CS540 2024 年秋季作业 3

到期日：2024 年 9 月 30 日上午 9:30

1 作业目标

- 探索主成分分析（PCA）和相关的 Python 软件包（numpy、scipy 和 matplotlib)
- 制作漂亮的图片）

2 摘要

在本项目中，您将利用**线性代数 + PCA 讲座**中的技能，使用 **PCA** 实现一个面部分析程序。您还将继续学习 Python 技能。我们将引导你逐步完成整个过程（高水平）。

3 本项目所需的软件包

您只能使用 Python3 标准库以及 NumPy、SciPy 和 matplotlib（安装说明已链接）。您应使用 SciPy 版本 >= 1.5.0 和以下导入命令：

```
>>> from scipy.linalg import eigh
>>> import numpy as np
>>> import matplotlib.pyplot as plt
```

4 数据集

您将使用 "荒野中的面孔 "标签的一部分（已处理）。数据集保存在 "face_dataset.nyy "文件中，你可以在发布的 hw3 文件夹中找到该文件。.nyy "文件格式用于存储 numpy 数组。我们将仅使用提供的数据集测试你的代码。

数据集包含 13233 幅样本图像，每幅图像大小为 64 × 64。我们将用 n 来表示图像的数量（因此 $n = 13233$），用 m 来表示每个样本图像的特征数量（因此 $m = 4096 = 64 \times 64$）。注意，我们将使用 x_i 来指代第 i *张*样本图像，它是一个 m *维*特征向量。

注：此处的设置与 PCA 讲座中的示例不同。在讲座中，我们将*一张*大图像分割成多个斑块，将每个斑块视为一个数据点 x_i。在这里，我们有多幅（$n = 13233$）小图像，每幅图像都被视为一个数据点 x_i。

5 计划规范

在 hw3.py 中执行这**六个** Python 函数，对数据集进行 PCA 分析：

1. load_and_center_dataset(filename)：从提供的 .npy 文件中加载数据集，以原点为中心，并以浮点数的 numpy 数组形式**返回**。

2. get_covariance(数据集)：计算数据集的协方差矩阵，并以 numpy 格式**返回**矩阵（$d \times d$ 阵列）。

3. get_eig(S,k)：对协方差矩阵 S 进行特征分解，并**返回**对角线上*按降序排列*的最大 k *个*特征值的对角矩阵（numpy 数组），以及以相应特征向量为列的矩阵（numpy 数组）。

4. get_eig_prop(S,prop)：与 get_eig 类似，但不是返回前 k *个*特征值，而是以类似格式返回所有能解释方差比例以上的特征值和相应的特征向量（具体来说，请确保以降序返回特征值）。

5. project_image(image, U)：将每幅 $m \times 1$ 图像投射到 k 维子空间（由 k 大小为 $m \times 1$ 的向量），并将新表示**返回**为 $m \times 1$ 的 numpy 数组。

6. display_image(orig,proj)：使用 matplotlib 并排显示原始图像和投影图像的视觉效果。

7. perturb_image(image,U,sigma)：使用标准偏差为 σ 的高斯分布对给定图像进行扰动，以 $m \times 1$ 的 numpy 数组形式**返回**重建后的图像。

8. combine_image(image1,image2,U,lam)：使用凸组合将两个图像特征合二为一，以 $m \times 1$ 的 numpy 数组形式**返回**凸组合图像。

5.1 加载数据集并居中（[20] 分）

您需要使用 numpy 函数 load() 将 face_dataset.npy 文件加载到 Python 中。

```
>>> x = np.load(filename)
```

这样就会得到一个 $n \times m$ *的*数据集，用 n-d 矩阵/数组 x 表示（$n = 13233$ 是数据集中的图像数，$m = 4096$ 是每幅图像的维数）。换句话说，x 的每一行代表一个图像特征向量 x^T（作为行向量）。

下一步是将数据集围绕原点居中。回想一下这一步骤的目的 - 这是一个技术条件，它使 PCA 更容易执行，但不会丢失任何重要信息。

要使数据集居中，只需从每个数据点 x_i（本例中为图像）中减去平均值 μ_x，即 $x^{cent} = x_i - u_x$，其中

$$\mu_x = \frac{1}{n} \sum_{i=1}^{n} x_i$$

您可以利用 x（如上定义）是一个 numpy 数组这一事实，因为它具有这种方便的行为：

```
>>> x = np.array([[1,2,5],[3,4,7]])
>>> np.mean(x, axis=0)
array([2., 3., 6.])
>>> x - np.mean(x, axis=0)
array([[-1., -1., -1.],
       [ 1.,  1.,  1.]])
```

实现该功能后，它应该这样工作：

```
> > >    x = load_and_center_dataset('face_dataset.npy')
>>> len(x)
13233
>>> len(x[0])
4096
>>> np.average(x)
-2.61654556892539e-15
```

(它的中心并不*完全*为零，但考虑到 421 个 4096 个浮点数组的精度误差，它已经足够 "接近 "了）。

从现在起，我们将使用 x_i 来指代 x^{cent}_i。

5.2 求协方差矩阵（[15] 分）

请回顾讲义，PCA 的解释之一是它是样本协方差矩阵的多重分解。在本作业中，我们将根据这一解释来完成作业，您需要了解的所有信息如下。

协方差矩阵的定义是

$$S = \frac{1}{n-1}\sum_{i=1}^{n} x_i x_i^{\top}。$$

请注意，x_i 是（居中）数据集中的 n 幅图像之一，被视为一个大小为

因此，S 是一个 $m \times m$ 矩阵；用数学符号表示，我们称 $S \in \mathbb{R}^{m \times m}$。

要计算 S，你还需要一些 numpy 工具：

```
>>> x = np.array([[1,2,5],[3,4,7]])
>>> np.transpose(x)
array([[1, 3]、
       [2, 4],
       [5, 7]])
>>> np.dot(x, np.transpose(x))
array([[30, 46]、
       [46, 74]])
>>>  np.dot(np.transpose(x), x )
 array([[10  14,  26],
        、
       [14,  20,  38],
       [26,  38,  74]])
```

对于我们的样本数据集，该函数的结果应该是一个 $m \times m$（即 4096 × 4096）矩阵。

5.3 获取 *k* 个最大特征值及其特征向量（[17] 点）

请再次回顾讲义，特征值和特征向量是描述矩阵特征的有用对象。更妙的是，PCA 可以通过进行特征分解并提取最大特征值对应的特征向量来实现。

在编写此函数时，您可能会发现 scipy 库中的 scipy.linalg.eigh 非常有用。名为 subset_by_index 的可选参数可能特别有用。

将 S *的最大* k *个特征值*按降序以 k *乘* k *的*对角矩阵Λ 的形式返回，并将相应的归一化特征向量以 m *乘* k 矩阵 U *的*形式返回、

$$\Lambda = \begin{bmatrix} \lambda_1 & & \\ & \lambda_2 & \\ & & \ddots \\ & & & \lambda_k \end{bmatrix}$$

U 的第 j 列是 u_j，其中 λ_j 是 S 的第j 个最大特征值，$u_j \in R^{m\times 1}$ 是相应的特征向量（归一化为欧氏规范等于 1）。

在 Python 中，要从一个函数返回多个内容，可以这样做：

```
def multi_return （） ：
    返回 "一个字符串"，
    5
my_string , my_int = multi_return ()
```

确保*首先*返回特征值的对角矩阵，然后返回相应列的特征向量。您可能需要重新排列 eigh 的输出，以便按递减顺序获得特征值，并*确保*在重新排列后*保留相应列中的特征向量*。

```
>>> S = get_covariance(x)
>>> Lambda, U = get_eig(S, 2)
>>> print(Lambda)

[[4756883.1090128         0.          ]
 [         0.          1395990.52536811]]
```

5.4 获取能解释一定比例以上方差的所有特征值/特征向量（[8] 分）

与其预先确定 k（最高特征值/特征向量的个数），不如在选择 k 时考虑到所有 "重要 "特征向量。我们的做法如下。回忆一下，λ_i 是协方差矩阵 S 的第 i 个特征值、

$$\frac{\lambda_i}{\sum_{j=1}^{m} \lambda_j},$$

表示数据集中由第 i 个特征向量解释的*方差比例*。这个量越大，说明第 i 个特征值/特征向量在捕捉原始数据集中的信息方面越重要。

对于给定的数字 $0 \leq p \leq 1$，我们希望使用*所有*能解释方差比例 p 以上的特征值/特征向量。以对角矩阵的形式按降序返回特征值，以矩阵的列形式返回相应的特征向量。提示：subset_by_index 对前面的函数很有用，所以类似的函数在这里也许也能派上用场。提示：什么是矩阵的 trace?

同样，请确保*首先*返回特征值的对角矩阵，然后返回相应列的特征向量。您可能需要重新排列 eigh 的输出，以便按递减顺序获得特征值，并*确保*在重新排列后*保留相应列中的特征向量*。

下面是一个 $p = 0.07$ 的例子。

```
>>> Lambda, U = get_eig_prop(S, 0.07)
>>> print(Lambda)
[[4756883.1090128         0.          ]
[         0.          1395990.52536811]]
```

插曲了解 PCA

选读：对 PCA 的严谨数学探索。

在介绍作业中的下一个任务之前，我们先来正式解释一下 PCA。回顾一下，我们有 n 个数据点 $x_1, x_2, ..., x_n$，其中每个 x_i 是一个 m 维列向量（注：x_i 是 $m \times 1$ 或 $x_i \in \mathbb{R}^{m \times 1}$）。在本节中，我们将对

1. 每个数据点的 PCA 投影 $\alpha_i \in \mathbb{R}^{k \times 1}$
2. 每个数据点的 PCA 重构 $x^{pca} \in \mathbb{R}^{m \times 1}$ i

5.4.1 PCA 投影

回顾一下，我们数据的协方差矩阵为

$$S = \frac{1}{n-1} \sum_{i=1}^{n} x_i x_i^\top \in \mathbb{R}^{m \times m}$$

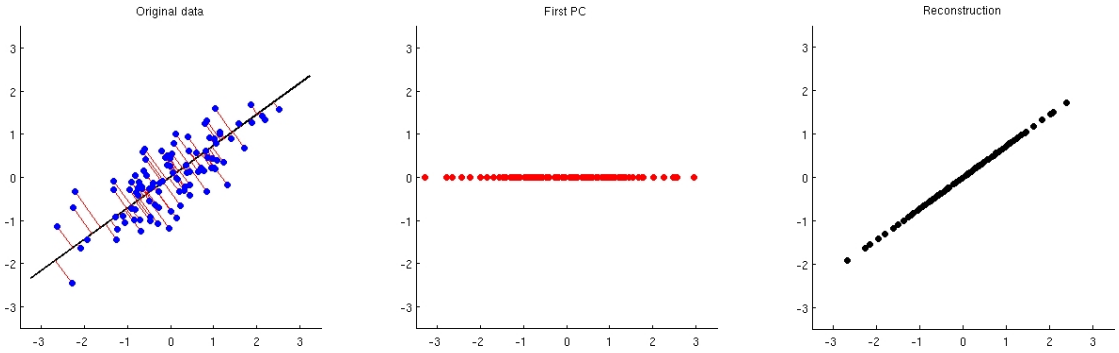


图 1: PCA 的可视化[1]，蓝点代表一些数据点 $x_1, \ldots, x_n \in \mathbb{R}^2$ 黑线是沿第一主成分 u_1 的轴线，它使投影误差平方和最小。第二幅图以红点表示 PCA 投影值，即 $\alpha_1, \ldots, \alpha_n \in \mathbb{R}^1$，我们将数据的维度从 2 降为 1。最后一幅图显示了重建后的数据点 $\boldsymbol{x}^{pca}_1, \ldots, \boldsymbol{x}^{pca}_n \in \mathbb{R}^2$ 的黑点，与原始数据相比有明显的重建误差。第一幅图中的数据。

回想一下，我们的目标是在降低数据维度 m 的同时，**最大限度地减少投影误差**（平方距离之和）。为此，我们将每个数据点 $x_i \in \mathbb{R}^{m\times 1}$ 投射到 S 的前 m 个特征向量所跨的线性子空间上。明确地说，x_i 的 PCA 投影（也称为 "得分"）由以下公式给出

$$\alpha_i = U x^{\mathsf{T}}{}_i \text{ 、}$$

其中 $U \in \mathbb{R}^{m\times k}$ 是矩阵，其列是顶部特征向量 $u_1, \ldots, u_k$，如前定义。作为正确性检验，可以验证 α_i 是一个 $k \times 1$ 列向量，其第 j 个元素是 $\alpha_{ij} = u_j^{\mathsf{T}} x_i$。

概括地说，PCA 投影就是将原始数据点 $x_i \in \mathbb{R}^{m\times 1}$ 投影到 $\alpha_i \in \mathbb{R}^{k\times 1}$，而这个投影是由 U 中包含的 S 的前 k 个特征向量决定的。

5.4.2 PCA 重建

我们如何才能（近似地）在原始 m 维特征空间中重建数据，即计算 $x^{pca}_i \in \mathbb{R}^{d\times 1}$ 从我们的 PCA 投影 α_i ?我们可以再次使用特征向量 U 并计算出

$$x^{pca}_i = U\alpha_i = UU\, x^{\mathsf{T}}{}_i = \sum_{j=1}^{k} u_j u_j^{\mathsf{T}} x_i = \sum_{j=1}^{k} \alpha_{ij} u_j \text{。}$$

证明上述三个右边项相等是线性代数的一个练习。(在本次作业中，您无需证明这一点，但可以在实现过程中随意使用这些等价表达式）。

图 1 展示了当我们将维度从 $m = 2$ 减小到 $m = 2$ 时的 PCA 投影和重建效果。到 $k = 1$。

提示：如果我们在 PCA 投影过程中没有减少特征向量的数量，即保持 $k = m$，那么根据线性代数，我们知道 $UU^{\mathsf{T}} = I$（同位矩阵），因此 $x^{pca}_i = UU\, x^{\mathsf{T}}{}_i = Ix_i = x_i$，在这种情况下，我们将完美地重建原始数据。您可以在调试代码时使用这一事实。

5.5 投射图像（[15] 分）

给定数据集中的一幅图像和来自 get_eig（或 get_eig_prop）的特征向量，计算图像的 PCA 表示。

回忆一下，u_j 是 U 的第 j 列。每个 u_j 都是 S 的特征向量，大小为 $m \times 1$。如果 U 有 k 特征向量，将图像 x_i 投影到 k 维子空间中。PCA 投影表示

图像的特征向量的加权和。这种投影只需存储每个特征向量（k 维）的权重，而无需存储整个图像（m 维）的权重。投影 $\alpha_i \in \mathbb{R}^k$ 的计算公式为 $\alpha_{ij} = u^T x_i$。

根据每个 α_i，我们可以计算重构 $x^{\text{pca}} = \sum_{j=1}^{m} \alpha_{ij} u_j$。注意，每个特征向量 u_j 乘以相应的权重 α_{ij}。重建后的图像 x^{pca} 不一定等于原始图像，因为在将 x_i 投影到较小的子空间时会丢失信息。使用的特征向量越少，信息损失就越大。执行 project_image，计算输入图像的 x^{pca}。

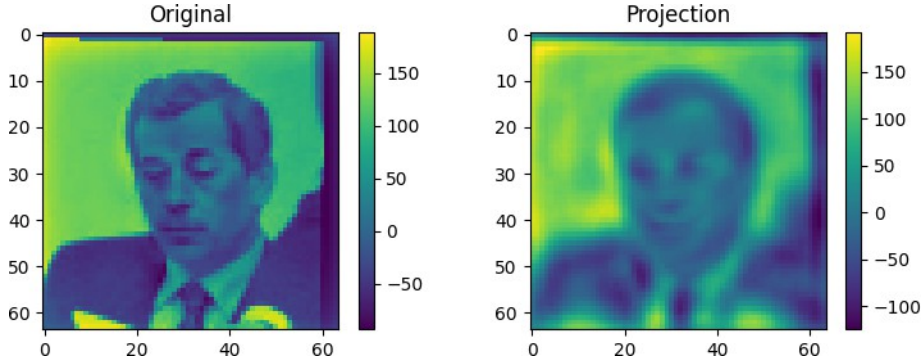
5.6 可视化（[10] 分）

请按照以下步骤使用 matplotlib 将图像可视化：

1. 将图像重塑为 64 × 64（在此之前，图像被认为是 1 维向量，即 $\mathbb{R}^{4096}$）.
2. 用 fig、ax1 和 ax2 对象创建一个包含两行子图的图形。ax1 代表原始图像，ax2 代表重建后的图像。在初始化图形时使用 figsize=(9,3)（请参阅启动代码）。
3. 将第一个分镜头（左边那个）命名为 "原始"（不带引号），将第二个分镜头（右边那个）命名为 "投影"（也不带引号）。
4. 使用带有可选参数 aspect='equal' 的 imshow，可以在正确的坐标轴上显示图像。
5. 为每幅图像创建一个颜色条，放在图像的右侧（请参阅下面的可视化绘图示例。 您的绘图应与此相匹配）。
6. 从 display_image() 返回步骤 2 中使用的 fig、ax1 和 ax2 对象。
7. 测试：渲染您的绘图。**请注意，如果生成的图形需要旋转，您可以对矩阵进行转置以正确渲染。**切勿在提交中包含此操作，这只是为了测试您的实现。我们建议在 main() 中调用 display_image()，并在此处渲染图像，如下面的示例代码段所示。

下面是供您测试功能的简单代码片段。**请勿**将其包含在您提交的文件中！

```
>>> x = load_and_center_dataset('face_dataset.npy')
>>> S = get_covariance(x)
>>> Lambda, U = get_eig(S, 100)
>>> projection = project_image(x[50], U )
>>> fig, ax1, ax2 = display_image(x[50], projection)
>>> plt.show()
```



5.7 使用扰动权重和组合权重进行重建（[15] 分）

在本节中，您将探索改变投影权重对图像重建的影响。具体来说，您将(i) 随机扰动投影权重，然后重建图像；(ii) 创建不同图像投影权重的凸组合，以创建 "混合 "草图。

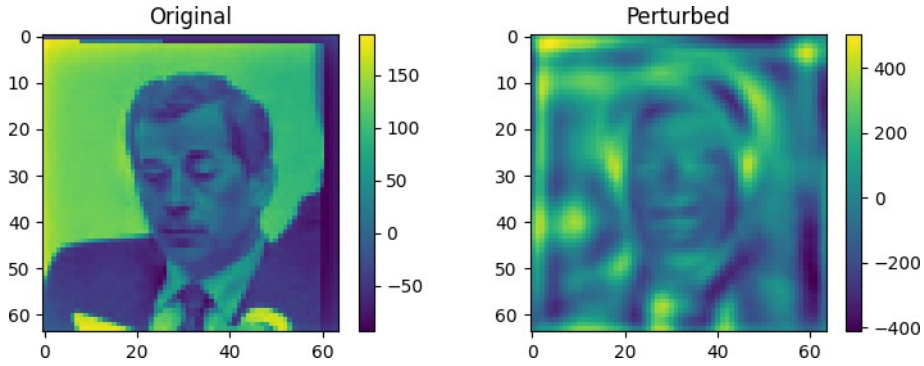
5.7.1 扰动权重重建

通过添加小的随机扰动，修改从图像的 PCA 投影中得到的权重 α。扰动应取自均值为 0、标准偏差为 σ 的高斯分布。使用这些扰动权重，利用 PCA 的特征向量重建图像。将重建后的图像与基于投影的原始重建图像和原始图像进行比较。(您无需在提交的文件中包含这些图像)

提示 1：您可能会发现 np.random.normal 在计算高斯分布时非常有用 提示 2：扰动权重是原始权重和扰动权重之和：扰动权重是原始权重和扰动的加法

下面是供您测试功能的简单代码片段。**请勿**将其包含在您提交的文件中！

```
> > >   x = load_and_center_dataset('face_dataset.npy')
>>> S = get_covariance(x)
>>> Lambda, U = get_eig(S, 100)
> > >  perturbed_image = perturb_projection(x[50], U , 1000)
>>> fig, ax1, ax2 = display_image_perturbed(x[50], perturbed_image)   # 该函数与 disp
>>> plt.show()
```



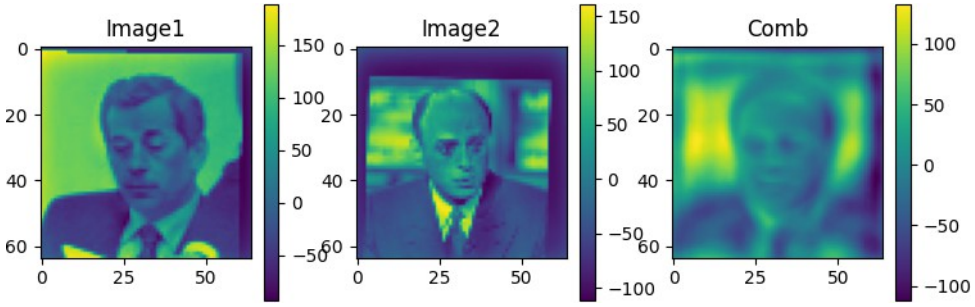
5.7.2 砝码的凸面组合

选择两幅不同的图像，如图像 1 和图像 2，分别计算它们的 PCA 投影权重 α_1 和 α_2。通过对两个权重向量进行凸组合，创建一组新的权重 α_{comb}，即 $\alpha_{comb} = \lambda \cdot \alpha_1 + (1 - \lambda) \cdot \alpha_2$，其中 $0 \leq \lambda \leq 1$。使用相同的特征向量从 α_{comb} 重建图像，并显示结果。(提示：您可以使用第 5.6 节中的函数来显示图像

下面是供您测试功能的简单代码片段。**请勿**将其包含在您提交的文件中！

```
> > >   x = load_and_center_dataset('face_dataset.npy')
>>> S = get_covariance(x)
>>> Lambda, U = get_eig(S, 100)
>>> combined_image = combine_projections(x[50], x[80], U, 0.5)
```

```
>>> fig, ax1, ax2, ax3 = display_image_combo(x[50], x[80], combined_image)     # 此函数类似
>>> plt.show()
```



6 提交说明

请向 Gradescope 提交一个名为 hw3.py 的文件。*请勿*提交 Jupyter notebook .ipynb 文件。

- 您的函数应静默运行（最后一个函数中的图像渲染窗口除外）。
- 任何代码都不得放在函数定义之外（导入语句除外；允许使用辅助函数）。

所有最好的！

参考资料

[1] 变形虫了解主成分分析、特征向量和特征值。交叉验证：https://stats.stackexchange.com/q/140579（版本：2022-08-31）。