模式识别与统计学习实验一：PCA与SVD图片压缩

张力群 2020080301001

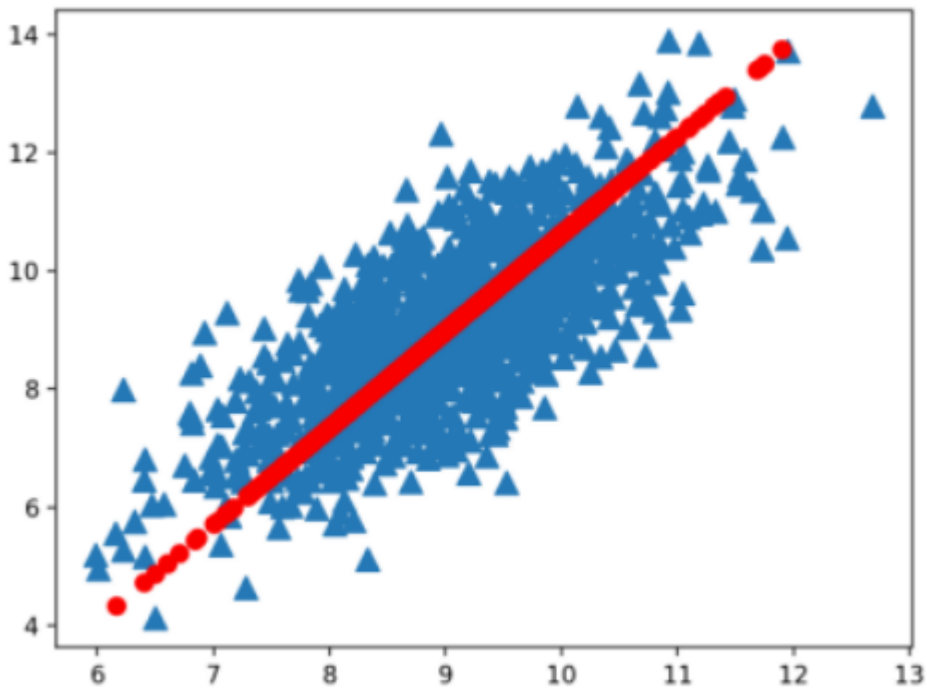
一、 实验原理

1.1 PCA原理概述

PCA（$PrincipalComponentAnalysis$，主成分分析）是一种常用的数据降维方法，可以将高维数据映射到低维空间。在图像压缩中，PCA可以用来对图像进行降维，以达到图像压缩的目的。

具体来说，PCA对于一幅彩色图像（假设图像大小为 $m \times n$）的压缩过程如下：

1. 将每个像素的 RGB 值看作一个三维向量，将整个图像表示为一个 $m \times n \times 3$ 的三维矩阵X。
2. 对该矩阵进行降维，即将其转换为一个 $k \times 1$ 的向量 y，其中 $k << m \times n \times 3$ 。这里采用了线性变换的方式进行降维，即 $y = W^T \cdot x$ ，其中 W为降维矩阵，x 为原始数据（即矩阵 X 的每一个像素）。
3. 对于每个像素，由于其 RGB 值在低维空间中的表示 y 只占原始 RGB 值在高维空间中表示的一小部分，因此用 y 来代替原始像素点可以实现图像的压缩。
4. 实际上，在进行 PCA 降维时，我们需要找到一个能够最大程度保留原始数据信息的投影矩阵W。这个矩阵可以通过以下步骤计算得到：
5. 对中心化后的数据（即在数据均值上减去每个数据点）进行协方差矩阵分解。
6. 对协方差矩阵进行特征值分解，得到特征值和特征向量。
7. 选择前 k 个最大的特征值所对应的特征向量，组成降维矩阵 W。



在这个过程中，特征向量对应了每个像素在低维空间中的投影。通过保留前 k 个最大的特征值，可以实现对原始图像的高维数据压缩。

1.2 SVD原理概述

SVD（$SingularValueDecomposition$，奇异值分解）是一种常用的矩阵分解方法，它可以将一个矩阵分解为三个矩阵的乘积。具体来说，对于矩阵 A（$m \times n$），它的 SVD 分解形式如下：

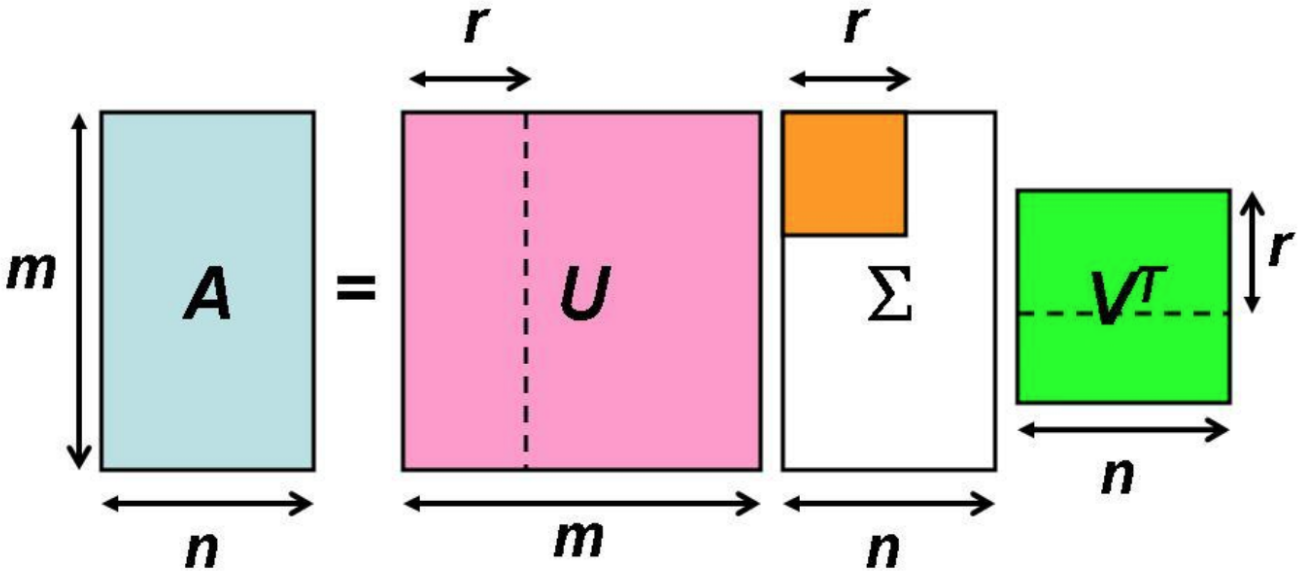
$$A_{m \times n} = U_{m \times m} \cdot \Sigma_{m \times n} \cdot V_{n \times n}^T$$

其中，U 和 V 分别是正交矩阵，Σ 是对角矩阵，对角线上的元素称为奇异值。在 SVD 中，奇异值代表了矩阵 A 的重要程度，奇异值越大，表示该方向上的信息越多，也就越重要。

在图像处理中，SVD 也可以用于图像压缩。我们可以将一个图像矩阵分解为三个矩阵的乘积，然后截取一部分奇异值，再将三个矩阵相乘，就得到了一张压缩后的图像。

具体来说，SVD对于一幅彩色图像（假设图像大小为 $m \times n$）的压缩过程如下：

1. 将每个像素的 RGB 值看作一个三维向量，将整个图像表示为一个 $m \times n \times 3$ 的三维矩阵 X。
2. 对该矩阵进行 SVD 分解，得到三个矩阵：U、S和V，即$X = U \cdot S \cdot V^T$ ，其中 U 和 V 是两个正交矩阵，S 是一个对角矩阵。
3. 对于每个像素，由于其 RGB 值在低维空间中的表示只占原始 RGB 值在高维空间中表示的一小部分，因此可以只保留 S 矩阵中的前 k 个对角元素，来实现图像的降维。
4. 通过将矩阵 U、S、V 中除前 k 个对角元素以外的其余元素置零，可以得到一个近似的压缩图像，恢复时只需要对三个矩阵再次相乘即可还原原始图像。



在这个过程中，SVD分解后的矩阵 S 中的对角元素即为原始图像在每个方向上的奇异值，其大小反映了该方向上图像信息的重要程度。通过保留前 k 个最大的奇异值，就可以实现对原始图像的高维数据压缩，从而实现图像的压缩和存储。

二、 python 代码实现

2.1 PCA代码

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

# 读取图片并转换为灰度图片
img = Image.open('../../database/homework1/butterfly.bmp')
img = img.convert('L')

# 将图片转换为数组
img_arr = np.array(img)

# 将数据类型转换为float64
img_arr = img_arr.astype('float64')

# 均值化处理
mean = np.mean(img_arr, axis=0)
img_arr -= mean

# 将数据类型转换为uint8
img_arr = img_arr.astype('uint8')

# 计算协方差矩阵
cov = np.cov(img_arr.T)
cov = np.atleast_2d(cov)

# 计算特征向量和特征值
eig_val, eig_vec = np.linalg.eig(cov)

# 将特征值从大到小排序，并选择前N个最大特征值的特征向量
N = 250  # 选择前250个最大特征值
idx = eig_val.argsort()[::-1]  # 从大到小排序
eig_vec = eig_vec[:, idx]
eig_vec = eig_vec[:, :N]

# 计算降维后的矩阵
img_pca = np.dot(img_arr, eig_vec)

# 将降维后的矩阵转换为图片矩阵
img_pca = np.dot(img_pca, eig_vec.T) + mean
img_pca = np.reshape(img_pca, (img_arr.shape[0], img_arr.shape[1]))

# 将图像数据类型转换为float
img_pca = np.real(img_pca)  # 取实部
img_pca -= np.min(img_pca)
img_pca /= np.max(img_pca)
img_pca *= 255
img_pca = img_pca.astype('uint8')
```

```
# 绘制原始图片和降维后的图片
plt.subplot(1, 2, 1)
plt.imshow(img, cmap='gray')
plt.title('Original Image')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(img_pca, cmap='gray')
plt.title('PCA Image')
plt.axis('off')

plt.show()
```

- 我们使用 PIL 库中的 Image 类读入一张图像，然后将图像转换为灰度图像，并将其存储为 numpy 数组 img_data。
- 对图片进行均值化处理，计算协方差矩阵，并求取其特征向量和特征值，将特征值从大到小排序，并选择前N个最大特征值的特征向量
- 计算降维后的矩阵，将降维后的矩阵转换为图片矩阵，将图像数据类型转换为float，并进行归一化和类型转换绘制原始图片和降维后的图片。 该代码的实现过程比较典型，典型的PCA代码框架也如下所示：
    - 去中心化
    - 计算样本协方差矩阵；
    - 计算协方差矩阵的特征向量和特征值；
    - 将特征值从大到小排序，并选择前K个最大特征值所对应的特征向量，作为降维后的子空间；
    - 在新的低维空间中，对样本进行投影。

这段代码实现了该框架的前四步，最后一步则是将降维后的矩阵转化为图片矩阵并进行归一化和类型转换，最终输出原始图片和降维后重建的图片。

需要注意的是，PCA 降维后的数据可能会失去一些细节和局部特征，与原始图像会存在一些差异。但是，PCA 经常用于在保留尽可能多的信息的同时，减少计算量和噪声，从而提高模型的稳定性和效率。

2.2 SVD代码

```
import numpy as np
from PIL import Image

# 读入原始图像，转换为灰度图像
img = Image.open("image.jpg").convert('L')
img_data = np.array(img)

# 使用 SVD 进行图像的分解
U, Sigma, V = np.linalg.svd(img_data)
```

- 我们使用 PIL 库中的 Image 类读入一张图像，然后将图像转换为灰度图像，并将其存储为 numpy 数组 img_data。
- 使用 numpy 的 linalg.svd 函数进行SVD分解，得到三个矩阵 U、Sigma 和 V。其中，U 矩阵（左奇异矩阵）包含了原始矩阵的行空间的信息、V 矩阵（右奇异矩阵）包含了原始矩阵的列空间的信息，Sigma 是一个对角矩阵，包含了奇异值。

```
# 按照需要保留的奇异值个数，进行矩阵截断
# 这里保留前 200 个奇异值
k = 200
U_truncated = U[:, :k]
Sigma_truncated = np.diag(Sigma[:k])
V_truncated = V[:k, :]
img_compressed = U_truncated @ Sigma_truncated @ V_truncated
img_reconstructed = np.uint8(img_compressed)
```

- 我们根据需要保留的奇异值个数，对三个矩阵进行截断，得到新的矩阵 U_truncated、Sigma_truncated 和 V_truncated。在本例中，我们保留前 200 个奇异值进行压缩。具体来说，我们将 U 矩阵的前 k 列、Sigma 矩阵的前 k 行前 k 列、V 矩阵的前 k 行保留下来，得到三个截断后的矩阵。然后将这三个矩阵相乘，得到一张压缩后的图像，存储在 img_compressed 变量中。
- 将 img_compressed 转换为整型 uint8 类型，即可将图像显示出来。

需要注意的是，经过矩阵截断和重构后，压缩后的图像可能和原始图像存在细微的差别，但是差别很小，通常人眼难以察觉。

三、实验结果及图形展示

1. PCA

压缩前

压缩后

2. SVD

压缩前

压缩后

四、总结体会

通过实验我们发现：基于PCA和SVD的降维能够有效地减小图像的存储空间；同时，通过选择合适的降维参数，可以在尽可能保留图像信息的同时，进一步提高压缩效果。

基于SVD的降维方法通常可以获得更好的压缩效果。这是因为SVD能够更好地适应图像数据的奇异分布，能够更精细地分解图像中的信息。

在选择降维参数时，需要兼顾压缩率和图像恢复质量两个方面。通常情况下，降维参数k的取值会影响到图像恢复质量，过大过小都可能导致不理想的结果。

总之，基于PCA和SVD的图像降维方法能够有效地实现图像压缩，并在一定程度上保留了图像的信息。在实际应用中，需要根据具体需求选择合适的降维方法和参数，以达到最佳的压缩效果和图像恢复质量。