# Logistic回归

1. 模型概述
2. 概念描述

logistic回归是一种广义线性回归（generalized linear model），因此与多重线性回归分析有很多相同之处。它们的模型形式基本上相同，都具有 w’x+b，其中w和b是待求参数，其区别在于他们的因变量不同，多重线性回归直接将w‘x+b作为因变量，即y =w‘x+b，而logistic回归则通过函数L将w‘x+b对应一个隐状态p，p =L(w‘x+b),然后根据p 与1-p的大小决定因变量的值。如果L是logistic函数，就是logistic回归，如果L是多项式函数就是多项式回归。

logistic回归的因变量可以是二分类的，也可以是多分类的，但是二分类的更为常用，也更加容易解释，多类可以使用softmax方法进行处理。实际中最为常用的就是二分类的logistic回归。

1. 公式描述

|  |  |  |  |
| --- | --- | --- | --- |
| **数据集** | x= | y= | =[] |
| **拟合函数** |  | | |
| **损失函数** |  | | |
| **梯度下降法** | = - | |  |

1. 重要代码详解

logistic函数：返回值为logistic函数的值，这里返回的是列向量，每个值为对应的

def sigmoid(z):

    return 1/(1+np.exp(-z))

损失函数：np.multiply()表示两个矩阵对应元素相乘，要求两个矩阵行数列数相等，如果不等则会扩展其中一个矩阵至形状相同。这里分别表示。

返回值为np.sum()表示将矩阵中所有元素加起来，这里表示最后除以m得出损失函数的值。

def cost(x,y,theta):

    left=np.multiply(-y,np.log(sigmoid(x\*theta.T)))

    right=np.multiply(1-y,np.log(1-sigmoid(x\*theta.T)))

    return np.sum(left-right)/len(x)

梯度下降算法：error表示。

然后使用梯度下降算法 = -，通过迭代最后得出最优解。

def graddecent(x,y,theta,alpha,count):

    J=np.matrix(np.zeros(count))

    for i in range(count):

        error=sigmoid(x\*theta.T)-y

        theta=theta-alpha\*(x.T\*error).T

        J[:,i]=cost(x,y,theta)

    return theta,J

评估函数：由于没有设定测试集，所以直接把原来的x乘以对应的参数来计算预测值，然后设定阈值和真实的类别进行比对，得出正确率。

def predict(x,theta):

    res=sigmoid(x\*theta.T)

    prediction=[]

    for i in res:

        if i>=0.5:

            prediction.append(1)

        else:

            prediction.append(0)

    accuracy=0

    for i in range(len(prediction)):

        if prediction[i]==y[i]:

            accuracy+=1

    accuracy\_rating=accuracy/len(prediction)

    return accuracy\_rating

绘图：分别得到不同类别的点，然后分颜色和形状画出，然后画出分类的直线查看结果。

pos=data[data['y'].isin([1])]

neg=data[data['y'].isin([0])]

x1=np.arange(15,110,step=0.1)

x2 = -(theta[0,0] + x1\*theta[0,1]) / theta[0,2]

plt.plot(pos['x1'],pos['x2'],'o',c='b')

plt.plot(neg['x1'],neg['x2'],'x',c='r')

plt.plot(x1,x2,c='g')

plt.show()

1. 结果分析

参数：=[]

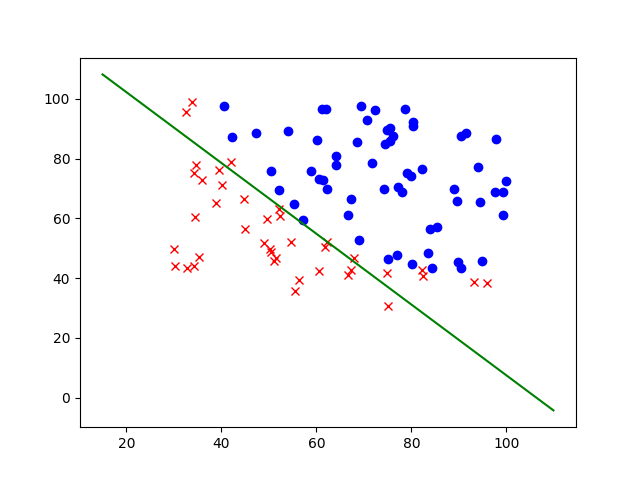
-4.81180027 0.04528064 0.03819149

损失函数最后为：0.38738952

与原分类别对比正确率为：准确率:0.91

我这里设定学习率为0.00001，迭代次数为100000，训练速度很慢，但是如果将学习率调大一点，我发现梯度下降不会收敛，所以只能将学习率设定非常低。

图示：因为没有使用多项式函数来作为特征，所以只是线性关系，拟合出分类直线，根据图例来看的话分类效果一般，有的点并没有很好的分出来。



四.源码

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

path = 'Logistic回归\data.txt'

data = pd.read\_csv(path, names=['x1', 'x2', 'y'])

data.insert(0, 'Ones', 1)

cols = data.shape[1]

x=np.matrix(data.iloc[:,0:cols-1])

y=np.matrix(data.iloc[:,cols-1:cols])

theta=np.matrix(np.zeros(3))

#看一下x,y,theta的形状

#print(x.shape,y.shape,theta.shape)

def sigmoid(z):

    return 1/(1+np.exp(-z))

def cost(x,y,theta):

    left=np.multiply(-y,np.log(sigmoid(x\*theta.T)))

    right=np.multiply(1-y,np.log(1-sigmoid(x\*theta.T)))

    return np.sum(left-right)/len(x)

def graddecent(x,y,theta,alpha,count):

    J=np.matrix(np.zeros(count))

    for i in range(count):

        error=sigmoid(x\*theta.T)-y

        theta=theta-alpha\*(x.T\*error).T

        J[:,i]=cost(x,y,theta)

    return theta,J

alpha=0.00001

count=100000

theta,J=graddecent(x,y,theta,alpha,count)

def predict(x,theta):

    res=sigmoid(x\*theta.T)

    prediction=[]

    for i in res:

        if i>=0.5:

            prediction.append(1)

        else:

            prediction.append(0)

    accuracy=0

    for i in range(len(prediction)):

        if prediction[i]==y[i]:

            accuracy+=1

    accuracy\_rating=accuracy/len(prediction)

    return accuracy\_rating

print(theta,J[:,-1])

print('准确率:%.2f'%predict(x,theta))

pos=data[data['y'].isin([1])]

neg=data[data['y'].isin([0])]

x1=np.arange(15,110,step=0.1)

x2 = -(theta[0,0] + x1\*theta[0,1]) / theta[0,2]

plt.plot(pos['x1'],pos['x2'],'o',c='b')

plt.plot(neg['x1'],neg['x2'],'x',c='r')

plt.plot(x1,x2,c='g')

plt.show()