris Virginica（维吉尼亚鸢尾）。

1. 实验步骤及代码：
2. 从本地导入iris数据集，选择要用的数据，存放在矩阵data里面，观察数据，自变量包含（花萼长度，花瓣长度）两个变量，因变量为山鸢尾和杂色鸢尾，在Python中用Numpy读取数据并采用matplotlib库可视化数据。
3. 引入对数，解出p(x),将其带入到sigmoid函数 中,得到需要的sigmoid函数，和线性回归一样，拟合出所以的参数。
4. 定步长批量梯度下降法
5. 编写predict()函数进行预测（两个数组分别变为两个个函数，采用sigmoind放到predict()函数中）
6. 训练和预测

采用model\_selection.train\_test\_split()实现训练集和测试集的划分，然后调用predict()函数，拟合出逻辑回归模型，然后在测试集上评估模型。

1. 代码：

#!/usr/bin/python

# -\*- coding: UTF-8 -\*-

import numpy as np

import matplotlib.pyplot as plt

from sklearn import model\_selection

######################################数据导入########################

data\_t=[]

data\_f=np.loadtxt('D:/Python/iris.data',delimiter=",",usecols=(0,1,2,3),dtype=float)

data\_s=np.loadtxt('D/Pychon /iris.data',delimiter=",",usecols=4,dtype=str)

for i in range(0, len(data\_s)):

if data\_s[i] =='Iris-setosa':

data\_t.append(1)

else:

data\_t.append(0)

data=np.insert(data\_f,4,values=data\_t,axis=1)

x=data[:100,1:3]

y=data[:100,4]

m,n=np.shape(x)

####################################数据显示################

seto=data[:50]

vers=data[51:100]

f1=plt.figure(1)

plt.title('Iris')

plt.xlabel('Width')

plt.ylabel('Length')

plt.scatter(seto[:,0],seto[:,2],marker='o',color='r',s=100,label='seto')

plt.scatter(vers[:,0],vers[:,2],marker='o',color='k',s=100,label='vers')

plt.legend(loc='upper right')

plt.show()

############### ##Sigmoid函数############################

def sigmoid(x,beta): #课本3.18

return 1.0/(1+np.math.exp(-np.dot(beta,x)))

####################梯度下降法####################

def gradDscent\_1(X, y): #

h = 0.1 # 步长

max\_times = 500 # 迭代的最大次数

m, n = np.shape(X)

#显示参数的收敛曲线

beta = np.zeros(n) # 初始化

delta\_beta = np.ones(n) \* h

llh = 0

llh\_temp = 0

for i in range(max\_times):

beta\_temp = beta

for j in range(n):

# 偏导数

beta[j] += delta\_beta[j]

llh\_tmp = likelihood(X, y, beta)

delta\_beta[j] = -h \* (llh\_tmp - llh) / delta\_beta[j]

beta = beta\_temp + delta\_beta

llh = likelihood(X, y, beta)

return beta

################预测函数#####################

def predict (x,theta):

m,n=np.shape(x)

xTest=np.ones((m,n+1))

xTest[:,:-1]=x

yp=np.dot(xTest,theta)

return yp

#################损失函数##############

def runExpe(x,y,theta):

costs=Gradientdescent(x,y,theta)

fig, ax = plt.subplots(figsize=(12, 4))

ax.plot(np.arange(len(costs)), costs,'r')

plt.show()

return costs

#############划分训练集与测试集###########

m,n=np.shape(x)

theta = np.zeros(n)

costs= np.zeros(n)

np.ones(n)

#x\_hat=np.c\_[x,np.ones(m)]

x\_train,x\_test,y\_train,y\_test=model\_selection.train\_test\_split(x,y,test\_size=0.5,random\_state=0)

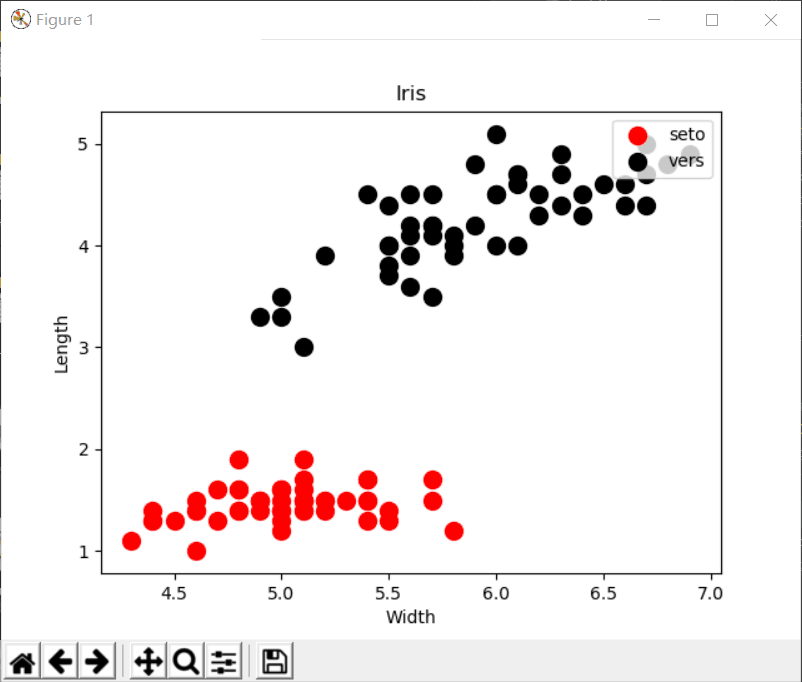
theat=Gradientdescent(x\_test,y\_test,theta)

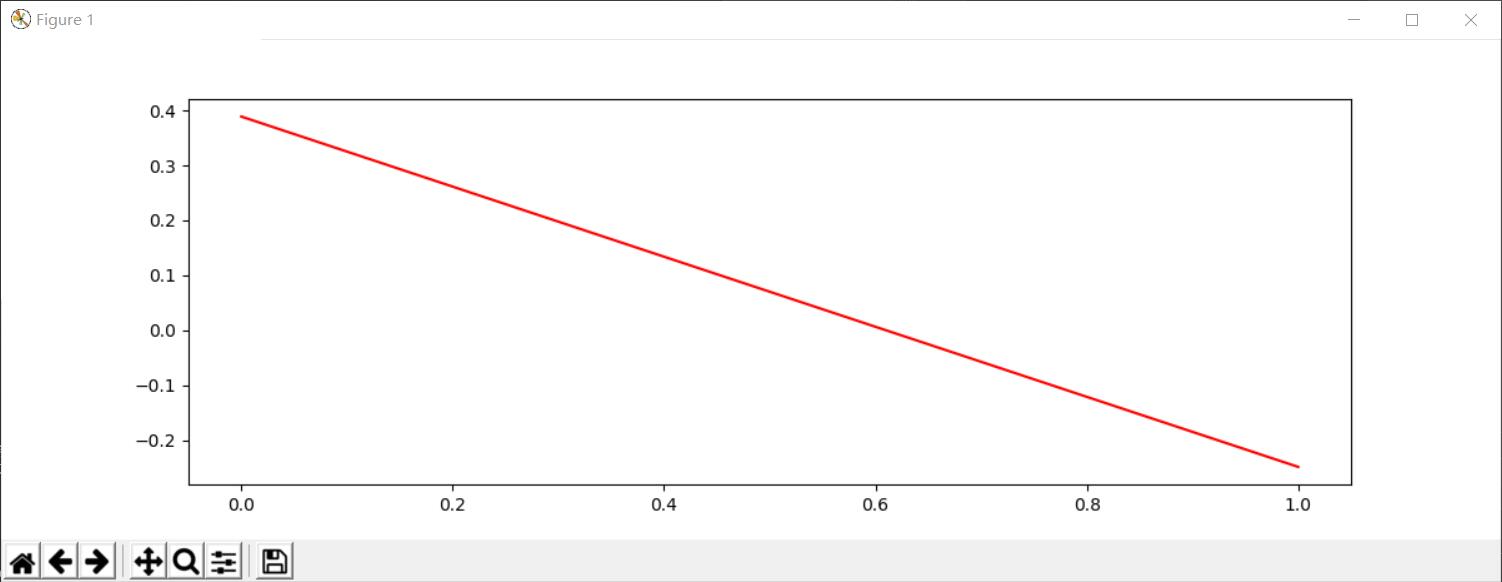
costs=runExpe(x,y,theat)

y\_p=predict(x\_test,theat)

print(y\_p)

1. 实验截屏：





1. 总结分析：

虽然线性回归的算法实现简单，计算简单，但是在做的过程中困难重重，刚开始几乎无从下手，不知道这一大堆的数学公式到底怎么用，经过查阅网上各种实现的方法，对比各种梯度下降法，才有了一点眉头，本次实验作业做的很慢，结果也不尽人意，需要更认真的学习哦，多做练习。