实验五 逻辑回归

#!/usr/bin/python  
# coding=utf-8  
*'''  
 Logistic Regression Working Module  
 Created by PyCharm  
 Date: 2018/7/28  
'''*from numpy import \*  
import matplotlib.pyplot as plt  
  
  
def loadDataSet(path,training\_sample):  
 *'''  
 从文件中读入训练样本的数据，同上面给出的示例数据  
 下面第20行代码中的1.0表示x0 = 1  
 @param filename 存放训练数据的文件路径  
 @return dataMat 存储训练数据的前两列  
 @return labelMat 存放给出的标准答案（0,1）  
 '''* dataMat = []; labelMat = []  
 filename=path+training\_sample  
 fr = open(filename)  
 for line in fr.readlines():  
 line = line.strip('\n')  
 lineArr = line.strip().split(' ') #文件中数据的分隔符  
 dataMat.append([1.0, float(lineArr[0]), float(lineArr[1])]) #前两列数据  
 labelMat.append(int(lineArr[2])) # 标准答案  
 return dataMat,labelMat  
  
  
def sigmoid(inX):  
 *'''  
 定义激活函数  
 '''* return 1.0/(1+exp(-inX))  
  
  
def gradAscent(dataMatIn, classLabels):  
 *'''  
 梯度上升求最优参数a，学习率0.001，迭代次数1000次  
 @****:param*** *dataMatIn 文件中数据的前两列  
 @****:param*** *classLabels 标准答案  
 @****:return*** *weights 训练后的参数 3 x 1  
 '''* dataMatrix = mat(dataMatIn) #转化成矩阵  
 labelMat = mat(classLabels).transpose() #矩阵转置  
 m,n = shape(dataMatrix)  
 alpha = 0.001 #学习率  
 maxCycles = 500  
 weights = ones((n,1)) #3行 1列  
 for k in range(maxCycles): # 计算权重  
 h = sigmoid(dataMatrix\*weights) # 模型预测值, n x 1  
 error = (labelMat - h) # 真实值与预测值之间的误差, n x 1  
 temp = dataMatrix.transpose()\* error # 交叉熵代价函数对所有参数求偏导数, 3 x 1  
 weights = weights + alpha \* temp # 更新权重  
 return weights  
  
  
def plotBestFit(weights,dataMat,labelMat1,labelMat2):  
 *'''  
 分类效果展示，画图部分  
 @****:param*** *weights 回归系数  
 @****:param*** *path 数据文件路径  
 @****:return*** *null  
 '''* # dataMat,labelMat1=loadDataSet(path,testing\_sample)  
 # dataMat1,labelMat1=loadDataSet(path,training\_sample)  
 dataArr = array(dataMat)  
 n = shape(dataArr)[0] #取行数  
 xcord1 = []; ycord1 = []  
 xcord2 = []; ycord2 = []  
 xcord3 = []; ycord3 = []  
 xcord4 = []; ycord4 = []  
 for i in range(n): #将训练前的数据分类存储  
 if int(labelMat1[i])== 1:  
 xcord1.append(dataArr[i,1]); ycord1.append(dataArr[i,2])  
 else:  
 xcord2.append(dataArr[i,1]); ycord2.append(dataArr[i,2])  
 for i in range(n): #将训练后的数据分类存储  
 if int(labelMat2[i])== 1:  
 xcord3.append(dataArr[i,1]); ycord3.append(dataArr[i,2])  
 else:  
 xcord4.append(dataArr[i,1]); ycord4.append(dataArr[i,2])  
 fig = plt.figure("LogisticRegression") #新建一个画图窗口  
 ax = fig.add\_subplot(111) #添加一个子窗口  
 ax.set\_title('Original')  
 ax.scatter(xcord1, ycord1, s=30, c='red', marker='s')  
 ax.scatter(xcord2, ycord2, s=30, c='green')  
 x = arange(-3.0, 3.0, 0.1) #定义x轴  
 y = (-weights[0] - weights[1]\*x) / weights[2] # x2 = f(x1) 定义y轴 a0\*1+a1\*x+a2\*y  
 ax.plot(x, y) #画一条直线  
 plt.xlabel('X1'); plt.ylabel('X2')  
  
 plt.figure("logisticRegression")  
 plt.title('Forecast')  
 plt.scatter(xcord3, ycord3, s=30, c='red', marker='s')  
 plt.scatter(xcord4, ycord4, s=30, c='green')  
 plt.plot(x,y)  
 plt.xlabel('X1');plt.ylabel('X2')  
 plt.show()  
  
def getResult(dataArr,A):  
 h = sigmoid(mat(dataArr)\*A) #预测结果h(a)的值  
 H = []  
 for i in range(shape(h)[0]):  
 if h[i,0] > 0.5:  
 H.append(1)  
 else:  
 H.append(0)  
 return H