第四次实训程序报告

学号: 2310764 姓名: 王亦辉

1 问题重述

本实验要求使用特征脸（Eigenface）算法进行人脸识别和图像重建。核心任务包括:

1. 基于ORL人脸库或自建人脸库（需确保图像大小一致、眼睛中心对齐），通过主成分分析（PCA）计算特征值和特征向量，构建特征脸模型。
2. 使用前K个特征脸进行人脸识别和图像重建，比较不同K值对识别准确率和重建效果的影响。
3. 使用Python实现，需参考特征脸相关资料，最终以人脸识别准确率为评分标准。

2 设计思想

使用主成分分析法

1. 求出平均脸
2. 计算训练数据里每张脸与平均脸的差异
3. 求差异矩阵的特征值和特征向量
4. 取前 K 个特征向量，计算出 K 张特征脸，然后就可以利用这 K 个特征脸对测试人脸进行识别了。

3 代码内容

3.1 特征人脸算法

```

def eigen_train(trainset, k=20):
    """
    训练特征脸（eigenface）算法的实现

    :param trainset: 使用 get_images 函数得到的处理好的人脸数据训练集
    :param K: 希望提取的主特征数
    :return: 训练数据的平均脸，特征脸向量，中心化训练数据
    """
    # 1. 计算平均人脸
```

```
    avg_img = np.mean(trainset, axis=0)
    # 2. 中心化训练数据
    norm_img = trainset - avg_img
    # 3. 计算协方差矩阵
    cov_matrix = np.dot(norm_img, norm_img.T) / norm_img.shape[0]

    # 4. 计算特征值和特征向量
    eigenvalues, eigenvectors = np.linalg.eig(cov_matrix)

    # 5. 选择前k个特征向量
    sorted_indices = np.argsort(eigenvalues)[::-1]
    eigenvectors = eigenvectors[:, sorted_indices]
    feature = np.dot(norm_img.T, eigenvectors[:, :k])
    # 6. 归一化特征脸
    feature = feature / np.linalg.norm(feature, axis=0)
    feature = feature.T

    # 返回：平均人脸、特征人脸、中心化人脸
    return avg_img, feature, norm_img
```

3.2 人脸识别模型

```
def rep_face(image, avg_img, eigenface_vects, numComponents = 0):
    """
    用特征脸（eigenface）算法对输入数据进行投影映射，得到使用特征脸向量表示的数据

    :param image: 输入数据
    :param avg_img: 训练集的平均人脸数据
    :param eigenface_vects: 特征脸向量
    :param numComponents: 选用的特征脸数量
    :return: 输入数据的特征向量表示，最终使用的特征脸数量
    """
    # 中心化输入图像
    norm_image = image - avg_img

    if numComponents == 0:
        numComponents = eigenface_vects.shape[0]

    # 投影到特征脸
    representation = np.dot(norm_image, eigenface_vects[0:numComponents,].T)

    # 返回：输入数据的特征向量表示，特征脸使用数量
```

```
    return representation, numComponents
```

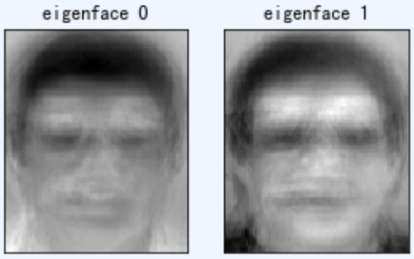
3.3 人脸重建模型

```
def recFace(representations, avg_img, eigenVectors, numComponents, sz=(112,92)):
    """
    利用特征人脸重建原始人脸

    :param representations: 表征数据
    :param avg_img: 训练集的平均人脸数据
    :param eigenface_vects: 特征脸向量
    :param numComponents: 选用的特征脸数量
    :param sz: 原始图片大小
    :return: 重建人脸，str 使用的特征人脸数量
    """
    # 使用特征向量和特征脸的线性组合重建中心化人脸
    reconstructed_norm = np.dot(eigenVectors[0 :numComponents,].T, representations)

    # 加上平均人脸
    face = reconstructed_norm + avg_img

    # reshape到原始图像大小
    face = face.reshape(sz)

    # 返回：重建人脸，str 使用的特征人脸数量
    return face, 'numEigenFaces_{}'.format(numComponents)
```

4 实验结果

打印出的特征人脸，虽然模糊不清，但是有人脸的形状。

[23]:
```
# 返回平均人脸、特征人脸、中心化人脸
avg_img, eigenface_vects, trainset_vects = eigen_train(train_vectors, num_eigenface)

# 打印两张特征人脸作为展示
eigenface_vects = eigenface_vects.T

eigenfaces = eigenface_vects.reshape((num_eigenface, 112, 92))
eigenface_titles = ["eigenface %d" % i for i in range(eigenfaces.shape[0])]
plot_gallery(eigenfaces, eigenface_titles, n_row=1, n_col=2)
```



人脸识别的准确率如下，超过 90%，算是比较高的。

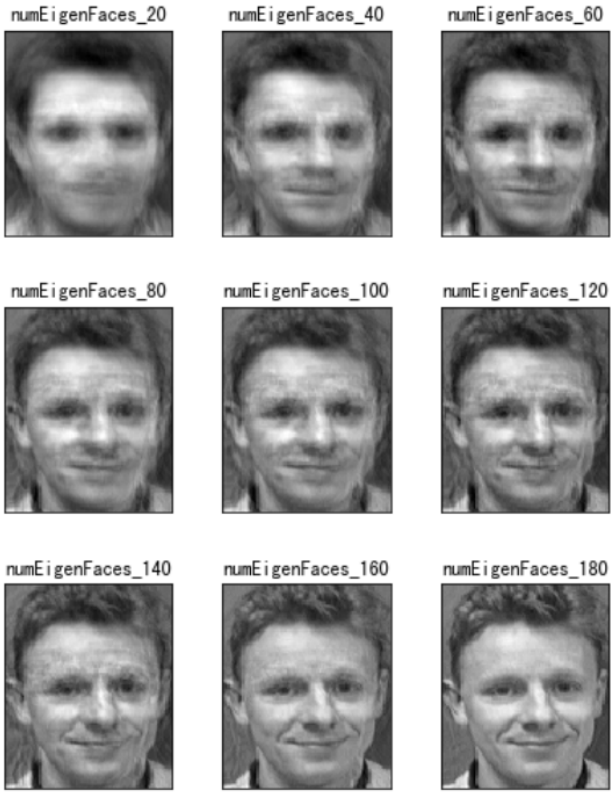
[33]:
```
train_reps = []
for img in train_vectors:
    train_rep, _ = rep_face(img, avg_img, eigenface_vects, num_eigenface)
    train_reps.append(train_rep)

num = 0
for idx, image in enumerate(test_vectors):
    label = test_labels[idx]
    test_rep, _ = rep_face(image, avg_img, eigenface_vects, num_eigenface)

    results = []
    for train_rep in train_reps:
        similarity = np.sum(np.square(train_rep - test_rep))
        results.append(similarity)
    results = np.array(results)

    if label == np.argmin(results) // 5 + 1:
        num = num + 1

print("人脸识别准确率: {}%".format(num / 80 * 100))
```

```
人脸识别准确率: 91.25%
```

重建人脸测试，可以看到使用更多特征人脸进行重建，得到的图像更清晰。

[37]:
```
print("重建训练集人脸")
# 读取train数据
image = train_vectors[100]

faces = []
names = []
# 选用不同数量的特征人脸重建人脸
for i in range(20, 200, 20):
    representations, numEigenFaces = rep_face(image, avg_img, eigenface_vects, i)
    face, name = recFace(representations, avg_img, eigenface_vects, numEigenFaces)
    faces.append(face)
    names.append(name)

plot_gallery(faces, names, n_row=3, n_col=3)

print("-"*55)
print("重建测试集人脸")
# 读取test数据
image = test_vectors[54]

faces = []
names = []
# 选用不同数量的特征人脸重建人脸
for i in range(20, 200, 20):
    representations, numEigenFaces = rep_face(image, avg_img, eigenface_vects, i)
    face, name = recFace(representations, avg_img, eigenface_vects, numEigenFaces)
    faces.append(face)
    names.append(name)

plot_gallery(faces, names, n_row=3, n_col=3)
```

重建训练集人脸



重建测试集人脸

系统测试通过

[ ]:

5 总结

[ ]:

不太熟悉 numpy 的操作和理念，操作行列向量、矩阵等比较艰难。

对数学方面的求解不太熟悉，想来是线代忘光光的缘故，不过好歹 numpy 提供了特征值、特征向量自动求解的函数，感慨工具的强大。以及我们在使用数学工具的时候，把原理搞懂并不是必要的，只要知道可以用来做什么、怎么用就行，感受到数学的这种工具性，毕竟结论只需要有人证一次，只要证出来就可以反复使用，即使把中间过程当作黑盒，这就是数学的美好啊。