**华中科技大学计算机科学与技术学院**

**机器学习报告**



专 业： 计算机科学与技术

班 级： 计卓1801

学 号： U201814472

姓 名： 江军

成 绩：

指导教师： 何琨

**完成日期： 2020年 6 月 20 日**

## 实验题目：人脸识别

1.实验背景

人脸识别功能包括人脸检测(从给定图片中用矩形框框住人脸位置，并裁剪出 人脸照片)、人脸特征提取(输入人脸照片，输出是高维矢量特征)、人脸匹配 (通常用 K-NN 算法，找出待查询人脸的最近邻居，在具体应用中，一般是 1-NN)。 可利用现有开源框架，完成一个人脸识别系统，能够在自己台式机或笔记本上实现

实时人脸识别。

2.实验要求

1 可以调整人脸库中单个人注册的人脸照片数;

2 采用不同距离函数(L2-distance)或和核函数(余弦函数)的设置

3 测试不同参数设置情况下，参数变化对人脸识别系统影响。

3.评测标准

分类正确率。

## 实验数据以及预处理

1.实验数据

经过搜索，我们这里使用中国科学院自动化研究所开源的人脸数据库<http://www.idealtest.org/dbDetailForUser.do?id=9>

数据结构为000～499一共500个人，每个人的文件夹下有5张自身在不同角度带着不同面部饰品的照片。文件目录如下

…

…

…

├── 498

│   ├── 498\_0.bmp

│   ├── 498\_1.bmp

│   ├── 498\_2.bmp

│   ├── 498\_3.bmp

│   └── 498\_4.bmp

└── 499

├── 499\_0.bmp

├── 499\_1.bmp

├── 499\_2.bmp

├── 499\_3.bmp

└── 499\_4.bmp

2.数据处理

Bmp文件占用空间过大，我们将文件格式转化为jpg格式。同时，将5张照片中的4张随机划分为训练集，1张划分为测试集。处理后的文件目录如下

└── TEST

├── 000

│   └── 000\_1.jpg

├── 001

│   └── 001\_2.jpg

├── 002

│   └── 002\_0.jpg

├── 003

│   └── 003\_1.jpg

………..

………..

└── TRAIN

├── 000

│   ├── 000\_0.jpg

│   ├── 000\_2.jpg

│   ├── 000\_3.jpg

│   └── 000\_4.jpg

├── 001

│   ├── 001\_0.jpg

│   ├── 001\_1.jpg

│   ├── 001\_3.jpg

│   └── 001\_4.jpg

├── 002

│   ├── 002\_1.jpg

│   ├── 002\_2.jpg

│   ├── 002\_3.jpg

│   └── 002\_4.jpg

├── 003

│   ├── 003\_0.jpg

……………

为了方便起见，这里我们将文件夹名认定为人名，例如000\_1.jpg对应的人名为000。

## 三、算法设计

**1. 使用数据结构以及库**

每张图片通过face\_recognize进行人脸检测，特征提取后转化为128维的向量。人脸匹配模块对这个向量进行knn处理，因为knn算法之前作业已经实现，较为简单，这里使用skleearn的neighbors模块进行处理，将处理的好的模型存盘方便检查。

**2. 核心算法设计**

实验的核心算法是knn使用的距离函数，我使用了以下这些距离函数

1. 欧式距离
2. 曼哈顿距离
3. 切比雪夫距离
4. 高斯核函数
5. 拉普拉斯核函数
6. 多项式核函数
7. 线性核函数

这些算法的具体实现如下

def chebyshev(X, Y):

res = np.max(np.abs(X - Y))

return res

def manhattan(X, Y):

res = np.sum(np.abs(X - Y))

return res

def euclidean(X, Y):

res = np.sum((X - Y) \*\* 2) \*\* 0.5

return res

def rbf(X, Y):

tmp = np.linalg.norm(X - Y) \*\* 2

return np.exp(-tmp / (2 \* rbf\_para \*\* 2))

def laplace(X, Y):

tmp = np.linalg.norm(X - Y)

return np.exp(-tmp / laplace\_para)

def poly(X, Y):

tmp = np.dot(X, Y)

return tmp \*\* poly\_para

def lin(X, Y):

tmp = np.dot(X, Y)

return tmp

def rbf\_kernel(X, Y):

method = rbf

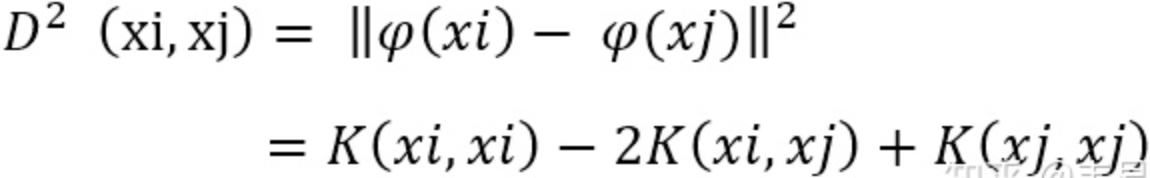
tmp = method(X, X) - 2 \* method(X, Y) + method(Y, Y)

return tmp \*\* 0.5

**3. 核心算法分析**

常见距离函数部分在第一次knn作业里已经完成，这里略过。这里主要讨论核函数。

核函数常常搭配svm一起出现，但核函数本质上是一个快速计算低维矢量映射到高纬矢量的内积的优化算法，knn在计算距离的时候可以通过两个矢量的内积来表示，所以我们利用高维空间中两点的距离公式：



其中K为我们选择的核函数。

可以看出，通过核函数，我们可以直接计算高维空间中的欧式距离，避免将低维映射到高维的转化以及高维度运算较高的复杂度。我们采用了较常见的几个核函数：线性核函数，多项式核函数，高斯核函数，拉普拉斯核函数。

## 四、实验环境与平台

1. 软件环境

python版本：python3.8.2-64bit。

开发工具：pycharm 2020.1，

操作系统：Archlinux

2. 硬件环境

CPU: AMD 3700X

Memory: 32GB

Graphics: NVIDIA GeForce RTX 2070

## 五、程序实现

1.preData.py

这是预处理部分，划分训练集和测试集，并将bmp格式改为jpg格式。

trainDir = './data/TRAIN'

testDir = './data/TEST'

dataDir = './64\_CASIA-FaceV5'

data = list()

for classDir in os.listdir(dataDir):

label = classDir

classDir = os.path.join(dataDir, classDir)

if not os.path.isdir(classDir):

continue

imgPaths = list()

for imgPath in [os.path.join(classDir, f) for f in os.listdir(classDir) if

re.match(r'.\*\.(jpg|jpeg|png|bmp)', f, flags=re.I)]:

imgPaths.append(imgPath)

random.shuffle(imgPaths)

data.append([label, imgPaths])

Q = queue.Queue()

for label, path in data:

Q.put([label, path])

核心代码如上，具体可见提交程序。

2.face\_recognition\_knn.py

这是这次实验的核心，knn部分。考虑到人脸识别以及knn训练性能要求较大，我们使用pool进行多线程操作，将label和图片到路径组合,append入tasks数组，作为train\_worker的参数：

train\_dir = './data/TRAIN'

for class\_dir in listdir\_nohidden(train\_dir):

if not os.path.isdir(os.path.join(train\_dir, class\_dir)):

continue

for img\_path in image\_files\_in\_folder(os.path.join(train\_dir, class\_dir)):

tasks.append((img\_path, verbose, class\_dir))

然后通过pool.starmap,将返回图片通过face\_recognize库生成的128维数组以及该图片的标签，输入KNeighborsClasssifier进行训练，保存训练的结果

for (class\_dir, encoding) in pool.starmap(train\_worker, tasks):

if (class\_dir == -1):

continue

X.append(encoding)

y.append(class\_dir)

测试时也使用pool进行多线程操作,将测试图片的label和路径append入tasks数组，作为test\_worker的参数。将test\_worker返回的预测label与实际label进行比对，得到正确次数数组：

for (label, predictions) in pool.starmap(test\_worker, tasks):

test\_result.append(label == predictions[0][0])

result\_labels.append(predictions[0][0])

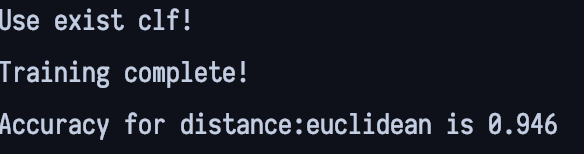
通过对result数组进行运算，得到准确率

accuracy = np.mean(test\_result)

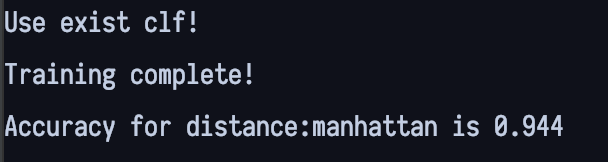
## 六、实验结果与分析

实验结果：

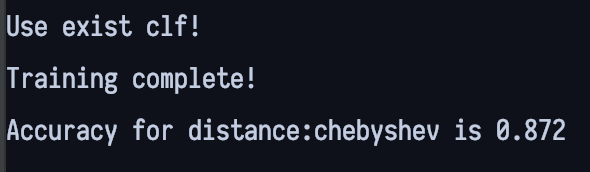
1. 欧式距离：0.946



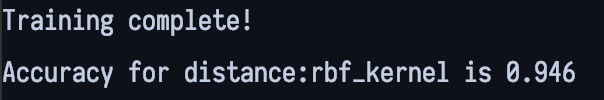
1. 曼哈顿距离：0.944



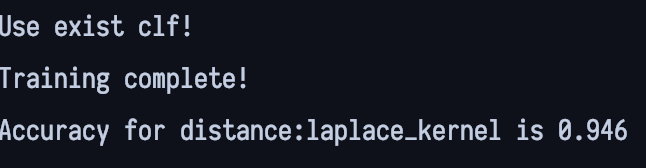
1. 切比雪夫距离：0.872



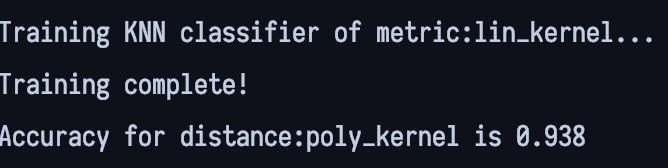
1. 高斯核函数：0.946（参数为0.2）



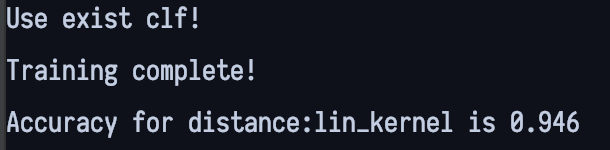
1. 拉普拉斯核函数：0.946



1. 多项式核函数：0.938



1. 线性核函数：0.946



结果分析：

对于常见距离函数，欧式距离的表现好于其他距离，切比雪夫距离的表现最差。

对于核函数，线性核函数本质上与欧式距离等价，所以准确率相等。高斯核函数在参数0.1~1的范围内都能保持0.946的准确率，低于0.1的时候准确率出现断崖式下跌。欧拉函数参数取值和高斯核函数基本一致，效果也一致。多项式核函数在参数取3～5的时候效果较好，准确率为0.938。

## 八、实验小结

这次测试了在knn中运用核函数以及常见距离函数进行人脸识别，发现核函数在人脸识别的时候效果与直接取欧式距离效果十分接近。这次实验我发现自己对于参数的选取还是有很大的盲点，下次要对如何选取核函数参数进行进一步学习。这样手动选取参数的效率以及稳定性还是太糟糕了。

这次实验对于机器学习的knn算法的各种衍生有了更深入的理解，学习了sklearn等库的使用，可以说收益颇丰。