基于 PCA 的图像压缩

实验内容

1. 按照主成分分析的原理实现 PCA 函数接口
2. 利用实现的 PCA 函数对图像数据进行压缩和重建
3. 利用实现的 PCA 函数对高维数据进行低维可视化

实验过程

1. PCA函数接口

实现`my_pca`函数，对输入二维矩阵计算协方差矩阵，再进行SVD分解，得到特征值，特征向量。以及根据输入参数`k`提取前 k 个主成分。

```
def my_pca(X, k):
    # 中心化数据
    X_mean = np.mean(X, axis=0)
    X_centered = X - X_mean
    # 计算协方差矩阵
    X_covariance = np.cov(X_centered, rowvar=False)
    # SVD 奇异值分解
    U, _, _ = np.linalg.svd(X_covariance)
    # 提取前 k 个主成分
    coeff = U[:, :k]
    # 计算主成分值
    score = np.dot(X_centered, coeff)
    return coeff, score
```

2. PCA基本应用

a) 对 Eigen Face 数据集中的灰度人脸数据进行压缩和重建

1. 利用 PCA 对这些人脸图像进行主成分分析，展示前 49 个的主成分

```
mat = scipy.io.loadmat('data/faces.mat')
X = mat['X']
k = 49
coeff, score = my_pca(X, k)
# 展示前49个主成分
fig, axes = plt.subplots(7, 7, figsize=(8, 8))
for i, ax in enumerate(axes.flat):
    component = coeff[:, i].reshape(32, 32)
    component = np.rot90(component, k=-1, axes=(0, 1))
    ax.imshow(component, cmap='gray')
    ax.axis('off')
plt.savefig('results/PCA/eigen_faces.jpg')
plt.show()
```

展示如下：

2. 将数据降维到不同维度(10，50, 100, 150)进行压缩，然后再重建，对比不同的压缩和重建效果

重建图像代码如下

```
coeff, score = my_pca(X, k)
# 重建原特征空间的图像
re_X = np.dot(score, coeff.T) + np.mean(X, axis=0)
```

对比不同维度的压缩和重建效果：可以明显看出随着提取主成分的增加，重建效果越来越接近原图像

b) 对 scenery.jpg 彩色 RGB 图进行压缩和重建。对该图片分布降维到不同维度(10，50, 100, 150)进行压缩，然后再重建，对比不同的压缩和重建效果

```
image = Image.open('data/scenery.jpg')
    img = np.array(image)
    h,w,_ = img.shape
    # RGB通道分开进行压缩重建
    R = img[:, :, 0]
    G = img[:, :, 1]
    B = img[:, :, 2]
    ks = [10,50,100,150]
    fig, axes = plt.subplots(1,5, figsize=(10, 5))
    for i, k in enumerate(ks):
        # 压缩     图像的每一行视作一个样本
        coeff_R, score_R = my_pca(R, k)
        coeff_G, score_G = my_pca(G, k)
        coeff_B, score_B = my_pca(B, k)
        # 重建
        re_R = np.dot(score_R, coeff_R.T) + np.mean(R, axis=0)
        re_G = np.dot(score_G, coeff_G.T) + np.mean(G, axis=0)
        re_B = np.dot(score_B, coeff_B.T) + np.mean(B, axis=0)
        # 重组成完整RGB图像
        re_X = np.stack((re_R, re_G, re_B), axis=-1).astype(np.uint8)
        # 展示
        axes[i+1].set_title(f'image k = {k}')
        axes[i+1].imshow(re_X)
```

```
        axes[i+1].axis('off')
    axes[0].set_title(f'original image')
    axes[0].imshow(img)
    axes[0].axis('off')
    # plt.savefig(f'results/PCA/compare_recovered_scenery.jpg')
    plt.show()
```

对比不同维度的压缩重建效果如下：