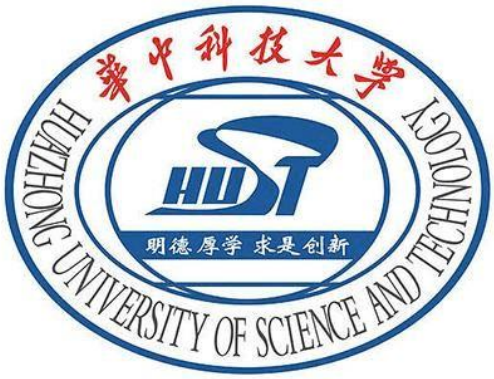



华中科技大学

HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

# 数据挖掘实验报告

## PCA 人脸识别

学　　院：电子信息与通信学院

班　　级：提高 2101 班

姓　　名：杨筠松

学　　号：U202115980

实验时间：2024/4/14

# 一、PCA 算法原理介绍

首先需要知道几个相关的数学概念，这是我们进行 PCA 分析的基础，标准差（Standard Deviation）、方差（Variance）、协方差（Covariance）、特征向量（eigenvectors）、特征值（eigenvalues）

## Standard Deviation（标准差）

标准差就是用来描述一组数据与平均值得偏离程度，反映了一组数据的波动情况，平均值数学表达公式如下：

$$\bar{X} = \frac{\sum_{i=1}^{n} X_i}{n}$$

标准差的表达公式如下：

$$s = \sqrt{\frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{(n-1)}}$$

需要注意的是分母是选择 `n` 还是 `n-1`，取决于你选取的数据是整个完整数据还是数据中的一部分

## Variance(方差)

方差是数据集中数据分布的另一种度量。实际上，它几乎与标准差相同

方差的数学表达公式如下：

$$s^2 = \frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{(n-1)}$$

## Covariance(协方差)

标准差与方差只针对一维数据进行衡量的指标，协方差是针对二维数据或者是更高维的数据进行的衡量指标，主要用来表示多维度与平均值的偏离程度。
协方差的数学表达公式如下：

$$cov(X,Y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{(n-1)}$$

## The covariance Matrix（协方差矩阵）

协方差矩阵主要是用于当数据的维度超过 3 或者更多的时候，我们可以通过一个矩阵来存储各个维度的协方差，这个矩阵就被称为“协方差矩阵”。
用数学方法来表示一个 N 为数据的协方差矩阵可以表示为：

$$C^{n\times n} = (c_{i,j},\ c_{i,j} = cov(Dim_i, Dim_j)),$$

现在假设我们有一个三个维度的数据，使用一个协方差矩阵将这三维数据的协方差表示如下：

$$C = \begin{pmatrix} cov(x,x) & cov(x,y) & cov(x,z) \\ cov(y,x) & cov(y,y) & cov(y,z) \\ cov(z,x) & cov(z,y) & cov(z,z) \end{pmatrix}$$

## Eigenvectors（特征向量）

在矩阵论中，我们可以这样去理解特征值和特征向量，一个矩阵由一个变换到另一个矩阵，$A\alpha=\lambda\alpha$，其中 α 称为矩阵 A 的一个特征向量，λ 称为矩阵 A 的一个特征值。特征向量确定了矩阵变换的方向，特征值确定了矩阵变换的比例。
协方差矩阵的特征向量代表的意思是方差最大的数据所在的方向。在 n 维数据空间中，第一特征向量指向的是数据方差最大的方向，第二特征向量是与第一特征向量垂直的
数据方差最大的方向，第三特征向量是与第二特征向量垂直的数据方差最大的方向，以此类推。

通常我们还需要对特征向量进行标准化处理，即求模长，然后将向量中每一个元素除以该模长，即为标准化处理。

## Choosing components and forming a feature vector

## （选择主成分并生成特征向量）

一个协方差矩阵有着不同的特征值与特征向量，事实上最高特征值的对应的特征向量就是这个数据集的主成分。
通常来说，一旦协方差矩阵的特征值和特征向量被计算出来了之后，就是按照特征值的大小从高到低依次排列。特征值的大小确定了主成分的重要性。
主成分分析的基本原理就是：选择特征值较大的作为主成分，从而进行降维。比如：一开始你的数据集是 N 维的，在进行了协方差矩阵的特征值计算后，你得到了 N 个特征值和与这些特征值相对应的特征向量。然后在主成分分析时，你选取了前 P 个较大的特征值，如此一来，就将原来 N 维的数据降维到只有 P 维。这样就起到了降维的效果了。

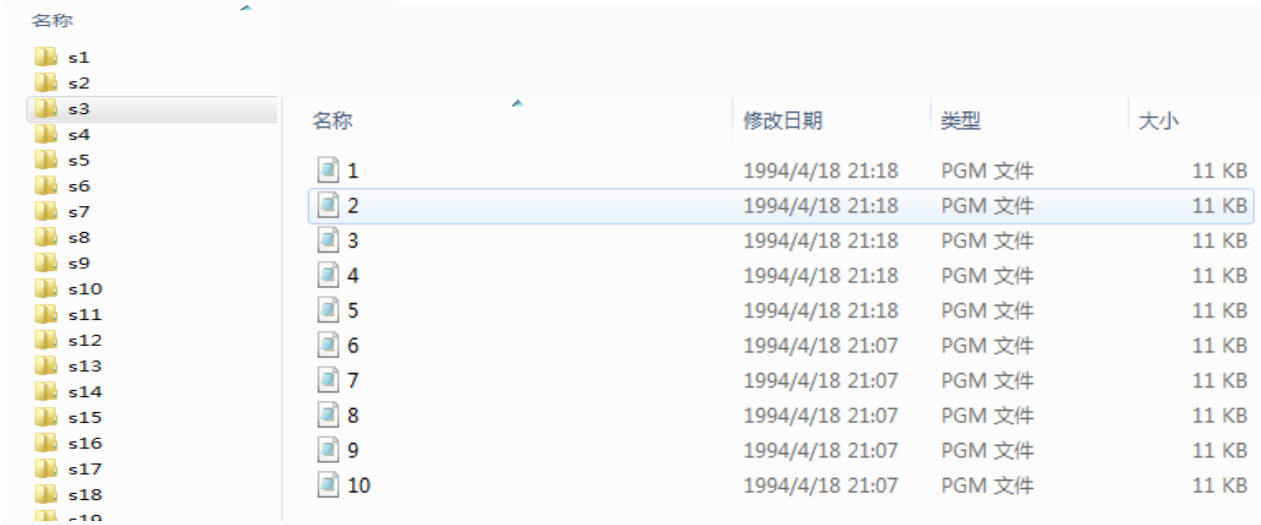
## Deriving the new data set（选择特征向量生成新的数据集）

$$FinalData = RowFeatureVector \times RowDataAdjust,$$

其中 rowFeatureVector 是由模式矢量作为列组成的矩阵的转置，因此它的行就是原来的模式矢量，而且对应最大特征值的特征矢量在该矩阵的最上一行。rowdataAdjust 是每一维数据减去均值后，所组成矩阵的转置，即数据项目在每一列中，每一行是一维，对我们的样本来说即是，第一行为 x 维上数据，第二行为 y 维上的数据
正是由于特征向量是两两正交的，那么我们就可以使用任何的特征向量来将原始数据变换到正交的这些坐标轴上。
接下来，我们选取一个较大的特征值对应的特征向量将原始数据降到一维，即将较大的特征值对应的特征向量转置然后乘以原始数据集，这样就得到新的降维后的一维数据。

# 二、PCA 的人脸识别算法（基于 Python 实现）

我使用的是 ORL 官方数据集




下载后的数据集如上图所示，其中包括了 s1-s40 总计 40 个数据包，每个包括 10 张照片和对应的 PGM 文件作为说明

首先定义一个函数用于将人脸图像矢量化为一个向量，向量的大小与图片的像素有关，代码如下：

```
# 图片矢量化
def img2vector(image):
    img = cv2.imread(image, 0)  # 读取图片
    rows, cols = img.shape  #获取图片的像素
    imgVector = np.zeros((1, rows * cols)) #初始值均设置为 0，大小就是图片像素的大小
    imgVector = np.reshape(img, (1, rows * cols))#使用 imgVector 变量作为一个向量存储图片矢量化信息
    return imgVector
```

接下来定义一个函数用来选取训练图片，并对每张图片进行前面定义过的矢量化处理

```
def load_orl(k):#参数 K 代表选择 K 张图片作为训练图片使用
    '''''
    对训练数据集进行数组初始化，用 0 填充，每张图片尺寸都定为 112*92,
    现在共有 40 个人，每个人都选择 k 张，则整个训练集大小为 40*k,112*92
    '''
    train_face = np.zeros((40 * k, 112 * 92))
    train_label = np.zeros(40 * k)  # [0,0,.....0](共 40*k 个 0)
    test_face = np.zeros((40 * (10 - k), 112 * 92))
    test_label = np.zeros(40 * (10 - k))
    # sample=random.sample(range(10),k)#每个人都有的 10 张照片中，随机选取 k 张作为训练样本(10 个里面随机选取 K 个成为一个列表)
    sample = random.permutation(10) + 1  # 随机排序 1-10 (0-9）+1
    for i in range(40):  # 共有 40 个人
        people_num = i + 1
        for j in range(10):  # 每个人都有 10 张照片
            image = orlpath + '/s' + str(people_num) + '/' + str(sample[j]) + '.jpg'
            # 读取图片并进行矢量化
            img = img2vector(image)
            if j < k:
```

```
                # 构成训练集
                train_face[i * k + j, :] = img
                train_label[i * k + j] = people_num
            else:
                # 构成测试集
                test_face[i * (10 - k) + (j - k), :] = img
                test_label[i * (10 - k) + (j - k)] = people_num

    return train_face, train_label, test_face, test_label
```

前期将所有训练图片矢量化之后，开始进行 PCA 算法的降维操作

```
def PCA(data, r):#参数 r 代表降低到 r 维
    data = np.float32(np.mat(data))
    rows, cols = np.shape(data)
    data_mean = np.mean(data, 0)  # 对列求平均值
    A = data - np.tile(data_mean, (rows, 1))  # 将所有样例减去对应均值得到 A
    C = A * A.T  # 得到协方差矩阵
    D, V = np.linalg.eig(C)  # 求协方差矩阵的特征值和特征向量
    V_r = V[:, 0:r]  # 按列取前 r 个特征向量
    V_r = A.T * V_r  # 小矩阵特征向量向大矩阵特征向量过渡
    for i in range(r):
        V_r[:, i] = V_r[:, i] / np.linalg.norm(V_r[:, i])  # 特征向量归一化

    final_data = A * V_r
    return final_data, data_mean, V_r
```

最后我们进行初次训练，随机选取每个人物的五张图片作为训练图片使用。将降低的维数设定为 10 维，查看一下训练效果如何。

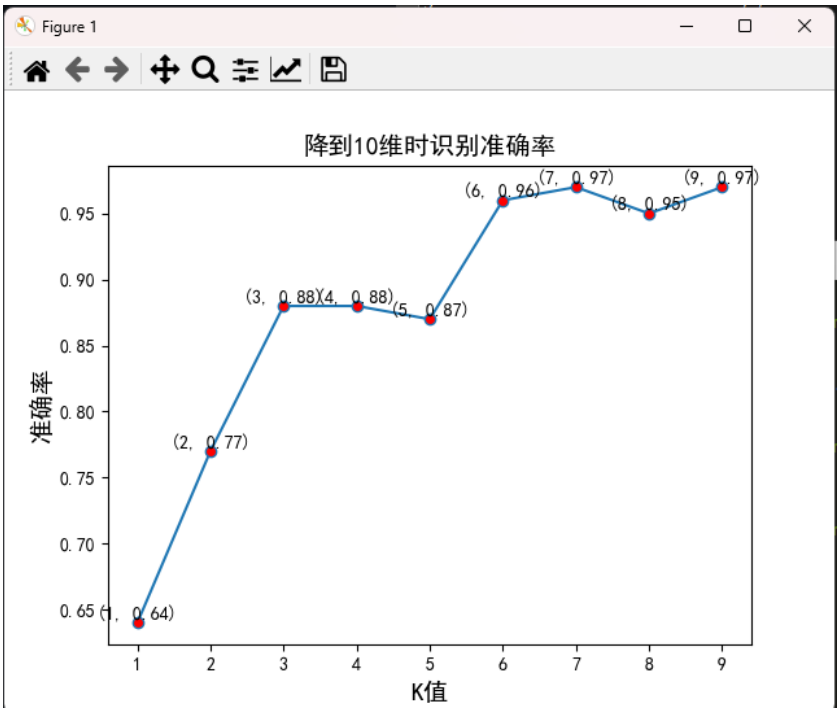
```
def face_recongize():
  #对每一个人随机选取 5 张照片作为训练数据
  train_face, train_label, test_face, test_label = load_orl(5)#随机选择每个人物的 5 张图片作为训练数据

  x_value = []
  y_value = []
  #将图片降维到 10 维
  data_train_new, data_mean, V_r = PCA(train_face, 10)
  num_train = data_train_new.shape[0]  # 训练脸总数
  num_test = test_face.shape[0]  # 测试脸总数
  temp_face = test_face - np.tile(data_mean, (num_test, 1))
  data_test_new = temp_face * V_r  # 得到测试脸在特征向量下的数据
  data_test_new = np.array(data_test_new)  # mat change to array
  data_train_new = np.array(data_train_new)

  true_num = 0
  for i in range(num_test):
      testFace = data_test_new[i, :]
      diffMat = data_train_new - np.tile(testFace, (num_train, 1))  # 训练数据与测试脸之间距离
      sqDiffMat = diffMat ** 2
      sqDistances = sqDiffMat.sum(axis=1)  # 按行求和
      sortedDistIndicies = sqDistances.argsort()  # 对向量从小到大排序，使用的是索引值,得到一个向量
      indexMin = sortedDistIndicies[0]  # 距离最近的索引
      if train_label[indexMin] == test_label[i]:
          true_num += 1
      else:
          pass

  accuracy = float(true_num) / num_test
  x_value.append(5)
  y_value.append(round(accuracy, 2))

  print('当对每个人随机选择%d 张照片降低至%d 维进行训练时，
The classify accuracy is: %.2f%%' % (5,10, accuracy * 100))
```




为了对比实验，我们分别选取 5 张，还是降低到 10 维，20 维，30 维进行对比实验：

```
λ python PCA_face_recongize.py
当降维到10时
当每个人选择5张照片进行训练时，The classify accuracy is: 90.50%
当降维到20时
当每个人选择5张照片进行训练时，The classify accuracy is: 93.50%
当降维到30时
当每个人选择5张照片进行训练时，The classify accuracy is: 95.00%
当降维到40时
当每个人选择5张照片进行训练时，The classify accuracy is: 97.50%
```

可以看出来随着训练集的图片选取的不断增多，训练准确率在不断增加。这是因为训练的样本多了，但是我们如果选择全部的 10 张图片作为训练样本的话就会导致训练结果过拟合。

再一次进行对比实验，我们在选用同样是 7 张图片作为训练样本的情况下，将降低的维数改为 10 维、20 维、30 维，查看训练效果如何。

```
λ python PCA_face_recongize.py
当降维到10时
当每个人选择7张照片进行训练时，The classify accuracy is: 95.83%
当降维到20时
当每个人选择7张照片进行训练时，The classify accuracy is: 96.67%
当降维到30时
当每个人选择7张照片进行训练时，The classify accuracy is: 97.50%
当降维到40时
当每个人选择7张照片进行训练时，The classify accuracy is: 97.50%
```

经过多次试验总结发现，训练样本越多训练效果越好，训练维数越高效果越好，但并不是绝对的，本次试验就发现，在选取的训练样本相同的情况下，降低至 40 维的效果反而不如降低至 30 维的效果