□ □ python □ □ □ □ □ □

□ □ □ □ : love you three

□ □ □ □ : □ □ □ , □ □ , □ □ □

□ □ □ □ : □ □

9th/6/2019

□ □

0 □ □ □ □ □ □ □

1 □ □ □ □ □ □ □ □ □

1.1 □ □ □ □

1.2 □ □ □ □

1.2.1 pandas

1.2.2 NumPy

1.2.3 Matplotlib

1.2.4 PIL

1.2.5 os

2 □ □ □ □

2.1 □ □ □

2.2 □ □ □ □ □ □ □ □

2.3 PCA □ □ □ □

2.4 KNN □ □

2.5 □ □ □ □

2.6 □ □ □ □ □ □ □ □

2.7 □ □ softmax □ □ □ □ □ □ □

3 □ □ 、□ □ □ □ □

4 □ □

5 □ □

0□ □ □ □ □ □ □

由于在程序设计课上老师对人脸识别介绍的启发，我们小组试图通过学习和对 python 知识的应用，自己建立一个人脸识别的模型。在同学的推荐下，我们选择了 Faces94 数据集作为我们的训练与检验模型的数据来源。该数据库中主要有 153*20=3060 张，大小为 200*180 的图片，包含了男性和女性的图像。

我们小组尝试在 Faces94 数据集上利用 python3 实现人脸识别模型，使得该模型能够在给一张图片后输出是哪个人，并通过测试统计在测试集上的整体识别率。

1□ □ □ □ □ □ □ □ □

1.1□ □ □ □

利用 python3 实现人脸识别模型，并统计整体识别率，为解决该问题，主要需要将问题分为四个部分。

1.数据获取与预处理

为了后续建立和测试评价模型，需要将 Faces94 数据集中的数据划分为训练集和测试集，各随机取数据集中的一半数据，以分别用来训练模型，和测试统计模型的识别率。

2.建模与代码实现

对人脸识别问题进行建模，利用数学公式来表达出如何识别出是哪一个人的面部。我们使用 KNN 来建模，即对于图片 X_i 与数据集中其他图片 X_j(j 不等于 i)，判断图片 X_i 属于哪一类，可以求使得(X_i - X_j)^2 最小的 j，X_i 则判别为和这个 X_j 的类别一样，即为 K_j 所对应的人为同一人。在实现代码的过程中，我们尽可能不调用现成的库，若库中自带了我们选取的方法的函数，则作为实现代码的参考。实现代码后，则利用训练集的数据对该模型进行训练。

3.模型测试与评价

模型训练后，利用测试集中的数据，对模型进行测试与评价。主要检验：给出一张图片后该模型是否能够输出其是哪一个人；在测试集上，该模型的识别率是多少？能否调整超参数使识别率尽可能高。

4.思考总结

在建立完成该模型的基础上，分析该模型的优缺点，分析其适用与不适用情况，并提出解决方案与可以拓展提升的地方。

1.2 □ □ □ □

该模型的代码实现与测试过程中，我们主要应用到了 3 个库，分别为 pandas，NumPy，Matplotlib。

1.2.1 pandas

pandas 主要用于数据分析和建模，填补 python 在这一方面的空白，使得在 python 中更方便地执行整个数据分析工作流程。利用 pandas 在读取写入数据、集成索引数据等功能，我们可以更便捷地完成对 Faces94 数据集的获取和预处理操作，便于后续步骤的实施。我们主要通过 http://pandas.pydata.org/学习相关的知识。

1.2.2 NumPy

由于在建模和代码实现过程中，我们需要运用到很多数学方法，因此我们选择了 NumPy 作为应用的库。NumPy(Numerical Python) 是 Python 语言的一个扩展程序库，支持大量的维度数组与矩阵运算，此外也针对数组运算提供大量的数学函数库。NumPy 是一个运行速度非常快的数学库，主要用于数组计算。NumPy 通常与 SciPy 和 Matplotlib 一起使用， 这种组合广泛用于替代 MatLab。利用 NumPy，可以简化代码实现中的操作，更为便捷地达到模型判断的目的。

1.2.3 Matplotlib

Matplotlib 是 Python 的绘图库，可以在各种平台上以各种硬拷贝格式和交互式环境生成出具有出版品质的图形，生成绘图，直方图，功率谱，条形图，错误图，散点图等。在解决该问题的过程中，我们主要借助 Matplotlib 对该人脸识别模型进行测试评价，针对不同的超参数，都可以获取一个对应的识别率，利用该绘图库，我们可以作出反映识别率与超参数关系的图表，更直观地看出使识别率尽可能高的超参数，来达到对该模型的测试评价操作。

1.2.4 PIL

PIL(Python Imaging Library)□ Python □□□□□□□□□□□□。□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□。□□□□□□□□□□□□□□□□□□□□□□□□□。□□□□□□□□□□□□□□□□□□□□。

1.2.5 os

□□□□□□□□□□□□□□□□□□□□□□□□□。□□□。□□□□□□□□□□□□□□□□□□□□□□□□□□□□、□□□、□□□□、□□□□□□□□

2 解决方法

实现方法分成以下步骤：

（1）【数据获取与预处理】

首先获取数据，从数据集 Faces94 中获取数据。接着，划分训练集和测试集，我们这里选取的比例是训练集：测试集=1:1。并且思考如何获得标注信息(类别可以用什么表示)

（2）【问题分析与建模】

对人脸识别问题进行建模(用数学公式，表达如何识别出是哪一个人的面部)。

运用 PCA,(Principal Component Analysis)进行降维。基本思想是用更少的特征（维数）概括数据集的大量特征。

以 KNN 为基础，探索其他更好的方法。这里运用了 softmax 函数的方法。

（3）【模型测试与评价】

a）给一张图片输出其是哪个人

b）在测试集上的识别率是多少?

（4）【思考总结】

模型有什么优缺点，什么情况下适用，什么情况下不适用。

数据集非常大，会出现内存问题，如何解决?

2.1 导入库

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from PIL import Image

import os
```

我们需要用到 numpy 库中的矩阵和向量化运算等，以及 pandas 库中的 DataFrame 对象和方法等，所以在程序开始时导入 Numpy 和 Pandas 库。Matplotlib 库是进行参数（KNN 中的 k）优化时，用来可视化的工具。PIL 和 os 库是处理图像，并从图像得到数据集这一过程中使用的库。

2.2 数据获取与预处理

从图片中获得 data.csv 文件

这段代码是从图片中获得 data.csv 文件

```
path = "/Users/arthas/Desktop/py_project/faces94/faces94/malestaff"


def toGrayVec(path):
    image = Image.open(path)  # 创建图像
    r,g,b = image.split()     # Splitting bands
    size = image.size[0]*image.size[1]  # 像素大小，长*宽
```

```
    gv = np.round(np.array(r).reshape(1,size)* 299/1000+np.array(g).reshape(1,size)* 587/1000+np.array(b).reshape(1,size)*114/1000)

    return gv            # 更改数组形状；每个 band（r,g,b)都变成行数为 1，列数为 size 的一个数组，然后相加。具体形成结果可见 data.csv。这其中的数据是红绿蓝这三种颜色权重的算法。

df = pd.DataFrame({})

files = os.listdir(path)  # 打开文件

for file in files:   # 遍历所有文件

    fpath = path + '/' + file

    ffiles = os.listdir(fpath)

    for ffile in ffiles:  # 这里是每个小文件夹里还有很多人的图片，所以遍历两次。

        if ffile.endswith('.jpg'):

            ffpath = fpath + '/' + ffile

            label = ffile.split('.')[0]

            gv = toGrayVec(ffpath)

            na = np.append(np.array([label]),gv)   # 添加

            df = df.append([na])    # 形成 data.csv 文件

df.to_csv('data_csv.csv')
```

压缩过程：（将 180*200 压缩到 18*20）

```
data = pd.read_csv('data.csv').iloc[:, 1:]

data_new = data.iloc[:, :361].copy(deep=True)

for i in range(360):

    # print(i)

    data_new.iloc[:, i + 1] = np.mean(data.iloc[:, 100 * i + 1: 100 * i + 101], axis=1)

data_new.to_csv('data_new_1.csv')

#  数据获取

data = pd.read_csv('data.csv').iloc[:, 1:]
```

train_data, test_data = separation(data_pca)

read_csv 是 pandas 中的方法，读取 csv 文件并返回 DataFrame 对象。Iloc 方法表示取 DataFrame 对象的行和列。这里 iloc[:, 1:]表示取所有行中的第二列及以后。（剔除第一列的很多 0）

```
# 数据处理

def separation(data, ratio=0.5):

    lst = [data[data.iloc[:, 1] == item].sample(frac=ratio) for item in set(data.iloc[:, 1])]

    train_data = pd.concat(lst)

    test_data = data.drop(train_data.index)

    return train_data, test_data
```

函数的第一句实现的是，在数据集里每个不同的标签类对应的数据点都各取百分之五十作为训练集。这是一个解析式的写法，在 data 里的第一列是标签分类，item 遍历标签分类，每一个 item 里都有很多数据点，我们从中取样，取样比例 frac=ratio=0.5，这样就可以实现训练集和测试集的划分。

第二句是将 lst 作为列表传入，拼接起来，拼接方法是列对齐。这样就形成了我们的测试集。第三句就是将测试集从数据集中删去，完成训练集划分。最后返回 train_data 和 test_data 这两个 DataFrame 对象。

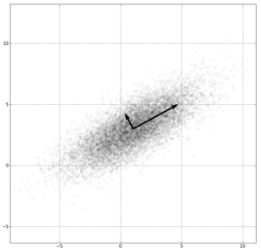
2.3 PCA 降维方法

```
def pca(data, d=2):

    # PCA,(Principal Component Analysis),是一个降维方法，基本思想是用更少的特征（维数）概括数据集的大量特征。

    # 默认参数 d=2，表示 pca 方法降维的维数。

    data_matrix = np.matrix(data)

    y = data.shape[1]   # 取列数 shape[0]与 shape[1]分别表示取行和列

    mean = np.matrix([np.mean(data_matrix[:, i]) for i in range(y)]) # 取平均值

    data_matrix_0 = data_matrix - mean   # 归一化

    square_matrix = data_matrix_0.T * data_matrix_0   # 求转置

    lamda, vector = np.linalg.eig(square_matrix.A)   # 求特征值与特征向量
```

```
    lamda_vector = [(np.abs(lamda[i]), vector[:, i]) for i in range(y)]  # 将特征值与特征向量配对

    lamda_vector.sort(key=lambda x: x[0], reverse=True)  # 按照特征值排序

    vector_s = np.matrix([t[1] for t in lamda_vector[:d]])  # 把特征值最大的几个取出来（取前 d 个）

    data_new = (data_matrix_0 * vector_s.T).A  # 得到刚才的特征向量，新的数据是降过维的数组

    return data_new
```

在多元统计分析中，主成分分析（Principal components analysis，PCA）是一种分析、简化数据集的技术。主成分分析经常用于减少数据集的维数，同时保持数据集中的对方差贡献最大的特征。这是通过保留低阶主成分，忽略高阶主成分做到的。这样低阶成分往往能够保留住数据的最重要方面。由于主成分分析依赖所给数据，所以数据的准确性对分析结果影响很大。主成分分析在分析复杂数据时尤为有用，比如人脸识别。

其方法主要是通过对协方差矩阵进行特征分解，以得出数据的主成分（即特征向量）与它们的权值（即特征值）。其结果可以理解为对原数据中的方差做出解释：哪一个方向上的数据值对方差的影响最大？换而言之，PCA 提供了一种降低数据维度的有效办法；如果分析者在原数据中除掉最小的特征值所对应的成分，那么所得的低维度数据必定是最优化的（也即，这样降低维度必定是失去讯息最少的方法）。



如果是一个主成分分析实例，这是一个平均值为(1, 3)、标准差在(0.878, 0.478)方向上为 3、在其正交方向为 1 的高斯分布。这里以黑色显示的两个向量是这个分布的协方差矩阵的特征向量，其长度按对应的特征值之平方根为比例，并且移动到以原分布的平均值为原点。

有了对 PCA 方法的基本了解，我们可以得到如上所示的代码。更加具体的数学原理过于复杂，在这里不作详细描述。

2.4 KNN 方法

```
def KNN(train_data, test_data, k=3):
```

```
        predict_data = test_data.copy(deep=True)  #深拷贝
        for i in range(test_data.shape[0]):
            d = []
            for j in range(train_data.shape[0]):      #计算距离
                    d.append((distance(test_data.iloc[i, 1:], train_data.iloc[j, 1:]), train_data.iloc[j,
0]))
           d.sort(key=lambda x: x[0])   # 按照距离排序
        dic = {}
        for j in range(k):
            dic[d[j][1]] = dic.get(d[j][1], 0) + 1   #计数
        mx = -1
        maxitem = ''
        for item in dic.items():
            if dic[item[0]] > mx:
                mx = dic[item[0]]
                maxitem = item[0]
        predict_data.iloc[i, 0] = maxitem   # 出现次数最多的作为数据点的标签
    return predict_data


def distance(x, y):
    return np.sqrt(np.sum((x - y) ** 2))
```

模式识别领域中，KNN 算法是一种用于分类和回归的非参数统计方法。在这两种情况下，输入包含特征空间（Feature Space）中的 k 个最接近的训练样本。在 k-NN 分类中，输出是一个分类族群。一个对象的分类是由其邻居的“多数表决”确定的，k 个最近邻居（k 为正整数，通常较小）中最常见的分类决定了赋予该对象的类别。若 k = 1，则该对象的类别直接由最近的一个节点赋予。

代码段 1 的作用是计算出每个对象与其周围点的距离，并按照距离排序。代码段 2 的作用是计数，也就是对排好序的 d 来说，取前 k 个点的标签，在这 k 个标签类中，出现最多的标签类就是我们估计的对象点的类别。

KNN 算法是机器学习中最简单的算法之一，我们可以通过调整 k 的数值来进行进一步研究。

```
def parameter_optimization(train_data,test_data):
    k_range = list(range(1, 26))
    scores = []
    for k_x in k_range:
        predict_data = KNN(train_data, test_data, k=k_x);
        error = evaluation(predict_data, test_data)
        accuracy = 1 - error
        scores.append(accuracy)
    # plot the relationship between K and testing accuracy
    plt.plot(k_range, scores)
    plt.xlabel('Value of K for KNN')
    plt.ylabel('Testing Accuracy')
```

这是通过迭代 k=1 到 k=25 来尝试寻找一个结果最好的 k，并利用 matplotlib 可视化。

2.5 评估函数

```
def evaluation(predict_data, data):
#评估函数，如果结果 predict_data 与正确答案不符合，则为 error
    total = predict_data.shape[0]
    error = len([index for index in range(total) if not predict_data.iloc[index, 0] ==
data.iloc[index, 0]])
    return error / total
```

这里返回的是错误率。第二条语句用了比较复杂的解析式的语法，如果 predict_data 与 data 中的数据不符合，则表示预测出来的结果是错误的。

2.6 参数传入及主程序

```
d = 10    #表示我们用 PCA 模型把维数降到 10

data = pd.read_csv('data_new_1.csv').iloc[:, 1:]   #读取数据

data_p = pca(data.iloc[:, 1:].values.astype('float'), d)

data_pca = pd.DataFrame(data_p)

data_pca.insert(0, 'name', data.iloc[:, 0])

train_data, test_data = separation(data_pca)

predict_data = KNN(train_data, test_data, 5)  #这里我们使用的 k 设为 5，也是受到老师推荐视频的启发

error = evaluation(predict_data, test_data)

print(predict_data)

print(test_data)

print(error)
```

主程序，包括了读取数据、处理数据、划分训练集和测试集、模型训练以及评估。最后输出评估结果。函数在之前已经介绍过，这里不多赘述。

2.7 运用 softmax 方法的程序实现

softmax 函数用于多分类过程中，它将多个神经元的输出，映射到（0,1）区间内，可以看成概率来理解，从而来进行多分类。

以下是全部程序：

```
import numpy as np
import pandas as pd
```

导入库

```
def separation(data, ratio=0.8):
    total = data.shape[0]
    sep = int(total * ratio)
    return data[:sep], data[sep:]
```

分割数据集的另一种方式

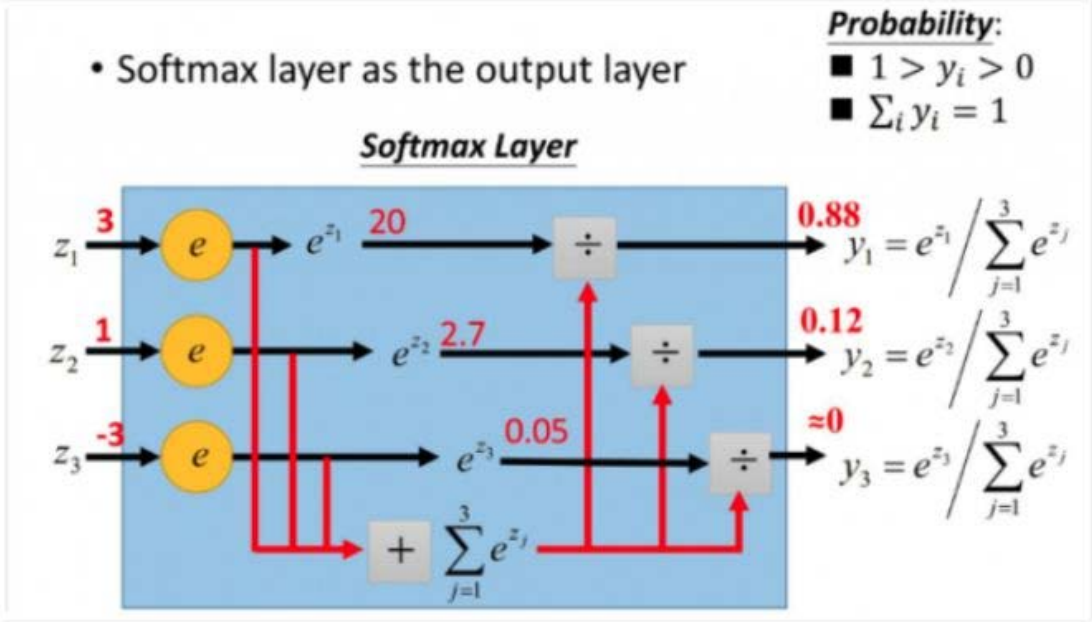
```
def probability(omega, data_x, k):  # 概率的归一化
```

```
    total = sum([np.exp(np.dot(omega[i].T, data_x)) for i in
range(omega.shape[0])])
    return np.exp(np.dot(omega[k].T, data_x)) / total
```

假设我们有一个数组，V，Vi 表示 V 中的第 i 个元素，那么这个元素的 softmax 值就是：

$$S_i = \frac{e^i}{\sum_j e^j}$$

更形象的可以举个例子：



如图，softmax 直白来说就是将 z1、z2、z3（这里分别是 3,1,-3）进行作用，把他们映射成为(0,1)的值，而这些值的累和为 1（满足概率的性质），那么我们就可以将它理解成概率。在最后选取输出结点的时候，我们就可以选取概率最大（也就是值对应最大的）结点，作为我们的预测目标。

```
def deri(omega, data, k, label):
    array = np.array([(int(data.iloc[i, 0] == label[k]) - probability(omega,
data.iloc[i, 1:], k)) * data.iloc[i, 1:] for i in range(data.shape[0])])
    return - np.sum(array, axis=0) / data.shape[0]
```

对第 k 个标签类，进行这样的算法变换（softmax 算法）

```
def softmax(data):   # softmax 函数
    label = list(set(data.iloc[:, 0]))   # 标签类
    K = len(set(label))     # 标签类数量
```

```
    row, line = data.shape    # 行、列
    line = line - 1
    alpha = 0.002        # 参数
    omega = np.zeros((K, line))  #构造全 0 数组
    iteration = 100
    thershold = 1e-12
    for i in range(iteration):
        der = np.array([deri(omega, data, k, label) for k in range(K)])
        omega1 = omega - alpha * der     #遍历并获取 der
        if np.sum(omega1 - omega) / (K * line) < thershold:
            print(i)
            return omega1
        omega = omega1
    return omega          #这一段数学算法，原理比较复杂，不过多叙述。

def predict(omega, data):
    K = len(set(data.iloc[:, 0]))
    p = np.array([[probability(omega, data.iloc[i, 1:], k) for k in range(K)] for i in
range(data.shape[0])])
    label_p = [np.unravel_index(np.argmax(p[i]), p.shape[1])[0] for i in
range(p.shape[0])]
    label = list(set(data.iloc[:, 0]))
    return [label[i] for i in label_p]


def evaluation(prediction, data):
    return sum(data.iloc[:, 0] == prediction) / data.shape[0]
```
评估函数

```
names = ['category'] + ['x{}'.format(i) for i in range(11)]
data = pd.read_csv('dataset.csv', names=names).iloc[:, :-1]
train, test = separation(data)
omega = softmax(train)
prediction = predict(omega, test)
accuracy = evaluation(prediction, test)
print(accuracy)
```

主程序，包括了读取数据、处理数据、划分训练集和测试集、模型训练以及评估。最后输出评估结果。

3 □□、□□□□□

□□□□□□□□□□□□400*36000□□□□□□□□□□□KNN□□□□□□□□95%。□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□99.6%。□□□□□□□□□□□□□□0.0□□□□□□□□softmax□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□15%□□。□□□□□□。。。□KNN□□□。

□□□□□□□□□□□。

□、□□□□□□□□KNN□□□□□□□□□□□□□f□.□□□□□□□□□□□□□□□□□□□。□□□□□□□□□f□□□□□□□□□□□□□□□□□□□□□□□□□。□□□□□□□f□□□□□□□□□□□□□□□□□□。□□□□□□□□□□□□□□□□□□□□。□□□□□□□□□□□□□□□□□□□□□□□□□□□□CNN□□□□□□□□□□□□□□。□□□□□□□□□□□□□□□□□□□□□□□□□□□KNN□□□□□□□□□□□□□□□□□□□□□□□□□□□。

□□□□□□□□KNN□□□□□□□。□□□□□□□□□□□□□□□□k□□□□□□□。□□□□□□□□□□□□□□□k。□□□□□□□□□k□5-15□□□□□□□□□□□□□□□□。□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□。

□、□□□□□□□□□□□□□□□□

□□□□□□□□data.csv□□□□400*36000□□180*200=36000□□□□□□□□□□□□。□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□8□□□。□□□□□□□□□□□□□□□□□□□□□□□□k□。

□□□□□□□□□□□□□□□□□□□180*200□□□□□□18*20□□□□data.csv□□□□□71M□□□□800k□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□。□□□□□□□□□□□□□□□□□□□□3060□□□□□400□□□□□□□。□□□□□□□□□□□□□□□□□□□□□□。

□□。

4 □ □

本次大作业，我们小组用 python3 实现了人脸识别。我们初步了解了应用于人脸识别领域的机器学习的基本算法，接触了 Numpy、Pandas、Matplotlib 等库中的方法，也了解了 Face94 数据集。

5 致谢

我们真诚地感谢鲍老师，他非常耐心，即使我们选择中途更换题目，他也始终热情地帮助、鼓励我们。他的辅导课程为我们这个项目的成功做出了很多贡献！