文本表示：其实就是文本的向量化问题。

向量空间模型的思想是把文档简化为特征项的权重为分量的向量表示，其中选取词作为特征项，权重用词频表示。

其主要用的是**TF-IDF算法**来计算：TF（词频）是一个词语出现的次数除以该文件的总词语数。IDF（文件频率）的方法是测定有多少文件出现过某个词，然后除以文件集里的文件数。注意这里IDF用的是逆向文件频率，即假如“汽车”一词在100份文件中出现过，总文件数是10000份，这时的逆向文件频率为log10(10000/100)=2log10(10000/100)=2，而得到的TF是0.05，此时的TF-IDF分数为0.05x2=0.1。

也即是，如果某个词在一篇文章中出现的频率TF高，并且在其他文章中很少出现，则认为这个词有很好的类别区分能力，适合用来分类；IDF表示含有该词汇的文档，比例越低IDF越大，则说明该词汇具有良好的类别区分能力。

**zip([iterable, …])函数使用：**其中iterabl 代表一个或多个迭代器，zip() 函数用于将可迭代的对象作为参数，将对象中对应的元素打包成一个个元组，然后返回由这些元组组成的列表。

如果各个迭代器的元素个数不一致，则返回列表长度与最短的对象相同，利用 \* 号操作符，可以将元组解压为列表。

>>>a = [1,2,3]

>>>b = [4,5,6]

>>>c = [4,5,6,7,8]

>>> zipped = zip(a,b) # 打包为元组的列表

[(1, 4), (2, 5), (3, 6)]

>>> zip(a,c) # 元素个数与最短的列表一致

[(1, 4), (2, 5), (3, 6)]

>>> zip(\*zipped) # 与 zip 相反，可理解为解压，返回二维矩阵式

[(1, 2, 3), (4, 5, 6)]

**# Rocchio的示例代码**

from sklearn.neighbors.nearest\_centroid import NearestCentroid

# 现在把TF-IDF转换后的结果和每条结果对应的主题编号twenty\_train.target放入分类器进行训练

clf = NearestCentroid().fit(X\_train\_tfidf, twenty\_train.target)

# 创建测试集合，这里有2条数据，每条数据一行内容，进行向量化和tf-idf转换

docs\_new = ['God is love', 'OpenGL on the GPU is fast']

X\_new\_counts = count\_vect.transform(docs\_new)

X\_new\_tfidf = tfidf\_transformer.transform(X\_new\_counts)

# 预测

predicted = clf.predict(X\_new\_tfidf)

# 打印结果

for doc, category in zip(docs\_new, predicted):

print('%r => %s' % (doc, twenty\_train.target\_names[category]))

answer是测试集；

[对复旦语料库的预处理](https://www.cnblogs.com/zhangchaoyang/articles/2232205.html)

在感谢复旦语料库整理人员辛勤劳动的同时，也要指出其工作上的瑕疵。

1. 采用了gbk编码而不是UTF-8，这导致大多Linux用户不能直接使用。
2. 语料库包含训练集和测试集，分别包含9000多个文档，却分别有近1500个文档是重复的。
3. 训练集和测试集中的C35-Law中的部分文件是已经经过分词处理了的（分词结果很差），且部分又不是采用的GBK编码（这给编码转换工作带来麻烦）。
4. 有些文章只有文章头部，而没有实际的内容。

step1.下载复旦语料库的[训练集](http://www.nlp.org.cn/docs/download.php?doc_id=294)和[测试集](http://www.nlp.org.cn/docs/download.php?doc_id=295)。解压。

step2.分别删除train和answer文件夹下的C35-Law文件夹。

step3.使用FindDupFile工具分别找到train和answer文件夹的重复文件，删除之。

step4.使用[iconv.exe](http://www.w3cool.com/2007/12/18/iconv_for_windows.html)工具把文件从gbk编码转换到utf-8编码。

step5.为提高文本分类精度，删除所有长度小于500的文档。

[+ View Code](https://www.cnblogs.com/zhangchaoyang/articles/2232205.html)

step6.删除文档中每行开头和结尾的空格，把文档合并为一行。同时给每个文档赋予一个全局编号，该编号除要写入文件名中外，还要写入文档内容的开头。用Java实现非常简单：

[+ View Code](https://www.cnblogs.com/zhangchaoyang/articles/2232205.html)

step7.最后删除文档数小于100的分类。

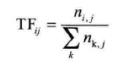
最后剩余的语料情况是：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 训练集train | | begin | end | 测试集answer | | begin | end |
| C39-Sports | 1070 | 0 | 1069 | C39-Sports | 1066 | 0 | 1065 |
| C7-History | 440 | 1070 | 1509 | C7-History | 447 | 1066 | 1512 |
| C3-Art | 513 | 1510 | 2022 | C3-Art | 508 | 1513 | 2020 |
| C32-Agriculture | 816 | 2023 | 2838 | C32-Agriculture | 822 | 2021 | 2842 |
| C31-Enviornment | 750 | 2839 | 3588 | C31-Enviornment | 741 | 2843 | 3583 |
| C38-Politics | 756 | 3589 | 4344 | C38-Politics | 758 | 3584 | 4341 |
| C34-Economy | 1392 | 4345 | 5736 | C34-Economy | 1404 | 4342 | 5745 |
| C11-Space | 473 | 5737 | 6209 | C11-Space | 463 | 5746 | 6208 |
| C19-Computer | 986 | 6210 | 7195 | C19-Computer | 984 | 6209 | 7192 |
| Total | 7196 |  |  | Total | 7193 |  |  |

**2.2.5 权重策略：TD-IDF方法**

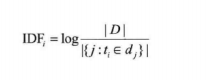
　　含义：如果某个词或短语在一篇文章中出现的频率越高，并且在其他文章中很少出现，则认为此词或者短语具有很好的类别区分能力，适合用来分类。

　　词频（Term Frequency，TF）指的是某一个给定的词语在该文件中出现的频率。这个数字是对词数（Term Count）的归一化，以防止它偏向长的文件。对于在某一特定文件里的词语来说，它的重要性可以表示为：



　　其中，分子是该词在文件中出现的次数，分母是文件中所有字词的出现次数之和：

　　逆向文件频率（Inverse Document Frequency，IDF）是一个词语普遍重要性的度量。某一特定词语的IDF，可以由总文件数目除以包含该词语的文件的数目，再将得到的商取对数得到：



其中|D|：语料库中的文件总数。

j：包含词语的文件数目。如果该词语不在语料库中，就会导致分母为零，因此一般情况下使用1+j作为分母

　　TF-IDF = TF \*IDF

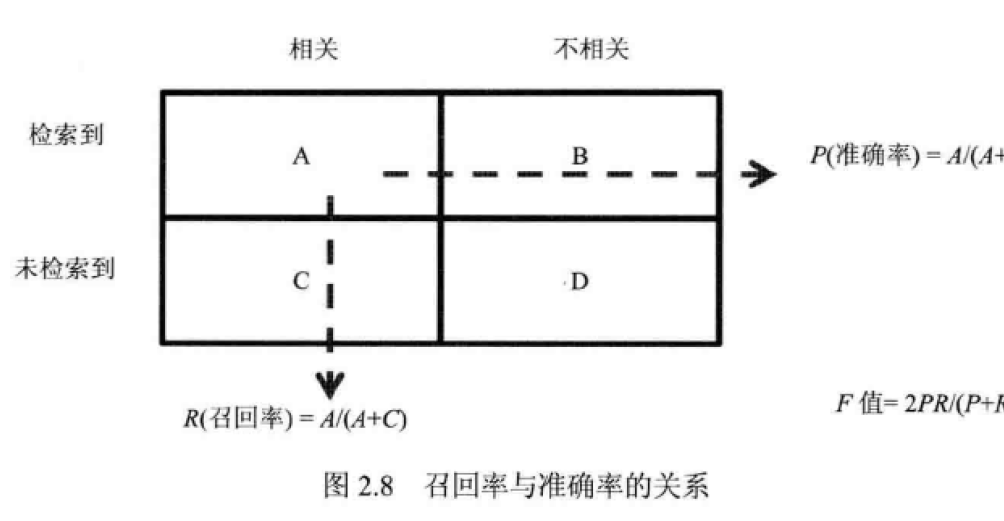
**2.2.7 分类结果评估**

　　（1）召回率（Recall Rate，也叫查全率）：是检索出相关文档数和文档库中所有相关文档的比率，衡量的是检索系统的查全率

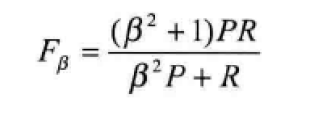
　　　　召回率（Recall） = 系统检索到的相关文件/系统所有相关的文件的总数

　　（2）准确率（Precision，也成称为精度）：是检索出的相关文档数与检索出的文档总数的比率，衡量的是检索系统的查准率。

　　　　准确率（Precision） = 系统检索到的相关文件/系统所有检索到的文档总数

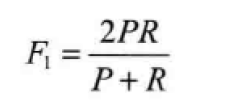


　　（3）Fβ-Mesure(又称为F-Score）：是机器学习领域常用的评价标准，计算公式：



　　其中，β是参数，p是准确率，R是召回率

　　当β=1时，就是最常见的F1-Mesure了：



>>> lst = [1,2,3]

>>> lst.append(["qiwsir","github"])

>>> lst

[1, 2, 3, ['qiwsir', 'github']] #append的结果

>>> len(lst)

4

>>> lst2 = [1,2,3]

>>> lst2.extend(["qiwsir","github"])

>>> lst2

[1, 2, 3, 'qiwsir', 'github'] #extend的结果

>>> len(lst2)

5

append是整建制地追加，extend是个体化扩编。

     extend将它的参数视为 list，extend的行为是把这两个list接到一起，append是将它的参数视为element，作为一个整体添加上去的。  
List里可以有任意的数据类型，所以，要分清这俩函数的区别。

# [sklearn.metrics](http://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics).precision\_score

sklearn.metrics.**precision\_score**（y\_true，y\_pred，labels = None，pos\_label = 1，average ='binary'，sample\_weight = None ）[[source]](https://github.com/scikit-learn/scikit-learn/blob/bac89c2/sklearn/metrics/classification.py#L1173)

计算精度

精度是比其中为真阳性的数量和假阳性的数量。精确度直观地说是分类器不能标记为阴性样本的能力。tp / (tp + fp)tpfp

最佳值为1，最差值为0。

阅读[用户指南中的更多内容](http://scikit-learn.org/stable/modules/model_evaluation.html" \l "precision-recall-f-measure-metrics)。

|  |  |
| --- | --- |
| **参数：** | **y\_true ： *1d数组，或标签指示符数组/稀疏矩阵***  基本事实（正确）目标值。  **y\_pred ： *1d数组，或标签指示符数组/稀疏矩阵***  分类器返回的估计目标。  **标签 ： *列表，可选***  要包含的标签集，以及它们的顺序if 。可以排除数据中存在的标签，例如计算忽略多数阴性类的多类平均值，而数据中不存在的标签将导致宏观平均值中的0个分量。对于多标记目标，标签是列索引。默认情况下，所有的标签，并 在排序顺序使用。average != 'binary'average is Noney\_truey\_pred  *在版本0.17中更改：*参数标签针对多类问题进行了改进。  **pos\_label ： *str或int，默认为1***  报告if average='binary'和数据是二进制的类。如果数据是多类或多标签，则会被忽略; 设置labels=[pos\_label]并仅报告该标签的分数。average != 'binary'  **average ： *string，[None，'binary'（默认），'micro'，'macro'，'samples'，'weighted']***  多类/多标记目标需要此参数。如果None，则返回每个班级的分数。否则，这将确定对数据执行的平均类型：  **'binary'：**  仅报告由指定的类的结果pos\_label。仅当targets（y\_{true,pred}）是二进制时才适用。  **'micro'：**  通过计算总真阳性，假阴性和误报来全球计算指标。  **'macro'：**  计算每个标签的指标，找出它们的未加权平均值。这不会考虑标签不平衡。  **'weighted'：**  计算每个标签的指标，并找到按支持加权的平均值（每个标签的真实实例数）。这会改变“宏观”以解决标签不平衡问题; 它可能导致F分数不在精确度和召回之间。  **'samples'：**  计算每个实例的指标，并找出它们的平均值（仅对于不同的多标记分类有意义 **[accuracy\_score](http://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html" \l "sklearn.metrics.accuracy_score" \o "sklearn.metrics.accuracy_score)**）。  **sample\_weight ： *shape = array = [n\_samples]，可选***  样品重量。 |
| **返回：** | **precision ： *float（如果average不是None）或float数组，shape = [n\_unique\_labels]***  二进制分类中正类的精度或多类任务的每个类的精度的加权平均值。 |

**例子**

>>>

**>>> 从** **sklearn.metrics**  **导入** precision\_score

**>>>** y\_true = [ 0 ， 1 ， 2 ， 0 ， 1 ， 2 ]

**>>>** y\_pred = [ 0 ， 2 ， 1 ， 0 ， 0 ， 1 ]

**>>>** precision\_score （y\_true ， y\_pred ， average = 'macro' ）

0.22 ...

**>>>** precision\_score （y\_true ， y\_pred ， average = 'micro' ）

0.33 ...

**>>>** precision\_score （y\_true ， y\_pred ， average = 'weighted' ）

**...**

0.22 ...

**>>>** precision\_score （y\_true ， y\_pred ， average = **None** ）

array （[0.66 ...，0.，0。]）

python中时间日期格式化符号：

%y 两位数的年份表示（00-99）

%Y 四位数的年份表示（000-9999）

%m 月份（01-12）

%d 月内中的一天（0-31）

%H 24小时制小时数（0-23）

%I 12小时制小时数（01-12）

%M 分钟数（00=59）

%S 秒（00-59）

%a 本地简化星期名称

%A 本地完整星期名称

%b 本地简化的月份名称

%B 本地完整的月份名称

%c 本地相应的日期表示和时间表示

%j 年内的一天（001-366）

%p 本地A.M.或P.M.的等价符

%U 一年中的星期数（00-53）星期天为星期的开始

%w 星期（0-6），星期天为星期的开始

%W 一年中的星期数（00-53）星期一为星期的开始

%x 本地相应的日期表示

%X 本地相应的时间表示

%Z 当前时区的名称

%% %号本身

正则表达式是一种用来匹配字符串的强有力的武器。它的设计思想是用一种描述性的语言来给字符串定义一个规则，凡是符合规则的字符串，我们就认为它“匹配”了，否则，该字符串就是不合法的：

用\d可以匹配一个数字，\w可以匹配一个字母或数字，所以：

* '00\d'可以匹配'007'，但无法匹配'00A'；
* '\d\d\d'可以匹配'010'；
* '\w\w\d'可以匹配'py3'；

.可以匹配任意字符，所以：

* 'py.'可以匹配'pyc'、'pyo'、'py!'等等。

要匹配变长的字符，在正则表达式中，用\*表示任意个字符（包括0个），用+表示至少一个字符，用?表示0个或1个字符，用{n}表示n个字符，用{n,m}表示n-m个字符：

来看一个复杂的例子：\d{3}\s+\d{3,8}。

我们来从左到右解读一下：

1. \d{3}表示匹配3个数字，例如'010'；
2. \s可以匹配一个空格（也包括Tab等空白符），所以\s+表示至少有一个空格，例如匹配' '，' '等；
3. \d{3,8}表示3-8个数字，例如'1234567'。

综合起来，上面的正则表达式可以匹配以任意个空格隔开的带区号的电话号码。

如果要匹配'010-12345'这样的号码呢？由于'-'是特殊字符，在正则表达式中，要用'\'转义，所以，上面的正则是\d{3}\-\d{3,8}。

但是，仍然无法匹配'010 - 12345'，因为带有空格。所以我们需要更复杂的匹配方式。

### 进阶

要做更精确地匹配，可以用[]表示范围，比如：

* [0-9a-zA-Z\\_]可以匹配一个数字、字母或者下划线；
* [0-9a-zA-Z\\_]+可以匹配至少由一个数字、字母或者下划线组成的字符串，比如'a100'，'0\_Z'，'Py3000'等等；
* [a-zA-Z\\_][0-9a-zA-Z\\_]\*可以匹配由字母或下划线开头，后接任意个由一个数字、字母或者下划线组成的字符串，也就是Python合法的变量；
* [a-zA-Z\\_][0-9a-zA-Z\\_]{0, 19}更精确地限制了变量的长度是1-20个字符（前面1个字符+后面最多19个字符）。

A|B可以匹配A或B，所以(P|p)ython可以匹配'Python'或者'python'。

^表示行的开头，^\d表示必须以数字开头。

$表示行的结束，\d$表示必须以数字结束。

你可能注意到了，py也可以匹配'python'，但是加上^py$就变成了整行匹配，就只能匹配'py'了。

re.match(r'^\d{3}\-\d{3,8}$', '010-12345')

match()方法判断是否匹配，如果匹配成功，返回一个Match对象，否则返回None。常见的判断方法就是：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 处理前 | | | | 删除重复文档后 | | | |  |
| 训练集train | | 测试集answer | | 训练集train | | 测试集answer | |  |
| C3-Art | 740 | C3-Art | 742 | C3-Art | 548 | C3-Art | 552 | C3艺术 |
| C4-Literature | 33 | C4-Literature | 34 | C4-Literature | 33 | C4-Literature | 34 | C4文学 |
| C5-Education | 59 | C5-Education | 61 | C5-Education | 58 | C5-Education | 59 | C5教育 |
| C6-Philosophy | 44 | C6-Philosophy | 45 | C6-Philosophy | 40 | C6-Philosophy | 42 | C6哲学 |
| C7-History | 466 | C7-History | 468 | C7-History | 465 | C7-History | 468 | C7历史 |
| C11-Space | 640 | C11-Space | 642 | C11-Space | 506 | C11-Space | 500 | C11航空 |
| C15-Energy | 32 | C15-Energy | 33 | C15-Energy | 30 | C15-Energy | 33 | C15能量 |
| C16-Electronics | 27 | C16-Electronics | 28 | C16-Electronics | 26 | C16-Electronics | 27 | C16电子学 |
| C17-Communication | 25 | C17-Communication | 27 | C17-Communication | 25 | C17-Communication | 27 | C17通信 |
| C19-Computer | 1357 | C19-Computer | 1358 | C19-Computer | 1022 | C19-Computer | 1023 | C19计算机 |
| C23-Mine | 33 | C23-Mine | 34 | C23-Mine | 33 | C23-Mine | 31 | C23矿 |
| C29-Transport | 57 | C29-Transport | 59 | C29-Transport | 57 | C29-Transport | 59 | C29交通 |
| C31-Enviornment | 1217 | C31-Enviornment | 1218 | C31-Enviornment | 805 | C31-Enviornment | 801 | C31环境 |
| C32-Agriculture | 1021 | C32-Agriculture | 1022 | C32-Agriculture | 847 | C32-Agriculture | 845 | C32农业 |
| C34-Economy | 1600 | C34-Economy | 1601 | C34-Economy | 1471 | C34-Economy | 1468 | C34经济 |
| C35-Law | 51 | C35-Law | 52 | C35-Law | 51 | C35-Law | 52 | C35法律 |
| C36-Medical | 51 | C36-Medical | 53 | C36-Medical | 51 | C36-Medical | 53 | C36医学 |
| C37-Military | 74 | C37-Military | 76 | C37-Military | 74 | C37-Military | 76 | C37军事 |
| C38-Politics | 1024 | C38-Politics | 1026 | C38-Politics | 1010 | C38-Politics | 1013 | C38政治 |
| C39-Sports | 1253 | C39-Sports | 1254 | C39-Sports | 1204 | C39-Sports | 1205 | C39体育 |

step5.为提高文本分类精度，删除所有长度小于500的文档：

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

public class DelSmallText {

    public void delSmallText(File srcFile){

        if(srcFile.isDirectory()){

            File[] childFiles=srcFile.listFiles();

            for(File child:childFiles){

                delSmallText(child);

            }

        }

        else if(srcFile.isFile()){

            StringBuffer content=new StringBuffer();

            try{

                FileReader fr=new FileReader(srcFile);

                BufferedReader br=new BufferedReader(fr);

                String line;

                while((line=br.readLine())!=null){      //readLine()并不读取末尾的换行符

                    content.append(line.trim());    //只去除行首尾的空格，因为去除空格的话连续的英文单词合并到了一起

                }

                br.close();

                if(content.toString().length()<500){

                    srcFile.delete();

                }

            }catch (Exception e){

                e.printStackTrace();

            }

        }

    }

    public static void main(String[] args){

        DelSmallText inst=new DelSmallText();

        File file=new File("/home/orisun/master/fudan\_corpus/train");

        inst.delSmallText(file);

    }

}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 删除重复文档和<500文档后 | | | |  |
| 训练集train | | 测试集answer | |  |
| C3-Art | 512 | C3-Art | 508 | C3艺术 |
| C7-History | 439 | C7-History | 447 | C7历史 |
| C11-Space | 473 | C11-Space | 463 | C11航空 |
| C19-Computer | 986 | C19-Computer | 984 | C19计算机 |
| C31-Enviornment | 749 | C31-Enviornment | 741 | C31环境 |
| C32-Agriculture | 816 | C32-Agriculture | 820 | C32农业 |
| C34-Economy | 1392 | C34-Economy | 1402 | C34经济 |
| C38-Politics | 751 | C38-Politics | 756 | C38政治 |
| C39-Sports | 1069 | C39-Sports | 1064 | C39体育 |

**HW3 Report**

Rohit Gaikwad

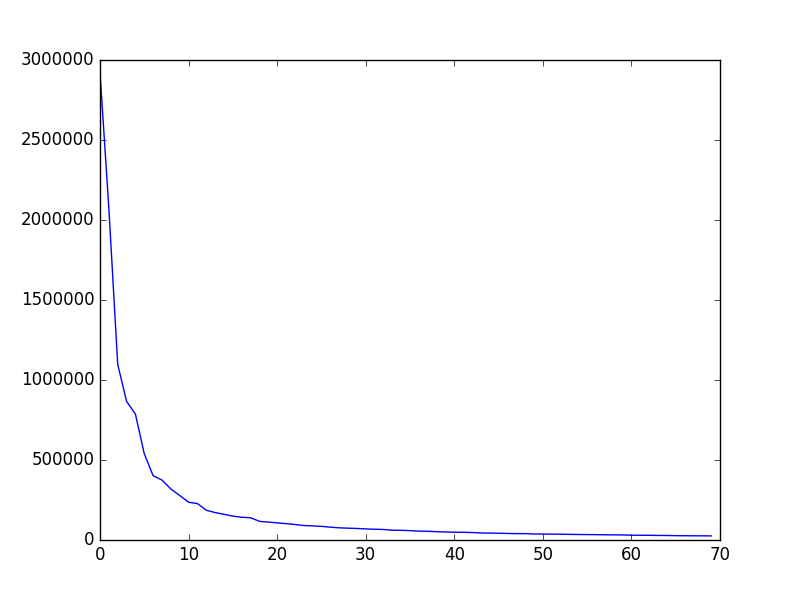
1001236285

Task 1: Classification using PCA+1NN with 5-fold cross validation

For PCA I took the following steps:

* Centerized the images
* Calculated Covariance matrix
* Calculated eigen values and eigen vectors
* Sorted the eigen values in descending order and then sorted the eigen vectors by the same index
* Took the first 70 principal components and plotted the data into new dimensions.

I used value for k=70 as I saw that as the best value by experimenting and by looking at the following eigen values graph.

For 1NN I took the following steps:

* Calculated the euclidean distance for each test image with every train image
* Sorted the distance in ascending order and took the 1st one as the result
* Compare the result with original class and computed the accuracy

After the 5-fold cross validation, the average accuracy for **PCA + 1NN** was ***96.75%***

Task 2: Repeating Task 1 after resizing the images from 112 x 92 to 56 x 46

The average accuracy after **Resizing Images + PCA + 1NN** was ***96.0%***

Comparing this with the original dimensions accuracy, there does not seem to be much change in the accuracy.

Task 3: Classification using LDA+1NN with 5-fold cross validation

For LDA I took the following steps:

* Calculated class wise mean vectors
* Calculated in-class scatter matrix
* Calculated between-class scatter matrix
* Calculated eigen values and eigen vectors
* Sorted the eigen values in descending order and then sorted the eigen vectors by the same index
* Took the first 70 principal components and plotted the data into new dimensions.

After the 5-fold cross validation, the average accuracy for **LDA + 1NN** was ***96.0%***

Task 4: Classification using PCA+LDA+1NN with 5-fold cross validation

After the 5-fold cross validation, the average accuracy for **PCA + LDA + 1NN** was ***98.0%***

Comparing this with previous results we can see that PCA before LDA helps increasing the accuracy of 1NN.

Task 5: Classification using SVM with 5-fold cross validation

For SVM I took the following steps:

* Generated Gramian Matrix
* Calculated values of P, q, A, b, G, h and solved quadratic equation
* Determined support vectors
* Calculatied bias and weights
* Prediction was done using one-vs-rest approach

After the 5-fold cross validation, the average accuracy for **SVM** was ***97.4%***

Task 6: Classification using PCA+SVM with 5-fold cross validation

After the 5-fold cross validation, the average accuracy for **PCA + SVM** was ***92.7%***

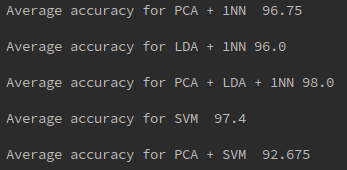
Comparing this with previous SVM results we can see that PCA decreases the accuracy of SVM drastically.

Answering the question whether KNN or SVM is more sensitive to high dimensionality of data.

Answer: SVM is more sensitive to high dimensionality as reducing the dimensions through PCA affects negatively to SVM, on the other hand PCA and LDA helps KNN to increase its accuracy with lower dimensions.

This is because PCA retrieves the features with highest information gain and discards the other, but SVM does not work feature wise, it relies on data-space dimensions which is reduced by PCA.

This is a snapshot of my results :-



HW3报告

罗希特盖克瓦德

十亿零一百二十三万六千二百八十五

任务1：使用五重交叉验证的PCA+1NN进行分类

对于PCA，我采取了以下步骤：

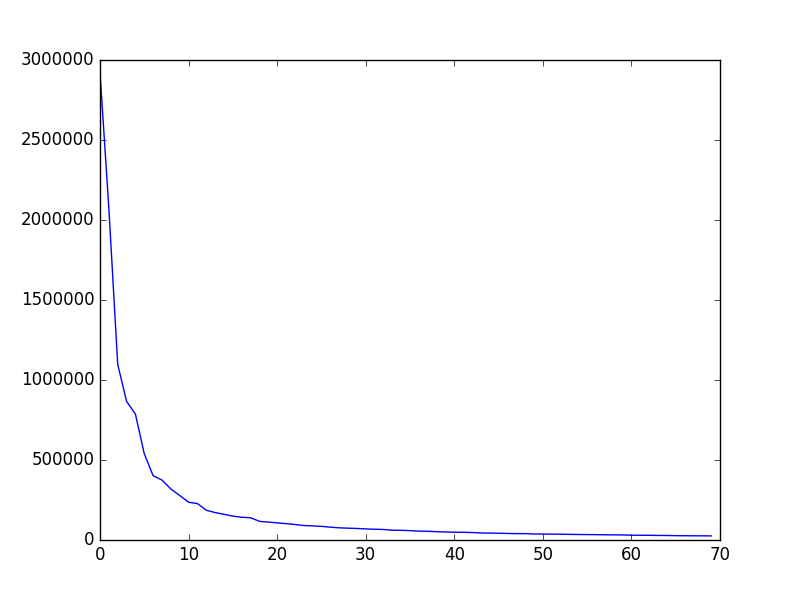
●将图像集中

•计算协方差矩阵

•计算的特征值和特征向量

·按降序对特征值进行排序，然后按照相同的索引对特征向量进行排序

·取了前70个主要组成部分并将数据绘制成新的维度。

我使用了k=70的值，因为我通过实验和观察下面的特征值图看到了这个值。

对于1NN，我采取以下步骤：

●针对每个火车图像计算每个测试图像的欧几里德距离

·按升序对距离进行排序，得到第一个结果

●将结果与原始类进行比较，并计算精度

经5次交叉验证，PCA+1NN的平均准确率为96.75%。

任务2：在将图像从112×92调整到56×46之后，重复任务1

图像缩放+PCA+1NN的平均准确率为96.0%。

与原尺寸精度相比，精度似乎没有太大变化。

任务3：使用LDA+1NN进行5次交叉验证的分类

对于LDA，我采取了以下步骤：

●计算类明智平均向量

·计算类内散射矩阵

●计算类间散布矩阵

•计算的特征值和特征向量

·按降序对特征值进行排序，然后按照相同的索引对特征向量进行排序

·取了前70个主要组成部分并将数据绘制成新的维度。

经5次交叉验证，LDA+1NN的平均准确率为96.0%。

任务4：使用PCA+LDA+1NN进行5次交叉验证的分类

经过5次交叉验证，PCA+LDA+1NN的平均准确率为98.0%。

通过与以往结果的比较，可以看出，LDA之前的PCA有助于提高1NN的精度。

任务5：使用具有5重交叉验证的SVM进行分类

对于SVM，我采取了以下步骤：

·生成的格雷米矩阵

·P、q、A、b、G、h的计算值和求解的二次方程

·确定的支持向量

•计算的偏差和权重

•使用一对一的方法进行预测

经过5次交叉验证，支持向量机的平均准确率为97.4%。

任务6：使用五重交叉验证的PCA+SVM进行分类

经过5次交叉验证，主成分分析+支持向量机的平均准确率为92.7%。

与已有的SVM结果相比，PCA算法显著降低了SVM的精度。

回答了KNN和SVM是否对数据的高维更加敏感的问题。

答：支持向量机对高维数据更为敏感，因为通过主成分分析降低维数对支持向量机有负面影响，而主成分分析和LDA有助于KN在较低维数下提高精度。

这是因为PCA检索信息增益最大的特征而丢弃了其他特征，但是SVM并不明智地处理特征，它依赖于PCA所减少的数据空间维数。

这是我的结果快照：-

