hw3 Machine Learning Fall 2018 | Part II - Programming | amc1354 & ads798

Problem 2

(d) Validate results in Python

In [4]:
```
import numpy as np
from numpy import linalg as la

A = np.array([[0, 14], [6, 9]])
[V, D] = la.eig(A)

print("V=", V, "\nD=", D)
```

```
V= [-5.71028893 14.71028893]
D= [[-0.9259401  -0.6894021 ]
 [ 0.37767039 -0.72437887]]
```

Problem 3

(c) Show sample covariance matrix and its largest eigenvalue

In [22]:
```
B0 = np.array([[5, 2, 4],
               [9, 6, 4],
               [7, 1, 0],
               [2, 5, 6]
               ])
S0 = np.cov(B0.T)

mean = B0.mean(0)
B = np.subtract(B0, mean)

S = np.cov(B.T)
```

The Covariance matrix S is:

In [23]:
```
print(S)
```

```
[[ 8.91666667  0.16666667 -4.16666667]
 [ 0.16666667  5.66666667  4.33333333]
 [-4.16666667  4.33333333  6.33333333]]
```

And the element in position 1,3 (or 3,1) matches what we manually calculated in part *(b)*. The covariance matrix's largest eigenvalue is:

In [24]:
```
[V, D] = la.eig(S)
print(max(V))
```

```
12.978818870145862
```

(d) Show first two columns of Z

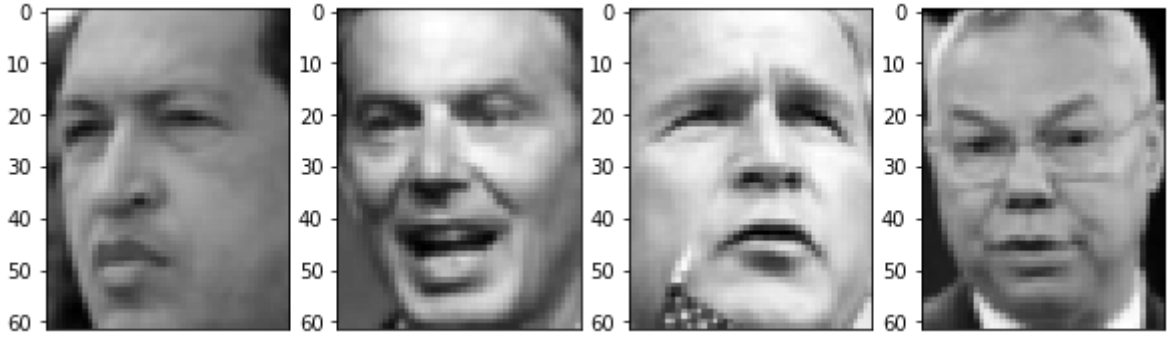
In [25]:
```
import sklearn.decomposition as skd
# .fit computes the principal components  (n_components of them)
# The columns of W are the eigenvectors of the covariance matrix of X
pca = skd.PCA(n_components = 3)
skd.PCA.fit(pca,B)
W1 = pca.components_
W = W1.transpose()
Z = pca.transform(B)

print(Z[:,:2])
```

```
[[ 0.26018674 -1.41900435]
 [-0.87353472  4.03721245]
 [-4.04749635 -1.8486773 ]
 [ 4.66084433 -0.7695308 ]]
```

Problem 5 (it should actually be #4)

In [26]:
```
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.datasets import fetch_lfw_people
lfw_people = fetch_lfw_people(min_faces_per_person=70)
n_samples, h, w = lfw_people.images.shape
npix = h*w
fea = lfw_people.data
def plt_face(x):
    global h,w
    plt.imshow(x.reshape((h, w)), cmap=plt.cm.gray)
    plt.xticks([])
plt.figure(figsize=(10,20))
nplt = 4
for i in range(nplt):
    plt.subplot(1,nplt,i+1)
    plt_face(fea[i])
```

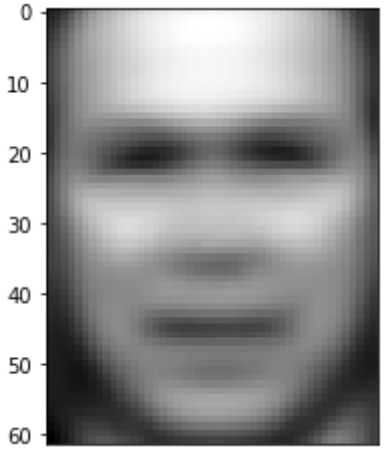


(a) Display the fourth face in the dataset

In [27]:
```
plt_face(fea[3])
```

(b) Compute and display the mean of all examples in the dataset

In [28]:
```
mean_faces = fea.mean(0)
plt_face(mean_faces)
```



(c) Compute the 5 top principal components and show the values of the associated 5 attributes of the fourth image in the dataset?

In [29]:
```
import sklearn.decomposition as skd
pca = skd.PCA(n_components = 5)
skd.PCA.fit(pca, fea)
W1 = pca.components_
W = W1.transpose()
Z = pca.transform(fea)

print(Z[3,:])
```

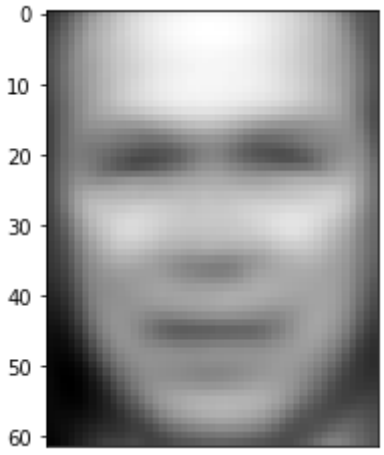
```
[ 202.54237  -261.4773    418.97363   -29.399895   39.7872  ]
```

(d) Representations of the fourth image, based on 5 or 50 features instead of the original 2914 pixel features

In [30]:
```
pca = skd.PCA(n_components = 50)
skd.PCA.fit(pca, fea)
W1 = pca.components_
W = W1.transpose()
Z = pca.transform(fea)
```

For 5 features:

In [31]:
```
fea_approx_5 = np.dot(W[:,:5], Z[3, :5]) + fea[:].mean(0)
plt_face(fea_approx_5)
```



For 50 features:

In [21]:
```
fea_approx_50 = np.dot(W, Z[3]) + fea[:].mean(0)
plt_face(fea_approx_50)
```