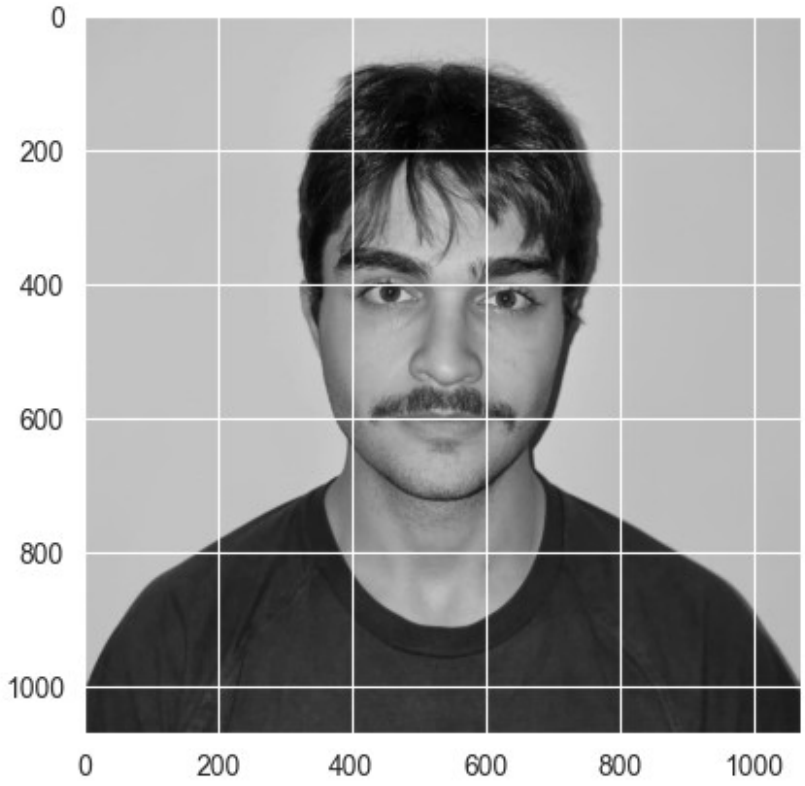
Numerical Methods on a Profile Picture

Preprocessing

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import sklearn

# load the (already b/w) image and convert it to a np array
propic = cv2.imread('data/propic.jpg', cv2.IMREAD_GRAYSCALE)

A =  np.array(propic)
A.shape

plt.imshow(A, cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x27d0be2c050>
```



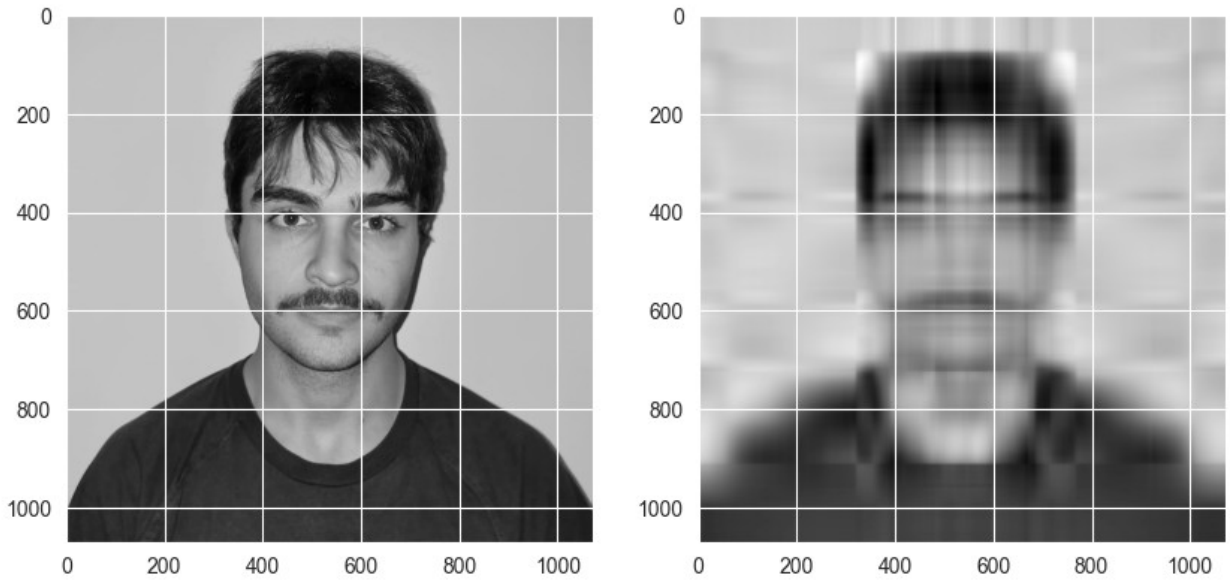
SVD

```
# Singular Value Decomposition and Rank-k approximation
# we can try with different values of k
```

```
# create a function to do this automatically

U, S, Vt = np.linalg.svd(A)

def lowrank_approx(k):
    U_k = U[:, :k]
    S_k = np.diag(S[:k])
    Vt_k = Vt[:k, :]

    A_k = np.dot(U_k, np.dot(S_k, Vt_k))

    return A_k

# plotting function

def plot_propic(original, approximation):
    plt.figure(figsize=(10, 5))

    plt.subplot(1, 2, 1)
    plt.imshow(original, cmap='gray')

    plt.subplot(1, 2, 2)
    plt.imshow(approximation, cmap='gray')

    plt.show()
```

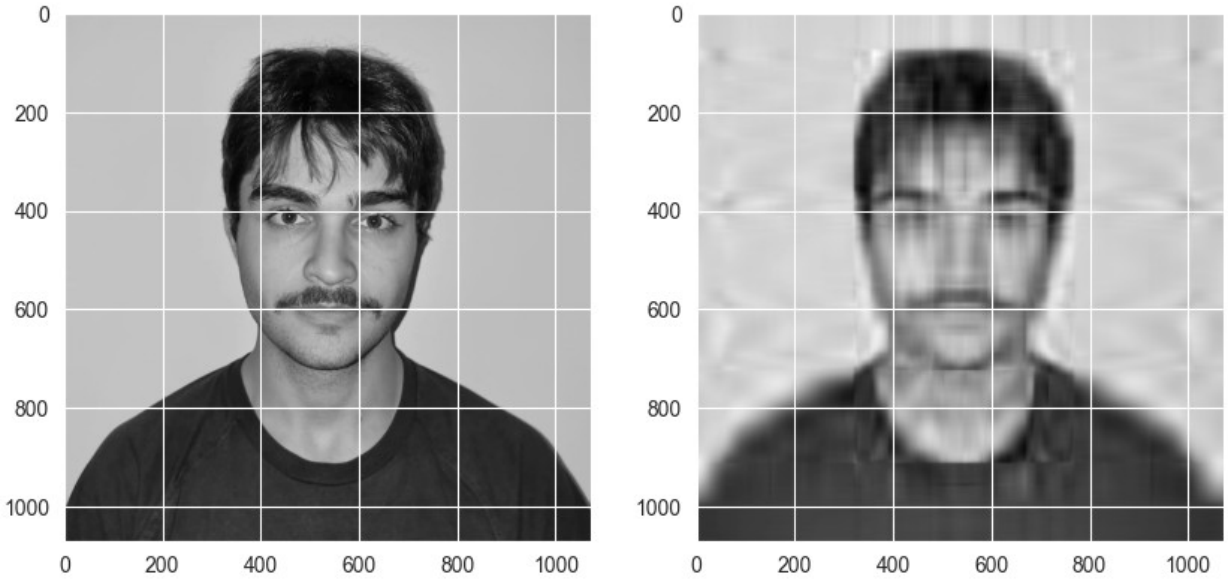
Rank-5 approximation

```
A_5 = lowrank_approx(5)

plot_propic(A, A_5)
```



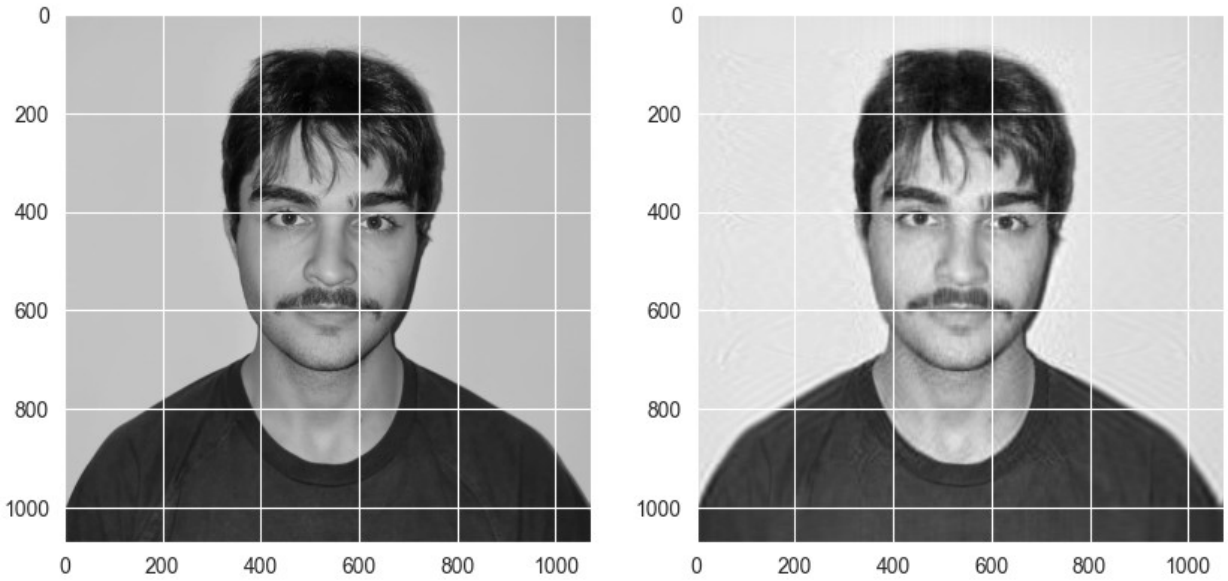
Rank-10 approximation

```
A_10 = lowrank_approx(10)
plot_propic(A, A_10)
```



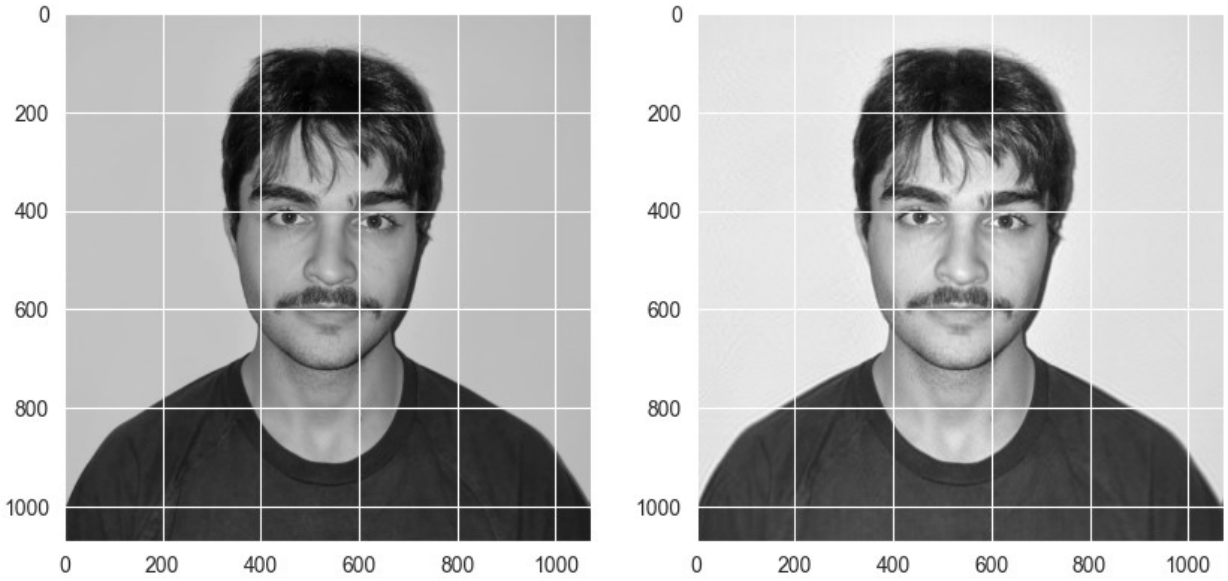
Rank-50 approximation

```
A_50 = lowrank_approx(50)
plot_propic(A, A_50)
```



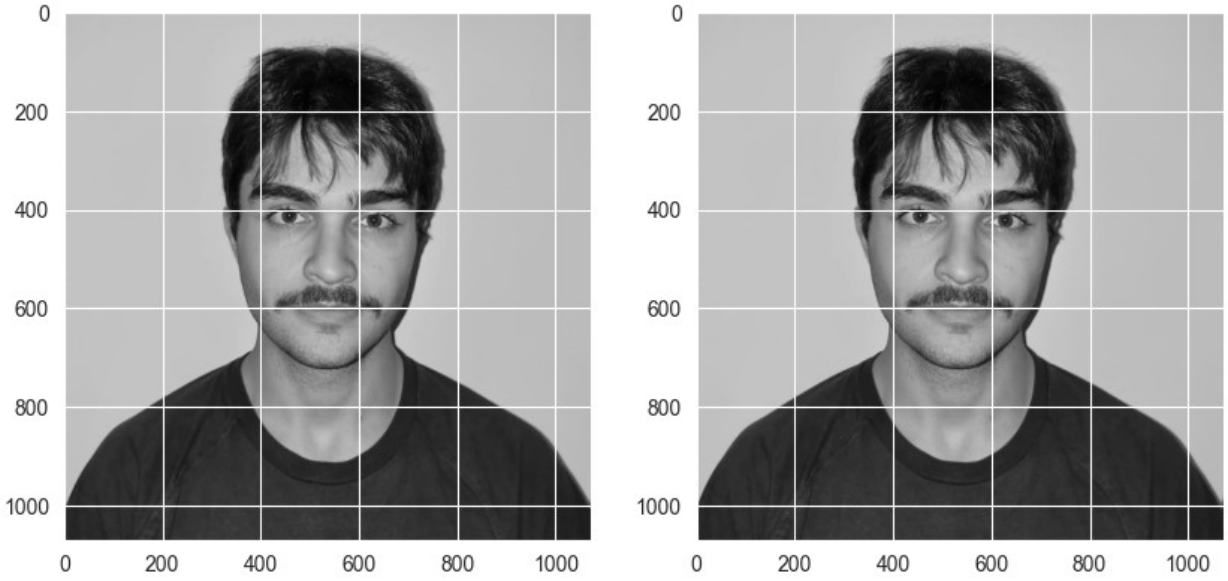
Rank-100 approximation

```
A_100 = lowrank_approx(100)
plot_propic(A, A_100)
```



Rank-500 approximation

```
A_500 = lowrank_approx(500)
plot_propic(A, A_500)
```

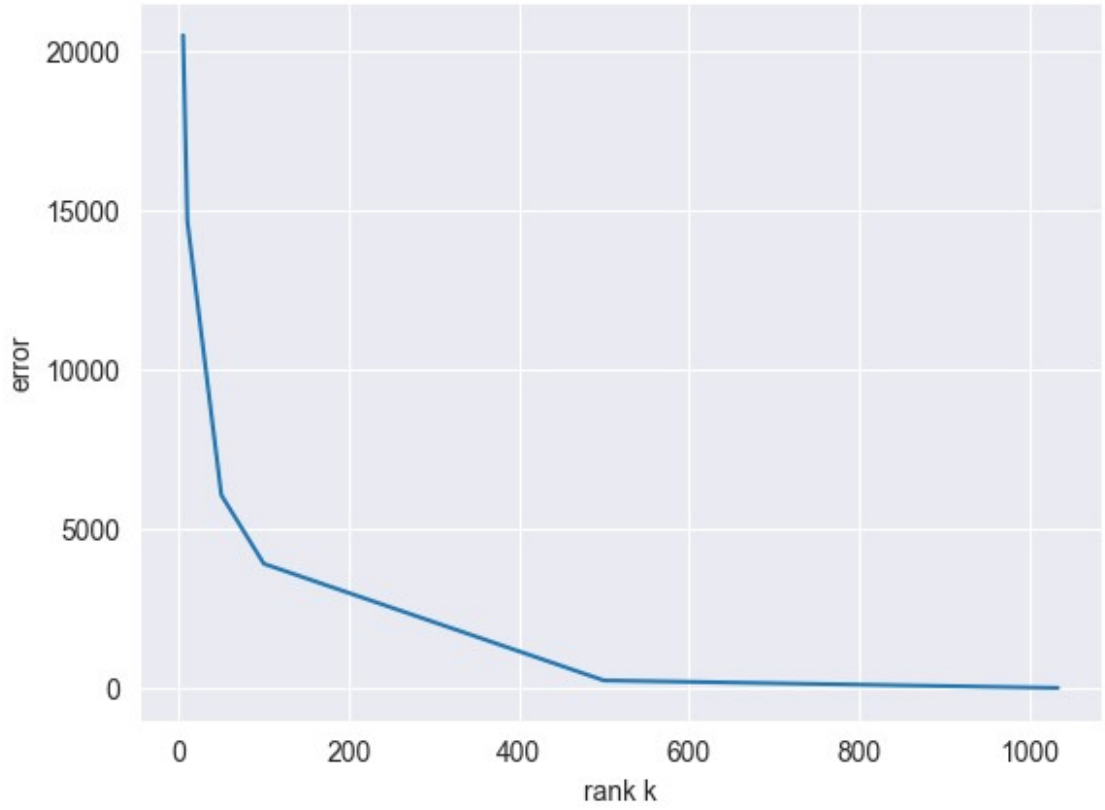


Comments

```
# Get rank of original A

print('Original rank of A:', np.linalg.matrix_rank(A))
```

```
Original rank of A: 1034
```

```
errors = [np.linalg.norm(A - lowrank_approx(k)) for k in [5, 10, 50,
100, 500, 1034]]
plt.plot([5, 10, 50, 100, 500, 1034], errors)
plt.xlabel('rank k')
plt.ylabel('error')
plt.show()
```




```
# although we see that a rank-200 approximation is near perfect, not
all the features are captured up until rank-500
# if we want to just compress some image data for display or storage,
we can use a rank-200 approximation
# if we want to do some analysis, or train a neural network or
whatever pixel/noise-sensitive architecture, we'll use a rank-500
approximation
```