数据挖掘课程实验报告

Homework 2: NBC

201834879　王士东　2018/11/14

一．贝叶斯定理

概率论中的经典条件概率公式：

$$P(Y|X)=\frac{P(X|Y)P(Y)}{P(X)}$$

其中，P（X，Y）= P（Y，X）<=> P（X|Y）P（Y）= P（Y|X）P（X），即 X 和 Y 同时发生的概率与 Y 和 X 同时发生的概率一样。

二．朴素贝叶斯定理

朴素贝叶斯的经典应用是对垃圾邮件的过滤，以及对文本格式的数据进行处理，因此这里以此为背景讲解朴素贝叶斯定理。

设 D 是训练样本和相关联的类的集合，其中训练样本的属性集为 X{ X1,X2, ... , Xn}，共有 n 个属性；类集为 C{ C1,C2, ... ,Cm }，有 m 种类别。

朴素贝叶斯定理：

$$P(C_i/\boldsymbol{X})=\frac{P(\boldsymbol{X}/C_i)P(C_i)}{P(\boldsymbol{X})}.$$

其中，P（Ci|X）为后验概率，P（Ci）为先验概率，P（X|Ci）为条件概率。由于对于所有的测试集计算时，上式的分母都是一样的，都是 P(X=X(test))，所以只需考虑分子的最大值。

又由于朴素贝叶斯的两个假设：1、属性之间相互独立；2、每个属性同等重要。通过假设 1 知，条件概率 P（X|Ci）可以简化为：

$$\mathrm{P}\left(\mathrm{X}|\mathrm{C_i}\right)=\prod_{k=1}^{k=n}P(\mathrm{X_k}|\mathrm{C_i})=P(X_1|C_i)\times P(X_2|C_i)\times\cdots\times P(X_n|C_i)$$

三．朴素贝叶斯算法实现

朴素贝叶斯算法的核心思想，是选择训练集中具有最高后验概率的类别作为

测试数据的预测类别。下面介绍利用 Python 语言实现朴素贝叶斯算法的过程，其本质是利用词和类别的联合概率来预测给定文档属于某个类别。

1. 数据集的准备

本实验所用数据集已经预处理完成，共包含 20 个类别，每个类别中包含若干文本文档，每个文本文档里包含若干单词，每个单词占一行。

1）生成训练集向量

此过程返回一个训练集向量集合，list 的格式为：[[str,int,float,dict],[],[]]。其中，str 为某个类别，int 为该类别所有文档的单词总数（一个单词在一个文档中只计算一次），float 为 P(Ci)的值（该类别的文档总数/数据集文档总数），dict 为该类别的字典，字典的 value 值为该类别所有单词的 df 值（包含该单词的文档总数）。

```
#返回训练集向量集合
def gettrainlist():

    #list格式为：[[str,int,float,{}],  [],[]]，
    #其中，str为某个类别，int为该类别所有文档的单词总数（一个单词在一个文档中只计算一次），
    #float为P(Ci)的值（该类别的文档总数/数据集文档总数），
    #dict为该类别的字典，字典的value值为该类别所有单词的df值（包含该单词的文档总数）。
    rootlist = []
    numfiles = getnumfiles(traindata) #总文档个数

    rootpathlist = os.listdir(traindata)
    for i in rootpathlist:
        rootpath = traindata + os.path.sep + i #目录名+路径切割符+文件名;i为每一个主文件夹的路径
        subspathlist = os.listdir(rootpath) #子文件夹列表

        dfdict = {} #存放值为df的类别字典
        subslist = [] #每一个类别的向量list,格式为[str,int,float,{}]
        #存类名
        subslist.append(i)
        #存各文档的单词总数（每个单词在一个文档中只计算一次）
        wordsum = 0
        for j in subspathlist:
            subspath = rootpath + os.path.sep + j
            lines = open(subspath).readlines() #此时每一行都为一个单词
            coun = collections.Counter(lines) #counter函数
            wordsum = wordsum + len(coun)
            for key,value in coun.items():  # counter函数的items()转化成(元素，计数值)组成的列表
                key = key.strip('\n')   #去除换行符，此时的数据是带着换行符的
                dfdict[key] = dfdict.get(key,0) + 1
        subslist.append(wordsum)
        #存P(Xi=c)
        pc = len(subspathlist)/numfiles #P(Xi=c)为该类别文档个数/总文档个数
        subslist.append(pc)
        #存字典
        subslist.append(dfdict)
        rootlist.append(subslist)
    return rootlist
```

2）生成测试集向量

此过程返回一个测试集向量集合，list 的格式为：[[str,list], [],[]]。其中，str 为测试集该向量所属的类别，list 为测试集该向量所包含的所有单词集合（无重复单词）。

```
#返回测试集向量集合
def gettestlist():

    #list格式为[[str,[]], [],[]],
    #其中，str为测试集该向量所属的类别，list为测试集该向量所包含的所有单词集合（无重复单词）。
    rootlist = []

    rootpathlist = os.listdir(testdata)
    for i in rootpathlist:
        rootpath = traindata + os.path.sep + i
        subspathlist = os.listdir(rootpath)

        for j in subspathlist:
            subslist = []   #格式为[str,[]]
            #存类名
            subslist.append(i)
            templist = []  #第一次subslist和templist写到j循环的外边，所以运行了三小时。。。。

            subspath = rootpath + os.path.sep + j
            lines = open(subspath).readlines() #此时每一行都为一个单词
            coun = collections.Counter(lines) #counter函数
            for key,value in coun.items():
                key = key.strip('\n')
                templist.append(key)

            #存单词list
            subslist.append(templist)
            rootlist.append(subslist)
    return rootlist
```

2.NBC 算法实现

1）装载数据：导入第一步已经预处理完成的训练集和测试集的向量集合

2）模型应用（伯努利朴素贝叶斯模型）：本实验采用的伯努利朴素贝叶斯模型，统计（训练）时和判断（测试）时均不考虑重复单词；

3)Laplace 平滑：若测试数据中有的单词在训练集没有出现，其概率就是 0，会十分影响分类器的性能，所以采取 Laplace 平滑，让分子各单词的出现次数默认加 1，让分母单词出现总数加上测试数据的单词总数，这样处理后不影响相对大小。

4）解决下溢问题：若很小是数字相乘，则结果会更小，再四舍五入存在误差，而且会造成下溢出。所以对概率值取 log，乘法变为加法，并且相对大小趋

势不变。

5）模型性能评测：记录模型预测成功与失败次数，计算并输出模型预测准确率。

```
def NBC():
    #装载数据
    trainlist = gettrainlist() #[[str,int,float,{}],  [],[]]
    print('训练集装载完成！ 数据集大小： ',len(trainlist))
    testlist = gettestlist() #[[str,[]],  [],[]]
    print('测试集装载完成！ 数据集大小： ',len(testlist))

    success = 0 #记录模型预测成功的次数
    failure = 0 #记录模型预测失败的次数

    print('预测开始：')
    for i in range(len(testlist)):
        maxp = 0          #与训练集向量比较后，最大的概率值P
        maxclass = ' '    #最大概率值P所属类的类名

        for j in range(len(trainlist)):
            #对P做log处理，不影响大小关系，不然的话用乘积，超过了计算机下限，最后全是0
            p = math.log10(trainlist[j][2])
            #print(p)
            for key in testlist[i][1]:
                #Laplace平滑，分子加1，分母加单词总数
                tempdp = trainlist[j][3].get(key,0) + 1
                tempfenmu = trainlist[j][1] + len(testlist[i][1])
                tempp = math.log10(tempdp / tempfenmu)

                p = p + tempp

            #让maxp,和maxclass初始值默认为第一个测试数据的值，不可以直接设为0，因为maxp的值可能小于0
            if j == 0:
                maxp = p
                maxclass = trainlist[j][0]
            elif p > maxp:
                maxp = p
                maxclass = trainlist[j][0]


        #print(maxclass,' == ', testlist[i][0])
        if maxclass == testlist[i][0]:
            success = success + 1
            #print('预测成功')
        else:
            failure = failure + 1
            #print('预测失败')

        if (i % 1000) == 0:
            print('已完成测试次数： ' ,i+1 ) #打印程序运行程度

    #输出模型的性能
    successp = success / (success + failure)
    print('预测结束！')
    print('该模型的性能：')
    print('总测试次数： ' , (success + failure))
    print('预测成功次数： ' , success)
    print('预测失败次数： ' , failure)
    print('预测准确率： ' , successp)
```

3.模型预测结果

```
In [2]: runfile('D:/code/AnacondaCod
AnacondaCodes/Homework 2 NBC')
训练集装载完成！  数据集大小：   20
测试集装载完成！  数据集大小：   15074
预测开始：
已完成测试次数：  1
已完成测试次数：  1001
已完成测试次数：  2001
已完成测试次数：  3001
已完成测试次数：  4001
已完成测试次数：  5001
已完成测试次数：  6001
已完成测试次数：  7001
已完成测试次数：  8001
已完成测试次数：  9001
已完成测试次数：  10001
已完成测试次数：  11001
已完成测试次数：  12001
已完成测试次数：  13001
已完成测试次数：  14001
已完成测试次数：  15001
预测结束！
该模型的性能：
总测试次数：  15074
预测成功次数：  14168
预测失败次数：  906
预测准确率：  0.9398965105479634
```

四.总结

不同于其它分类器，朴素贝叶斯是一种基于概率理论的分类算法。特征之间的条件独立性假设，显然这种假设显得“粗鲁”而不符合实际，这也是名称中“朴素”的由来，然而事实证明，朴素贝叶斯在有些领域很有用，比如垃圾邮件过滤。

在具体的算法实施中，要考虑很多实际问题。比如因为“下溢”问题，需要对概率乘积取对数；再比如词集模型和词袋模型，还有停用词和无意义的高频词的剔除，以及大量的数据预处理问题等。

总的来说，朴素贝叶斯原理和实现都比较简单，学习和预测的效率都很高，是一种经典而常用的分类算法。