山东大学 软件 学院

模式识别 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：201500301229 | 姓名：李洪德 | | 班级：15级二班 |
| 实验题目：朴素贝叶斯算法 | | | |
| 实验学时：4 | | 实验日期：2017.12.11 | |
| 实验目的：  掌握朴素贝叶斯的思想及相应的编程实现 | | | |
| 硬件环境：  Dell laptop  Intel i7 5500U | | | |
| 软件环境：  操作系统：Windows 8.1  使用语言：Python 3.6  开发软件包：Numpy  编译器：pycharm | | | |
| 实验步骤与内容：  利用贝叶斯算法进行数据分类操作，并统计其预测正确率。  一、算法设计说明：  1．实验数据分析  **汽车评价数据集**  共1728个数据，每个数据特征为6维，分为4类，类别标记为unacc，acc，good，vgood  四个类别标记分别表示汽车性价比等级（由低到高）  unacc：1210个  acc：384个  good：69个  vgood：65个  6个特征分别为：（6个属性）   1. buying （取值：vhigh、high、med、low） 表示购买价格 2. maint （取值： vhigh、high、med、low） 表示维修价格 3. door （取值：2、3、4、5more） 车门数量 4. Persons （取值：2、4、more） 可容纳人数 5. Lug\_boot （取值：small、med、big） 行李箱大小   Safety （取值：low、med、high） 安全系数  在car.txt文件中，每辆汽车数据占一行，中间用逗号分隔，最后一列是汽车评价。  将实验数据随机分为两个文件，一个car.txt保存的是训练集，另一个car\_test.txt保存测试集数据。  2.算法设计  贝叶斯公式:p(y|x)=p(x|y)p(y)/p(x)，可以用比较适合于这个问题的方法，p(y)已知，只需要挨个算出类条件概率密度。但是为了算法的一般性，我还是使用词汇表生成向量的方法。  生成一个4＋4＋4＋3＋3＋3的词汇表，用于计算条件概率密度。  **def** createVocabList():  vocabList = []  vocabList.append([**'vhigh'**,**'high'**,**'med'**,**'low'**])  vocabList.append([**'vhigh'**,**'high'**,**'med'**,**'low'**])  vocabList.append([**'2'**,**'3'**,**'4'**,**'5more'**])  vocabList.append([**'2'**,**'4'**,**'more'**])  vocabList.append([**'small'**,**'med'**,**'big'**])  vocabList.append([**'low'**,**'med'**,**'high'**])  **return** vocabList  然后编写读取训练数据和测试数据的函数：  **def** loadDataSet(filename):  **with** open(filename) **as** fr:  arrayLines = fr.readlines()postingList = []  classVec = []  **for** line **in** arrayLines:  line = line.strip()  listFromLine = line.split(**','**)  postingList.append(listFromLine[0:6])  classVec.append(listFromLine[-1])  **return** postingList,classVec  编写将测试数据集根据前面得到的词汇表生成用于计算的向量的函数：  *#vocabList是一个嵌套的list，所以处理的时候要注意下标* **def** setOfWords2Vec(vocabList, inputSet):  returnVec = [0]\*21  index = 0  count = 0  **for** word **in** inputSet:  **if** word **in** vocabList[index]:  returnVec[count + vocabList[index].index(word)] = 1  count += len(vocabList[index])  index += 1  **return** returnVec  根据训练向量（在testBayes函数里）生成训练矩阵，并将其用于训练贝叶斯分类器：  **def** trainBayes(trainMatrix,trainCatagory):  numTrainDocs = len(trainMatrix)  numWords = len(trainMatrix[0])  *#pA = 0.70023; pB = 0.22222; pC = 0.03993; pD = 0.03762* p1Num = ones(numWords); p2Num = ones(numWords)  p3Num = ones(numWords); p4Num = ones(numWords)  p1Denom = 4.0; p2Denom = 4.0; p3Denom = 4.0; p4Denom = 4.0  **for** i **in** range(numTrainDocs):  **if** trainCatagory[i] == **'unacc'**:  p1Num += trainMatrix[i]  p1Denom += sum(trainMatrix[i])  **elif** trainCatagory[i] == **'acc'**:  p2Num += trainMatrix[i]  p2Denom += sum(trainMatrix[i])  **elif** trainCatagory[i] == **'good'**:  p3Num += trainMatrix[i]  p3Denom += sum(trainMatrix[i])  **else**:  p4Num += trainMatrix[i]  p4Denom += sum(trainMatrix[i])  p1Vec = log(p1Num/p1Denom)  p2Vec = log(p2Num/p2Denom)  p3Vec = log(p3Num/p3Denom)  p4Vec = log(p4Num/p4Denom)  **return** p1Vec,p2Vec,p3Vec,p4Vec  编写根据所给测试向量得到概率的函数：  **def** classifyB(vec,p1Vec,p2Vec,p3Vec,p4Vec):  p1 = sum(vec\*p1Vec)+log(pA)  p2 = sum(vec\*p2Vec)+log(pB)  p3 = sum(vec\*p3Vec)+log(pC)  p4 = sum(vec\*p4Vec)+log(pD)  **if** (p1>p2 **and** p1>p3 **and** p1>p4): **return 'unacc'  elif** (p2>p1 **and** p2>p3 **and** p2>p4): **return 'acc'  elif** (p3>p2 **and** p3>p1 **and** p3>p4): **return 'good'  else**: **return 'vgood'**  最后编写调用函数：  **def** testBayes():  *#读进训练集* list, listClass = loadDataSet(**'car.txt'**)  *#得到词汇表* vocabList = createVocabList()  *#生成训练矩阵* trainMat = []  **for** doc **in** list:  trainMat.append(setOfWords2Vec(vocabList,doc))  *#训练贝叶斯分类器* p1,p2,p3,p4 = trainBayes(trainMat,listClass)   *#读进测试集数据  #进行预测，并与标签进行比较得到错误率* err = 0  index = 0  testList,testListClass = loadDataSet(**'car\_test.txt'**)  **for** line **in** testList:  testDoc = setOfWords2Vec(vocabList,line)  ans = classifyB(testDoc,p1,p2,p3,p4)  print(line,**'classified as: '**,ans)  **if** (ans != testListClass[index]): err += 1  index += 1  print(**'the errors %d the error rate %f'** % (err,err/len(testListClass)))  其中类概率在数据集中给出了也可以利用代码重新统计，这里我们用文件中给出的类概率：  pA = 0.70023; pB = 0.22222; pC = 0.03993; pD = 0.03762  3.实验结果    省略了好多输出，用了八百多个测试数据，得到正确率只有83.83%。  4.实验结果分析  可以看到，这个代码得到的正确率并不高，可能是由于一下几个原因造成的  首先，朴素贝叶斯的前提是属性之间不相关，但是这六个属性相关，购买价格和维修价格一般是呈正相关关系，车门数量和可容纳人数以及行李箱大小这三个属性之间也有相关关系，所以说，这几个属性的选取并不是那么合适。  其次，测试集和训练集划分的不够随机，这两个数据集的分布有可能不同。  参考书籍：《machine learning in action》 Peter Harrington  《机器学习》周志华  程序运行方法：  将car.txt和car\_test.txt文件与python文件放到一个文件夹下  进入到py文件所在目录，输入python开启python shell程序  输入：import bayes  bayes.testBayes() | | | |
| 结论分析与体会： | | | |