# Szeregi czasowe - Eigenfaces

## Sprawozdanie z laboratorium

*Wykonali:*

*Paweł Suchanicz*

*Elżbieta Dziedzic*

```python
In [19]: from matplotlib import pyplot as plt
         from matplotlib.image import imread
         import numpy as np
         import os
         import math
```

```python
In [20]: images_path = 'images/'
         images_files = os.listdir(images_path)
```

```python
In [21]: # zamiana na rgb
         def rgb_to_grayscale(rgb):
             return np.dot(rgb[...,:3], [0.2989, 0.5870, 0.1140])
```

```python
In [22]: # wyświetlenie obrazków
         def display_images(images):
             for i in range(images.shape[0]):
                 img = images[i].reshape(height,width)
                 plt.subplot(math.ceil(images_count / 5), 5, i+1)
                 plt.imshow(img, cmap='gray')
                 plt.subplots_adjust(right=1.2, top=1.2)
             plt.show()
```

Wczytanie obrazków i zamian na skalę szarości

```python
In [23]: width  = 250
         height = 250
         images_count = len(images_files)
         images = np.zeros((images_count, height*width))
         for i in range(images_count):
             img = rgb_to_grayscale(imread(images_path + images_files[i]))
             images[i] = np.array(img.flatten('C'), dtype='float64').flatten()

         display_images(images)
```
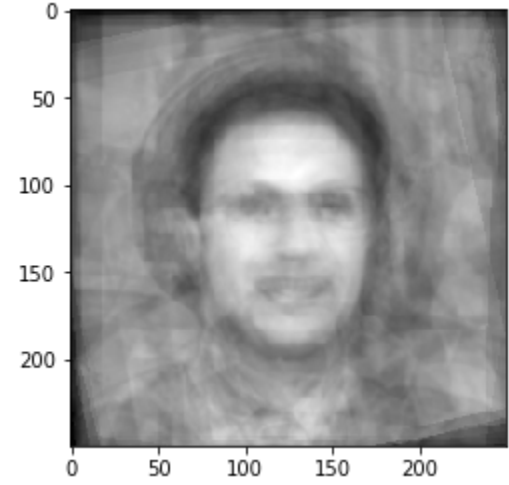


Wyliczenie średniego obrazu

```python
In [24]: mean_face = np.zeros((1, height * width))

         for image in images:
             mean_face = mean_face + image

         mean_face = mean_face / images.shape[0]

         plt.imshow(mean_face.reshape(height, width), cmap='gray')
         plt.show()
```



Znormalizowanie pozostałych obrazków (odjęcie średniego obrazu)

```python
In [25]: normalised_images = np.ndarray(shape=images.shape)

         for i in range(normalised_images.shape[0]):
             normalised_images[i] = images[i] - mean_face

         display_images(normalised_images)
```



```python
In [26]: covariance_matrix = np.cov(normalised_images.transpose())
```

```
---------------------------------------------------------------------------
MemoryError                               Traceback (most recent call last)
<ipython-input-26-6d3ddeba22c9> in <module>
----> 1 covariance_matrix = np.cov(normalised_images.transpose())

C:\ProgramData\Anaconda3\lib\site-packages\numpy\lib\function_base.py in cov(m, y, rowvar, bias, ddof, fweights, aweights)
   2448         else:
   2449             X_T = (X*w).T
-> 2450         c = dot(X, X_T.conj())
   2451         c *= np.true_divide(1, fact)
   2452     return c.squeeze()

MemoryError:
```

Nie można obliczyć macierzy kowariancji z powodu zbyt dużej ilości danych (brak pamięci)

```python
In [ ]:
```