中山大学计算机学院

人工智能

本科生实验报告

（2022学年春季学期）
课程名称：Artificial Intelligence

教学班级	专业(方向)	学号	姓名
计科三班	计算机科学与技术	21307185	张礼贤

一、实验题目

利用朴素贝叶斯方法完成文本信息情感分类训练

二、实验内容

1. 算法原理

- 数据预处理：将文本数据转换为数字向量，通常使用词袋模型将每个文档表示为一个向量，其中每个元素表示对应单词在文档中出现的频率或者重要性等。
- 计算每个类别的先验概率：先验概率是指在没有考虑任何特征的情况下，每个类别出现的概率。通常可以根据训练集中每个类别的文档数量来估计其先验概率。
- 计算每个特征在各个类别下的条件概率：条件概率是指在给定类别的情况下，某个特征出现的概率。在朴素贝叶斯算法中，条件概率可以被分解为各个特征的乘积，假设各个特征之间是独立的。这也是“朴素”一词的来源。
- 拉普拉斯平滑技巧是一种常用于朴素贝叶斯分类器中的平滑技术，用于解决零概率问题。在分类器的训练过程中，由于某些特征在训练样本中从未出现过，导致它们的概率为0。这会使得分类器无法正确预测包含这些特征的测试样本。其作用在于减小了特征数量很多但是训

练集中很少出现的特征对模型的影响，从而提高了模型的鲁棒性和泛化能力。但是，拉普拉斯平滑技巧也会引入一定的噪声

拉普拉斯平滑技巧通过给所有特征值出现次数加上一个正整数（通常是1）来解决这个问题，从而避免出现概率为0的情况

- 计算后验概率：根据训练集中的样本特征和类别，可以计算出给定一个样本特征，属于每个类别的后验概率。根据贝叶斯定理，后验概率可以表示为：P(类别|特征) = P(特征|类别) * P(类别) / P(特征)。其中，P(类别)是先验概率，P(特征|类别)是条件概率，P(特征)是文档中所有特征出现的概率。

- 通过后验概率进行预测：在实际计算中，由于条件概率求的是每个单词所在的类中的条件概率，所以需要将文本中的每个单词的后验概率相加，通过遍历每一个给定的类得到相应的概率，并且取最大值所对应的类返回，作为预测值

---

2. 关键代码即注释

```
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from timeit import default_timer as timer
import math
from collections import defaultdict

label_dict = {"joy": 4,"disgust": 2,"anger": 1 ,"fear": 3,"sad": 5
,"surprise":6}

def read_file(filename):
    """读取并分离输入文件中的信息"""
    texts = []  #存储测试文本信息
    labels = [] #存储测试用的标签
    with open(filename, 'r') as f:
        for line in f:
            s = line.strip().split(' ')
            if s[0] == 'documentId':
                continue
            text = s[3]
            for c in s[4:]:
                text += ' ' + c
            label = label_dict[s[2]]
            texts.append(text)
            labels.append(label)
    return texts, labels

def calculate_prior(train_labels):
    """计算每个类出现的先验概率"""
    class_count = np.bincount(train_labels)[1:]  # 统计每个类别的样本数量
    class_prior = class_count / len(train_labels)  # 计算每个类别的先验概率
    return class_prior

def calculate_case(train_labels,train_texts,vocabulary):
    """计算每个文本特征出现在各自情感类的条件概率"""
    class_probs = defaultdict(dict)  # 存储每个类别下每个特征的条件概率
    num_features = train_set.shape[1]
    word_count = {1: defaultdict(int), 2: defaultdict(int), 3:
defaultdict(int),
                  4: defaultdict(int), 5: defaultdict(int), 6:
```

```
defaultdict(int)}
    class_count = {1:0,2:0,3:0,4:0,5:0,6:0}
    for i in range(len(train_labels)):
        label = train_labels[i]
        s = train_texts[i].split(' ')
        for word in s:
            word_count[label][vocabulary.get(word)] += 1
        class_count[label] += len(s)
    for c in range(1,7):
        # 统计训练集中属于类别 c 的文本数目
        sum = 0
        num_docs_in_class = class_count[c]
        for j in range(num_features):
            # 统计属于类别 c 的文本中特征 j 出现的次数
            count = word_count[c][j]
            sum += count
            # 计算条件概率 P(x_j|c_i)
            class_probs[c][j] = (count + 1) / (num_docs_in_class + 2)
    return class_probs

def predict(test_texts,test_labels,vocabulary,class_prior,class_probs):
    """预测函数,根据测试集的样本预测并计算正确率"""
    test_probs = test_set.toarray()
    count = [0,0,0,0,0,0]
    size = [0,0,0,0,0,0]
    accuracy = [0.0,0.0,0.0,0.0,0.0,0.0]    #统计每一个情感类的预测正确率
    cnt = 0
    for k in range(len(test_texts)):
        t = test_texts[k]
        words = t.split(' ')
        probs = []     #存储每个类的后验概率，选择最大的返回
        for c in range(1,7):
            sum = 0.0   #存储语句中单词的后验概率之和
            for word in words:
                if(word not in vocabulary):continue #如果在训练集中的单词未在测试集中出现，则跳过
                index = vocabulary.get(word)
                sum += class_prior[c-1] * class_probs[c][index]  #计算后验概率，分母一致就不用管了
            probs.append(sum)
        res = np.argmax(probs) + 1  #返回最大概率的下标，不要忘了加一表示相应的类
        if(res == test_labels[k]):
```

```
            count[res-1] += 1
            cnt += 1
        size[test_labels[k] - 1] += 1
    for i in range(6):
        accuracy[i] = count[i] / size[i]
        print(f'The {i+1}th class Accuracy: {accuracy[i]:.4f}')

    print(f'The total Accuracy: {cnt/len(test_texts):.4f}')

if __name__ == '__main__':
    """主函数"""
    train_texts, train_labels = read_file('train.txt')  #通过读文件获取测试集和标签集
    test_texts, test_labels = read_file('test.txt')
    train_labels = np.array(train_labels)   #将标签集合转换为np数组，才能作为参数传入knn函数中

    tfidf = TfidfVectorizer()   #创建tfidf类
    train_set = tfidf.fit_transform(train_texts)#前者调用fit_transform函数固定化匹配模板
    vocabulary = tfidf.vocabulary_     #生成反映射到索引的单词的字典
    test_set = tfidf.transform(test_texts)  #后者直接调用transform函数直接套用前面的模板，达到对齐的效果

    probs_prior = calculate_prior(train_labels) #计算先验概率
    probs_case = calculate_case(train_labels,train_texts,vocabulary)    #计算条件概率
    predict(test_texts, test_labels, vocabulary, probs_prior, probs_case)  #根据给定的文本输入进行预测，并打印预测正确率
```

三、实验结果展示及分析

实验结果展示：

- **lamda = 0.01**
  - The 1th class Accuracy: 0.1061
  - The 2th class Accuracy: 0.1923

- The 3th class Accuracy: 0.1375
- The 4th class Accuracy: 0.5525
- The 5th class Accuracy: 0.4950
- The 6th class Accuracy: 0.0489
- The total Accuracy: 0.3430
- **总体预测值准确率**：0.3440

- **lamda = 0.02**
  - The 1th class Accuracy: 0.1061
  - The 2th class Accuracy: 0.1923
  - The 3th class Accuracy: 0.1375
  - The 4th class Accuracy: 0.5525
  - The 5th class Accuracy: 0.4950
  - The 6th class Accuracy: 0.0489
  - The total Accuracy: 0.3430

- **lamda = 1**
  - The 1th class Accuracy: 0.1061
  - The 2th class Accuracy: 0.1923
  - The 3th class Accuracy: 0.1375
  - The 4th class Accuracy: 0.5580
  - The 5th class Accuracy: 0.4950
  - The 6th class Accuracy: 0.0489
  - The total Accuracy: 0.3450

- **lamda = 10**
  - The 1th class Accuracy: 0.3182
  - The 2th class Accuracy: 0.2692
  - The 3th class Accuracy: 0.0063
  - The 4th class Accuracy: 0.6215
  - The 5th class Accuracy: 0.1980
  - The 6th class Accuracy: 0.0000
  - The total Accuracy: 0.2940

- **lamda = 100**
  - The 1th class Accuracy: 0.7727
  - The 2th class Accuracy: 0.1538
  - The 3th class Accuracy: 0.0000
  - The 4th class Accuracy: 0.1906
  - The 5th class Accuracy: 0.0000
  - The 6th class Accuracy: 0.0000
  - The total Accuracy: 0.1240

实验结果分析

- 由实验结果可知，六个类的预测值的准确率各不相同且各有高低，可能是以下原因导致的：

  1. 样本分布不均匀：不同的样本可能分布不相同，每个类的样本数量各有差异
  2. 样本特征分布不均匀：不同的样本特征对于每个类的分布可能并不相同，在某些模型的预测上可能表现较差

- 调整lamda参数，可以发现随着lamda的增大，预测的准确率也随之降低，取相对较小的值能够提高准确率