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：1170301007
姓名：沈子鸣

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：

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）。

实验环境：

Windows 10 专业教育版；python 3.7.4；jupyter notebook 6.0.1

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

   本次实验用到的算法是PCA(主成分分析)，其作用是从高维数据中提取一部分特征（主成分优先），根据这些特征向低维变换，常用于数据压缩和高维数据可视化。

   PCA一共有两种形式，最大方差形式和最小误差形式。本次实验中用到的是最大方差形式。

   最大方差形式的通俗描述，是将高维数据向高维空间中的某一个平面投影，以求得投影得到的数据点的方差最大，从而尽可能多的保留原数据的特征。如：一个三维空间的椭球体，投影为椭圆的方差最大，而不是侧投影的圆方差最大，投影为椭圆可以更多的保留原数据的特征。

   最大方差形式推导过程如下：

   设有一组数据$X=\{x_1,x_2,\dots,x_n\}$，其中$x_i,i\in[1,n]$是D维空间中的向量，即X的大小是$D\times N$的。又规定μ_1是一个投影方向，则投影距离为$z=x^T\mu_1$。

   从而有，投影均值为

   $$\overline{x}=\frac{1}{N}\Sigma_{n=1}^{N}\mu_1^Tx_n$$

   投影方差为

   $$\frac{1}{N}\Sigma_{n=1}^{N}(\mu_1^Tx_n-\mu_1^T\overline{x})^2$$
   $$=\mu_1^T(\frac{1}{N}\Sigma_{n=1}^{N}(x_n-\overline{x})(x_n-\overline{x})^T)\mu_1$$

   设$S=\frac{1}{N}(x_n-\overline{x})(x_n-\overline{x})^T$，则方差为$\mu_1^TS\mu_1$。

   因为μ_1是投影方向，做归一化后，设它是单位向量，则有$\mu_1^T\mu_1=1$，用拉格朗日乘子法最大化目标函数：

   $$L(\mu_1)=\mu_1^TS\mu+\lambda(1-\mu_1^T\mu_1)$$

   解的

   $$S\mu_1=\lambda\mu_1$$

   显然，μ_1和λ是一组对应的S的特征向量和特征值。又

   $$\mu_1^TS\mu_1=\lambda$$

   故求得最大化方差，即求最大的特征值。要将D维的数据降维到P维度，只需计算前P个最大的特征值，将其对应的特征向量组合成特征向量矩阵U(DxP)，然后用U右乘数据矩阵的转置即可实现降维压缩。

2. 算法的实现

详细过程请见附录源代码。这里只给出关键代码的解释。

PCA算法实现如下：

```
'''
    对数据data用PCA降至k维
    data.shape = (N, D)
    返回值：
    c_data，中心化数据，shape=(N, D)
    eigVectsReduce，特征向量矩阵，shape=(D, k)
    data_mean，降维前数据均值，shape=(1, D)
'''
def PCA(data, k):
    rows, cols = data.shape
    data_mean = np.sum(data, 0) / rows
    c_data = data - data_mean # 中心化
    covMat = np.dot(c_data.T, c_data)
    eigVals, eigVects = np.linalg.eig(covMat) # 对协方差矩阵(D,D)求特征值和特征向量
    eigValIndex = np.argsort(eigVals) # 特征值排序
    eigVectsReduce = eigVects[:,eigValIndex[:-(k+1):-1]] # 取前k个特征值对应的特征向量
    return c_data, eigVectsReduce, data_mean
```

按照公式，先将得到原数据的均值（按维度取均值），并以此对原数据做中心化，然后生成中心化数据的协方差矩阵`covMat`，然后求`covMat`的特征值和特征向量，对特征值排序，取前`k`个最大的特征值，并选取它们对应的特征向量组成特征向量矩阵，最后返回中心化数据`c_data`，特征向量矩阵`eigVectsReduce`，降维前数据均值`data_mean`。

```
c_data, eigVectsReduce, data_mean = PCA(data, 1) # PCA降维
eigVectsReduce = np.real(eigVectsReduce) # 一旦降维维度超过某个值，特征向量矩阵将出现复向量，对其保留实部
pca_data = np.dot(c_data, eigVectsReduce) # 计算降维后的数据
recon_data = np.dot(pca_data, eigVectsReduce.T) + data_mean # 重构数据
```

得到这三个变量后，若要对原数据降维，则先对特征向量矩阵做去除虚部处理（可能存在虚部）。然后，用中心化数据左乘特征向量矩阵，就得到了投影数据（即降维后数据）`pca_data`。要重构回原数据，只需将`pca_data`右乘一个特征向量矩阵的转置，最后再加上原数据均值即可。

四、实验结果与分析

1. 人工生成数据

本次实验人工生成数据采用的是瑞士卷数据，瑞士卷数据是分布在三维空间的瑞士卷结构，正投影为漩涡状，侧投影为矩形。根据瑞士卷的“厚度”变化，其特征投影面将从“薄瑞士卷”的正投影，变为“厚瑞士卷”的侧投影。

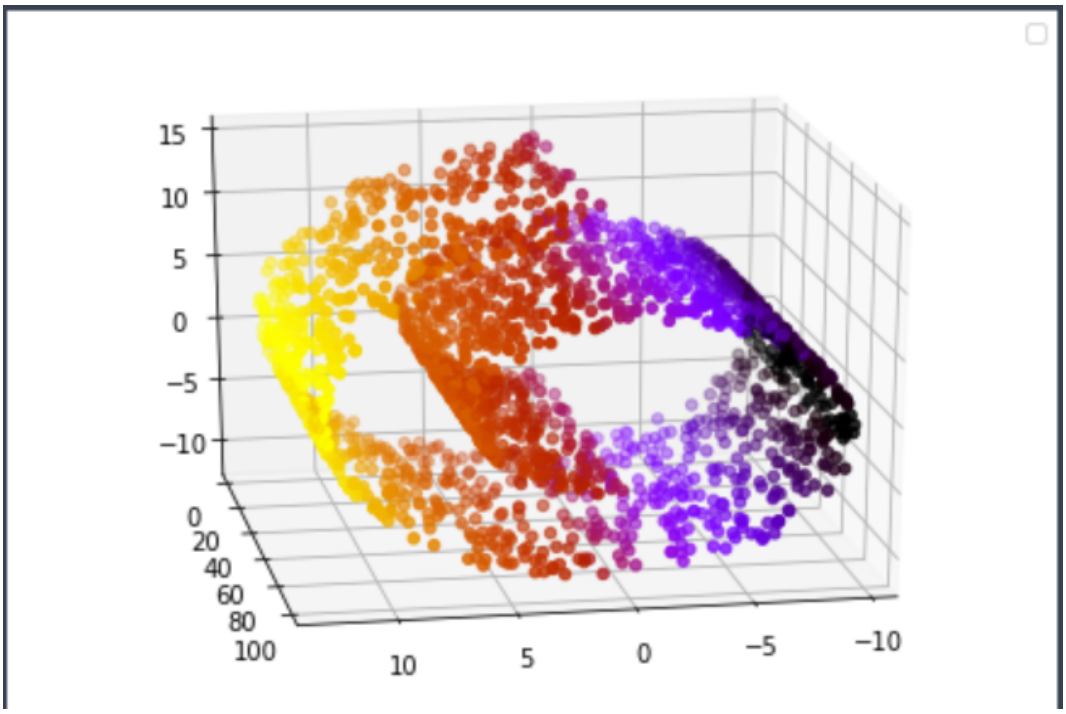
数据生成代码如下：

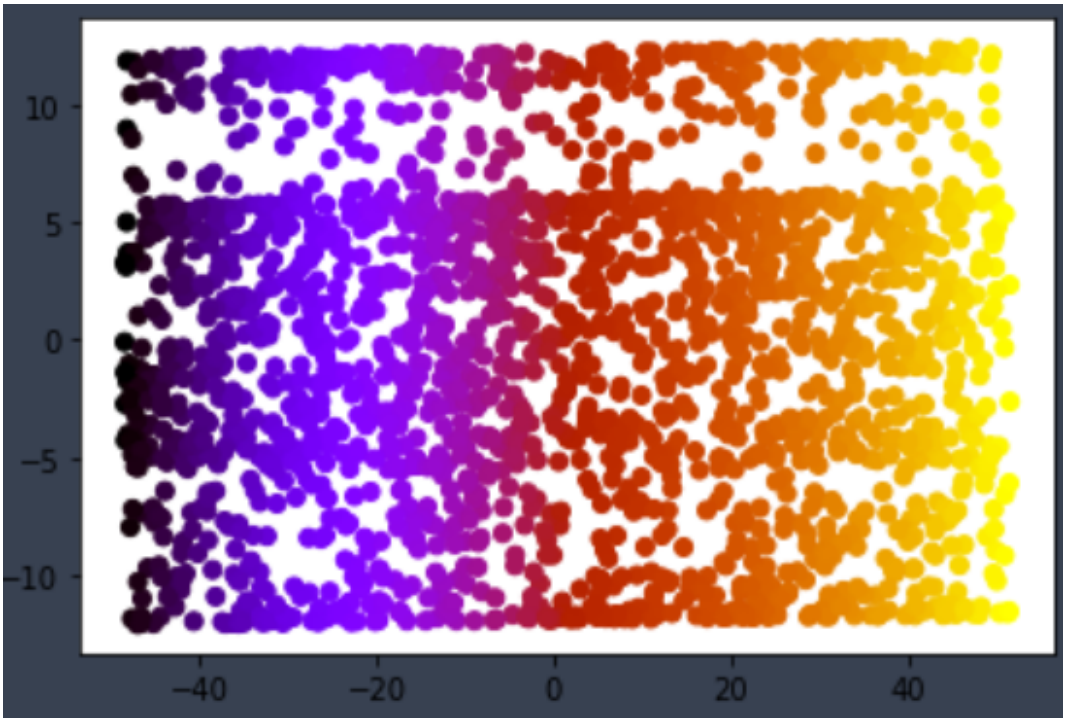
```python
def make_swiss_roll(n_sample=100, noise=0.0, y_scale=100):
    t = 1.5 * np.pi * (1 + 2 * np.random.rand(1, n_sample))
    x = t * np.cos(t)
    y = y_scale * np.random.rand(1, n_sample)
    z = t * np.sin(t)
    X = np.concatenate((x, y, z))
    X += noise * np.random.randn(3, n_sample)
    X = X.T
    return X
```

n_sample为每个维度上生成的数据点数，默认值为100。noise为噪声值，即标准瑞士卷在三个维度上分别加上噪声，默认值为0. y_scale决定了瑞士卷的“厚度”，即结构的y值区间，默认为100.

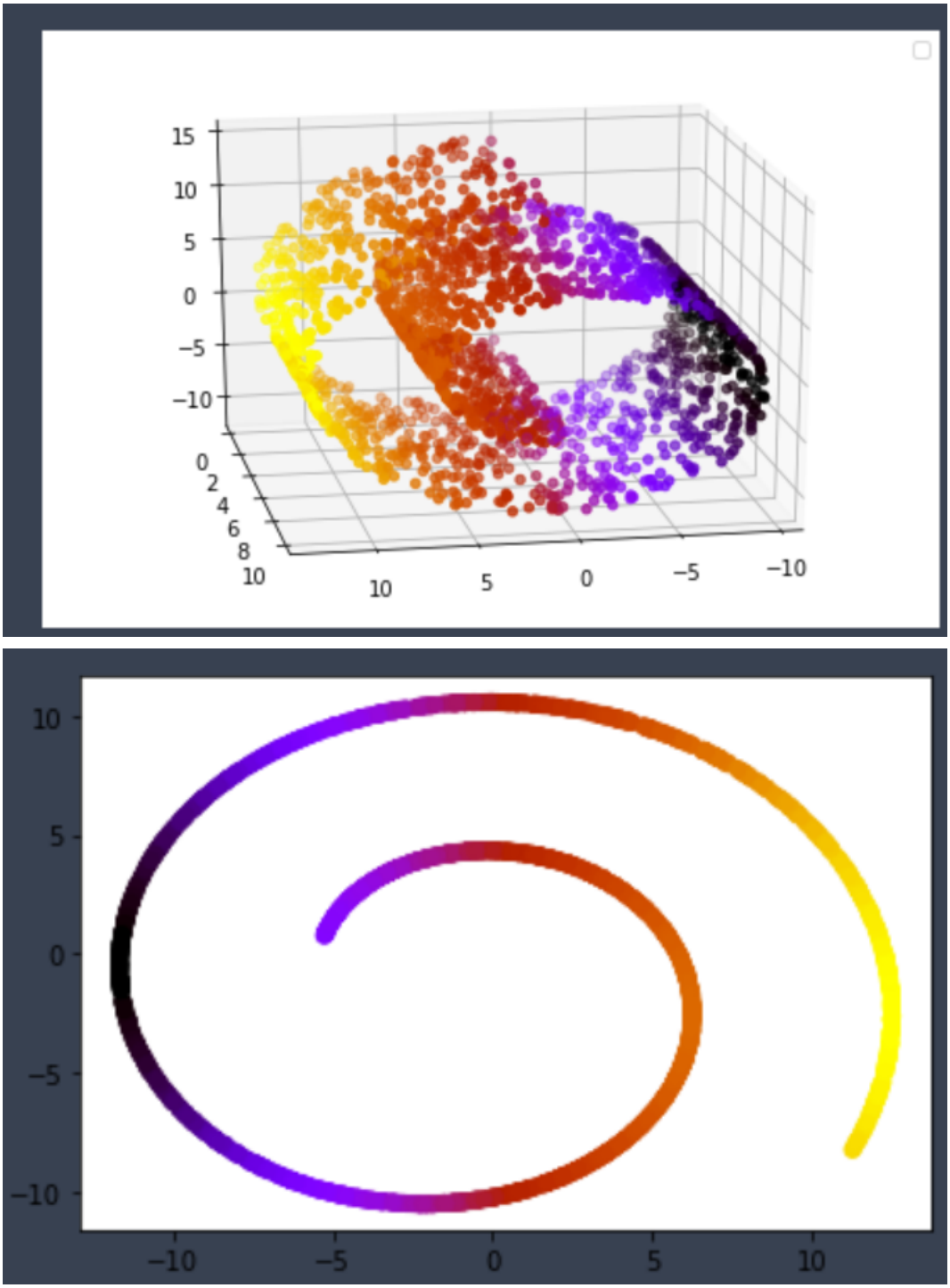
实验结果如下：

1. n_sample=2000, noise=0.0, y_scale=100

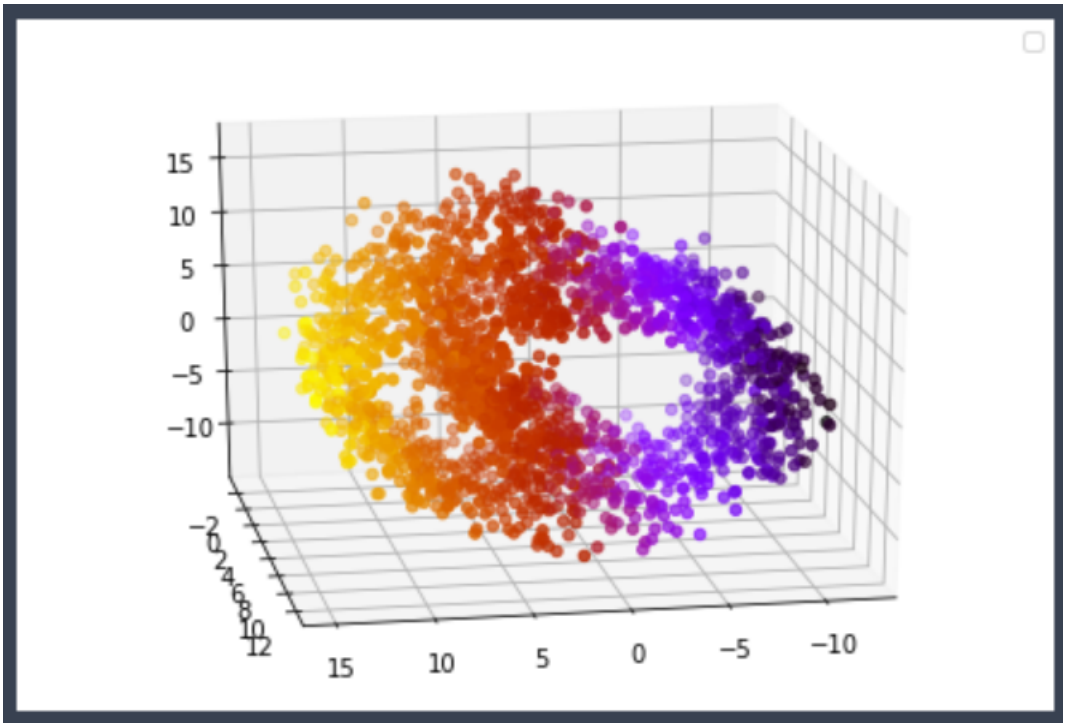


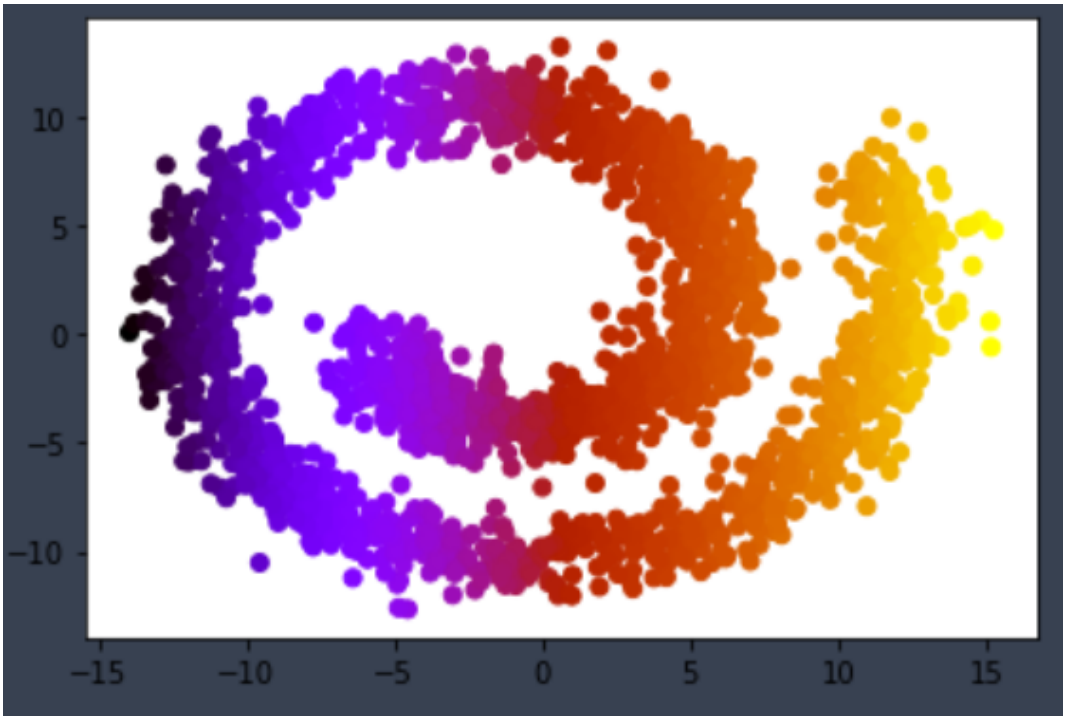


2. n_sample=2000, noise=0.0, y_scale=100



3. n_sample=2000, noise=0.0, y_scale=100





分析：

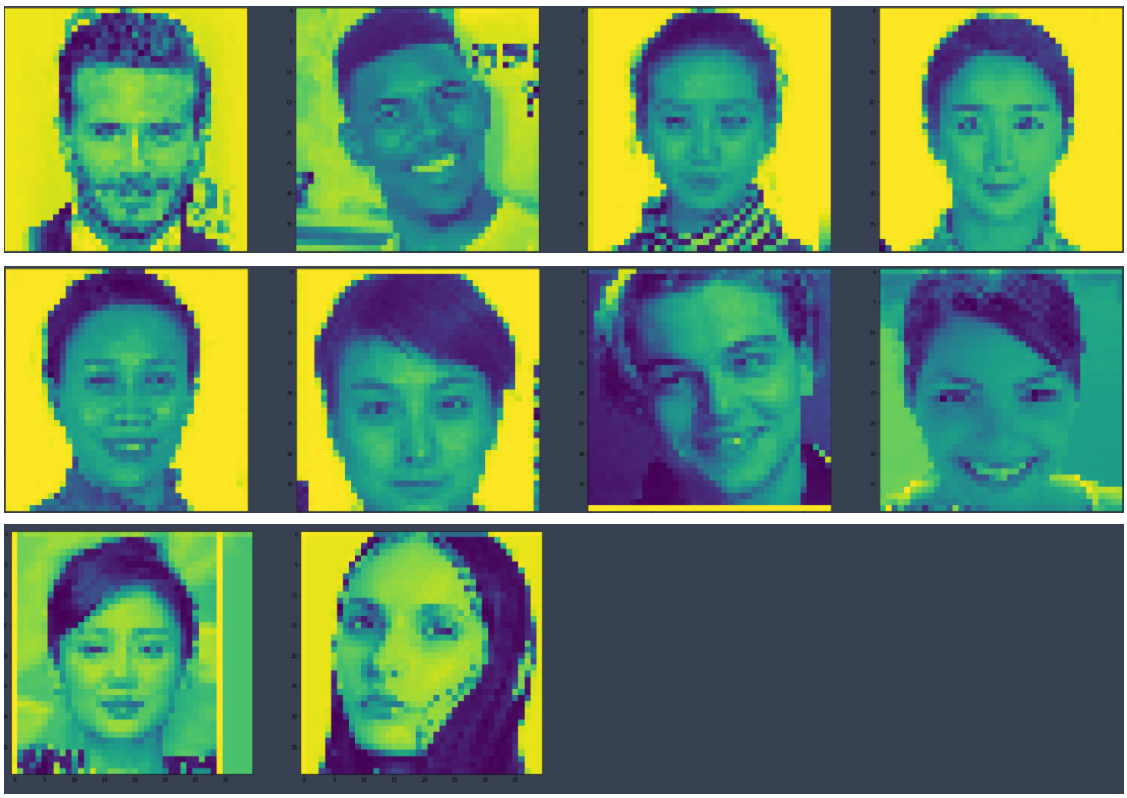
从上面三次实验中可以看出，y_scale=100时和y_scale=10时的主成分提取结果有本质上的区别，当y_scale=100时，瑞士卷较厚，纵向方差较大，投影面为侧投影，而y_scale=10时瑞士卷较薄，只有薄薄的一层，所以相比是漩涡状的正投影方差较大。不同的厚度（纵向方差）导致了不同的投影面，但都保留了原数据尽可能多的特征。

此外，当给y_scale=10时加上noise=1的噪声前后对比，加噪声前，正投影面是平滑的漩涡，加噪声后，正投影面虽然变得参差不齐，但也可以看出是漩涡状。

4. 人脸数据压缩

本次人脸数据压缩所用的数据集均来自于百度图片，搜索的十张明星人脸（此数据为本人自行搜索并上传到QQ群中，tu.zip）。人工将十张图片裁剪为250 * 250像素大小，但是在实验中出现了方差矩阵过大，求解特征值非常缓慢的问题，因此，调用cv2（或人工压缩），将250 * 250压缩至size=(40,40)大小，以求得更快的运行速度。

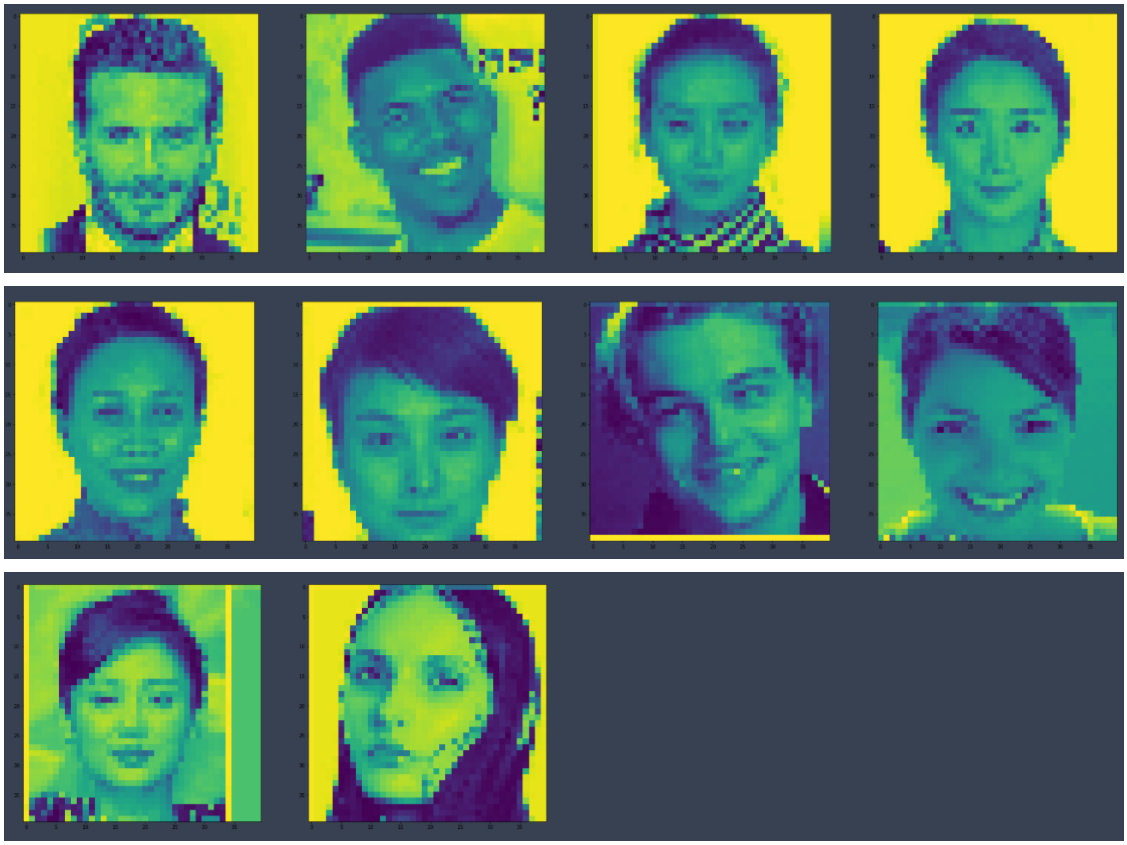
十张人脸原图如下：（如果用cmap="Greys"等灰度图形式输出图像过于恐怖，因此输出默认的图像，绿色偏色严重）



下面做特征值提取，将分几次不同的降维程度，给出实验结果。

降维至20维：

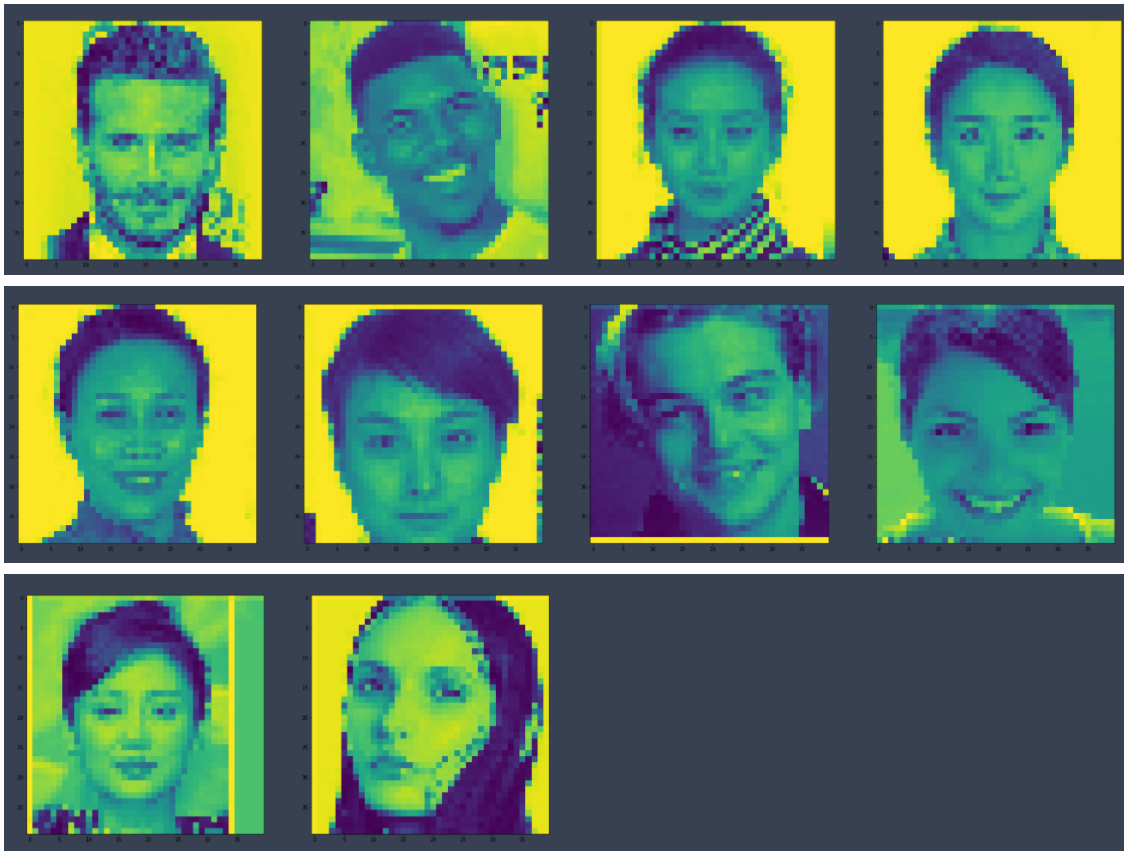
```
c_data, eigVectsReduce, data_mean = PCA(data, 20)
```



```
信噪比如下:
图 0 的信噪比:  100
图 1 的信噪比:  100
图 2 的信噪比:  100
图 3 的信噪比:  100
图 4 的信噪比:  100
图 5 的信噪比:  100
图 6 的信噪比:  100
图 7 的信噪比:  100
图 8 的信噪比:  100
图 9 的信噪比:  100
```

降维至10维:

```
c_data, eigVectsReduce, data_mean = PCA(data, 10)
```



```
信噪比如下：
图 0 的信噪比:  100
图 1 的信噪比:  100
图 2 的信噪比:  100
图 3 的信噪比:  100
图 4 的信噪比:  100
图 5 的信噪比:  100
图 6 的信噪比:  100
图 7 的信噪比:  100
图 8 的信噪比:  100
图 9 的信噪比:  100
```

降维至8维：

```
c_data, eigVectsReduce, data_mean = PCA(data, 8)
```

```
信噪比如下：
图 0 的信噪比:  32.60924422324026
图 1 的信噪比:  34.09373002495726
图 2 的信噪比:  29.632799200219996
图 3 的信噪比:  24.975497288159296
图 4 的信噪比:  20.938772353320857
图 5 的信噪比:  53.64887295675984
图 6 的信噪比:  46.05723898380169
图 7 的信噪比:  46.184987487337125
图 8 的信噪比:  58.22735538044612
图 9 的信噪比:  49.696660143746456
```

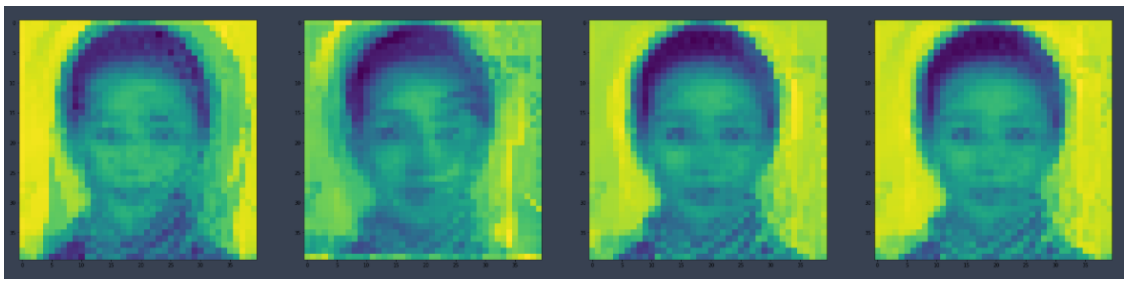
降维至5维：

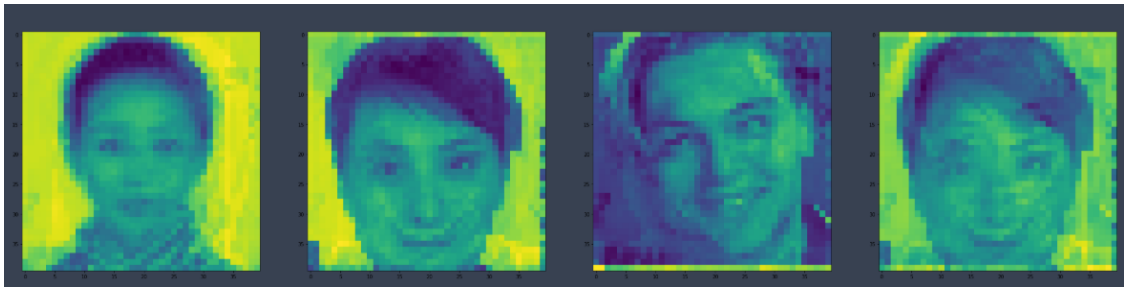
```
c_data, eigVectsReduce, data_mean = PCA(data, 5)
```

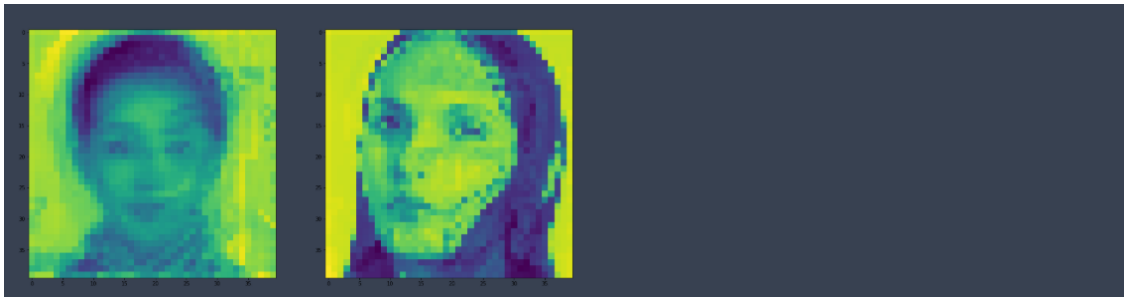
```
信噪比如下:
图 0 的信噪比:  28.37987040271324
图 1 的信噪比:  16.200939954625955
图 2 的信噪比:  19.034534896466447
图 3 的信噪比:  18.76828920400277
图 4 的信噪比:  20.71330032882016
图 5 的信噪比:  22.84689165015511
图 6 的信噪比:  23.644103553491043
图 7 的信噪比:  18.351003033504632
图 8 的信噪比:  26.477910226433178
图 9 的信噪比:  43.82883415326377
```

降维至3维:

```
c_data, eigVectsReduce, data_mean = PCA(data, 3)
```



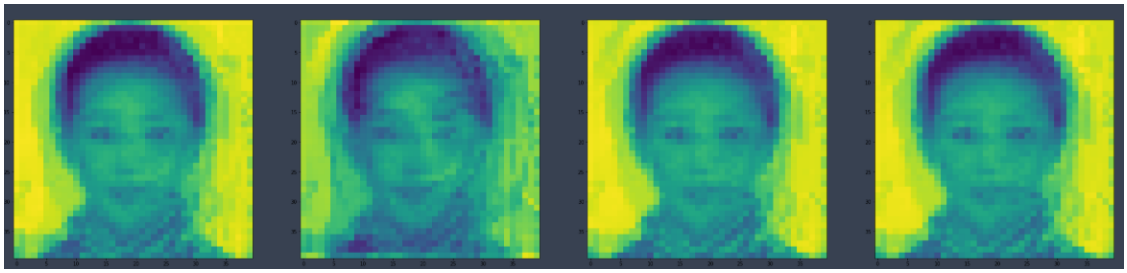


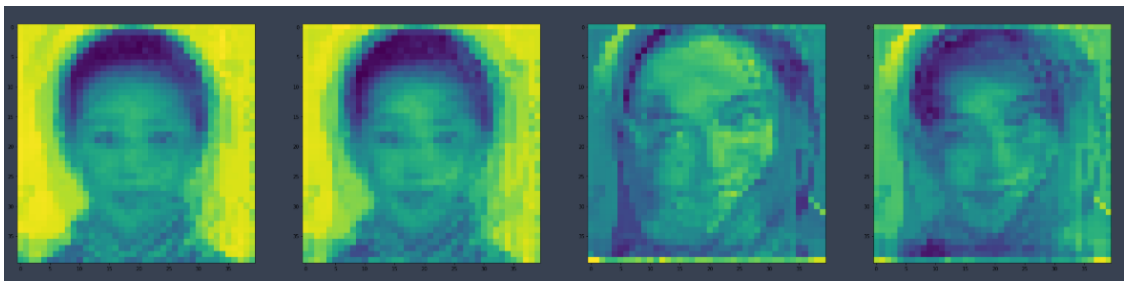


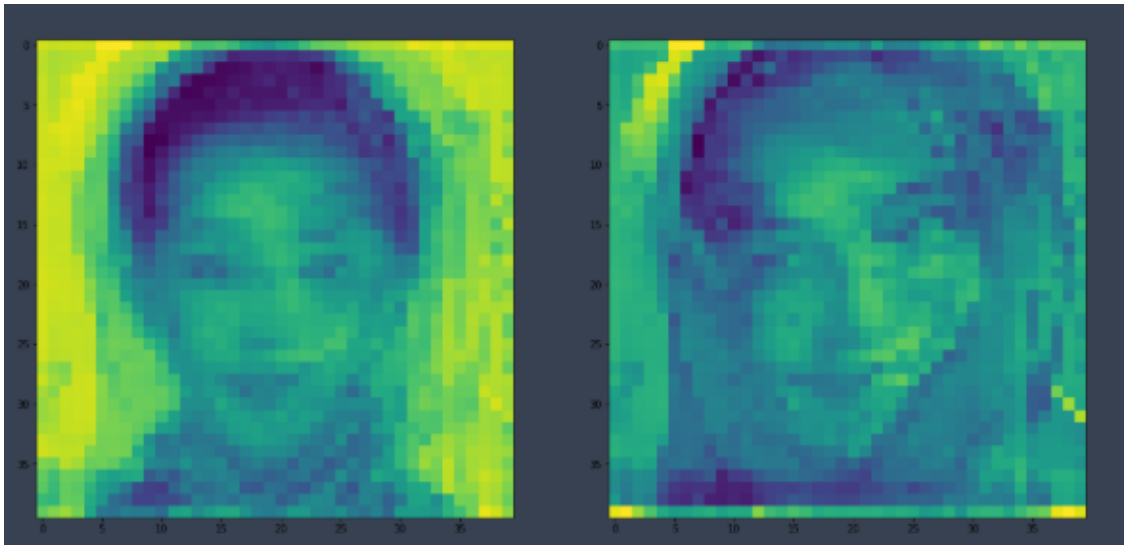
```
信噪比如下:
图 0 的信噪比:  15.215015719101316
图 1 的信噪比:  15.42970290080552
图 2 的信噪比:  18.626868798946973
图 3 的信噪比:  18.270874500088528
图 4 的信噪比:  18.295906419425993
图 5 的信噪比:  22.327697022526294
图 6 的信噪比:  23.34023690818676
图 7 的信噪比:  18.112636437892647
图 8 的信噪比:  14.814133557330377
图 9 的信噪比:  26.320317240367444
```

降维至1维:

```
c_data, eigVectsReduce, data_mean = PCA(data, 1)
```







```
信噪比如下：
图 0 的信噪比：  14.471213830525642
图 1 的信噪比：  14.894908823379382
图 2 的信噪比：  17.192307069896465
图 3 的信噪比：  17.659561442661257
图 4 的信噪比：  17.693964235186453
图 5 的信噪比：  13.090294808158605
图 6 的信噪比：  15.522278429142029
图 7 的信噪比：  15.729392099451731
图 8 的信噪比：  14.653351894095398
图 9 的信噪比：  10.364558804486636
```

分析：

可以看出，随着维度的降低，重构图像越来越恐怖。。信噪比也越来越低。这十张图片中，我特意找了几张长得相似的和差别较大的，可以看到，从降维幅度10维到8维的变化，其余图像仍保存较好的还原度，但图四和图五却混在了一起。。紧接着，图六和图八也逐渐混在了一起，都是有斜刘海的。而最后一张图，直到降至3维，仍保留着主要特征，没有和其他图混在一起，说明它和其他图的特征差别较大，这在直观上也很容易看出。

上述结论从信噪比也可以看出。按照代码逻辑，如果压缩图像和原图几乎不差别，则输出psnr=100，在降维程度为10维和20维时，几乎都无差别，在这之后压缩效果越来越差，但在一次压缩的10张图中，有些图片psnr值高，有些图片psnr值低。

在实验的过程中，还会遇到求解特征向量时，特征向量出现虚部的问题，但是发现只有在降维程度较高的时候（比如降维至10维以上），特征向量矩阵的后面才会出现虚部。如果只保留虚特征向量的实部，实验还是能得到很好的结果。在10维以内的特征向量矩阵求解结果，全部都是实特征向量。

五、结论

1. PCA是数据压缩和高维数据可视化的有效手段。
2. PCA对图像的压缩效果是显著的。比如有10000个30x30像素大小的图片，在做PCA降维时，将其降至50维保存（假设50维特征可以很好的还原图像），那么压缩后，我们需要保存的数据只有——10000x50的压缩数据，900x50的特征向量矩阵，1x900的原数据均值，相比10000x30x30的原数据大小，有显著的压缩效果。
3. psnr是衡量图像压缩信号重建质量的指标。

六、参考文献

七、附录：源代码（带注释）

编辑器为jupyter notebook

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

```
'''
    对矩阵X进行旋转变换
    X.shape = (D, N)
```

```
    theta为旋转的弧度
    axis为旋转的轴，合法值为'x','y'或'z'
'''
def rotate(X, theta=0, axis='x'):
    if axis == 'x':
        rotate = [[1, 0, 0], [0, np.cos(theta), -np.sin(theta)], [0, np.sin(theta), np.cos(theta)]]
        return np.dot(rotate, X)
    elif axis == 'y':
        rotate = [[np.cos(theta), 0, np.sin(theta)], [0, 1, 0], [-np.sin(theta), 0, np.cos(theta)]]
        return np.dot(rotate, X)
    elif axis == 'z':
        rotate = [[np.cos(theta), -np.sin(theta), 0], [np.sin(theta), np.cos(theta), 0], [0, 0, 1]]
        return np.dot(rotate, X)
    else:
        print('错误的旋转轴')
        return X
```

```
'''
    生成瑞士卷数据
    n_sample，生成的数据点数量，default=100
    noise，生成数据点的噪声程度，default=0.0
    y_scale，瑞士卷的厚度，default=100
'''
def make_swiss_roll(n_sample=100, noise=0.0, y_scale=100):
    t = 1.5 * np.pi * (1 + 2 * np.random.rand(1, n_sample))
    x = t * np.cos(t)
    y = y_scale * np.random.rand(1, n_sample)
    z = t * np.sin(t)
    X = np.concatenate((x, y, z))
    X += noise * np.random.randn(3, n_sample)
    X = rotate(X, 40 * np.pi / 180, 'z') # 绕x轴旋转数据点20°
    X = X.T
    return X
```

```
def show_3D(X):
    fig = plt.figure()
    ax = Axes3D(fig)
    ax.view_init(elev=20, azim=80)
    ax.scatter(X[:,0], X[:,1], X[:,2], c=X[:,0], cmap=plt.cm.gnuplot)
    ax.legend(loc='best')
    plt.show()
```

```
def show_2D(X):
    plt.scatter(X[:,0].tolist(), X[:,1].tolist(), c=X[:,0].tolist(), cmap=plt.cm.gnuplot)
    plt.show
```

```
'''
    对数据data用PCA降至k维
    data.shape = (N, D)
    返回值：
    c_data，中心化数据，shape=(N, D)
```

```
    eigVectsReduce，特征向量矩阵，shape=(D, k)
    data_mean，降维前数据均值，shape=(1, D)
'''
def PCA(data, k):
    rows, cols = data.shape
    data_mean = np.sum(data, 0) / rows
    c_data = data - data_mean # 中心化
    covMat = np.dot(c_data.T, c_data)
    eigVals, eigVects = np.linalg.eig(covMat) # 对协方差矩阵(D,D)求特征值和特征向量
    eigValIndex = np.argsort(eigVals) # 特征值排序
    eigVectsReduce = eigVects[:,eigValIndex[:-(k+1):-1]] # 取前k个特征值对应的特征向量
    return c_data, eigVectsReduce, data_mean
```

```
'''
    人工数据PCA实验
'''
def self_exp(X):
    show_3D(X)
    c_data, eigVectsReduce, data_mean = PCA(X, 2)
    pca_data = np.dot(c_data, eigVectsReduce)
    show_2D(pca_data)
```

```
X1 = make_swiss_roll(2000, 0, 100)
X2 = make_swiss_roll(2000, 0, 10)
X3 = make_swiss_roll(2000, 1, 10)
self_exp(X1)
self_exp(X2)
self_exp(X3)
```

```
import os
import matplotlib.image as mpimg
import cv2
from PIL import Image
import math
size = (40, 40) # 由于较大的数据在求解特征值和特征向量时很慢，故统一压缩图像为size大小
```

```
'''
    从file_path中读取面部图像数据
'''
def read_faces(file_path):
    file_list = os.listdir(file_path)
    data = []
    i = 1
    plt.figure(figsize=size)
    for file in file_list:
        path = os.path.join(file_path, file)
        plt.subplot(3, 4, i)
        with open(path) as f:
            img = cv2.imread(path) # 读取图像
            img = cv2.resize(img, size) # 压缩图像至size大小
            img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) # 三通道转换为灰度图
            plt.imshow(img_gray) # 预览
```

```
            h, w = img_gray.shape
            img_col = img_gray.reshape(h * w) # 对(h,w)的图像数据拉平
            data.append(img_col)
        i += 1
    plt.show()
    return np.array(data)
```

```
'''
    计算峰值信噪比psnr
'''
def psnr(img1, img2):
   mse = np.mean((img1 / 255. - img2 / 255.) ** 2 )
   if mse < 1.0e-10:
      return 100
   PIXEL_MAX = 1
   return 20 * math.log10(PIXEL_MAX / math.sqrt(mse))
```

```
data = read_faces('PCA-FACE')
```

```
n_samples, n_features = data.shape
c_data, eigVectsReduce, data_mean = PCA(data, 1) # PCA降维
print(eigVectsReduce)
eigVectsReduce = np.real(eigVectsReduce) # 一旦降维维度超过某个值，特征向量矩阵将出现复向量，对其保留实部
pca_data = np.dot(c_data, eigVectsReduce) # 计算降维后的数据
recon_data = np.dot(pca_data, eigVectsReduce.T) + data_mean # 重构数据
plt.figure(figsize=size)
for i in range(n_samples):
    plt.subplot(3,4,i+1)
    plt.imshow(recon_data[i].reshape(size))
plt.show()

```

```
print("信噪比如下：")
for i in range(n_samples):
    a = psnr(data[i], recon_data[i])
    print('图', i, '的信噪比：', a)
```