实验四 朴素贝叶斯

'''

Naive Bayes Algorithm

Created by PyCharm

Date: 2018/8/7

'''

import numpy as np

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

@param path 存放训练数据的文件路径

@:param training\_sample 文件名

@return dataMat 存储训练数据集

"""

dataMat = [];labelMat = []#定义列表

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([float(lineArr[0]), float(lineArr[1]),float(lineArr[2])]) #前三列数据

labelMat.append(int(lineArr[2])) # 标准答案

return dataMat,labelMat

def getSubCol(dataSet,col1,col2):

"""

取列表的部分列

@:param dataSet 数据列表

@:param col1 第col1列

@:param col2 第col2列

@:return list 返回列表子集

"""

rownum = len(dataSet)

list = []

for featVec in dataSet: # 统计每一类的数量

list.append([featVec[col1],featVec[col2]])

return list

def getSubRow(dataSet,value):

"""

取列表的部分行

@:param dataSet 数据列表

@:param value 要取的条件

@:return list 返回列表子集

"""

rownum = len(dataSet)

list = []

for featVec in dataSet:

if featVec[-1] == value:

list.append(featVec)

return list

def sample\_average(data\_sample):

"""

计算样本均值

@:param data\_sample 样本数据

@:return (sum/num) 样本均值

"""

num = len(data\_sample)

sum = 0

for i in range(num):

sum += data\_sample[i][0]

return sum / num

def sample\_variance(data\_sample, mean\_value):

"""

计算样本方差

@:param data\_sample 样本数据

@:param mean\_value 样本方差

@:return sum/(num-1) 返回方差

"""

num = len(data\_sample)

sum = 0

for i in range(num):

sum += np.square(data\_sample[i][0]-mean\_value)

return sum/(num-1)

def Gaussian\_distribution(data\_sample,mean\_value,variance):

"""

高斯分布函数

@:param data\_sample 样本数据

@:param mean\_value 样本均值

@:param variance 样本方差

@:return equation 结果

"""

molecule = 0 # 分子

denominator = 0 # 分母

equation = 0

molecule = np.exp(-(np.square(data\_sample - mean\_value)) / (2 \* variance)) #分子部分

denominator = np.sqrt(2\*np.pi\*variance) #分母部分

equation = (molecule/denominator)

return equation

def percentage(dataSet,value):

"""

计算样本中分类值的概率值

@:param dataSet 数据集

@:param value 分类值

@:param (count/num) 概率

"""

num = len(dataSet)

count = 0

for featVec in dataSet:

if featVec[-1] == value:

count += 1

return (count/num)

def plotBestFit(dataArr,labelMat1,labelMat2):

"""

分类效果展示

@:param dataArr 测试数据集

@:param labelMat1 标准结果

@:param labelMat2 预测结果

"""

n = len(dataArr) #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: #分类为1

xcord1.append(dataArr[i][0]); ycord1.append(dataArr[i][1])

else:

xcord2.append(dataArr[i][0]); ycord2.append(dataArr[i][1])

for i in range(n): #将训练后的数据分类存储

if int(labelMat2[i]) == 1: # 分类为1

xcord3.append(dataArr[i][0]);ycord3.append(dataArr[i][1])

else:

xcord4.append(dataArr[i][0]);ycord4.append(dataArr[i][1])

fig = plt.figure("Naive Bayes1") #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') #画点并标记颜色

ax.scatter(xcord2, ycord2, s=30, c='green') #画点并标记颜色

plt.xlabel('X1'); plt.ylabel('X2')

plt.figure("Naive Bayes2")

plt.title('Forecast')

plt.scatter(xcord3, ycord3, s=30, c='red', marker='s')

plt.scatter(xcord4, ycord4, s=30, c='green')

plt.xlabel('X1');plt.ylabel('X2')

plt.show()

def getResult(trainingSet,testingSet):

"""

对数据集进行朴素贝叶斯分类

@:param trainingSet 训练数据集，用于求均值和方差

@:param testingSet 测试数据集，预测结果

@:return h 结果向量

"""

p0 = percentage(trainingSet,0) #初始0的频率

p1 = percentage(trainingSet,1) #初始1的频率

h = []

mean\_value0 = [1,1]

variance0 = [1,1]

mean\_value1 = [1,1]

variance1 = [1,1]

for i in range(2): #求均值和方差

featList = getSubCol(trainingSet, i, 2) # 取部分特征

featList0 = getSubRow(featList, 0) # 取结果值为0的行

featList1 = getSubRow(featList, 1) # 取结果值为1的行

mean\_value0[i] = sample\_average(featList0) # 值为0的均值

variance0[i] = sample\_variance(featList0, mean\_value0[i]) # 值为0的方差

mean\_value1[i] = sample\_average(featList1) # 值为1的均值

variance1[i] = sample\_variance(featList1, mean\_value1[i]) # 值为1的方差

for featVec in testingSet: #计算数据样本的高斯值

result0 = 1 #初始化

result1 = 1 #初始化

for j in range(2):

Gaussian0 = Gaussian\_distribution(featVec[j],mean\_value0[j],variance0[j]) #计算结果为0的高斯值

Gaussian1 = Gaussian\_distribution(featVec[j], mean\_value1[j], variance1[j]) #计算结果为1的高斯值

result0 \*= Gaussian0 #迭乘运算

result1 \*= Gaussian1 #迭乘运算

result0 \*= result0\*p0 #为0的可能值

result1 \*= result1\*p1 #为1的可能值

if(result0 > result1): #分类

h.append(0)

else:

h.append(1)

return h

from NaiveBayes import \*

'''

主函数

'''

def main():

path = "D:\\AI\\data\\" #文件目录

training\_sample = 'trainingSet.txt' #训练数据文件

testing\_sample = 'testingSet.txt' #测试数据文件

trainingSet,label = loadDataSet(path,training\_sample) #获取训练数据

testingSet,label = loadDataSet(path,testing\_sample) #获取测试数据

h = getResult(trainingSet,testingSet) #计算结果向量

plotBestFit(testingSet,label,h) #图形化展示

'''

程序入口

'''

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

main()