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模型实验

学号：1190202401

姓名：陈豪

# 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

## 2.1 实验要求

（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。

（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）。

## 2.2 实验环境

- Windows 10
- python3.9.2
- vscode

# 三、设计思想（本程序中的用到的主要算法及数据结构）

## 3.1 算法原理

PCA(主成分分析，Principal Component Analysis)是最常用的一种降维方法。PCA的主要思想是将D维特征通过一组投影向量映射到K维上，这K维是全新的正交特征，称之为主成分，采用主成分作为数据的代表，有效地降低了数据维度，且保留了最多的信息。关于PCA的推导有多种方式：最大投影方差和最小投影距离，svd奇异值分解等等。这里以前两种视角来分析，不过无论从哪一种视角分析，最终得到的结果都是一致的。

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

### 3.1.1 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$\mathbf{X}=\{\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_m\}$，其中$\mathbf{x}_i\in\mathbb{R}^n$。对每个样本均进行如下操作：

$$\mathbf{x}_i\leftarrow\mathbf{x}_i-\mu$$

其中$\mu=\frac{1}{m}\sum_{j=1}^{m}\mathbf{x}_j$被称为样本集的$X$的中心向量。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$\frac{\sum_{i=1}^{m}\mathbf{x}_i\mathbf{x}_i^T=\mathbf{X}^T\mathbf{X}}{m}$就是样本集的协方差矩阵。经过中心化后的数据，有$\sum_{j=1}^{m}\mathbf{x}_j=\mathbf{0}$。设使用的投影坐标系的标准正交向量基为$\mathbf{U}=\{\mathbf{u}_1,\mathbf{u}_2,\ldots,\mathbf{u}_d\},\quad d<n$，每个样本降维后得到的坐标为:$\mathbf{z}=\{z_1,z_2,\ldots,z_d\}=\mathbf{U}^T\mathbf{x}$
因此，样本集与降维后的样本集表示为：

$$\mathbf{X}=\begin{bmatrix}\mathbf{x}_1^T\\ \vdots\\ \mathbf{x}_m^T\end{bmatrix}=\begin{bmatrix}x_{1,1}&x_{1,2}&\cdots&x_{1,n}\\ x_{2,1}&x_{2,2}&\cdots&x_{2,n}\\ \vdots&\vdots&\ddots&\vdots\\ x_{m,1}&x_{m,2}&\cdots&x_{m,n}\end{bmatrix},\mathbf{Z}=\begin{bmatrix}\mathbf{z}_1^T\\ \vdots\\ \mathbf{z}_m^T\end{bmatrix}=\begin{bmatrix}z_{1,1}&z_{1,2}&\cdots&z_{1,d}\\ z_{2,1}&z_{2,2}&\cdots&z_{2,d}\\ \vdots&\vdots&\ddots&\vdots\\ z_{m,1}&z_{m,2}&\cdots&z_{m,d}\end{bmatrix}$$

## 3.1.2 最大投影方差

对于原始数据样本点$\mathbf{x}_i$在降维后在新空间的超平面上的投影为$\mathbf{W}^T\mathbf{x}_i$。若使样本点的投影尽可能分开，应该使样本点在投影后的方差最大化,即使下式最大化：

$$\begin{aligned}\arg\max_{\mathbf{U}}&=\arg\max_{\mathbf{U}}\frac{1}{m}\sum_{i=1}^{m}\mathbf{U}^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{U}\\&=\arg\max_{\mathbf{U}}\frac{1}{m}tr(\mathbf{U}^T\mathbf{X}^\mathbf{T}\mathbf{X}\mathbf{U})\\&\mathbf{s.t.}\ \mathbf{U}^T\mathbf{U}=\mathbf{I}\end{aligned}$$

使用拉格朗日乘子法对于W矩阵上的其中一维w1可得

$$L(u_1)=\frac{1}{u}u_1^TSu_1+\lambda({u_1}^Tu_1-1)$$

其中的S是中心化后X的协方差矩阵，也就是说$S=\frac{1}{m}\mathbf{X}^\mathbf{T}\mathbf{X}$，对其求导并令导数为0可得

$$Su_1=\lambda u_1$$

所以$\lambda$是S的特征值，要使得L最大，则取最大的$\lambda$。而对于样本总体的最大投影方差，就是要选择前d个最大的特征值。

## 3.1.3 最小投影距离

现在考虑整个样本集，希望所有的样本到这个超平面的距离足够近，也就是得到 $Y$ 后，与 原来的$X$ 的距离最小。即求：

$$\begin{aligned}\arg\min_U\sum_{i=1}^{n}||\hat{x}_i-x_i||_2^2&=\arg\min_U\sum_{i=1}^{n}||x_iU^TU-x_i||_2^2\\&=\arg\min_U\sum_{i=1}^{n}((x_iU^TU)(x_iU^TU)^T-2(x_iU^TU)x_i^T+x_ix_i^T)\\&=\arg\min_U\sum_{i=1}^{n}(x_iU^TUU^TUx_i^T-2x_iU^TUx_i^T+x_ix_i^T)\\&=\arg\min_U\sum_{i=1}^{n}(-x_iU^TUx_i^T+x_ix_i^T)\\&=\arg\min_U-\sum_{i=1}^{n}x_iU^TUx_i^T+\sum_{i=1}^{n}x_ix_i^T\\&\Leftrightarrow\arg\min_U-\sum_{i=1}^{n}x_iU^TUx_i^T\\&\Leftrightarrow\arg\max_U\sum_{i=1}^{n}x_iU^TUx_i^T\\&=\arg\max_U\ tr(U(\sum_{i=1}^{n}x_i^Tx_i)U^T)\\&=\arg\max_U\ tr(UX^TXU^T)\qquad s.t.\ UU^T=1\end{aligned}$$

上式的优化目标同最大投影方差一样，因此最终结果一致。

## 3.2算法实现

给定样本集$\mathbf{D}=\{\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_m\}$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$\mu=\frac{1}{m}\sum\limits_{j=1}^{m}\mathbf{x}_j$
    2. 所有样本减去均值$\mathbf{x}_j=\mathbf{x}_j-\mu,\ j\in\{1,2,\ldots,m\}$
2. 计算样本的协方差矩阵$\mathbf{X}^T\mathbf{X}$
3. 对协方差矩阵$\mathbf{X}^T\mathbf{X}$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$\mathbf{u}_1,\mathbf{u}_2,\ldots,\mathbf{u}_d$，构造投影矩阵$\mathbf{U}=(\mathbf{u}_1,\mathbf{u}_2,\ldots,\mathbf{u}_d)$
5. 输出投影矩阵$\mathbf{U}$与样本均值$\mu$

# 四、实验结果与分析

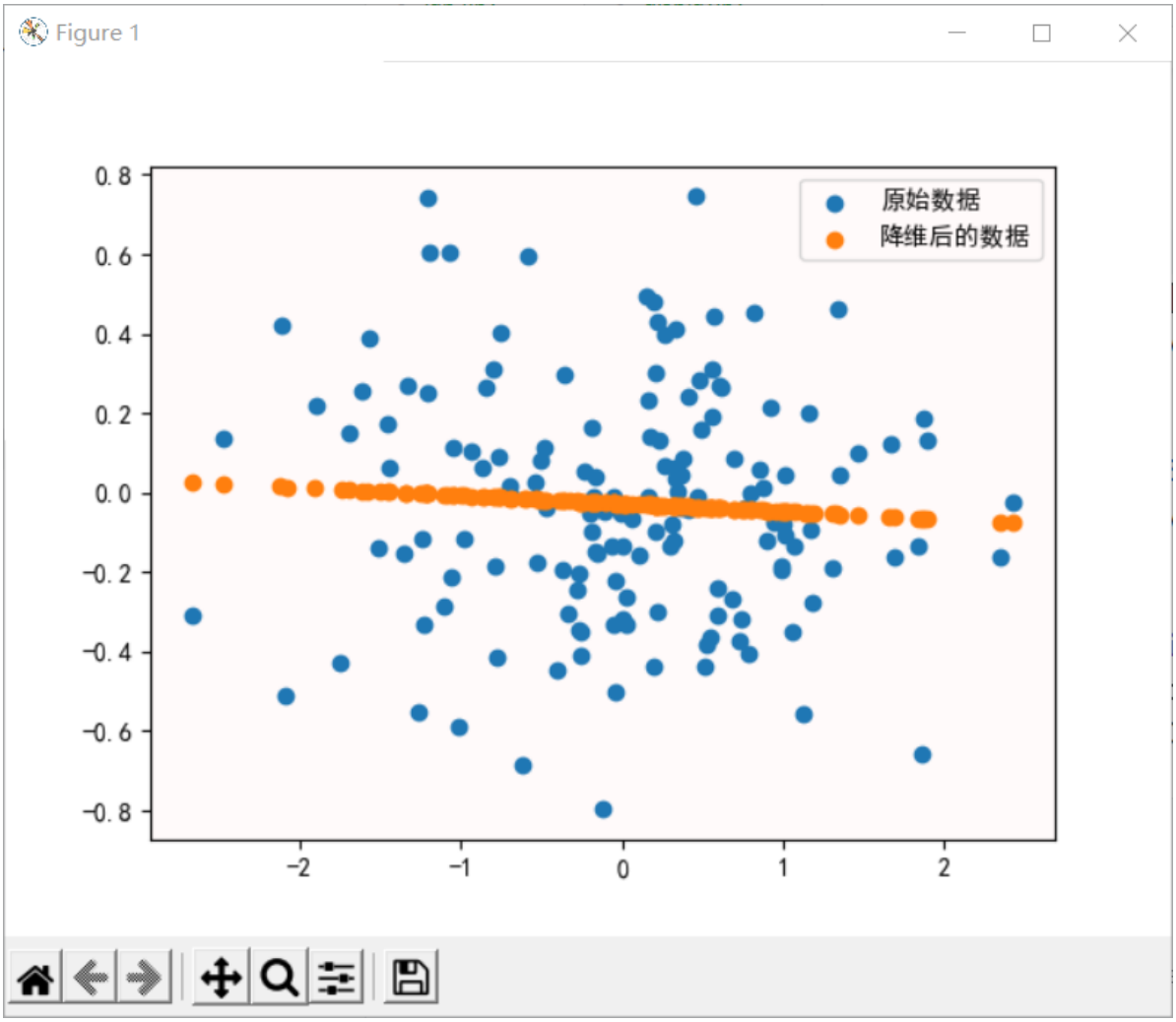
## 4.1 生成数据测试

### 4.1.1 二维降维到一维

在2维数据的测试中，选择使用2维高斯分布产生样本，使用的参数为：

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

使用PCA后得到的结果如图所示



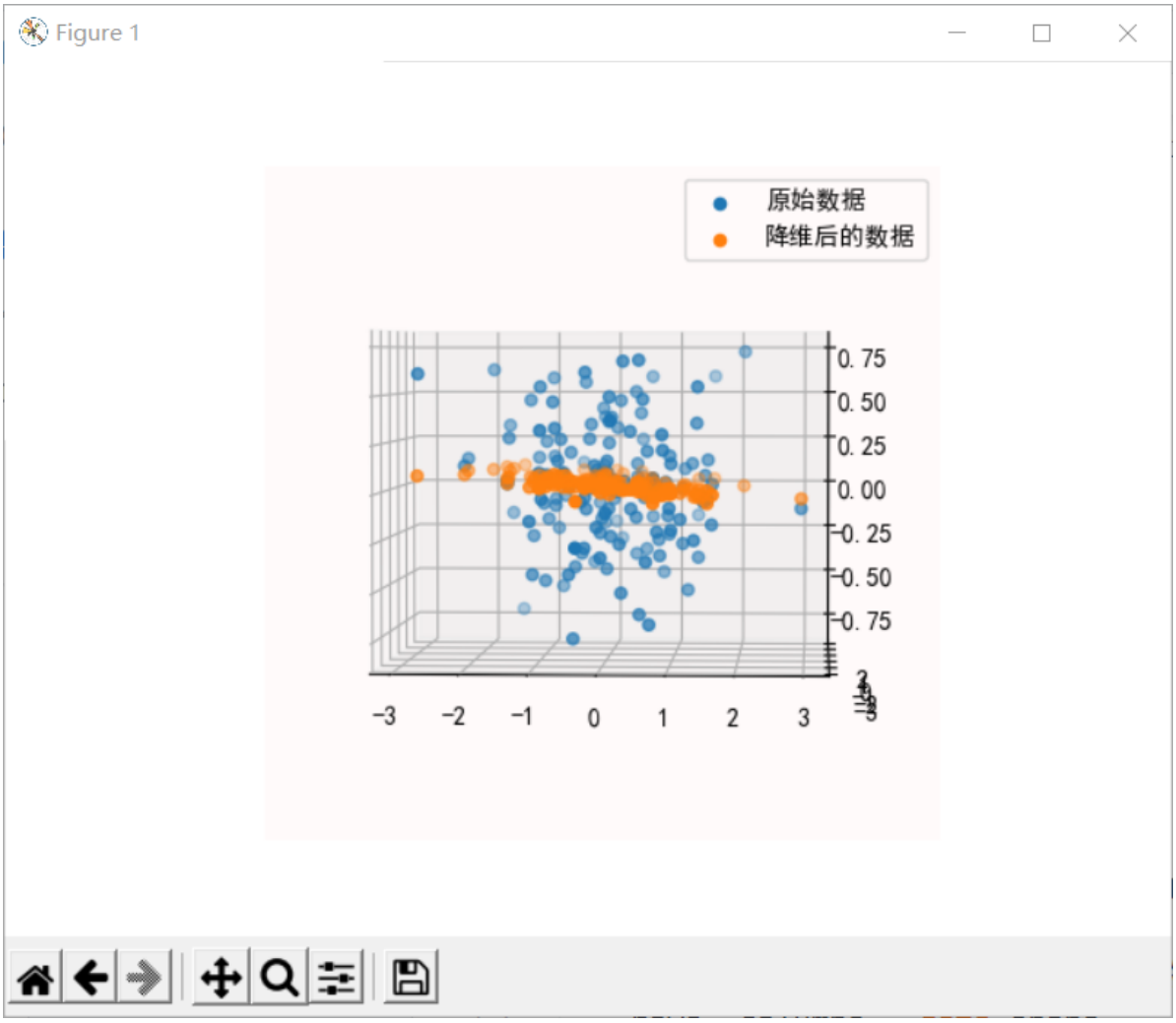

从上面的公式可以看出在横轴的方差大，纵轴的方差小，因此所以在进行PCA之后得到的直线与横轴接近。

### 4.1.2 三维降维到二维

在3维数据的测试中，使用3维高斯分布随机产生样本，使用的参数为：

$$\mathbf{mean} = [0, 0, 0],\ \mathbf{cov} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0.1 \end{bmatrix}$$

可以看出第三维的方差小，而前面两位的方差大于第三维。因此使用PCA后得到的结果应该垂直z轴。而事实正是如此。




## 4.2 人脸数据测试

这里使用了段然同学提供的50x50_grey_anime_face进行测试，分别演示不同的降维情况。

使用的信噪比公式如下

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

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

在这里M = 50，N=50，使用的MAX是255

### 4.2.1 降维到20维

```
总的信噪比:  (14.012396950255434+0j)
```

上面的是总的信噪比。

这里选择一例图片作为对比看看结果

原图片：

降维后的图片：

## 4.2.2 降维到100维

```
总的信噪比：  (17.236577951875244+0j)
```

同样选择一例图片查看对比结果

原图片：

降维后的图片：

## 4.2.2 降维到1250维

```
总的信噪比：  (44.529763759945936+0j)
```

同样选择一例图片查看对比结果

原图片：

降维后的图片：

从上面的实验结果可以看出，随着维度降低信噪比降低，同时图片变得模糊，这符合我们对PCA降维的直观印象。

# 五、结论

- PCA能够将特征数量降低，在这之中虽然会损失一些精度，但是只要降低的特征数量合理，就能够使得图片保存很好的模样，同时方便分析
- 在做PCA的时候先中心化能够方便计算
- pca压缩后的主成分之间是正交关系
- 一些样本存在多维特征，但事实上仅有部分特征是有意义的，其他的特征与这些特征存在线性关系，因此使用PCA能够降低这些无效的特征，方便找到更加合适的特征。

# 六、参考文献

https://www.bilibili.com/video/BV1aE411o7qd

# 七、附录：源代码（带注释）

## lab4.py

```
from getdata import generateData
from display import display2dim, display2dimWithResult, display3dim,
display3dimWithResult
import numpy as np
from pca import PCA
from PIL import Image
import numpy as np
import os

def readData(fileName):
    """
    从指定fileName中读取图片并转化成对应的ndarray
    """
    im = Image.open('50x50_grey_anime_face\data\\{}'.format(fileName))
    data = np.array(im).reshape(2500)
    return data

def saveData(data,fileName):
    """
    将处理后的数据保存到新的文件夹
    """
    im = Image.fromarray(np.uint8(data))
    im = im.convert('L')
    im.save('50x50_grey_anime_face\\newData\\{}'.format(fileName))

def processData(targetdim):
    file_dir = "50x50_grey_anime_face\data"
    sumPSNR = 0
    rawData = []
    for root, dirs, files in os.walk(file_dir,topdown=True):
        for fileName in files:
            rawData.append(readData(fileName))
    rawData = np.array(rawData)
    model = PCA(rawData,targetdim)
```

```
    featureVectors, mean = model.train()
    lowDimData = model.transform(featureVectors)
    i = 0
    for root, dirs, files in os.walk(file_dir,topdown=True):
        for fileName in files:
            sumPSNR = calPSNR(rawData[i],lowDimData[i])
            saveData(lowDimData[i].reshape(50,50),fileName)
            i += 1
    print('总的信噪比：',sumPSNR)


def calPSNR(source,target):
    """
    计算图片峰值信噪比
    """
    diff = source - target
    diff = diff ** 2
    rmse = np.sqrt(np.mean(diff))
    return 20 * np.log10(255.0 / rmse)


if __name__ == '__main__':
    #二维数据
    mean = np.array([0,0])
    cov_xy = np.array([[1,0],[0,0.1]])
    data2d = generateData(mean,cov_xy,150)
    #三维数据
    mean = np.array([0,0,0])
    cov_xy = np.array([[1,0,0],[0,1,0],[0,0,0.1]])
    data3d = generateData(mean,cov_xy,150)
    #对于人脸数据集
    targetdim = 2

    model = PCA(data2d,1)
    featureVectors, mean = model.train()
    lowDimData = model.transform(featureVectors)
    display2dimWithResult(data2d,lowDimData)

    model = PCA(data3d,2)
    featureVectors, mean = model.train()
    lowDimData = model.transform(featureVectors)
    display3dimWithResult(data3d,lowDimData)

    processData(targetdim)
```

## getdata.py

```
import numpy as np

def generateData(mean,cov_xy,size):
    """
    mean:均值
    cov_xy:方差矩阵
    size:数目
    """
    data = np.random.multivariate_normal(mean, cov_xy, size)
    return data
```

## pca.py

```
import numpy as np

class PCA(object):

    def __init__(self,data,targetDim) -> None:
        """
        data:数据
        targetDim:目标维数
        """
        self.data = self.__deCenter(data)
        self.size = self.data.shape[0]
        self.colums = self.data.shape[1]
        self.targetDim = targetDim
        self.__calVar()

    def __deCenter(self,data):
        """
        去中心化
        """
        self.mu = np.mean(data,axis=0)
        return data - self.mu

    def __calVar(self):
        """
        计算协方差矩阵
        """
        self.cov = np.dot(self.data.T,self.data)/self.size


    def train(self):
        """
        计算PCA
        """
        eigenvalues, featureVectors = np.linalg.eig(self.cov)  # 特征值分解
        sortedEigenvalues = np.argsort(eigenvalues)
        # 选取最大的特征值对应的特征向量
        featureVectors = np.delete(featureVectors,
sortedEigenvalues[:self.colums - self.targetDim], axis=1)
        return featureVectors, self.mu

    def transform(self,featureVectors):
        """
        将数据从x转化为x帽
```

```
        """
        return self.data.dot(featureVectors).dot(featureVectors.T)+self.mu
```

# display.py

```
import numpy as np
import matplotlib.pyplot as plt

plt.rcParams['axes.facecolor']='snow'
plt.rcParams['font.sans-serif'] = ['SimHei']  #显示中文
plt.rcParams['axes.unicode_minus']=False #用来正常显示负号

def display3dim(data):
    """
    需要一个n*3的矩阵
    """
    ax = plt.axes(projection='3d')
    ax.scatter(data[:,0],data[:,1],data[:,2],c=data[:,2])
    plt.show()


def display2dim(data):
    """
    需要一个n*2的矩阵
    """
    ax = plt.axes()
    ax.scatter(data[:,0],data[:,1])
    plt.show()

def display2dimWithResult(rawdata,transformedData):
    """
    在2d的情况下对比结果
    """
    ax = plt.axes()
    ax.scatter(rawdata[:,0],rawdata[:,1],label='原始数据')
    ax.scatter(transformedData[:,0],transformedData[:,1],label='降维后的数据')
    plt.legend()
    plt.show()

def display3dimWithResult(rawdata,transformedData):
    """
    在3d的情况下对比结果
    """
    ax = plt.axes(projection='3d')
    ax.scatter(rawdata[:,0],rawdata[:,1],rawdata[:,2],label='原始数据')

 ax.scatter(transformedData[:,0],transformedData[:,1],transformedData[:,2],label
='降维后的数据')
    plt.legend()
    plt.show()
```