

**计算机与信息 学院实验报告**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 实验课程： | Python机器学习实验 | | | | |
| 实验编号： | 实验一 | | | | |
| 实验名称： | **LR回归模型二分问题** | | | | |
| 实验人员： | 学号 | 18111207248 | | | |
| 姓名 | 吴钰 | | | |
| 班级 | 创新班 | | | |
| 实验日期： | 2021.3.22 | | | | |
| 实验室： | 2060301 | | | | |
|  |  | | | | |
| 实验评价： |  | | | | |
| 实验成绩： | |  | 评价日期： |  |
|  | 指导教师： | |  | | |

# 实验目的

掌握机器学习的几种模型

编程实现LR回归模型的二分类问题上的应用

# 实验要求

请编程实现LR回归模型的二分类问题上的应用，提交开题报告、训练集和测试集

# 实验内容

请编程实现LR回归模型的二分类问题上的应用，提交开题报告、训练集和测试集

# 实验过程及结果

代码：

# -\*- coding: utf-8 -\*-

"""

Created on Tue Mar 23 11:23:38 2021

@author: wuyu

"""

# 使用 Logistic 回归在简单数据集上的分类

import numpy as np

import matplotlib.pyplot as plt

def load\_data\_set(file\_path):

"""

加载dataset文件夹中的数据集

:param file\_path: 文件目录

:return:

"""

data\_matrix = []

label\_list = []

with open(file\_path, "r") as file:

for line in file:

line = line.strip().split()

data\_matrix.append([float(line[0]), float(line[1]), 1.0])

label\_list.append(int(line[2]))

return np.array(data\_matrix), np.array(label\_list)

def sigmoid(values):

"""

sigmoid函数

:param values: 列表类型 分别计算出值封装成列表返回

:return: 1.0 / (1.0 + np.exp(-values))

"""

return 1.0 / (1.0 + np.exp(-values))

def gradAscent(data\_matrix, label\_matrix, alpha=0.001, n=500):

"""

开始使用梯度上升训练weights

:param data\_matrix: np.array()类型 数据集矩阵

:param label\_matrix: np.array()类型 标签矩阵

:param alpha: 学习率

:param n: 训练次数

:return: weights

"""

# label\_matrix转为列向量

label\_matrix = np.mat(label\_matrix).transpose()

# 初始化weights = [3,1] = 1

weights = np.ones(shape=(len(data\_matrix[0]), 1))

for i in range(n):

# [n,1] = [n,m] \* [m,1] 矩阵乘法

# 要把data\_matrix从np.array 转为np.mat才能相乘

values = np.mat(data\_matrix) \* weights

# 激活函数后得到类别

activated\_values = sigmoid(values)

# 计算误差 (以2元函数举例)

error\_list = label\_matrix - activated\_values

print(activated\_values,error\_list)

# 调整权重weights

# 因为偏移量error\_list是(5,1)所以要把data\_matrix转为(3,5)，最终得到每个权重需要调整的量

weights = weights + alpha \* data\_matrix.transpose() \* error\_list

return np.array(weights)

def paint(data\_matrix, label\_matrix, weights):

"""

画出数据集和拟合直线

:param data\_matrix:

:param label\_matrix:

:param weights:

:return:

"""

point\_0 = []

point\_1 = []

plt.figure(figsize=(5, 5), dpi=200)

# 遍历把两类点分离出来

for i in range(len(label\_matrix)):

if label\_matrix[i] == 1:

point\_1.append(data\_matrix[i])

else:

point\_0.append(data\_matrix[i])

point\_0 = np.array(point\_0)

point\_1 = np.array(point\_1)

# 把离散点画出来

plt.scatter(x=point\_0[:, 0], y=point\_0[:, 1], c="red")

plt.scatter(x=point\_1[:, 0], y=point\_1[:, 1], c="blue")

# 画拟合曲线

x = np.arange(-3.0, 8.0, 0.1)#######

y = (-weights[2] - weights[0] \* x) / weights[1]

plt.plot(x, y)

plt.show()

from sklearn.model\_selection import train\_test\_split

# 1' 加载数据集

data\_matrix, label\_matrix = load\_data\_set("D:\\大三（下）专业课\\机器学习实验\\实验一\\机器学习\\逻辑回归数据\\data.txt")

data\_matrix\_train,data\_matrix\_test,\

label\_matrix\_train,label\_matrix\_test=\

train\_test\_split(data\_matrix,label\_matrix,

test\_size=0.2,random\_state=42)

# 2' 梯度上升训练weights参数

weights = gradAscent(data\_matrix\_train, label\_matrix\_train, n=5000)

# print(weights)

# 3' 把数据集和拟合直线画出来

paint(data\_matrix\_test, label\_matrix\_test, weights)

print('训练集数据的形状为：',data\_matrix\_train.shape)

print('训练集标签的形状为：',label\_matrix\_train.shape)

print('测试集数据的形状为：',data\_matrix\_test.shape)

print('测试集标签的形状为：',label\_matrix\_test.shape)

print('训练集数据为\n')

print(data\_matrix\_train)

print('测试集数据为\n')

print(data\_matrix\_test)

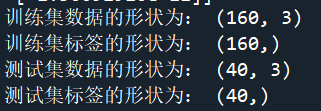
with open('D:\大三（下）专业课\机器学习实验\实验一\data\_test.txt', 'w') as f: # 打开test.txt 如果文件不存在，创建该文件。

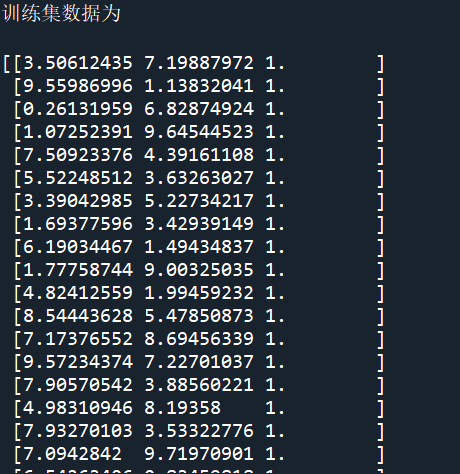
f.write(str(data\_matrix\_test)) # 把变量var写入test.txt。这里var必须是str格式，如果不是，则可以转一下。

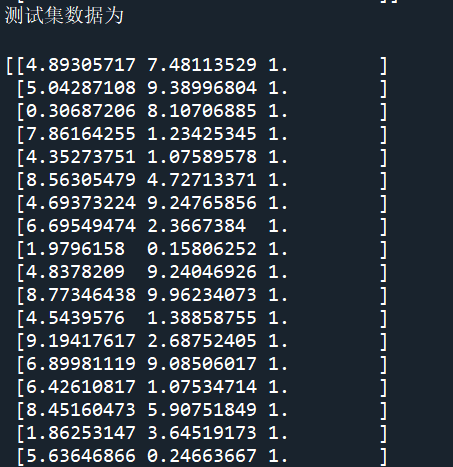
with open('D:\大三（下）专业课\机器学习实验\实验一\data\_train.txt', 'w') as f: # 打开test.txt 如果文件不存在，创建该文件。

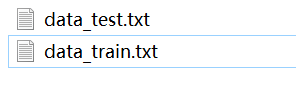
f.write(str(data\_matrix\_train)) # 把变量var写入test.txt。这里var必须是str格式，如果不是，则可以转一下。

结果截图：

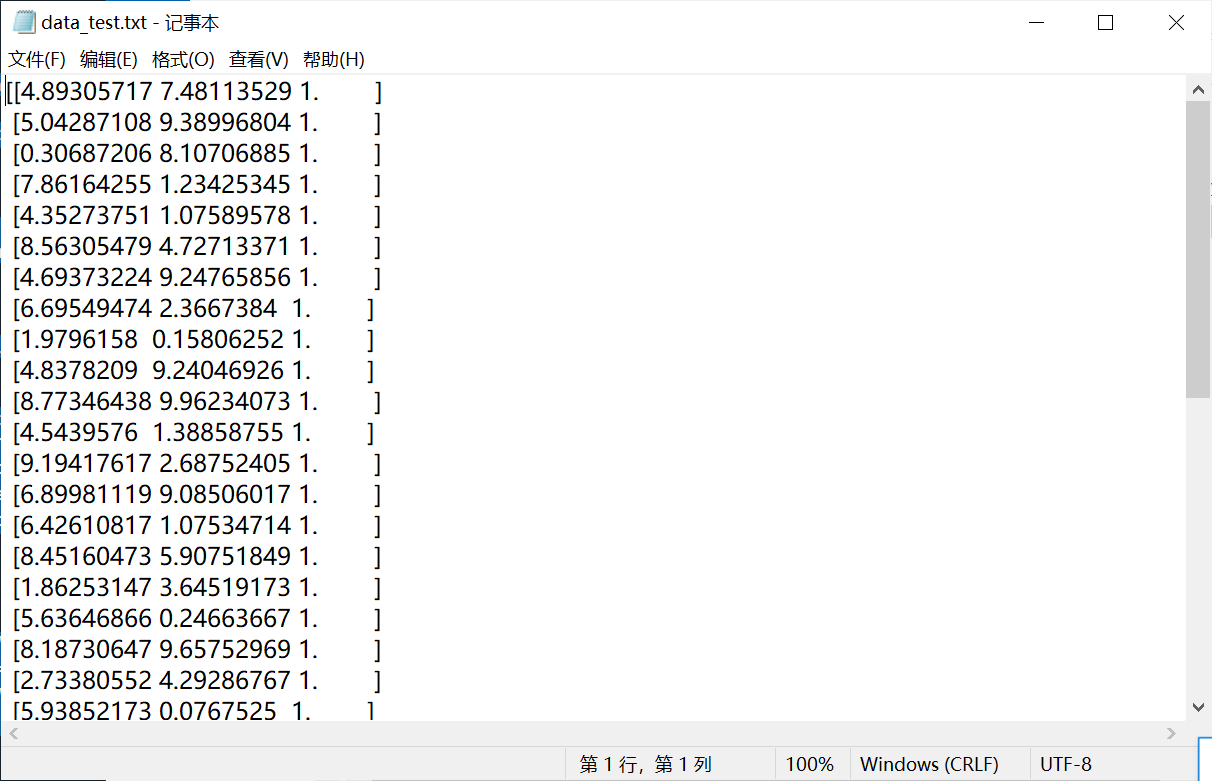












数据集拟合直线图：

