利用python实现人脸识别

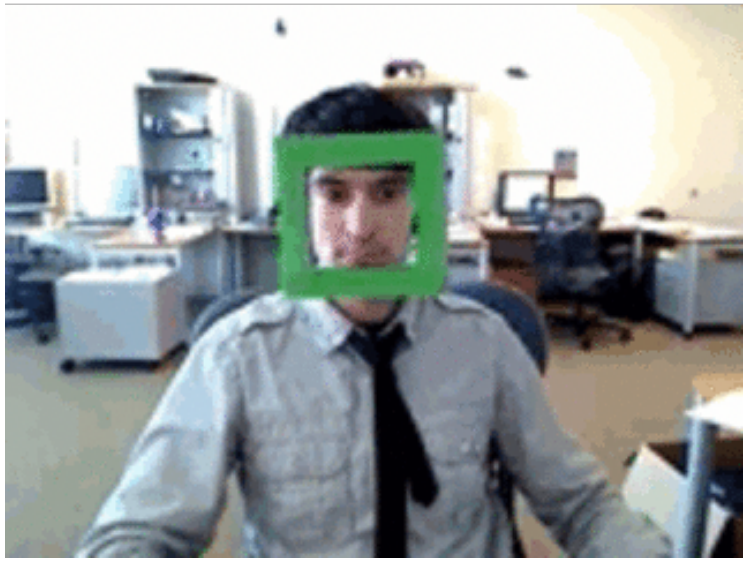
张芷馨，韩涵，刘奕涵

01

背景

人脸识别，特指利用分析比较人脸视觉特征信息进行身份鉴别的计算机技术。

人脸识别是一项热门的计算机技术研究领域，它属于生物特征识别技术，是对生物体（一般特指人）本身的生物特征来区分生物体个体。



提出问题：

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。

Faces94数据集

收购条件

受试者与摄像机保持固定距离并被要求说话，同时拍摄一系列图像。该演讲用于介绍面部表情变化。

数据库描述

- 人数：153
- 图像分辨率：180 x 200像素（纵向格式）
- 目录：女性（20），男性（113），男性（20）
- 包含单独目录中的男性和女性主题的图像

个人形象的变化

- 背景：背景是纯绿色
- 头部规模：无
- 转头，倾斜和倾斜：这些属性的微小变化
- 脸部在图像中的位置：微小的变化
- 图像照明变化：无
- 表达变异：相当大的表达变化
- 附加评论：由于图像是在一次会话中拍摄的，因此没有单独的hairstly变化。

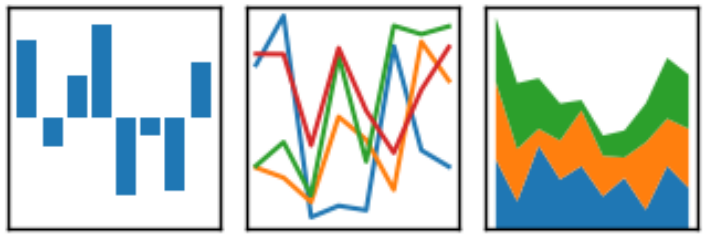
02

准备

应用的库

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



用于数据分析和建模，更方便地执行整个数据分析工作流程，完成对Faces94数据集的获取和预处理操作

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，用于数组计算

应用的库

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。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

PIL(Python Imaging Library)为Python解释器添加了图像处理功能。该库提供广泛的文件格式支持，高效的内部表示和相当强大的图像处理功能。核心图像库旨在快速访问以几种基本像素格式存储的数据。它应该为一般的图像处理工具提供坚实的基础。

03

问题解决


数据获取
预处理
建模
代码实现
模型测试
评价
评价
提升

数据获取与预处理

图像处理

利用PIL库和os库对数据集中的图像做处理，生成data.scv文件

数据获取

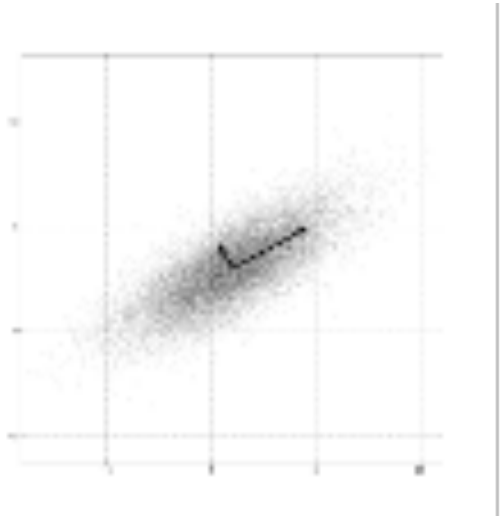
```
data = pd.read_csv('data_new_1.csv').iloc[:, 1:]
train_data, test_data = separation(data_pca)
```

数据处理

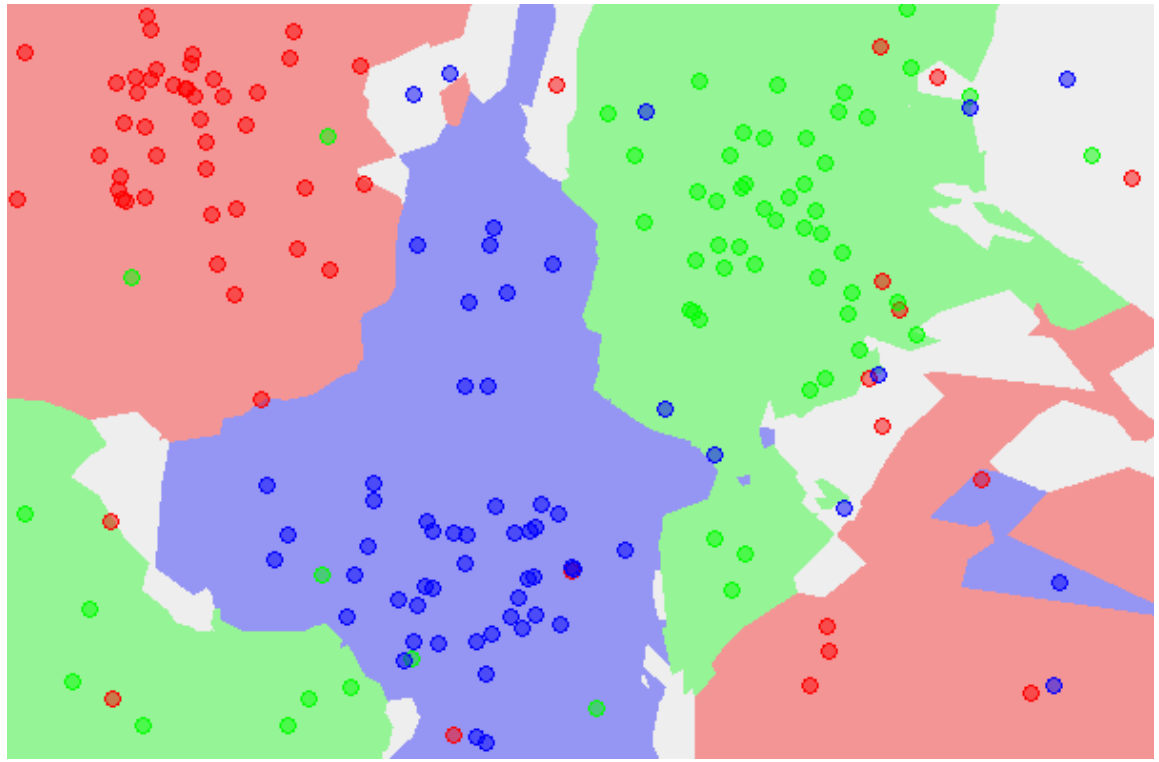
```
def separation(data, ratio=0.5):
    lst = [data[data.iloc[:, 1] == item].sample(frac=ratio) for item in set(data.iloc[:, 1])]
    train_data = pd.concat(lst)
    test_data = data.drop(train_data.index)
    return train_data, test_data
```

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
```



KNN方法



1.Pick a value for K.
2.Search for the K observations in the training data that are "nearest" to the measurements of the unknown dataset.
3.Use the most popular response value from the K nearest neighbors as the predicted response value for the unknown dataset.

KNN方法

```
def KNN(train_data, test_data, k=3):
    （代码段1）
    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

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

```
    （代码段2）
    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
```

参数优化

```
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')
```

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
```

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.csv').iloc[:, 1:]    #读取数据
data_p = pca(data.iloc[:, 1:].values.astype('float'), d)
data_pca = data.iloc[:, 0]
for i in range(d):
    data_pca['x{}'.format(i)] = data_p.T[i]
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)
```

04

评价提升

一、模型的简单，并没有能运用更加复杂、高效的模型

二、内存问题。随着存储数据方法的优化，这个问题得以解决。

Thanks!