

**模式识别大作业**

题 目 垃圾短信分类

学 院 信息科学与工程

专 业 控制科学与工程

组 员 王旭鹏

指导教师 赵海涛

**完成日期： 2018 年 10 月26日**

**模式识别作业报告——垃圾短信分类**

组员：王旭鹏

经过几次课程的模式识别学习，在赵海涛老师的辛勤指导下，我对模式识别的贝叶斯决策有了一定的了解，并通过本次英文垃圾短信分类实验来巩固所学内容。

经过一段时间的程序编写及调试，并尝试了各种主流的推荐系统算法，最终准确率达到0.96951，跑完一次程序约5秒左右。下面将详细说明我的解决过程。

**一、垃圾短信问题描述**

本题提供一个数据集, 它包括了5574条英文短信，每条短信内容由几个长短不一的句子组成。每条短信都标注好了是否为垃圾短信，通过该训练集训练出一个分类器，预测测试集中的短信内容是否为垃圾短信。

**二、整体解决方案**

概率论是许多机器学习算法的基础，所以深刻理解这一主题就显得十分重要。同时针对垃圾短信分类的具体问题，首先如何处理文本信息方便后面机器学习的算法进行操作是至关重要的。我们将充分利用Python的文本处理能力将文档切分成词向量，并构建TF-IDF特征来对文本数据进行预处理。

**2.1贝叶斯定理简述**

贝叶斯准则告诉我们如何交换条件概率中的条件与结果，即如果已知 P(x|c) ，要求 P(c|x) ，那么可以使用下面的计算方法：



假设一个二分类问题：给定某个由x、y表示的数据点，那么该数据点来自类别 c 1 的概率是多少？数据点来自类别 c 2 的概率又是多少？具体地，应用贝叶斯准则得到：



于是我们有：

如果 P(c 1 |x, y) > P(c 2 |x, y) ，那么属于类别 c 1 。

如果 P(c 1 |x, y) < P(c 2 |x, y) ，那么属于类别 c 2 。

使用贝叶斯准则，可以通过已知的三个概率值来计算未知的概率值。后面就会给出利用贝叶斯准则来计算概率并对数据进行分类的代码。

**2.2数据的读取与预处理**

在进行垃圾短信分类之前，我们需要对文本信息进行预处理。可以使用分词工具，然后将句子转换成词向量，建立词表。简单的，可以通过统计词汇表中词汇出现的次数构建向量。初步的清洗数据，包括把所有的单词都转换成小写，删除除了英文之外的字符，考虑到英语有简写，恢复简写的部分，把所有的单词恢复成词干。实现程序如下：

|  |
| --- |
| def read\_data(file):#此函数是读取CSV文的文件  train\_data = csv.reader(open(file, encoding="utf-8"))  lines = 0  for r in train\_data:  lines += 1  train\_data\_label = np.zeros([lines - 1, ])  train\_data\_content = []  train\_data = csv.reader(open(file, encoding="utf-8"))  i = 0  for data in train\_data:  if data[0] == "Label" or data[0] == "SmsId":  continue  if data[0] == "ham":  train\_data\_label[i] = 0  if data[0] == "spam":  train\_data\_label[i] = 1  train\_data\_content.append(data[1])  i += 1  print(train\_data\_label.shape, len(train\_data\_content))  return train\_data\_label,train\_data\_content  #下面函数是对文本内容进行简单的处理，比如将单词转化为小写，将常见的一些缩写恢复  def clean\_text(comment\_text):  comment\_list = []  for text in comment\_text:  # 将单词转换为小写  text = text.lower()  # 删除非字母、数字字符  text = re.sub(r"[^a-z']", " ", text)  # 恢复常见的简写  text = re.sub(r"what's", "what is ", text)  text = re.sub(r"\'s", " ", text)  text = re.sub(r"\'ve", " have ", text)  text = re.sub(r"can't", "can not ", text)  text = re.sub(r"cannot", "can not ", text)  text = re.sub(r"n't", " not ", text)  text = re.sub(r"\'m", " am ", text)  text = re.sub(r"\'re", " are ", text)  text = re.sub(r"\'d", " will ", text)  text = re.sub(r"ain\'t", " are not ", text)  text = re.sub(r"aren't", " are not ", text)  text = re.sub(r"couldn\'t", " can not ", text)  text = re.sub(r"didn't", " do not ", text)  text = re.sub(r"doesn't", " do not ", text)  text = re.sub(r"don't", " do not ", text)  text = re.sub(r"hadn't", " have not ", text)  text = re.sub(r"hasn't", " have not ", text)  text = re.sub(r"\'ll", " will ", text)  # text = re.sub(r"am", "", text)  #进行词干提取  new\_text = ""  s = nltk.stem.snowball.EnglishStemmer() # 英文词干提取器  for word in word\_tokenize(text):  new\_text = new\_text + " " + s.stem(word)  # 放回去  comment\_list.append(new\_text)  return comment\_list |

上述这种简单地构建向量的方式，会导致每个词的重要过于均衡，无法体现词向量中词汇的重要性，导致分类结果不理想。改进中，可以采用TF-IDF方式来构建词向量。这之后，可以使用条件概率来进行分类，实现贝叶斯分类器。

TF-IDF的基本思想是：词语的重要性与它在文件中出现的次数成正比，但同时会随着它在语料库中出现的频率成反比下降。 但无论如何，统计每个单词在文档中出现的次数是必要的操作。所以说，TF-IDF也是一种基于 bag-of-word 的方法。Python的机器学习包已经有相关的TF-IDF特征提取，所以这一步实现相对简单。实现程序如下：

|  |
| --- |
| # 载入数据  train\_y,train\_data\_content = read\_data("train.csv")  \_,test\_data\_content = read\_data("test.csv")  train\_data\_content = clean\_text(train\_data\_content)  test\_data\_content = clean\_text(test\_data\_content)  # 数据的TF-IDF信息计算  all\_comment\_list = list(train\_data\_content) + list(test\_data\_content)  text\_vector=TfidfVectorizer(sublinear\_tf=True, strip\_accents='unicode',token\_pattern=r'\w{1,}',max\_features=5000, ngram\_range=(1, 1), analyzer='word')#取所有文件中出现频率最高的前5000个词，作为衡量每个文本的特征  text\_vector.fit(all\_comment\_list)  train\_x = text\_vector.transform(train\_data\_content)  test\_x = text\_vector.transform(test\_data\_content)  train\_x = train\_x.toarray()  test\_x = test\_x.toarray()  #将训练和测试样本转化为词向量(5572, 5000) (1115, 5000)，矩阵中的数值跟单词在文本中出现次数以及重要程度都有关系 |

**2.3贝叶斯训练器的程序实现**

首先根据训练集上的数据集训练贝叶斯分类器，及求取每个特征的类条件概率密度以及先验概率。在求取特征的类条件概率密度时，我们简单的假设个特征之间是独立的，这样在求联合概率密度过程中，我们就可以将先求每个特征的条件概率再相乘即可。但是我们需要做一些修正。利用贝叶斯分类器对文档进行分类时，要计算多个概率的乘积以获得文档属于某个类别的概率，即计算 。如果其中一个概率值为0，那么最后的乘积也为0。为降低这种影响，可以将所有词的出现数初始化为1，并将分母初始化为2。另一个遇到的问题是下溢出，这是由于太多很小的数相乘造成的。当计算乘积 时，由于大部分因子都非常小，所以程序会下溢出或者得到不正确的答案。一种解决办法是对乘积取自然对数。在代数中有 ln(a\*b) = ln(a)+ln(b) ，于是通过求对数可以避免下溢出或者浮点数舍入导致的错误。同时，采用自然对数进行处理不会有任何损失它们的取值虽然不同，但不影响最终结果。以下是经过修正后的训练分类器的函数及进行分类的函数：

|  |
| --- |
| #训练朴素贝叶斯模型（得到所需后验概率）  def trainNB0(trainMatrix,trainCategory):  numTrainDocs = len(trainMatrix)  numWords = len(trainMatrix[0])  pAbusive = sum(trainCategory)/float(numTrainDocs)  p0Num = ones(numWords); p1Num = ones(numWords) #change to ones()  p0Denom = 2.0; p1Denom = 2.0 #change to 2.0  for i in range(numTrainDocs):  if trainCategory[i] == 1:  p1Num += trainMatrix[i]  p1Denom += sum(trainMatrix[i])  else:  p0Num += trainMatrix[i]  p0Denom += sum(trainMatrix[i])  p1Vect = log(p1Num/p1Denom) #change to log()  p0Vect = log(p0Num/p0Denom) #change to log()  return p0Vect,p1Vect,pAbusive  #朴素贝叶斯分类器  def classifyNB(vec2Classify, p0Vec, p1Vec, pClass1):  p1 = sum(vec2Classify \* p1Vec) + log(pClass1) #element-wise mult  p0 = sum(vec2Classify \* p0Vec) + log(1.0 - pClass1)  if p1 > p0:  return 1  else:  return 0 |

**三、小组分工**

程序设计及编写：王旭鹏

程序调试：王旭鹏

实验报告：王旭鹏

**四、作业总结**

由于这是第一次接触稍大的数据处理，所以在算法实现上走了不少弯路，首先碰到的问题就是如何处理文本数据，在查找了许多资料之后，我决定采用TF-IDF方法来提取文本信息。

这次实验我采用朴素贝叶斯分类器进行垃圾短信的分类。限于现阶段能力有限，并没有使用其他机器学习的模型，不过采用贝叶斯分类模型，在lintcode上的提交预测答案之后的准确率已经达到0.96951。所以从准确率角度上来说，贝叶斯模型已经很好地能完成该任务。

这次的大作业让增加了我对模式识别的兴趣，同时原来神秘的推荐算法变得立体清晰。当然我也感觉到了自身能力的不足，“practice makes perfect”，要想真正领略模式识别的精髓，需要我不断的理解和积累不同的机器学习模型并将它们融会贯通。“路漫漫其修远兮，吾将上下而求索”。最后感谢赵老师教学帮助，并且在作业过程中给了我们大家很多指导。

**附：文件说明**

本次附件一共包含有：

1 大作业报告；

2 最终的Python实现程序源码：lastMovieLens.py

3 导出了预测数据集：lastSVD0.91616.csv