

哈尔滨工业大学计算机科学与技术学院

# 实验报告

课程名称：机器学习

课程类型：选修

实验题目：PCA

学号：1170300418

姓名：于新蕊

# 一、实验目的

实现一个PCA模型，能够对给定数据进行降维（即找到其中的主成分）

# 二、实验要求及实验环境

## 实验要求

- 首先人工生成一些数据（如三维数据），让它们主要分布在低维空间中，如首先让某个维度的方差远小于其它唯独，然后对这些数据旋转。生成这些数据后，用你的PCA方法进行主成分提取。
- 找一个人脸数据（小点样本量），用你实现PCA方法对该数据降维，找出一些主成分，然后用这些主成分对每一副人脸图像进行重建，比较一些它们与原图像有多大差别（用信噪比衡量）。

## 实验环境

windows64, pycharm, python3.0, anaconda

# 三、设计思想（本程序中的用到的主要算法及数据结构）

## 1.算法原理

PCA(主成分分析, Principal Component Analysis)是最常用的一种统计分析、简化数据集的方法。通过PCA可以使要分析的数据的维度降低,且这些维度还会包含原数据集的主要信息。关于PCA的推导有两种方式:最小投影距离和最大投影方差。

- 最小投影距离: 样本点到这个超平面的距离都足够近
- 最大投影方差: 样本点在这个超平面上的投影尽可能分开

### 1.1 中心化和坐标变化

假设 $m$ 个 $n$ 维数据 $X = (x^{(1)}, x^{(2)}, \dots, x^{(m)})$ , 我们令 $x^{(i)} = x^{(i)} - \mu$ , 其中 $\mu = \frac{1}{m} \sum_{i=1}^m x^{(i)}$ 。

这样我们会得到 $\sum_{i=1}^m x^{(i)} = 0$ , 这就是中心化。

中心化的目的是方便求协方差矩阵: 中心化后 $X^T X$ 就是样本集的协方差矩阵。

我们知道 $x^{(i)}$ 是一个 $n$ 维向量, 这个 $n$ 维空间中的基底是 $(1, 0, \dots, 0)^T, (0, 1, \dots, 0)^T, \dots, (0, 0, \dots, 1)^T$ 。如果我们换一组基底,  $x^{(i)}$ 又是另一种表现形式。

现在假设我们 $n$ 维空间中的基底是 $W = (w^{(1)}, w^{(2)}, \dots, w^{(n)})$ , 那么 $Z = W^T X$ 。

假设我们只取 $W$ 中的前 $d$ 个向量, 即 $W = (w^{(1)}, w^{(2)}, \dots, w^{(d)})$ , 其中 $d < n$ 。那么 $Z = W^T X$ 就是 $d$ 维空间下的一个坐标, 即 $Z$ 是 $X$ 从 $n$ 维空间到 $d$ 维空间的一个投影。

那么我们从 $d$ 维空间下的坐标 $Z$ 变化到 $n$ 维空间, 就是 $X = ZW$ 。

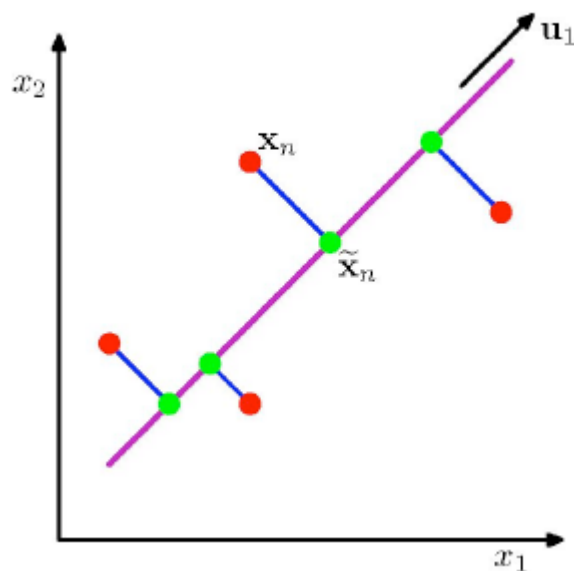
总结就是:

某种投影 $W = (w^{(1)}, w^{(2)}, \dots, w^{(d)})$

投影坐标:  $Z = W^T X$

还原坐标:  $X = ZW$

显然我们想要 $x^{(i)}$ 这些点的投影越分散越好、距离平面越近越好，就像这样：



因此我们有两种策略：最小投影距离和最大投影方差。事实上二者相当于优化同一个函数。

## 1.2 PCA的推导:基于最小投影距离

现在我们考虑整个样本集，我们希望所有的样本到这个超平面的距离足够近，即最小化

$$\arg \min_W \sum_{i=1}^m \|\bar{x}^{(i)} - x^{(i)}\|_2^2$$

将这个式子进行整理，可以得到：

$$\begin{aligned}
\sum_{i=1}^m \|\bar{x}^{(i)} - x^{(i)}\|_2^2 &= \sum_{i=1}^m \|Wz^{(i)} - x^{(i)}\|_2^2 \\
&= \sum_{i=1}^m (Wz^{(i)})^T (Wz^{(i)}) - 2 \sum_{i=1}^m (Wz^{(i)})^T x^{(i)} + \sum_{i=1}^m x^{(i)T} x^{(i)} \\
&= \sum_{i=1}^m z^{(i)T} z^{(i)} - 2 \sum_{i=1}^m z^{(i)T} W^T x^{(i)} + \sum_{i=1}^m x^{(i)T} x^{(i)} \\
&= \sum_{i=1}^m z^{(i)T} z^{(i)} - 2 \sum_{i=1}^m z^{(i)T} z^{(i)} + \sum_{i=1}^m x^{(i)T} x^{(i)} \\
&= - \sum_{i=1}^m z^{(i)T} z^{(i)} + \sum_{i=1}^m x^{(i)T} x^{(i)} \\
&= -\text{tr}(W^T (\sum_{i=1}^m x^{(i)} x^{(i)T}) W) + \sum_{i=1}^m x^{(i)T} x^{(i)} \\
&= -\text{tr}(W^T X X^T W) + \sum_{i=1}^m x^{(i)T} x^{(i)}
\end{aligned}$$

因此相当于

$$\arg \max_W \text{tr}(W^T X X^T W) \quad s.t. \quad W^T W = I$$

### 1.3 PCA的推导:基于最大投影方差

对于任意一个样本 $x^{(i)}$ ，在新的坐标系中的投影为 $W^T x^{(i)}$ ，在新坐标系中的投影方差为 $W^T x^{(i)} x^{(i)T} W$ 。要使所有的样本的投影方差和最大，也就是最大化 $\sum_{i=1}^m W^T x^{(i)} x^{(i)T} W$ ，即

$$\arg \max_W \text{tr}(W^T X X^T W) \quad s.t. \quad W^T W = I$$

### 1.4 参数求解

拉格朗日乘数法：

$$\begin{aligned}
J(W) &= \text{tr}(W^T X X^T W + \lambda(W^T W - I)) \\
\frac{\partial J(W)}{\partial W_i} &= 0 \\
X X^T W &= (-\lambda) W
\end{aligned}$$

可以看出 $W_i$ 是特征向量。

$$\text{tr}(W^T X X^T W) = \sum_{i=1}^d \lambda_i$$

因此 $W = (w^{(1)}, w^{(2)}, \dots, w^{(d)})$ ，其中 $w^{(i)}$ 是 $X X^T$ 第 $i$ 大的特征值对应的特征向量。

## 2.算法实现

给定样本集 $X = x^{(1)}, x^{(2)}, \dots, x^{(m)}$ 和低维空间的维数 $d$

1. 对所有的样本进行中心化操作:

1. 计算样本均值 $\mu = \frac{1}{m} \sum_{j=1}^m x^{(j)}$

2. 所有样本减去均值 $x^{(j)} = x^{(j)} - \mu, j \in 1, 2, \dots, m$

2. 计算样本的协方差矩阵 $X^T X$

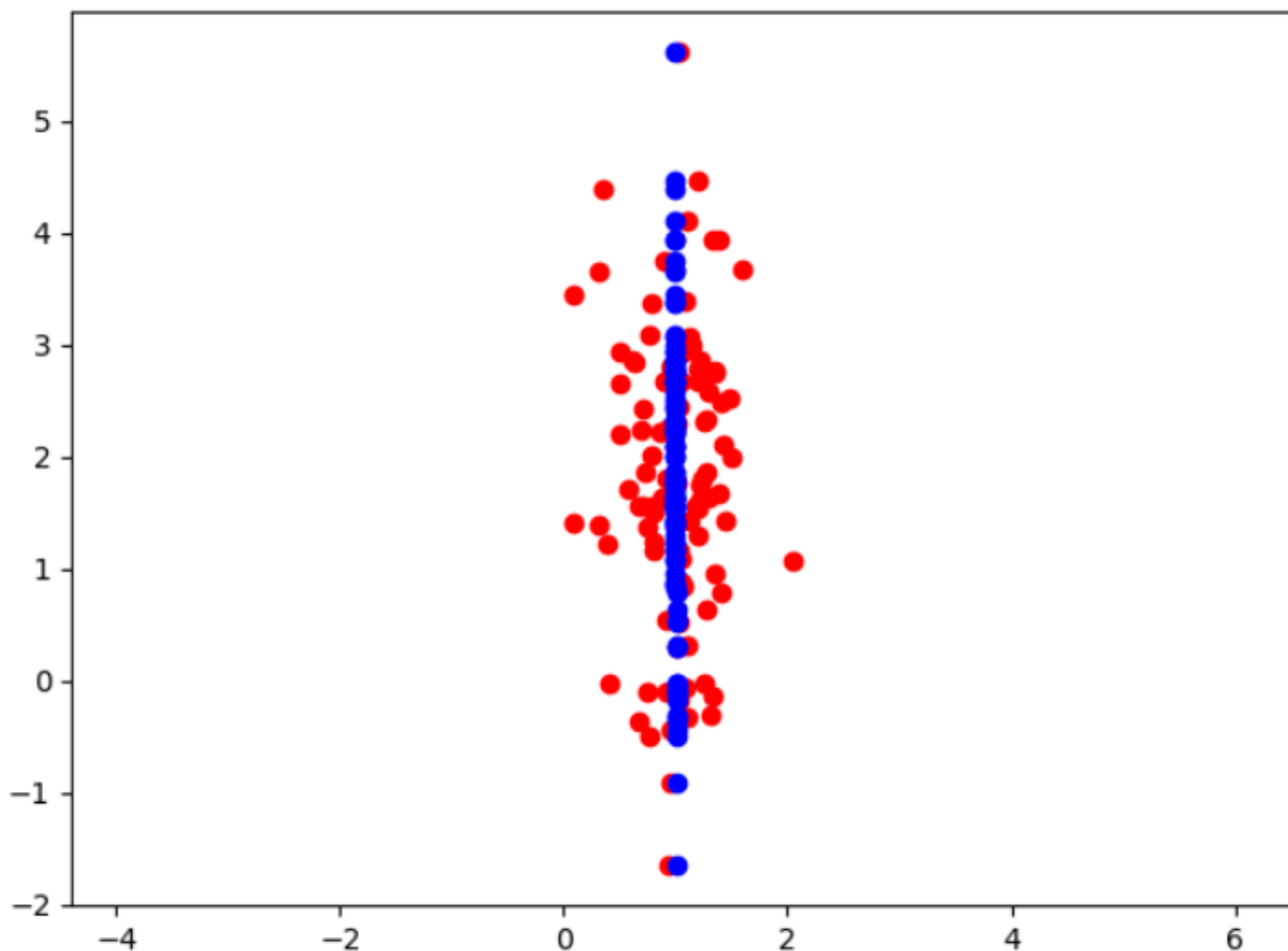
3. 对协方差矩阵 $X^T X$ 进行特征值分解

4. 取最大的 $d$ 个特征值对应的单位特征向量 $w_1, w_2, \dots, w_d$ , 构造投影矩阵 $W = (w_1, w_2, \dots, w_d)$

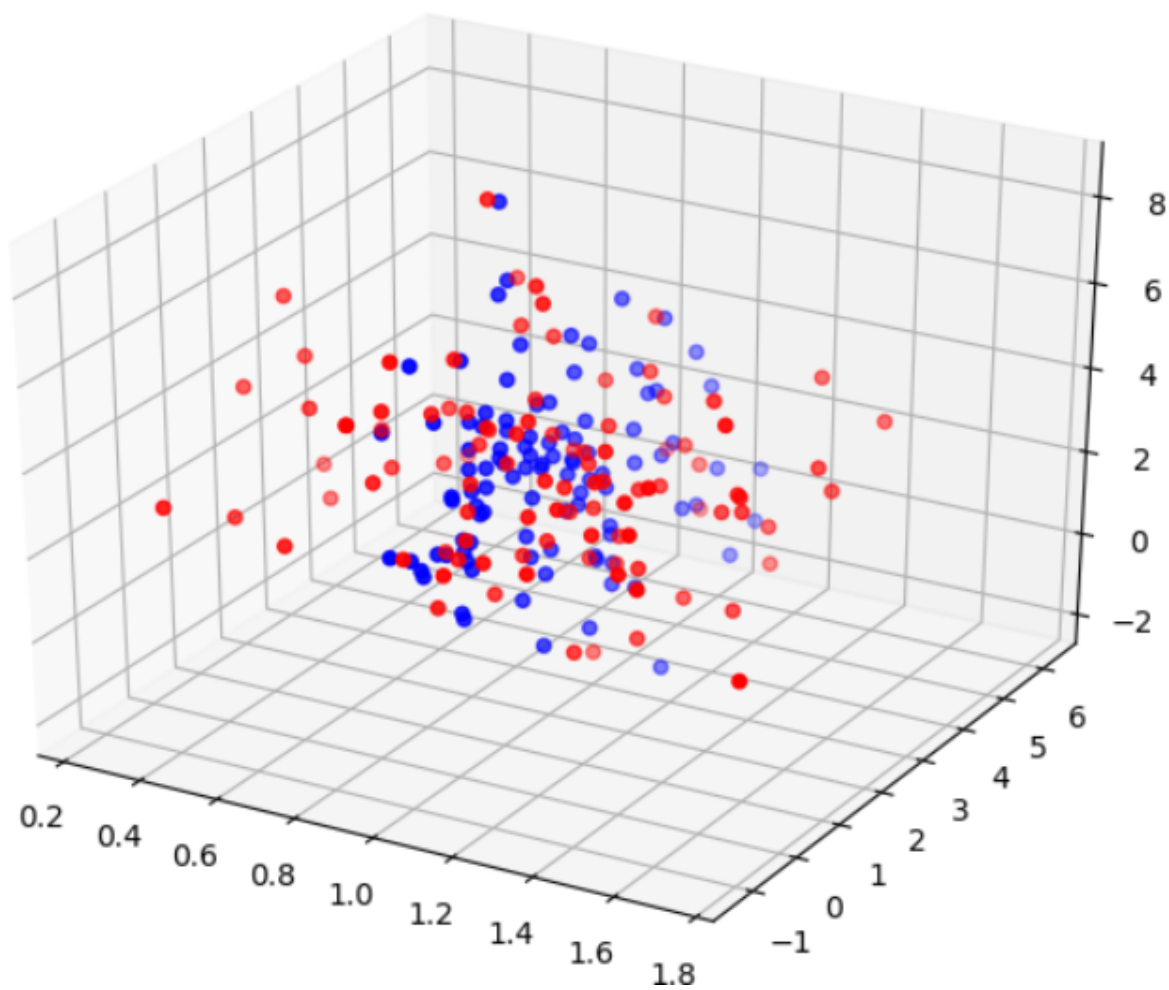
5. 输出投影矩阵 $W$

## 四、实验结果与分析

### 二维降一维



### 三维降二维



## 人脸数据

每一行单过一个样本，因此数据集大小是 $256 \times 256$ 。从256维降到20维的效果：



信噪比公式:

$$MSE = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{n-1} ||I(i, j) - K(i, j)||^2$$

$$PSNR = 10 \log_{10} \left( \frac{MAX_I^2}{MSE} \right) = 20 \log_{10} \left( \frac{MAX_I^2}{\sqrt{MSE}} \right)$$

人脸数据信噪比:

```
image 1: snr = 30.07384620805405
image 2: snr = 29.820386431657916
image 3: snr = 30.427266564547985
image 4: snr = 29.957863212811393
image 5: snr = 29.95718099520124
image 6: snr = 29.825134427527253
image 7: snr = 30.325307659609354
image 8: snr = 30.08843690753187
image 9: snr = 30.03770584756137
image 10: snr = 30.845106110036276
image 11: snr = 29.4130683257556
image 12: snr = 28.840226293974446
image 13: snr = 29.34407612583972
image 14: snr = 29.813708111686918
image 15: snr = 28.864170879874585
image 16: snr = 28.90971958775407
image 17: snr = 29.342122511604853
image 18: snr = 29.31517887778789
image 19: snr = 29.188209678501423
image 20: snr = 29.31632888939709
image 21: snr = 29.358984036995068
image 22: snr = 29.017930811989384
image 23: snr = 29.994222801517594
```

## 分析

三维数据降到二维时，降维后的数据分布在一个平面(2维)上，并且与方差最小的1维相垂直。二维数据同理。一个  $256 \times 256$  的数据，可以粗略用15维数据来表示，通过PCA降维的方法。

## 五、结论

---



PCA算法中舍弃了 $n - d$ 个最小的特征值对应的特征向量，用交大的 $d$ 特征值对应的特征向量来表示数据，称其为主成分。PCA在数据压缩消除冗余和数据噪音消除等领域都有广泛的应用，如图片压缩。

## 六、参考文献

[吴恩达机器学习\(Coursera\)](#)

## 七、附录：源代码（带注释）

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from PIL import Image

def generate_data(dim=3, size=100):
    if dim == 2:
        mean = [1, 2]
        cov = [[0.1, 0], [0, 2]]
    if dim == 3:
        mean = [1, 2, 3]
        cov = [[0.1, 0, 0], [0, 3, 0], [0, 0, 3]]
    data = []
    for index in range(size):
        data.append(np.random.multivariate_normal(mean, cov).tolist())
    data = np.array(data)
    data.reshape(size, dim)
    return data

def pca(data, reduced_dim=2):
    mean = np.mean(data, axis=0)
    data = data - mean
    cov = np.dot(data.T, data)
    eig_vals, eig_vecs = np.linalg.eig(cov)
    idx = np.argsort(-eig_vals, axis=0)[:reduced_dim:]
    # print(np.dot(np.dot(eig_vecs.T, cov), eig_vecs))
    eig_vecs = eig_vecs[:, idx]
    return mean, data, eig_vecs

def recover_data(w, centered_data, mean):
    return np.dot(np.dot(centered_data, w), w.T) + mean

def plot(origin_data, pca_data):
    fig = plt.figure()
    dim = origin_data.shape[1]
    if dim == 3:
        ax = Axes3D(fig)
        ax.scatter(origin_data[:, 0], origin_data[:, 1], origin_data[:, 2], c='r',
                    label='Original Data')
```

```

    ax.scatter(pca_data[:, 0], pca_data[:, 1], pca_data[:, 2], c='b', label='PCA Data')
if dim == 2:
    plt.axis('equal')
    plt.scatter(origin_data[:, 0], origin_data[:, 1], c='r', label='Original Data')
    plt.scatter(pca_data[:, 0], pca_data[:, 1], c='b', label='PCA Data')
plt.show()

def calc_snr(source, target):
    diff = source - target
    diff = diff ** 2
    mse = np.sqrt(np.mean(diff))
    return 20 * np.log10(255.0 / mse)

def test(path='./Japanese'):
    for i in range(1, 24):
        new_path = path + '/' + str(i) + '.tiff'
        img = np.array(Image.open(new_path).convert('L'), 'f').astype(np.float)
        mean, centered_data, w = pca(img, 20)
        # w = np.array(w, dtype=np.uint32)
        pca_data = recover_data(w, centered_data, mean)
        pca_data[pca_data < 0] = 0
        # print(pca_data)
        pca_data = pca_data.astype(np.uint8)
        new_image = Image.fromarray(pca_data, mode='L')
        new_image.save(path + '/' + str(i) + '_.tiff')
        print('image {}: snr = {}'.format(i, calc_snr(pca_data, img)))

if __name__ == '__main__':
    # data = generate_data(dim=3)
    # mean, centered_data, w = pca(data, reduced_dim=2)
    # pca_data = np.dot(np.dot(centered_data, w), w.T) + mean
    # plot(data, pca_data)
    test()

```