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：必修

实验题目：PCA模型实验

学号：1190202105

姓名：傅浩东

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

实验测试：

1. 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维度，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。
2. 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：Windows 11; Visual Studio Code; python 3.9.7

三. 设计实验（主要算法和数据结构）

1. 算法原理

(1) 基本介绍

在多元统计分析中，**主成分分析**（**PCA**）是一种统计分析、简化数据集的方法。它利用正交变换来对一系列可能相关的变量的观测值进行线性变换，从而投影为一系列线性不相关变量的值，这些不相关变量称为主成分。具体地，主成分可以看做一个线性方程，其包含一系列线性系数来指示投影方向。PCA对原始数据的正则化或预处理敏感（相对缩放）。

基本思想：

- 将坐标轴中心移到数据的中心，然后旋转坐标轴，使得数据在 C_1 轴上的方差最大，即全部n个数据个体在该方向上的投影最为分散。意味着更多的信息被保留下来。 C_1 成为**第一主成分**。
- C_2 **第二主成分**：找一个 C_2 ，使得 C_2 与 C_1 的协方差（相关系数）为0，以免与 C_1 信息重叠，并且使数据在该方向的方差尽量最大。
- 以此类推，找到第三主成分，第四主成分……第p个主成分。p个随机变量可以有p个主成分。

主成分分析经常用于减少数据集的维数，同时保留数据集当中对方差贡献最大的特征。这是通过保留低维主成分，忽略高维主成分做到的。这样低维成分往往能够保留住数据的最重要部分。但是，这也不是一定的，要视具体应用而定。由于主成分分析依赖所给数据，所以数据的准确性对分析结果影响很大。

主成分分析用于分析数据及建立数理模型，在原理上与主轴定理相似，其方法主要是通过对协方差矩阵进行特征分解，以得出数据的主成分（即特征向量）与它们的权值（即特征值）。PCA是最简单的以特征量分析多元统计分布的方法。其结果可以理解为对原数据中的方差做出解释：哪一个方向上的数据值对方差的影响最大？换而言之，PCA提供了一种降低数据维度的有效办法；如果分析者在原数据中除掉最小的特征值所对应的成分，那么所得的低维度数据必定是最优化的（也即，这样降低维度必定是失去信息最少的方法）。主成分分析在分析复杂数据时尤为有用，比如人脸识别。

PCA是最简单的以特征量分析多元统计分布的方法。通常，这种运算可以被看作是揭露数据的内部结构，从而更好地展现数据的变异度。如果一个多元数据集是用高维数据空间之坐标系来表示的，那么PCA能提供一幅较低维度的图像，相当于数据集在讯息量最多之角度上的一个投影。这样就可以利用少量的主成分让数据的维度降低了。

(2) 中心化

在PCA开始时都假设数据集进行了中心化，即：对于数据集 $D=\{x_1,x_2,\ldots,x_N\}$ ，其中 $\mathbf{x}_i \in \mathbb{R}^D$ 。对每个样本均进行如下操作：

$$\mathbf{x}_i \leftarrow \mathbf{x}_i - \frac{1}{N}\sum_{j=1}^{N}\mathbf{x}_j$$

$\overline{\mathbf{x}}=\frac{1}{N}\sum_{j=1}^{N}\mathbf{x}_j$ 称为样本集的 D 的均值向量。之所以进行中心化，是因为经过中心化之后的常规的线性变换就是绕原点的旋转变化，也就是坐标变换；以及 $\sum_{i=1}^{N}(\mathbf{x}_i-\overline{\mathbf{x}})(\mathbf{x}_i-\overline{\mathbf{x}})^T=S$ 就是样本集的协方差矩阵。经过中心化后的数据，有 $\sum_{j=1}^{N}\mathbf{x}_j=\mathbf{0}$ 。

PCA 的目标是找到 D 维数据下的 M 个主成分，将数据投影到维度为 M < D 的空间上：选择数据的最大的M个特征向量 $\{u_1,u_2,\ldots,u_M\}$，将每个输入向量 x 投影到这个子空间，即 $z_{n1}=\mathbf{x}_n^T u_1$.数据完全投影到 M 维的形式：

$$\begin{bmatrix}\mathbf{u}_1^\top \\ \cdots \\ \mathbf{u}_M^\top\end{bmatrix}\left[\mathbf{x}_1\cdots\mathbf{x}_N\right]=\left[\mathbf{z}_1\cdots\mathbf{z}_N\right]$$

(3) Maximize variance

使用 D 维单位向量 $\mathbf{u}_1$ 定义降维后空间的方向，那么有 $\mathbf{u}_1^\top\mathbf{u}_1=1$.目标是最大化投影数据相对于 $\mathbf{u}_1$ 的方差.

$$\frac{1}{N}\sum_{n=1}^{N}\left\{\mathbf{u}_1^T\mathbf{x}_n-\mathbf{u}_1^T\overline{\mathbf{x}}\right\}^2$$

$$=\frac{1}{N}\sum_{n=1}^{N}\left\{\mathbf{u}_1^T\mathbf{x}_n-\mathbf{u}_1^T\overline{\mathbf{x}}\right\}\left\{\mathbf{u}_1^T\mathbf{x}_n-\mathbf{u}_1^T\overline{\mathbf{x}}\right\}^\top$$

$$=\mathbf{u}_1^T\frac{1}{N}\sum_{n=1}^{N}\{\mathbf{x}_n-\overline{\mathbf{x}}\}\{\mathbf{x}_n-\overline{\mathbf{x}}\}^\top\mathbf{u}_1$$

$$=\mathbf{u}_1^T\mathbf{S}\mathbf{u}_1$$

考虑 $||\mathbf{u}_1||=1$，那么可以最大化下式：

$$\mathbf{u}_1^T\mathbf{S}\mathbf{u}_1+\lambda\left(1-\mathbf{u}_1^T\mathbf{u}_1\right)$$

求导，设导数为0，有：

$$\mathbf{S}\mathbf{u}_1=\lambda_1\mathbf{u}_1$$

易知，在这里 $\mathbf{u}_1$ 是数据的特征向量，而 λ_1 则是特征值。投影数据的最大方差由下式给出：

$$\mathbf{u}_1^T\mathbf{S}\mathbf{u}_1=\lambda_1$$

也就是说，最大方差就是在找最大特征值，而最大特征值所对应的特征向量就是投影后的方向。

(4) Minimize error

引入一组完整的 D 维基向量正交集，$\{\mathbf{u}_1,\ldots,\mathbf{u}_D\}$：

$$\mathbf{u}_i^T\mathbf{u}_j=\delta_{ij}$$

不失一般性地，可以有如下写法，其中 $\mathbf{x}_n$ 的表达式的含义是将坐标系旋转到由 $\mathbf{u}_i$ 定义的新系统：

$$\mathbf{x}_n = \sum_{i=1}^{D} \alpha_{ni}\mathbf{u}_i, \alpha_{ni} = \mathbf{x}_i^T\mathbf{u}_i.$$

用基向量的前 M 个表示 M 维线性子空间，其中 z_{ni} 取决于特定数据点，而 b_i 是常数：

$$\tilde{\mathbf{x}}_n = \sum_{i=1}^{M} z_{ni}\mathbf{u}_i + \sum_{i=M+1}^{D} b_i\mathbf{u}_i$$

我们的目标是最小化关于 $z_{ni}, \mathbf{u}_i$ 以及 b_i 的失真，其中 $z_{nj} = \mathbf{x}_n^T\mathbf{u}_j$，$b_j = \overline{\mathbf{x}}^T\mathbf{u}_j$：

$$\begin{aligned}
J &= \frac{1}{N}\sum_{n=1}^{N}\|\mathbf{x}_n - \tilde{\mathbf{x}}_n\|^2 \\
&= \frac{1}{N}\sum_{n=1}^{N}\left\|\sum_{i=1}^{M}(\alpha_{ni} - \mathbf{z}_{ni})\mathbf{u}_i + \sum_{j=M+1}^{D}(\alpha_{nj} - b_i)\mathbf{u}_j\right\|^2 \\
&= \sum_{n=1}^{N}\sum_{i=1}^{M}(\alpha_{ni} - \mathbf{z}_{ni})^2 + \sum_{j=M+1}^{D}(\alpha_{nj} - b_i)^2 \\
&= \frac{1}{N}\sum_{n=1}^{N}\sum_{i=M+1}^{D}\left(\mathbf{x}_n^T\mathbf{u}_i - \overline{\mathbf{x}}^T\mathbf{u}_i\right)^2 \\
&= \sum_{i=M+1}^{D}\mathbf{u}_i^T\mathbf{S}\mathbf{u}_i
\end{aligned}$$

通过选择 $\mathbf{u}_i$ 作为协方差矩阵的特征向量得到通用解：

$$\mathbf{S}\mathbf{u}_1 = \lambda_1\mathbf{u}_1$$

那么有下式：

$$J = \sum_{i=M+1}^{D}\lambda_i$$

当剩余的 **D-M** 个组成是具有最低特征值所对应的特征向量时，目标最小化，这个结果与 Maximize variance 相同。

2. 算法实现

```
输入：样本集 D = {x1, x2, ..., xm};
      低维空间维数 d.
过程：
     对所有样本进行中心化：xi <-- xi - 1/m sum(xi);
     计算样本的协方差矩阵XX^T;
     对协方差矩阵XX^T做特征值分解;
     取最大的 d 个特征值所对应的特征向量 w1, w2, ..., wd.
输出：投影矩阵 W = {w1, w2, ..., wd}.
```

3. 信噪比

峰值信噪比(PSNR) 是一个工程术语，表示信号的最大可能功率与影响其表示保真度的破坏噪声的功率之间的比率。由于许多信号具有非常宽的动态范围，因此通常使用分贝标度将PSNR 表示为对数量。PSNR 通常用于量化受有损压缩影响的图像和视频的重建质量。

PSNR 最容易通过**均方误差(MSE)** 定义。给定无噪声 $m \times n$ 单色图像 ***I*** 及其噪声近似值 **K**，*MSE*定义为：

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2$$

PSNR（以**dB**为单位）定义为如下，这里 MAX_I 是图像的最大可能像素值：

$$\begin{aligned} PSNR &= 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) \\ &= 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) \\ &= 20 \cdot \log_{10}(MAX_I) - 10 \cdot \log_{10}(MSE) \end{aligned}$$

PSNR 最常用于衡量有损压缩编解码器的重建质量（如图像压缩），这种情况下的信号是原始数据，噪声是压缩引入的误差。在比较压缩编解码器时，PSNR 是人类对重建质量感知的近似值。在没有噪声的情况下，两个图像是相同的，因此 MSE 为零，PSNR 是无限的。

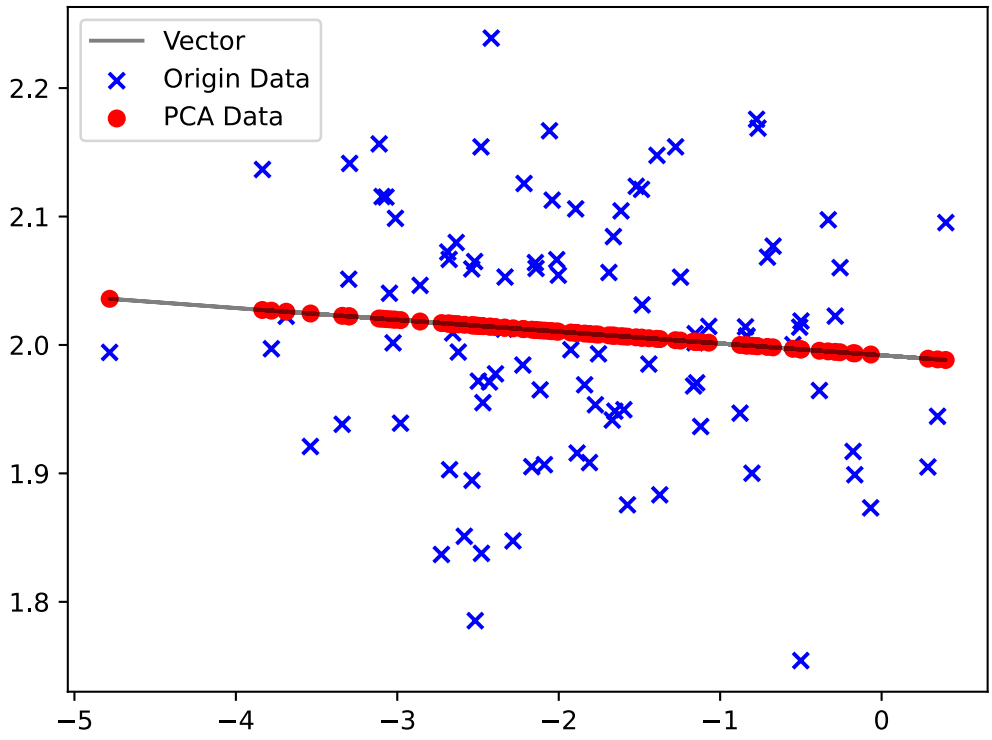
四. 实验结果与分析

1. 生成数据

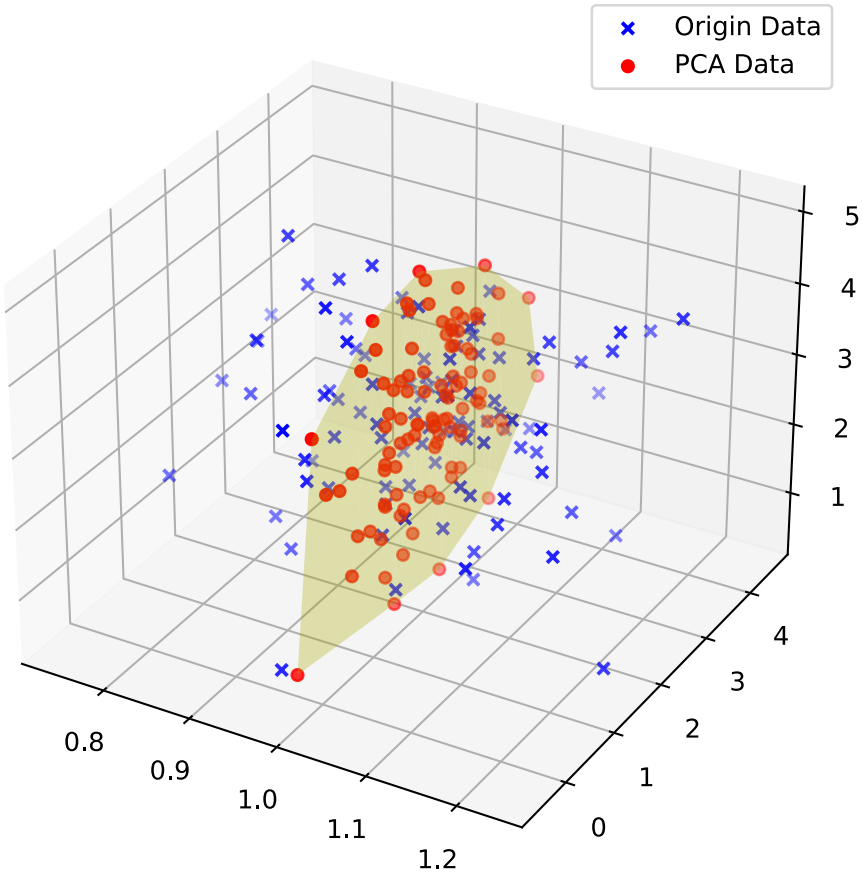
这里生成了一组二维和一组三维数据，并画图进行对比，结果如下。

首先，二维降为一维，其中采样时将某个维度的方差远小于其它维度，所以有下图所示结果，可以知道由于某一维方差很小，被视为了非主要信息，直观上基本上映射到了降维后的另一个维度（包含更多信息）：

$$\text{mean} = [-2, 2], \text{cov} = \begin{bmatrix} 1 & 0 \\ 0 & 0.01 \end{bmatrix}$$

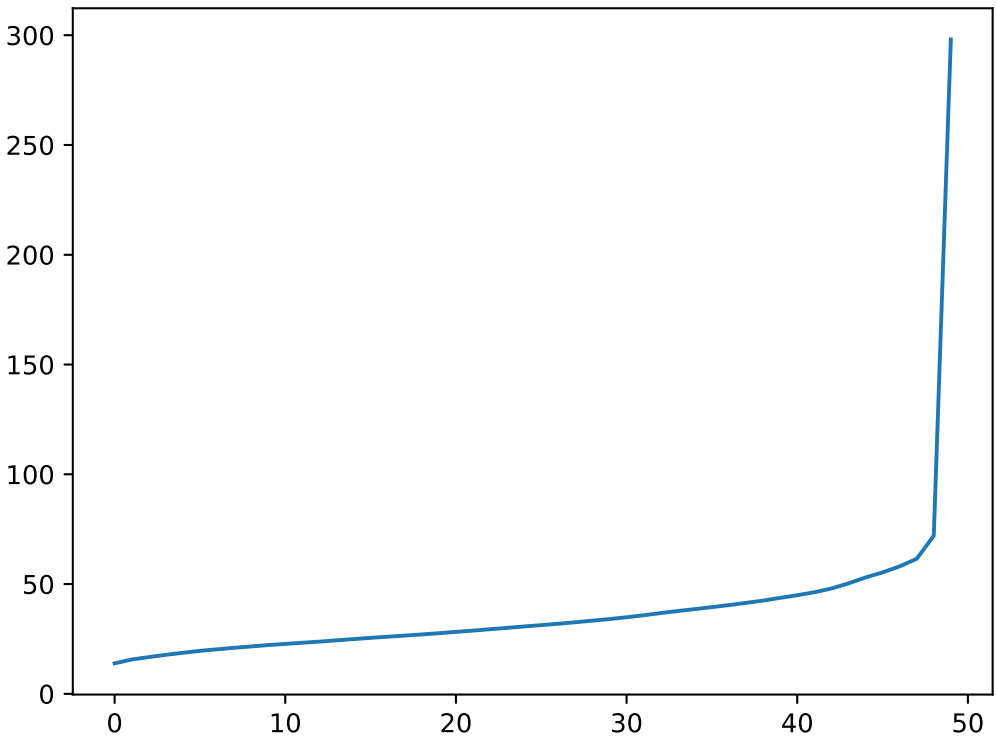


同理，有三维数据降至二维，被映射到一个平面上：



2. 实际二次元数据

这是一组全是50像素*50像素的灰度图片，降维后得到最大的M个特征向量和中心向量。首先对于峰值信噪比，当结果的维度越低，基本上PSNR就月底，也就是说与原图的相似度也就越低，这里给出一张图从 **0-49** 结果维度的PSNR变化图，可以看出峰值信噪比值随着维度升高逐渐增大。



对于全部的图，这里给出维度在 **[30, 10, 5, 3, 1]** 内的几个结果，可以看到当维度降低，清晰度也逐渐降低，这里不展示全部的图：

五. 结论

1. 可以使用**主成分分析 (PCA)** 进行降维、高维数据可视化、噪声过滤以及高维数据中的特征选择。由于 PCA 的多功能性和可解释性，它已被证明在各种环境和学科中都是有效的。给定任何高维数据集，从 PCA 开始，以便可视化点之间的关系（就像我们对数字所做的那样），了解数据中的主要方差（就像我们对特征脸所做的那样），并理解内在维度（通过绘制解释的方差比）。当然，PCA 并非对每个高维数据集都有用，但它提供了一种直接有效的途径来深入了解高维数据。
2. PCA 的主要弱点是它往往受数据中异常值的影响很大。

六. 参考文献

[1] https://en.wikipedia.org/wiki/Principal_component_analysis

[2] https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio

七. 附录：源代码（带注释）

PCA.py

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


def pca(data, reduced_dimension):
    rows, columns = data.shape
    x_mean = np.sum(data, axis=0) / rows
    decentralized_x = data - x_mean  # 去中心化
    cov = decentralized_x.T.dot(decentralized_x)  # 计算协方差
    eigenvalues, feature_vectors = np.linalg.eig(cov)  # 特征值分解
    index = np.argsort(eigenvalues) # 特征值从小到大排序后的下标序列
    # 选取最大的特征值对应的特征向量
    feature_vectors = np.delete(feature_vectors, index[:columns -
reduced_dimension], axis=1)
    return feature_vectors, x_mean

def psnr(source, target):
    # Peak signal-to-noise ratio 峰值噪音比
    rmse = np.sqrt(np.mean((source - target) ** 2))
    return 20 * np.log10(255.0 / rmse)
```

```
if __name__ == "__main__":

    number = 100
    # test for generation data
    mean_2 = [-2, 2]
    cov_2 = [[1, 0], [0, 0.01]]
    x = np.random.multivariate_normal(mean_2, cov_2, number)
    w, mu_x = pca(x, 1)
    pca_data = (x - mu_x).dot(w).dot(w.T) + mu_x
    # draw result
    plt.scatter(x[:, 0], x[:, 1], marker="x", color="b", label="Origin Data")
    plt.scatter(pca_data[:, 0], pca_data[:, 1], color='r', label='PCA Data')
    plt.plot(pca_data[:, 0], pca_data[:, 1], c="k", label="Vector", alpha=0.5)
    plt.legend(loc="best")
    plt.savefig('./images_result/2D21D.svg')
    plt.show()

    mean_3 = [1, 2, 3]
    cov_3 = [[0.01, 0, 0], [0, 1, 0], [0, 0, 1]]
    x = np.random.multivariate_normal(mean_3, cov_3, number)
    # PCA降维
    w, mu_x = pca(x, 2)
    # 重建数据
    pca_data = (x - mu_x).dot(w).dot(w.T) + mu_x
    # draw result
    fig = plt.figure()
    ax = Axes3D(fig, auto_add_to_figure=False)
    fig.add_axes(ax)
    ax.scatter(x[:, 0], x[:, 1], x[:, 2], marker="x", color="b", label='Origin
Data')
    ax.scatter(pca_data[:, 0], pca_data[:, 1], pca_data[:, 2], color='r',
label='PCA Data')
    ax.plot_trisurf(pca_data[:, 0], pca_data[:, 1], pca_data[:, 2], color="y",
alpha=0.3)
    plt.legend(loc="best")
    plt.savefig('./images_result/3D22D.svg')
    plt.show()

    # psnr of one picture
    pic = Image.open('./data/61853_2019.jpg')
    pic_array = np.array(pic)
    psnrS = list()
    for k in range(50):
        # PCA降维
        w, mu_x = pca(pic_array, k)
        # 重建数据
        pca_data = (pic_array - mu_x).dot(w).dot(w.T) + mu_x
        psnrS.append(psnr(pic_array, pca_data))
    # print(np.array(psnrS))
    plt.plot(list(np.arange(50)), psnrS)
    plt.savefig('./images_result/psnr.svg')
    plt.show()

    # picture data
    k_list = [30, 10, 5, 3, 1]
    size = len(k_list) + 1
```

```
    # list of all pictures
    file_list = os.listdir('data')
    f = open('PSNR.txt', 'w')
    for file in file_list:
        file_path = os.path.join('data', file)
        # Import picture
        pic = Image.open(file_path)
        # convert to ndarray
        pic_array = np.asarray(pic)
        x = pic_array
        res = file + " : "

        for k in k_list:
            w, mu_x = pca(pic_array, k)
            pca_data = (pic_array - mu_x).dot(w).dot(w.T) + mu_x
            x = np.concatenate((x, pca_data), axis=1)
            psnr_k = psnr(pic_array, pca_data)
            res = res + "k=" + str(k) + ", PSNR=" + "{:.2f}".format(psnr_k) +
"dB; "
        im = Image.fromarray(x)
        im = im.convert('L')
        im.save('./images/' + str(file_list[file_list.index(file)]))
        f.write(res + '\n')
```