**实验四：逻辑回归二分类算法**

**一、实验目的及要求**

1. 掌握对基本机器学习概念的理解

2. 掌握分类和逻辑回归二分类实现的原理和方法

3. 按照既定格式书写实验报告

**二、实验设备与平台**

1. 实验设备：计算机；

2. 平台：Windows 10操作系统

**三、实验内容**

**题目**：学习sigmoid函数和逻辑回归算法。将实验三.2中的样例数据用聚类的结果打标签{0，1}，并用逻辑回归模型拟合。

1. 学习并画出sigmoid函数
2. 设计梯度下降算法，实现逻辑回归模型的学习过程。
3. 根据给定数据（实验三.2），用梯度下降算法进行数据拟合，并用学习好的模型对(2,6)分类。

**实验实施**：

（在此详述平台，技术栈，思路，处理逻辑等等）

平台：基于win10系统的pycharm

# 该实验参考了网上的博客————地址：https://blog.csdn.net/qq\_21201267/article/details/104288855

#采取的是调库实现

'''

遇到不熟悉的库、模块、类、函数，可以依次：

1）百度（google确实靠谱一些），如"matplotlib.pyplot"，会有不错的博客供学习参考

2）"终端-->python-->import xx-->help(xx.yy)"，一开始的时候这么做没啥用，但作为资深工程师是必备技能

3）试着修改一些参数，观察其输出的变化，在后面的程序中，会不断的演示这种办法

'''

# written by hitskyer

# modified by Michael Ming on 2020.2.12

import sys

import numpy as np

import matplotlib.pyplot as plt

# sklearn 中文文档 https://sklearn.apachecn.org/docs/0.21.3/

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

from sklearn.pipeline import Pipeline

from sklearn.preprocessing import PolynomialFeatures

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LogisticRegression

from sklearn import metrics

def get\_parabolic\_curve\_data\_set():

# 设置随机数的种子，以保证每回运行程序的随机结果一致

# np.random.seed(520) # 520 可以随便写 Seed must be between 0 and 2\*\*32 - 1

# 随机生成200个样本，每个样本两维特征

# X = np.random.normal(0, 1, size=(n, 2)) # 正态分布，中心0，标准差1

# 更改X的分布（正态，均匀），看看结果有什么变化

# X = np.random.uniform(-4, 4, size=(n, 2)) # 均匀分布，区间[-4,4)

test\_x=[3.45,1.76,4.29,3.35,3.17,3.68,2.11,2.58,3.45,6.17,4.20,5.87,5.47,5.97,6.24,6.89,5.38,5.13,7.26,6.32]

test\_y=[7.08,7.24,9.55,6.65,6.41,5.99,4.08,7.10,7.88,5.40,6.46,3.87,2.21,3.62,3.06,2.41,2.32,2.73,4.19,3.62]

x=[]

for i in range(20):

x.append([test\_x[i],test\_y[i]])

x=np.array(x)

# 分类面（线）是y=-x^2+1.5，开口向下的抛物线，口内为1类，口外为0类

# y = np.array(X[:, 0] \*\* 2 + X[:, 1] < 1.5, dtype=int) # 满足关系的为1，否则为0

# 加入10%的噪声数据

# for \_ in range(n // 10): # //为整除

# y[np.random.randint(n)] = 1

return x

def show\_data\_set(X):

plt.scatter(X[:,0], X[:,1], c='r')

# 散点图，分量1,为y==0的行的0列，分量2，y==0的行的1列，c表示颜色

# plt.scatter(X[y == 1, 0], X[y == 1, 1], c='b')

plt.show()

def PolynomialLogisticRegression(degree=2, C=1.0, penalty='l2'):

# 对输入特征依次做 多项式转换、归一化转换、类别预测，可以尝试注释掉前2个操作，看看结果有什么不同

return Pipeline([

# Pipeline 可以把多个评估器链接成一个。例如特征选择、标准化和分类

# 以多项式的方式对原始特征做转换，degree次多项式

('poly', PolynomialFeatures(degree=degree)),

# 对多项式转换后的特征向量做归一化处理，例如（数据-均值）/标准差

('std\_scaler', StandardScaler()),

# 用转换后的特征向量做预测，penalty是正则化约束，C正则化强度，值越小，强度大

# solver 不同的求解器擅长的规模类型差异

# 正则化 https://blog.csdn.net/zouxy09/article/details/24971995/

('log\_reg', LogisticRegression(C=C, penalty=penalty, solver="liblinear", max\_iter=10000))

])

def plot\_decision\_boundary(x\_min, x\_max, y\_min, y\_max, pred\_func):

h = 0.01

# 产生网格

xx, yy = np.meshgrid(np.arange(x\_min, x\_max, h), np.arange(y\_min, y\_max, h))

# ravel将矩阵展平，np\_c[a,b]将a,b按列拼在一起

Z = pred\_func(np.c\_[xx.ravel(), yy.ravel()])

Z = Z.reshape(xx.shape)

plt.contourf(xx, yy, Z, cmap=plt.cm.Spectral) # 填充等高线

# 等高线参考 https://blog.csdn.net/lens\_\_\_/article/details/83960810

def test(X\_train, X\_test, y\_train, y\_test, degree=2, C=1.0, penalty='l2'):

poly\_log\_reg = PolynomialLogisticRegression(degree=degree, C=C, penalty=penalty)

# 训练模型

poly\_log\_reg.fit(X\_train, y\_train)

# 在训练数据上做测试

predict\_train = poly\_log\_reg.predict(X\_train)

sys.stdout.write("LR(degree = %d, C=%.2f, penalty=%s) Train Accuracy : %.4g\n" % (

degree, C, penalty, metrics.accuracy\_score(y\_train, predict\_train)))

# 在测试数据上做测试

predict\_test = poly\_log\_reg.predict(X\_test)

score = metrics.accuracy\_score(y\_test, predict\_test)

sys.stdout.write("LR(degree = %d, C=%.2f, penalty=%s) Test Accuracy : %.4g\n" % (

degree, C, penalty, score))

print("--------------------------------------")

# 展示分类边界

plot\_decision\_boundary(0, 10, 0, 10, lambda x: poly\_log\_reg.predict(x))

plt.scatter(X\_train[y\_train == 0, 0], X\_train[y\_train == 0, 1], color='r')

plt.scatter(X\_train[y\_train == 1, 0], X\_train[y\_train == 1, 1], color='b')

plt.xlabel("x1")

plt.ylabel("x2")

plt.rcParams['font.sans-serif'] = 'SimHei' # 消除中文乱码

plt.rcParams['axes.unicode\_minus'] = False # 用来正常显示负号

plt.title("参数：degree:%d, C:%.2f, penalty:%s -- 准确率: %.4f" % (degree, C, penalty, score))

plt.show()

if \_\_name\_\_ == '\_\_main\_\_':

# 随机生成200个拥有2维实数特征 且 分类面（线）为y=-x^2+1.5(换言之，x2=-x1^2+1.5)的语料

x = get\_parabolic\_curve\_data\_set() # 可以加大数据量查看对结果的影响

print(1)

# 预留30%作为测试语料

# X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=0)

# 展示所生成的数据

show\_data\_set(x)

# 测试不同的超参数组合

X\_train=x

X\_test=[[2,6]]

y\_train=[0,0,0,0,0,0,0,0,0,1,0,1,1,1,1,1,1,1,1,1]

y\_test=[0]

print("准确率高，比较恰当的模型")

test(X\_train, X\_test, y\_train, y\_test, degree=2, C=1.0, penalty='l2')

print("准确率高，且恰当的模型")

test(X\_train, X\_test, y\_train, y\_test, degree=2, C=0.1, penalty='l2')

print("准确率高，但是过拟合的模型")

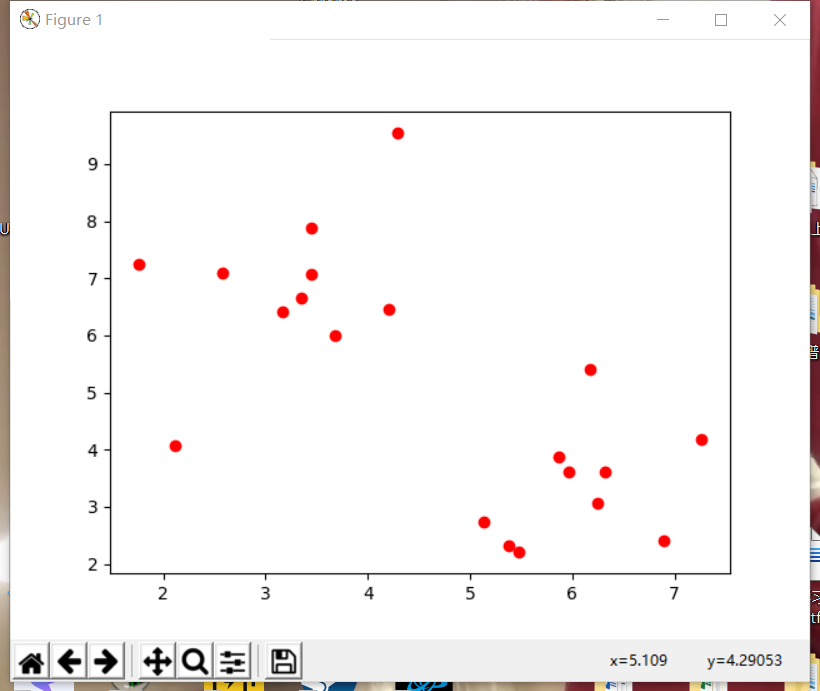
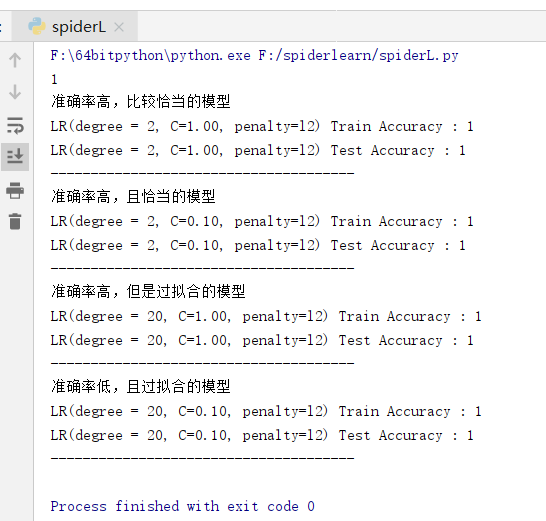
test(X\_train, X\_test, y\_train, y\_test, degree=20, C=1.0, penalty='l2')

print("准确率低，且过拟合的模型")

test(X\_train, X\_test, y\_train, y\_test, degree=20, C=0.1, penalty='l2')

**运行结果**：

（运行结果、现象的说明与截图）



在实验的参数不一样的情况下的结果图

