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	人工智能（陈川）	专业（方向）	计算机科学与技术人工智能与大数据
学号	20337025	姓名	崔璨明

一、实验题目

在给定的文本数据集完成文本情感分类训练，在测试集完成测试，计算准确率。需要对上次给的数据集进行重新划分，训练集：测试集为8：2

实验要求

1. 文本的特征可以使用TF-IDF（可以使用sklearn库提取特征）
2. 利用KNN完成对测试集的分类，并计算准确率
3. 报告中需探究超参K对分类准确率的影响
4. 需要提交简要报告+代码

二、实验内容

1、算法原理

KNN（K- Nearest Neighbor）即K最邻近法。该方法的思路非常简单直观：如果一个样本在特征空间中的K个最相似（即特征空间中最邻近）的样本中的大多数属于某一个类别，则该样本也属于这个类别。该方法在定类决策上只依据最邻近的一个或者几个样本的类别来决定待分样本所属的类别 。

该方法的不足之处是计算量较大，因为对每一个待分类的文本都要计算它到全体已知样本的距离，才能求得它的K个最邻近点。目前常用的解决方法是事先对已知样本点进行剪辑，事先去除对分类作用不大的样本。KNN算法比较适用于样本容量比较大的类域的自动分类，而那些样本容量较小的类域采用这种算法比较容易产生误分 。

总体来说，KNN分类算法包括以下4个步骤：

1. 准备数据，对数据进行预处理。
2. 计算测试样本点（也就是待分类点）到其他每个样本点的距离。
3. 对每个距离进行排序，然后选择出距离最小的K个点。
4. 对K个点所属的类别进行比较，根据少数服从多数的原则，将测试样本点归入在K个点中占比最高的那一类

2、算法伪代码

```
输入：训练集trian_set，每个句子对应的类别types，测试集test_set。
输出：测试集中每个句子对应的类别列表res
def classify_knn(trian_set,types,test_set):
        res=[]
        for i1 in range(len(test_set)): #对每个句子进行判断
                arr=[]
                for i2 in range(len(train_set)):
                        dis=caculate_cos(test_set[i1],train_set[i2]) #
计算测试句到该训练集中句子的余弦相似度
                        arr.push(dis,types[i2])#记录距离和类型
                arr.sort()#进行排序
                choose k values from arr#选择前k个
                res.push(the_max_appear_types(k))#选择出现次数最多的类型
        return res

输入：句子a和b,句子的TF_IDF矩阵TF_IDF
输出：两个句子之间的余弦相似度
def caculate_cos(a,b,TF_IDF):
        t1=TF_IDF.find(a)#找到该句子对应的行
        t2=TF_IDF.find(b)#找到该句子对应的行
        abst1=0
        abst2=0
        mul=0
        for i in range(0,len(t1)):
                abst1+=t1[i]*t1[i]
                abst2+=t2[i]*t2[i]
                mul+=t1[i]*t2[i]
        abst1=math.sqrt(abst1)
        abst2=math.sqrt(abst2)
        return mul/(abst1+abst2)
```

3、关键代码展示

在该次实验中，我采用了文本的tf-idf值作为文本特征，调用sklearn库计算tf-idf矩阵的代码实现如下：

```
vectorizer = CountVectorizer()
transformer = TfidfTransformer()
#得到tf_idf矩阵X1
X1 = transformer.fit_transform(vectorizer.fit_transform(sentances))
#矩阵中每一列对应的特征列表
name=vectorizer.get_feature_names_out().tolist()
#将矩阵转为列表
arr=X1.toarray().tolist()
```

对于knn算法的实现，经过多次实验，我最终选取了曼哈顿距离来计算相似度，并选取k值为14，具体的代码实现如下：

```
def classify_knn(sen,arr,types,sentances):
    all=[]#记录距离和对应类型的列表
    index=sentances.index(sen)#找到对应下标
    for i in range(0,1247):
        dis=0
        #计算欧式距离
        for j in range(len(arr[i])):
            hot=arr[index][j]
            tmp=abs(hot-arr[i][j])
            dis+=tmp
        #采用元组的方式记录
        t=(dis,types[i])
        all.append(t)
    all.sort()#对距离进行从小到大排序
    cnt=[0,0,0,0,0,0]#统计前k个中每个类型出现的次数
    #k取14
    for i in range(14):
        if all[i][1]=='1':
            cnt[0]+=1
        elif all[i][1]=='2':
            cnt[1]+=1
        elif all[i][1]=='3':
            cnt[2]+=1
        elif all[i][1]=='4':
            cnt[3]+=1
        elif all[i][1]=='5':
            cnt[4]+=1
        elif all[i][1]=='6':
            cnt[5]+=1
    max=-999
    flag=-1
    #找到出现次数最多的类型
    for i in range(6):
        if cnt[i]>max:
            max=cnt[i]
            flag=i
    return flag+1
```

三、实验结果分析&&对比

实验结果

将原来的测试集test.txt和训练集train.txt对换以保证8:2的比例，选取曼哈顿距离、k=14，进行实验，实验结果如下：

```
判断: 4 正确: 4 ['cher', 'item', 'snap', 'auction']
判断: 4 正确: 2 ['griffith', 'scorn', 'withnail', 'plai']
判断: 4 正确: 4 ['lili', 'allen', 'win', 'web', 'music', 'award']
判断: 4 正确: 1 ['costner', 'appeal', 'casino', 'wrangl']
判断: 6 正确: 1 ['beyonc', 'copyright', 'claim', 'reject']
判断: 4 正确: 4 ['cohn', 'gunman', 'jail']
判断: 4 正确: 4 ['melua', 'deep', 'sea', 'gig', 'set', 'record']
判断: 5 正确: 5 ['sri', 'lanka', 'clash', 'kill', 'troop']
判断: 5 正确: 5 ['gunmen', 'kill', 'iraq', 'tv', 'raid']
判断: 6 正确: 6 ['itali', 'drop', 'sicili', 'bridg', 'plan']
判断: 5 正确: 5 ['amish', 'kill', 'school', 'demolish']
判断: 4 正确: 4 ['pamuk', 'win', 'nobel', 'literatur', 'prize']
判断: 4 正确: 4 ['madonna', 'adopt', 'bid', 'approv']
判断: 3 正确: 3 ['admit', 'uk', 'terror', 'bomb', 'plot']
判断: 4 正确: 2 ['ar', 'call', 'fat']
rate: 53.44129554655870%
PS E:\VSCODE\py>
```

但是由于特征太多，程序运行时间较慢，且很多特征对分类的影响非常小甚至会有干扰的作用，因此我采取了主成分分析算法（PCA）来进行线性降维，将3000+的特征数降维到了147，运行时间大大提升，且正确率也有所增加：

```
判断: 4 正确: 5 ['dr', 'mcdreami', 'choke', 'grei', 'scuffl']
判断: 4 正确: 4 ['poetri', 'prize', 'award']
判断: 4 正确: 4 ['la', 'vega', 'honour', 'tiger', 'pair']
判断: 4 正确: 4 ['uk', 'charact', 'boost', 'box', 'offic']
判断: 5 正确: 4 ['cher', 'item', 'snap', 'auction']
判断: 4 正确: 2 ['griffith', 'scorn', 'withnail', 'plai']
判断: 4 正确: 4 ['lili', 'allen', 'win', 'web', 'music', 'award']
判断: 4 正确: 1 ['costner', 'appeal', 'casino', 'wrangl']
判断: 6 正确: 1 ['beyonc', 'copyright', 'claim', 'reject']
判断: 4 正确: 4 ['cohn', 'gunman', 'jail']
判断: 4 正确: 4 ['melua', 'deep', 'sea', 'gig', 'set', 'record']
判断: 5 正确: 5 ['sri', 'lanka', 'clash', 'kill', 'troop']
判断: 5 正确: 5 ['gunmen', 'kill', 'iraq', 'tv', 'raid']
判断: 4 正确: 6 ['itali', 'drop', 'sicili', 'bridg', 'plan']
判断: 5 正确: 5 ['amish', 'kill', 'school', 'demolish']
判断: 4 正确: 4 ['pamuk', 'win', 'nobel', 'literatur', 'prize']
判断: 4 正确: 4 ['madonna', 'adopt', 'bid', 'approv']
判断: 3 正确: 3 ['admit', 'uk', 'terror', 'bomb', 'plot']
判断: 4 正确: 2 ['ar', 'call', 'fat']
rate: 54.65587044534413%
PS E:\VSCODE\py>
```

超参K对分类准确率影响

运用控制变量法，在使用曼哈顿距离的程序下，选取k值为10,15,20,25,30进行实验，得到的结果如下：

k值	正确率
10	54.665%
15	49.392%
20	43.724%
25	40.080%
30	38.866%

可见超参k对分类的准确率有重要的影响，经过对比，最终选取了k=10来作为最终的值。

距离计算方式对分类准确率影响

同样运用控制变量法，在参数k=14的情况下，选取四种距离函数：曼哈顿距离、欧式距离、余弦相似度、向量内积进行实验，得到的结果如下：

计算方式	正确率
曼哈顿距离	53.441%
欧式距离	47.368%
余弦相似度	46.153%
向量内积	46.153%

可见使用曼哈顿距离时效果最好。

四、创新点&&优化：

1、使用了PCA来进行降维

利用主成分分析算法（PCA）来进行线性降维，将3000+的特征数降维到了147后，运行时间大大提升，且正确率也有所增加，提高了算法的效率和分类的准确率，具体代码实现如下：

```
def eigValPct(eigVals,percentage):
    sortArray=sort(eigVals) #使用numpy中的sort()对特征值按照从小到大排序
    sortArray=sortArray[-1::-1] #特征值从大到小排序
    arraySum=sum(sortArray) #数据全部的方差arraySum
    tempSum=0
    num=0
    for i in sortArray:
        tempSum+=i
        num+=1
        if tempSum >=arraySum*percentage:
            return num

'''pca函数有两个参数，其中dataMat是已经转换成矩阵matrix形式的数据集，列表示特征;
其中的percentage表示取前多少个特征需要达到的方差占比，默认为0.9'''
def pca(dataMat,percentage=0.99):
    meanVals=mean(dataMat,axis=0)  #对每一列求平均值，因为协方差的计算中需要减去均值
    meanRemoved=dataMat-meanVals
    covMat=cov(meanRemoved,rowvar=0)  #cov()计算方差
    eigVals,eigVects=linalg.eig(mat(covMat))  #利用numpy中寻找特征值和特征向量的模块linalg中的eig()方法
    #k=eigValPct(eigVals,percentage) #要达到方差的百分比percentage，需要前k个向量
    k=134  #选取k为134
    eigValInd=argsort(eigVals)  #对特征值eigVals从小到大排序
    eigValInd=eigValInd[:-(k+1):-1] #从排好序的特征值，从后往前取k个，这样就实现了特征值的从大到小排列
    redEigVects=eigVects[:,eigValInd]   #返回排序后特征值对应的特征向量redEigVects（主成分）
    lowDDataMat=meanRemoved*redEigVects #将原始数据投影到主成分上得到新的低维数据lowDDataMat
    reconMat=(lowDDataMat*redEigVects.T)+meanVals   #得到重构数据reconMat
    return lowDDataMat,reconMat,k
```

2、进行多次实验，选取最合适的k值

为选取最合适的k值，我编写了一个测试程序，运行了很长的时间，遍历5~100的每个k值，最终选取了正确率最高的k=14来作为最终的k值。具体测试程序见code文件夹

3、运用多个距离度量函数

如上文实验结果中所示，我采用了四种不同的距离度量函数，进行多次实验，最终选取效果最好的距离度量函数，这几种距离度量的具体代码如下：

```python
#向量内积的方法
def neiji(t1,t2):
    mul=0
    for i in range(0,len(t1)):
        mul+=t1[i]*t2[i]
    return mul

#余弦相似度的方法，使用时记得将sort函数的参数reverse设置为true
def caculate_cos(t1,t2):
    abst1=0
    abst2=0
    mul=0
    for i in range(0,len(t1)):
        abst1+=t1[i]*t1[i]
        abst2+=t2[i]*t2[i]
        mul+=t1[i]*t2[i]
    abst1=math.sqrt(abst1)
    abst2=math.sqrt(abst2)
    return mul/(abst1+abst2)
```

```python
for i in range(0,1247):
    dis=0
    #计算曼哈顿距离
    for j in range(len(arr[i])):
        hot=arr[index][j]
        tmp=abs(hot-arr[i][j])
        dis+=tmp
    #采用元组的方式记录
    t=(dis,types[i])
```

五、参考资料

- [哈工大停用词表](GitHub - goto456/stopwords: 中文常用停用词表（哈工大停用词表、百度停用词表等）)