

# FACE RECOGNITION

based on Principle Component Analysis

Weng,Xiangxiang Xiao,Chuanyi Xu,Ruixuan Group 1

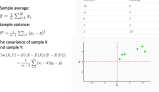
## Introduction

Principal Component Analysis (PCA) is a commonly used dimensionality reduction technology and data analysis method. It can convert high-dimensional data into low-dimensional representation while retaining the main information in the data. Due to these characteristics of PCA, it has wide applications in the field of face recognition

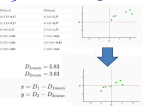
## Principles of mathematics

The idea of PCA is to map dimensionality features to dimensionality, which is a new orthogonal feature. This dimension feature is called the principal element and is a reconstructed dimension feature. In PCA, the data is transformed from the original coordinate system to the new coordinate system.

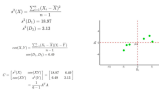
### 1) Initialization



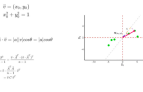
### 2) Deaverage



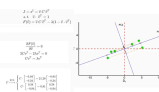
### 3) Covariance matrix



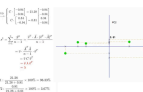
### 4) Projection



### 5) Eigenvectors



### 6) Results



The PCA algorithm, by reducing dimensions and discarding some information, can increase the sampling density of samples, thereby achieving a denoising effect to some extent.

## Conclusion

We loaded the training images using MATLAB code and computed their average face and features. Subsequently, we calculated the relevant features of the test image and compared them with the training image data to identify the identity, yielding the matching results. Finally, the results revealed that the accuracy of the face recognition achieved a rate of 90%.

## Code Implementation

We created two folders. One is called "face source", which is used to store the face photos of the training set. The other is called "face test", which is used to store the face photos of the test set.



☀️ "face\_source" ☀️ "face\_test"

## Matlab Code

```
clear all; %清除所有变量
load('face_source'); %加载训练集
load('face_test'); %加载测试集
%计算训练集的平均值和协方差矩阵
mean_face = mean(face_source, 2);
cov_matrix = cov(face_source - mean_face);
%计算协方差矩阵的特征值和特征向量
[eig_vec, eig_val] = eig(cov_matrix);
%对特征值进行排序
[sorted_eig_val, sorted_eig_idx] = sort(eig_val, 'descend');
%提取前k个特征向量
k = 10; %选择保留的特征数
principal_components = eig_vec(:, sorted_eig_idx(1:k));
%将训练集投影到主成分空间
proj_train = (face_source - mean_face) * principal_components;
%将测试集投影到主成分空间
proj_test = (face_test - mean_face) * principal_components;
%计算测试集与训练集之间的距离
dist = sqrt(sum((proj_test - proj_train).^2, 2));
%找出最近的训练集样本
[nearest_idx, min_dist] = min(dist, [], 2);
%输出识别结果
disp('识别结果:');
for i = 1:length(nearest_idx)
    disp(sprintf('测试集样本 %d 的最近邻训练集样本是 %d', i, nearest_idx(i)));
end
```

## Data preparation

## Calculation of eigenvalues

## Evaluation

```
clear all; %清除所有变量
load('face_source'); %加载训练集
load('face_test'); %加载测试集
%计算训练集的平均值和协方差矩阵
mean_face = mean(face_source, 2);
cov_matrix = cov(face_source - mean_face);
%计算协方差矩阵的特征值和特征向量
[eig_vec, eig_val] = eig(cov_matrix);
%对特征值进行排序
[sorted_eig_val, sorted_eig_idx] = sort(eig_val, 'descend');
%提取前k个特征向量
k = 10; %选择保留的特征数
principal_components = eig_vec(:, sorted_eig_idx(1:k));
%将训练集投影到主成分空间
proj_train = (face_source - mean_face) * principal_components;
%将测试集投影到主成分空间
proj_test = (face_test - mean_face) * principal_components;
%计算测试集与训练集之间的距离
dist = sqrt(sum((proj_test - proj_train).^2, 2));
%找出最近的训练集样本
[nearest_idx, min_dist] = min(dist, [], 2);
%输出识别结果
disp('识别结果:');
for i = 1:length(nearest_idx)
    disp(sprintf('测试集样本 %d 的最近邻训练集样本是 %d', i, nearest_idx(i)));
end
```