HW6

June 1, 2021

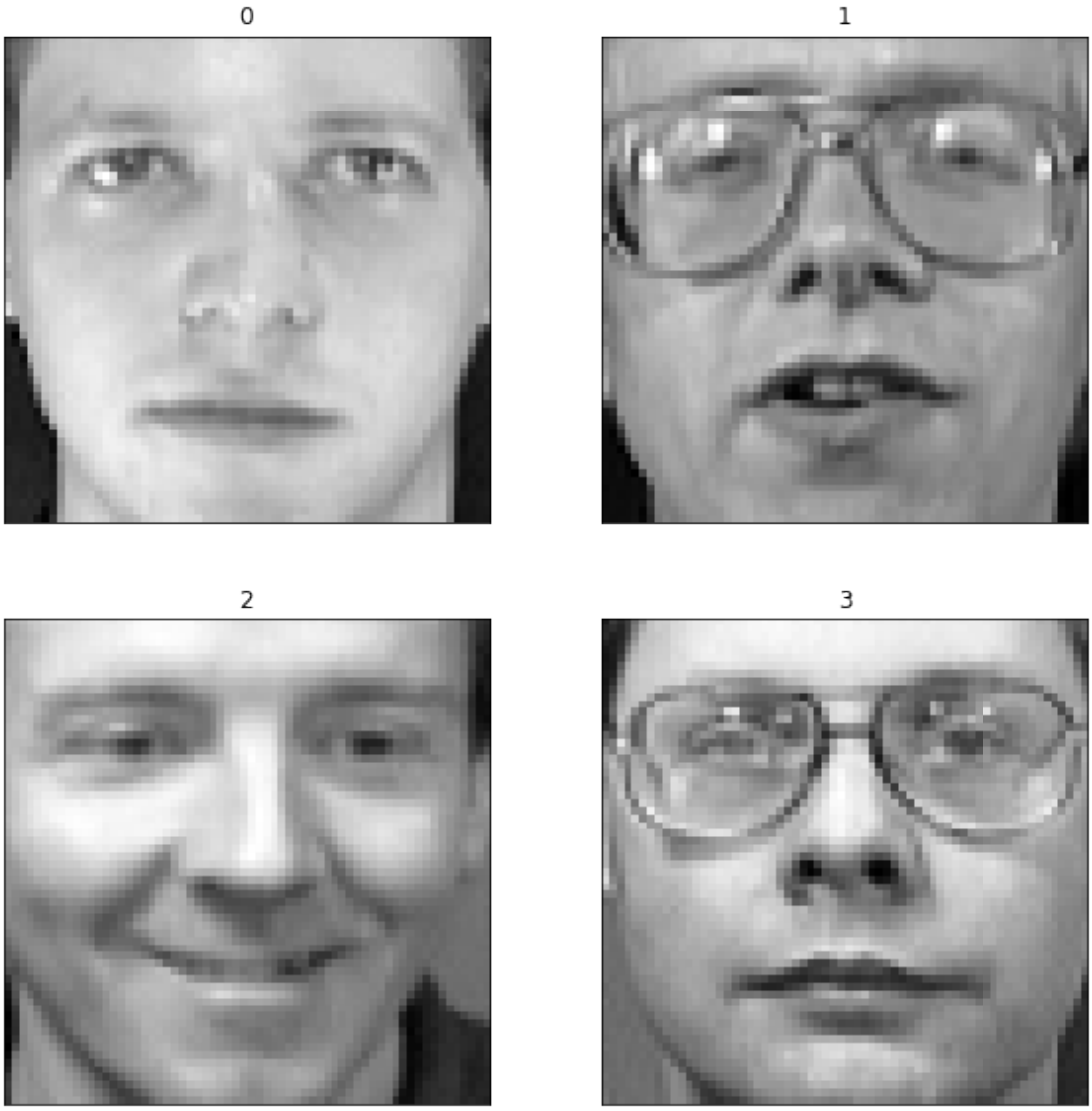
[115]:
```
from sklearn.datasets import fetch_olivetti_faces
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.decomposition import PCA
from sklearn.metrics import mean_squared_error
import pickle
```

[116]:
```
faces = fetch_olivetti_faces()
```

[117]:
```
X = faces.data
y = faces.target
```

1 0 wybrane zdjęcia

[118]:
```
nrows, ncols = 2, 2
plt.figure(figsize=(10,10))
plt.gray()
for i in range(ncols * nrows):
    ax = plt.subplot(nrows, ncols, i + 1)
    ax.matshow(faces.images[10*i,...])
    plt.xticks([]); plt.yticks([])
    plt.title(faces.target[10*i])
plt.show()
```

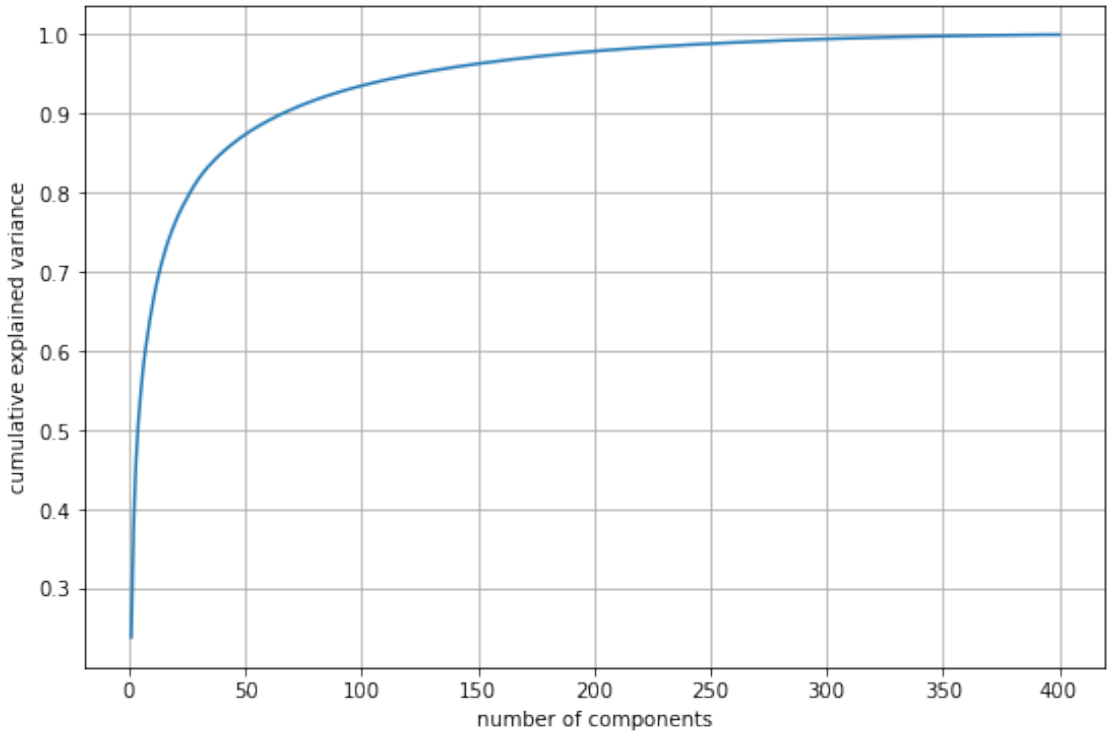


2 1 wybór liczby składowych

Na wykresie widać, że przegięcie jest ok 50.

[119]:
```
pca = PCA().fit(X)

plt.figure(figsize=(9,6))
plt.plot(range(1, len(pca.explained_variance_ratio_)+1), np.cumsum(pca.
 ↪explained_variance_ratio_))
plt.grid()
plt.xlabel('number of components')
plt.ylabel('cumulative explained variance');
```



[120]:
```
def scatter(x, colors):
    palette = np.array(sns.color_palette("hls", 40))

    f = plt.figure(figsize=(12, 12))
    ax = plt.subplot(aspect='equal')
    sc = ax.scatter(x[:,0], x[:,1], lw=0, s=40,
                    c=palette[colors.astype(np.int)])
    plt.xlim(-25, 25)
    plt.ylim(-25, 25)
    plt.legend()
    ax.axis('off')
    ax.axis('tight')

    return f, ax, sc,
```

2.1 Transformacja dzięki PCA

[121]:
```
pca = PCA(n_components = 50)
X_tr = pca.fit_transform(X)
scatter(X_tr, faces.target)
plt.show()
```

```
No handles with labels found to put in legend.
```

2.2 Stopień kompresji

[122]:
```
compression = X.shape[1] / X_tr.shape[1]
print('Stopien kompresji = ' + str(round(compression, 2)))
```

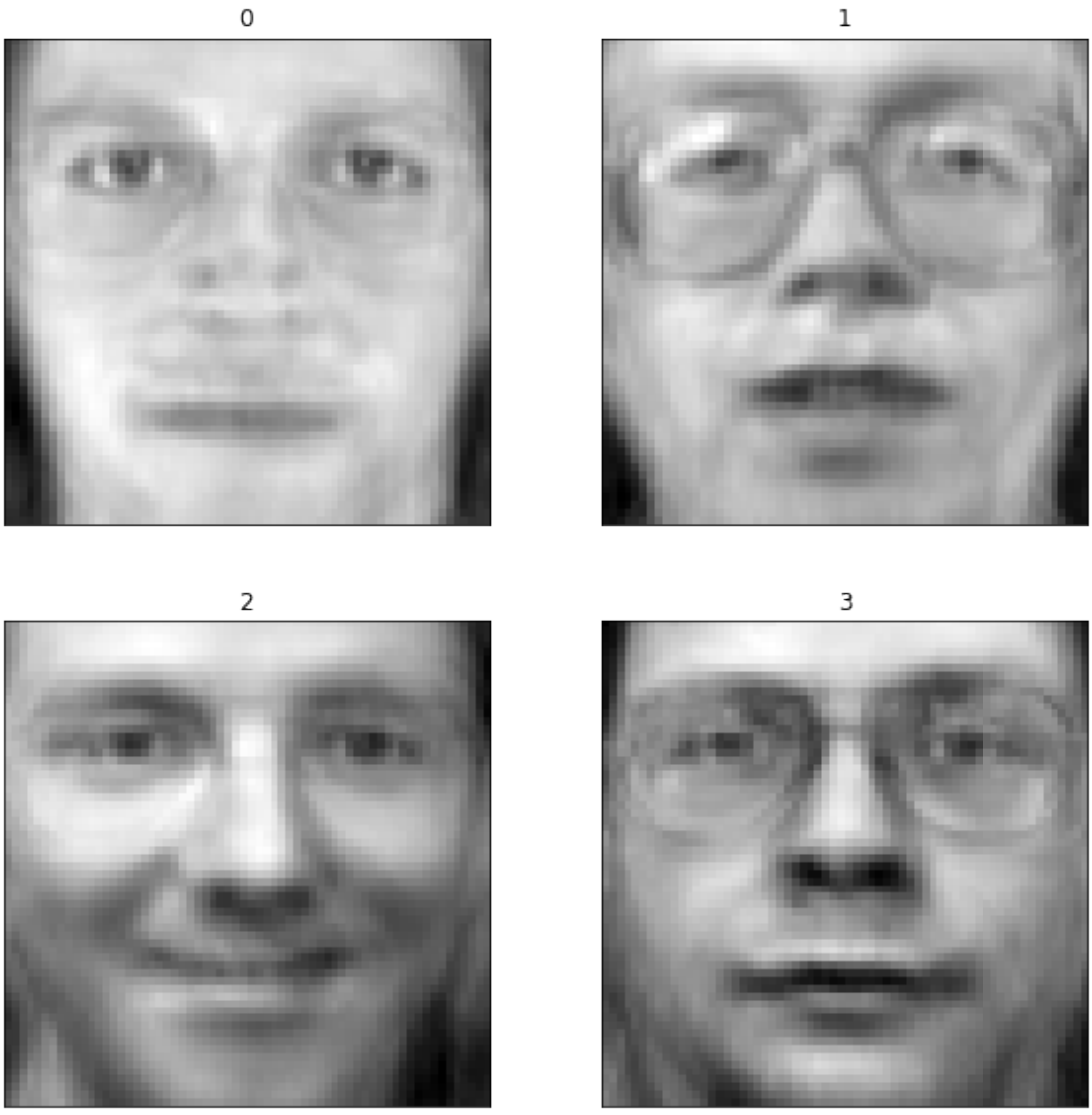
```
Stopien kompresji = 81.92
```

3 2 Transformacja odwrotna

[123]:
```
X_itr = pca.inverse_transform(X_tr)
```

3.1 Porównanie zdjęć

[124]:
```
nrows, ncols = 2, 2
plt.figure(figsize=(10,10))
plt.gray()
for i in range(ncols * nrows):
    ax = plt.subplot(nrows, ncols, i + 1)
    ax.matshow(X_itr[10*i,...].reshape(64,64))
    plt.xticks([]); plt.yticks([])
    plt.title(faces.target[10*i])
plt.show()
```



3.2 Błąd rekonstrukcji

[125]:
```
for i in range(4):
    print('RMSE dla zdjęcia: '+ str(i) + ' = ' +␣
 ↪str(mean_squared_error(X[10*i], X_itr[10*i], squared=False)))
```

```
RMSE dla zdjęcia: 0 = 0.04832953
RMSE dla zdjęcia: 1 = 0.05796484
RMSE dla zdjęcia: 2 = 0.043977305
RMSE dla zdjęcia: 3 = 0.053899612
```

[126]:
```
print('RMSE dla całego zbioru =  ' + str(mean_squared_error(X, X_itr,␣
 ↪squared=False)))
```

```
RMSE dla całego zbioru =  0.04784827
```

4 3 Modyfikacja obrazów

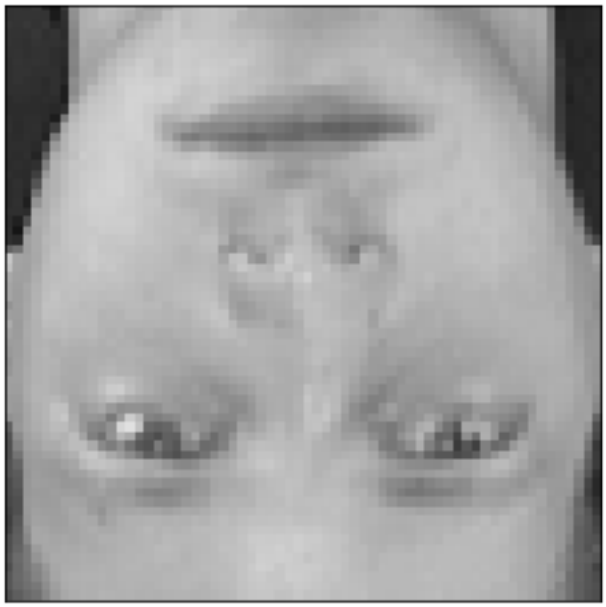
[127]:
```
def rotate(x):
    return x.reshape(64,64).T.reshape(4096)

def bright(x, factor):
    return x * factor

def flip(x):
    return np.flipud(x.reshape(64,64)).reshape(4096)

def picshow(x):
    plt.matshow(x.reshape(64,64),interpolation='nearest', vmin=0, vmax=1)
    plt.xticks([]); plt.yticks([])
```

[128]:
```
ax = picshow(rotate(X[0]))
ax = picshow(bright(X[0], 1.5))
ax = picshow(flip(X[0]))
plt.show()
```



[129]:
```
L = [X[5], X[25], X[45], X[65], X[85]]
X_r = list(map(rotate, L))
X_b = list(map(lambda x: bright(x, 1.5), L))
X_f = list(map(flip, L))
```

[130]:
```
LS = [L, X_r, X_b, X_f]
LS_t = list(map(pca.transform, LS))
LS_i = list(map(pca.inverse_transform, LS_t))
```

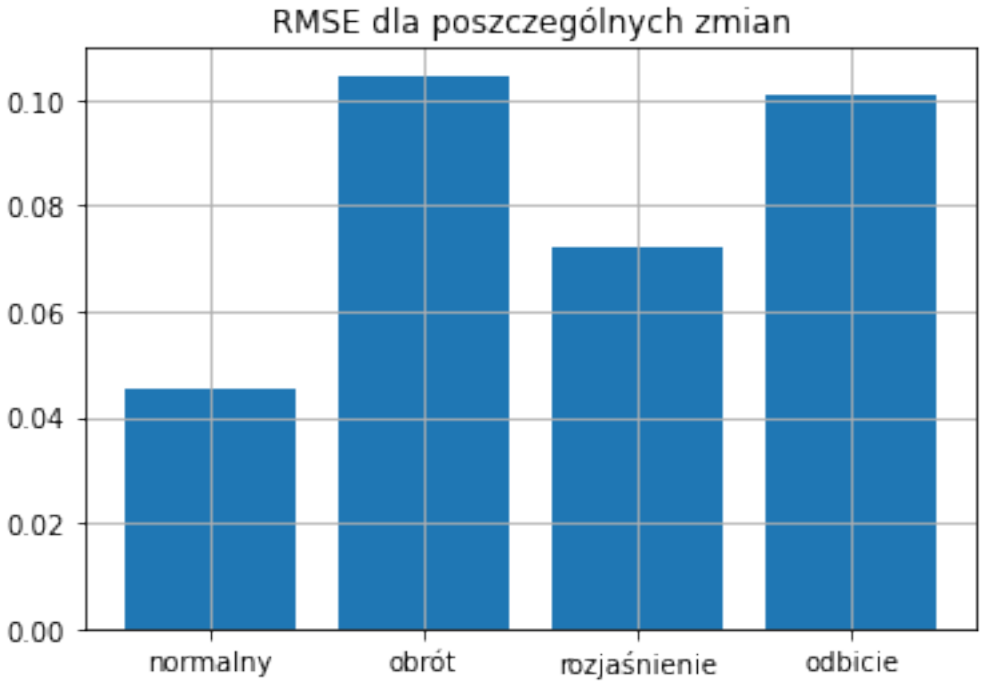
5 5 Wnioski

Jak widać wszystkie zmiany powodują znaczny wzrost RMSE, z czego najwięszy obroty i odbicia. Dzięki takim anomaliom możliwe jest wyłapanie zdjęć które odstają zostały zmienione w jeden z powyższych sposobów.

[131]:
```
rmse = []
names = ['normalny', 'obrót', 'rozjaśnienie', 'odbicie']
for i, j, k in zip(LS, LS_i, names):
    rmse.append(mean_squared_error(i, j, squared=False))
    print('RMSE dla zbioru ' + k + ' =  ' + str(round(mean_squared_error(i, j,␣
 ↪squared=False), 5)))
```

```
RMSE dla zbioru normalny =  0.0454
RMSE dla zbioru obrót =  0.10461
RMSE dla zbioru rozjaśnienie =  0.07204
RMSE dla zbioru odbicie =  0.10077
```

[132]:
```
plt.grid()
plt.bar(height= rmse, x = names)
plt.title('RMSE dla poszczególnych zmian')
plt.show()
```



[133]:
```
flatLSi = [item for sublist in LS_i for item in sublist]
```

[134]:
```
nrows, ncols = 4, 5
plt.figure(figsize=(10,10))
plt.gray()
for i in range(nrows * ncols):
    ax = plt.subplot(nrows, ncols, i + 1)
    ax.matshow(flatLSi[i].reshape(64,64))
    plt.xticks([]); plt.yticks([])
    plt.title(names[i//5])
plt.show()
```


normalny
normalny
normalny
normalny
normalny
obrót
obrót
obrót
obrót
obrót
rozjaśnienie
rozjaśnienie
rozjaśnienie
rozjaśnienie
rozjaśnienie
odbicie
odbicie
odbicie
odbicie
odbicie

[ ]: