CS 5785 Homework 4

Russell Bingham and Omer Ifrach (rcb366@cornell.edu and oi38@cornell.edu)

December 5, 2020

1. Clustering for Text Analysis

(a) Document-Wise Data

After loading the data and setting up the basic K-Means implementation (leveraging `sklearn`), we started playing around with the number of clusters. Determining the correct k value seems to be a largely qualitative and subjective process, but after testing various values between 5 and 20, we settled on $k = 10$ as a reasonable value for the rest of our tests. Using $k = 5$ tends to create very general clusters that don't illuminate much about the data, whereas $k = 20$ tends to create lots of clusters with only one element, which feels purpose-defeating. Thus we settled on $k = 10$ as the happy medium between these extremes (even so, this often creates some 1-element clusters).

We implemented processes to output the top 10 words and top documents for each cluster found by the model. This results in a lot of text output, so we've attached a full sample set of output as "documentWiseK10.pdf" in the submission folder. This is just one snapshot of the data groupings, as the output data changes each time the model is retrained.

A rough characterization of the sample 10 categories is as follows (top 3 words, qualitative subject assessment):

Cluster 1 – ("gene", "genes", "cells") **Cellular biology (fieldwork)** (240 articles)

Cluster 2 – ("residues", "crystal", "structural") **Organic Chemistry** (38 articles)

Cluster 3 – ("says", "researchers", "scientists") **News-Like Articles about Science** (94 articles)

Cluster 4 – ("energy", "fig", "electron") **Molecular Physics** (209 articles)

Cluster 5 – ("mating", "albicans", "mtl") **Albican Mating** (2 articles)

Cluster 6 – ("mail", "years", "news") **Corrections and Study of Science** (350 articles)

Cluster 7 – ("aptamer", "aptamers", "ligand") **Aptamers** (1 article)

Cluster 8 – ("ppy", "polymer", "conjugated") **Polymer Actuators** (1 article)

Cluster 9 – ("cells", "protein", "cell") **Cellular biology (labwork)** (280 articles)

Cluster 10 – ("values", "estimates", "global") **Geology/Climate** (158 articles)

In this example set, there are 3 negligible clusters and 7 well-defined topic areas. Across testing, even for higher k values, the most consistent of these clusters were those in cellular biology and corrections, suggesting that the journal Science tends to have articles on those things most consistently, and that the vocabulary of those articles is distinct. Cellular biology has so many articles that it even splits into two clusters, one encompassing studies on real species with behavioral implications, and one dealing mostly with more theoretical work on isolated cells. Unsurprisingly, the other general fields of science tend to split into their own clusters, as articles in similar topic areas will use similar words in their prose.

Two of the negligible clusters stand out: those on "albican mating" and "aptamers". At a glance, the words in the article titles are extremely niche (what even are albicans and aptamers?). Thus it makes sense that niche articles like these would be most likely to split into their own clusters.

This kind of algorithm seems most useful for deciding **what topic an article is about**.

(b) Term-Wise Data

Clustering this data using the term-wise dataset produces results that are different than the previous section, but that are also similar in interesting ways. We used the same code file, analyzing both the most relevant words and the most relevant documents for each of the 10 clusters. The full output of one test run is similarly attached, in the file named "termWiseK10.pdf".

A rough characterization of this output data is as follows (top 3 words, subjective name for cluster):

Cluster 1 – ("terminus", "cooh", "nh2") **Genetics** (120 words)

Cluster 2 – ("suggestion") **negligible** (1 word)

Cluster 3 – ("interglacial", "clim", "volcanism") **Climate** (384 words)

Cluster 4 – ("recalls", "clinton", "fight") **Politics and Society** (472 words)

Cluster 5 – ("vertical", "dashed", "parameters") **Data and Measurements** (89 words)

Cluster 6 – ("excitations", "coherence", "resonant") **Materials Physics** (600 words)

Cluster 7 – ("corresponding") **negligible** (1 word)

Cluster 8 – ("org", "sciencemag", "vol") **Acronyms (bio)** (878 words)

Cluster 9 – ("aptamers", "lcts", "dnag") **Acronyms (corrections)** (2820 words)

Cluster 10 – ("natl", "acad", "start") **Abbreviations (orgo)** (111 words)

Instead of being grouped by general topic area, these clusters end up being more specific and (mostly) coherent. The qualitative judgements for cluster titles were more clear when looking at this output, largely because the word groupings were able to pick out more specific subsections of scientific fields. The physics clusters were more clearly separated between particle physics and materials, and the biology sections were seemingly separated by what kinds of acronyms they use (simple molecules for genetics, complex acronyms for orgo and corrections). There is also a particularly interesting category based on words relevant to graphing and data analysis. Papers in this category come from several different science disciplines, including physics and biology and chemistry, suggesting that the word-based model groups things more strongly by the actual tasks and actions that are going on in the text.

On the flip side, the negligible cluster categories are far more nonsensical here. Before it was evident that the 1-element clusters were for articles on very niche topics, but it's not immediately clear why words like "suggestion" and "corresponding" are so unusual as to merit their own clusters.

This kind of algorithm seems most useful for picking out **what kinds of methods are used** in the paper.

2. EM Algorithm and Implementation

(a) Data Download/Display

We downloaded the data file as requested, and reformatted the file directly into a text file to avoid parsing the header section. We then loaded the data with `pandas.load-csv()`, using the "/s+" flag to get everything into the dataframe correctly. We then plotted the data on a 2D plane as specified – this plot can be seen below in Figure 1.

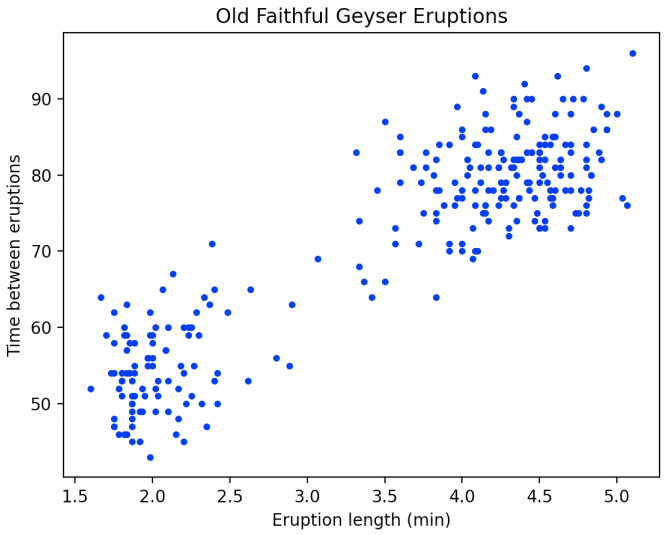


Figure 1: Raw data for Old Faithful Geyser eruptions

(b) Implement GMM

Here we leverage the `sklearn` implementation of GMM to accomplish the stated objective. This implementation uses the EM algorithm to conduct GMM, and all the parameters (spherical covariance, termination criteria, and initialization conditions) are set as function paramters to the `gmm()` constructor call. We conducted tests with the termination criteria and random initial conditions and found that a termination threshold of $1e-4$ produces repeatable mean results up to 2 decimal points of precision. To validate this repeatable mean result, we plotted it on top of the raw data, with the means shown in red on top of the data in Figure 2 below.

We ran the GMM model 50 times with random initial guesses, and produced a histogram of iteration count frequencies, included below in Figure 3. This data shows that the random initialization typically takes about a dozen iterations to converge, but occasionally makes a good guess and converges far quicker.

(c) K-Means Guessing

To implement this section, we again leveraged the `sklearn` GMM implementation. This function actually provides support for exactly the implementation described in the question, and since the homework says `sklearn` is fair game, we used that tool outright. Passing "kmeans" as the initialization parameter causes the GMM implementation to run k-means before GMM to come up with good initial guesses. We tried to graph a histogram of the iteration frequencies, but perhaps unsurprisingly

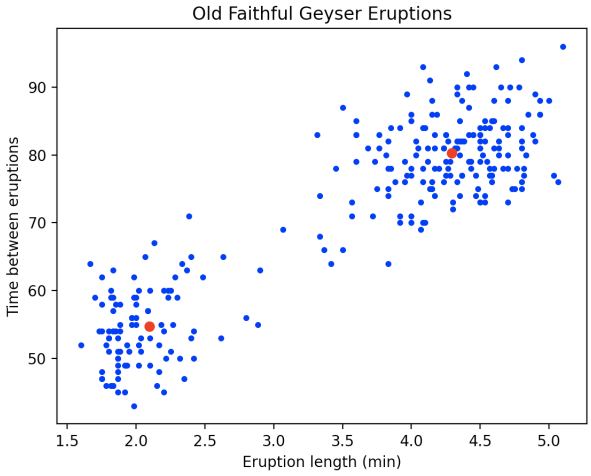


Figure 2: Raw data with calculated GMM means superimposed in red

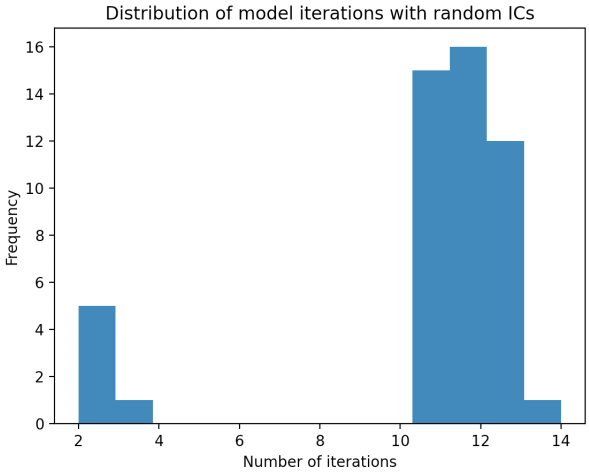


Figure 3: Raw data with calculated GMM means superimposed in red

the algorithm always uses the same number of iterations to converge when using non-random initial conditions: 2.

All is perhaps not lost for the randomized GMM version, and we tested the speed of both algorithms to get a more meaningful comparison (since both algorithms converge to the exact same mean values given a stop condition of $1e-4$). Over 50 trials, the randomized algorithm, though it conducts more GMM iterations by a factor of 6, is actually more than twice as fast as the k-means version. The random algorithm takes an average of 4.1 milliseconds, while the k-means algorithm takes an average of 8.6 milliseconds. This means the k-means initial condition finder is much more time consuming than each GMM iteration. Particularly for this simple dataset, it would be much more efficient to just enter rough guesses of what the mean of each cluster is by inspection–k-means is just too heavy-duty for 2D, easily separable data.

Thus we can say that randomized GMM is significantly more efficient than k-means GMM when time is of high importance. In fact, since the two algorithms converge to the same values, it can be said that k-means reduces the functional performance of GMM on this data set.

3. Eigenface for Face Recognition

See Appendix A for guidance to our answers to 3.

4. W1: SVD and Eigendecomposition

We know from theory that the singular values in the S matrix of the SVD are the square roots of the eigenvalues of M^TM, and that the V matrix of the SVD will give its eigenvectors. To demonstrate this, we define a generic test matrix:

$$M = \begin{bmatrix} 4 & 2 & 1 \\ 0 & 9 & 6 \\ -1 & -2 & 0 \\ -4 & 0 & -1 \end{bmatrix}$$

First, we find M^TM:

$$M^TM = \begin{bmatrix} 33 & 10 & 8 \\ 10 & 89 & 56 \\ 8 & 56 & 38 \end{bmatrix}$$

The eigendecomposition of M^TM is as follows:

$$eigenvalues(M^TM) = \begin{bmatrix} 1.9093 & 31.3274 & 126.7634 \end{bmatrix}$$

$$eigenvectors(M^TM) = \begin{bmatrix} 0.0442 & 0.9899 & 0.1346 \\ 0.5367 & -0.1372 & 0.8325 \\ -0.8426 & -0.0355 & 0.5374 \end{bmatrix}$$

We now take the SVD of the original test matrix. The S matrix from this is:

$$S = \begin{bmatrix} 11.2589 & 0 & 0 \\ 0 & 5.5971 & 0 \\ 0 & 0 & 1.3818 \\ 0 & 0 & 0 \end{bmatrix}$$

When element-wise squared, we get:

$$S_{squared} = \begin{bmatrix} 126.7634 & 0 & 0 \\ 0 & 31.3274 & 0 \\ 0 & 0 & 1.9093 \\ 0 & 0 & 0 \end{bmatrix}$$

Evidently, this last matrix contains the eigenvalues of M^TM. The V matrix from SVD is:

$$V = \begin{bmatrix} 0.1346 & 0.9899 & 0.0442 \\ 0.8325 & -0.1372 & 0.5367 \\ 0.5374 & -0.0355 & -0.8426 \end{bmatrix}$$

Here we can see that this matrix contains the eigenvectors of M^TM as shown above. Thus we can say that the SVD does give us the appropriate information about the eigendecomposition of M^TM. Another valid example of this can be seen in question W3 below, where the same relationship holds.

5. W2: Weights for Clustering

Here we simply need to prove that the given definitions are valid, and that when the definition of the "proper transformed data" is substituted into its euclidean distance formula, the factored result is equivalent to the weighted euclidean distance formula.

We begin by stating the givens:

$d_e^{(w)}(x_i, x_{i'}) = \frac{\sum_{l=1}^{p} w_l (x_{il} - x_{i'l})^2}{\sum_{l=1}^{p} w_l}$

$d_e(z_i, z_{i'}) = \sum_{l=1}^{p} (z_{il} - z_{i'l})^2$

$z_{il} = x_{il} (\frac{w_l}{\sum_{l=1}^{p} w_l})^{1/2}$

We substitute the definition of z_{il} into the second given and factor out the square-rooted weight term:

$d_e(z_i, z_{i'}) = \sum_{l=1}^{p} ((x_{il} - x_{i'l})(\frac{w_l}{\sum_{l=1}^{p} w_l})^{1/2})^2$

Since the weight factor is a scalar multiplication operation inside the parentheses, we can pull it out:

$d_e(z_i, z_{i'}) = \sum_{l=1}^{p} (\frac{w_l}{\sum_{l=1}^{p} w_l})(x_{il} - x_{i'l})^2$

Here, since the $\sum_{l=1}^{p} w_l$ term is a scalar division factor on every term of the outer sum, we can factor it out:

$d_e(z_i, z_{i'}) = \frac{\sum_{l=1}^{p} w_l (x_{il} - x_{i'l})^2}{\sum_{l=1}^{p} w_l}$

Since this form of $d_e(z_i, z_{i'})$ is equivalent to the given definition of $d_e^{(w)}(x_i, x_{i'})$, we can say the target statement holds:

$d_e^{(w)}(x_i, x_{i'}) = d_e(z_i, z_{i'})$

6. W3: SVD of Rank Deficient Matrix

Given $M = \begin{bmatrix} 1 & 0 & 3 \\ 3 & 7 & 2 \\ 2 & -2 & 8 \\ 0 & -1 & 1 \\ 5 & 8 & 7 \end{bmatrix}$:

(a) Compute the matrices $M^T M$ and MM^T.

$$M^T M = \begin{bmatrix} 39 & 57 & 60 \\ 57 & 118 & 53 \\ 60 & 53 & 127 \end{bmatrix}$$

$$MM^T = \begin{bmatrix} 10 & 9 & 26 & 3 & 26 \\ 9 & 62 & 8 & -5 & 85 \\ 26 & 8 & 72 & 10 & 50 \\ 3 & -5 & 10 & 2 & -1 \\ 26 & 85 & 50 & -1 & 138 \end{bmatrix}$$

(b) Find the eigenvalues for your matrices of part (a).

$eigenvalues(M^T M) = \begin{bmatrix} 0 & 69.3295 & 214.6705 \end{bmatrix}$

$eigenvalues(MM^T) = \begin{bmatrix} 0 & 0 & 0 & 69.3295 & 214.6705 \end{bmatrix}$

(c) Find the eigenvectors for your matrices of part (a).

The following eigenvectors are given as a matrix of column vectors, corresponding to the above eigenvalues from left to right:

$$eigenvectors(M^T M) = \begin{bmatrix} 0.9045 & 0.0146 & 0.4262 \\ -0.3015 & 0.7286 & 0.6150 \\ -0.3015 & -0.6848 & 0.6634 \end{bmatrix}$$

$$eigenvectors(MM^T) = \begin{bmatrix} 0.1432 & -0.9446 & 0.0045 & -0.2450 & 0.1649 \\ 0.5256 & 0.0470 & 0.5419 & 0.4533 & 0.4716 \\ 0.2997 & 0.3197 & 0.0826 & -0.8294 & 0.3365 \\ -0.6311 & -0.0475 & 0.7554 & -0.1697 & 0.0033 \\ -0.4638 & 0.0328 & -0.3591 & 0.1331 & 0.7982 \end{bmatrix}$$

(d) Find the SVD of matrix M.

The matrices U, S, and V that make up the SVD of M are as follows:

$$U = \begin{bmatrix} -0.1649 & -0.2450 & 0.9485 & 0.0212 & -0.1128 \\ -0.4716 & 0.4533 & -0.0561 & 0.1031 & -0.7472 \\ -0.3365 & -0.8294 & -0.3037 & -0.1519 & -0.2890 \\ -0.0033 & -0.1697 & -0.0617 & 0.9828 & 0.0393 \\ -0.7982 & 0.1331 & -0.0346 & -0.0053 & 0.5865 \end{bmatrix}$$

$$S = \begin{bmatrix} 14.6516 & 0 & 0 \\ 0 & 8.3264 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$V = \begin{bmatrix} -0.4262 & 0.0146 & -0.9045 \\ -0.6150 & 0.7286 & 0.3015 \\ -0.6634 & -0.6848 & 0.3015 \end{bmatrix}$$

When multiplied together in the form $U * S * V^T$, the above matrices correctly return exactly M.

(e) Compute a one-dimensional approximation of M by zeroing out one singular value.

When we zero out the smaller singular value (8.3264), multiply the result $U * S * V^T$, and collapse the matrix values into a column vector, we get:

$$M_{1D} = \begin{bmatrix} 4.1192 \\ 11.7795 \\ 8.4034 \\ 0.0826 \\ 19.9353 \end{bmatrix}$$

This is very close to the row-wise sum of M, which would be:

$$M_1 = \begin{bmatrix} 4 \\ 12 \\ 8 \\ 0 \\ 20 \end{bmatrix}$$

7. Appendix A: Coding 3 Writeup

See the following pages for an exported PDF of the notebook that is our Q3 submission. See the relevant code and other files in the "Q3" subfolder of the submission folder.

Question 3 - Eigenface for face recognition

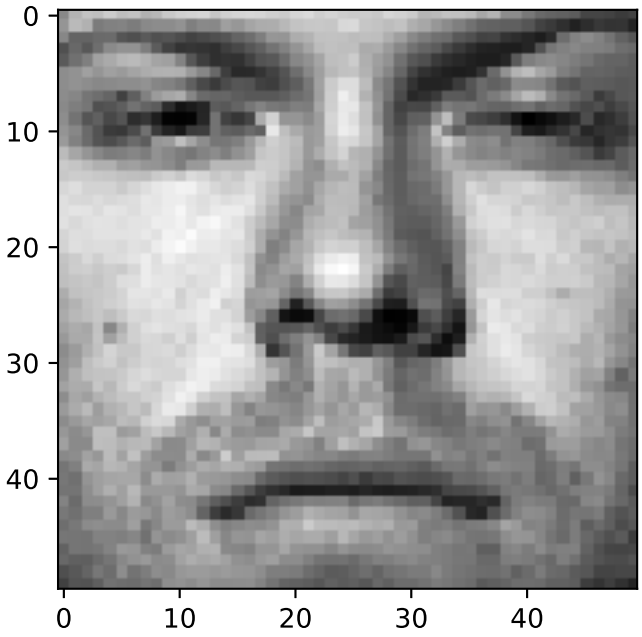
In [2]:
```
import numpy as np
from scipy import misc
from matplotlib import pylab as plt
import matplotlib.cm as cm
from sklearn.multiclass import OneVsRestClassifier
from sklearn.linear_model import LogisticRegression
%matplotlib inline
from PIL import Image
```

(a) Download and unzip The Face Dataset (faces.zip), You will find a folder called im ages which contains all the training and test images; train.txt and test.txt specifies the training set and test (validation) set split respectively, each line gives an image path and the corresponding label.

(b) Load the training set into a matrix X: there are 540 training images in total, each has 50 × 50 pixels that need to be concatenated into a 2500-dimensional vector. So the size of X should be 540×2500, where each row is a flattened face image. Pick a face image from X and display that image in grayscale. Do the same thing for the test set. The size of matrix Xtest for the test set should be 100×2500.

In [3]:
```
train_labels, train_data = [], []
for line in open('faces/train.txt'):
    im = Image.open(line.strip().split()[0])
    im = np.array(im)
    train_data.append(im.reshape(2500,))
    train_labels.append(line.strip().split()[1])
train_data, train_labels = np.array(train_data, dtype=float), np.array(train_l
abels, dtype=int)

print(train_data.shape, train_labels.shape)
plt.imshow(train_data[10, :].reshape(50,50), cmap = cm.Greys_r)
plt.show()

print(train_data[10, 10])
```

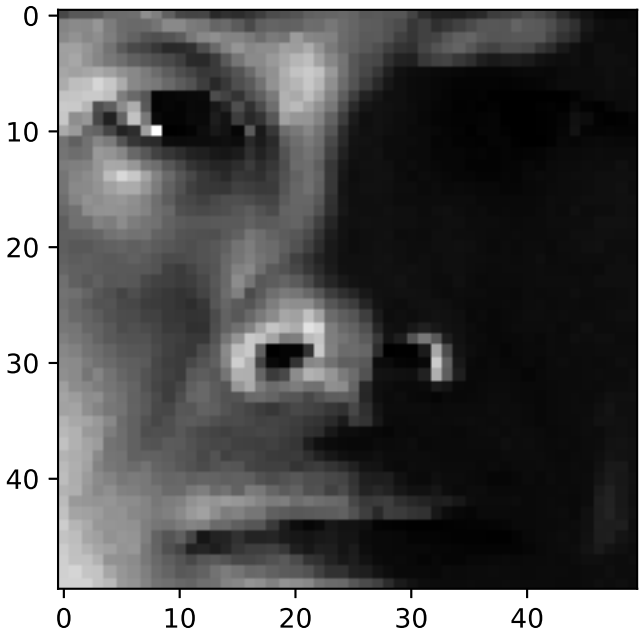
```
(540, 2500) (540,)
```




```
168.0
```

In [4]:
```
test_labels, test_data = [], []
for line in open('./faces/test.txt'):
    im = Image.open(line.strip().split()[0])
    im = np.array(im)
    test_data.append(im.reshape(2500,))
    test_labels.append(line.strip().split()[1])
test_data, test_labels = np.array(test_data, dtype=float), np.array(test_label
s, dtype=int)

print(test_data.shape, test_labels.shape)
plt.imshow(test_data[10, :].reshape(50,50), cmap = cm.Greys_r)
plt.show()

print(test_data[10, 10])
```

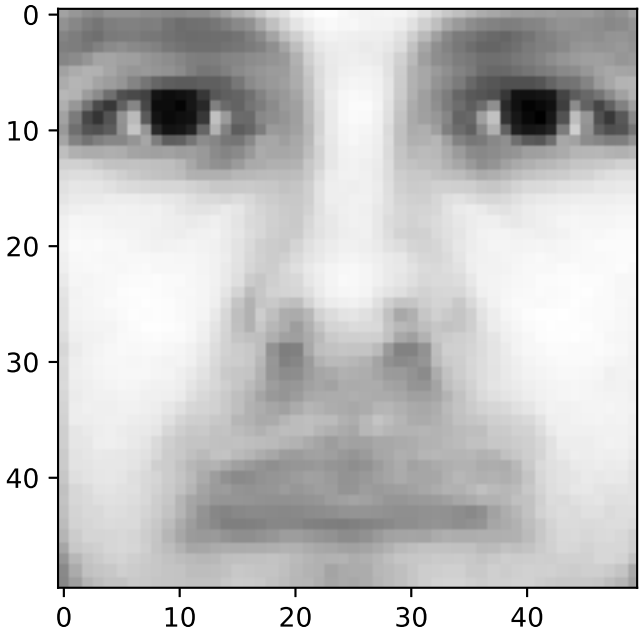
```
(100, 2500) (100,)
```




```
56.0
```

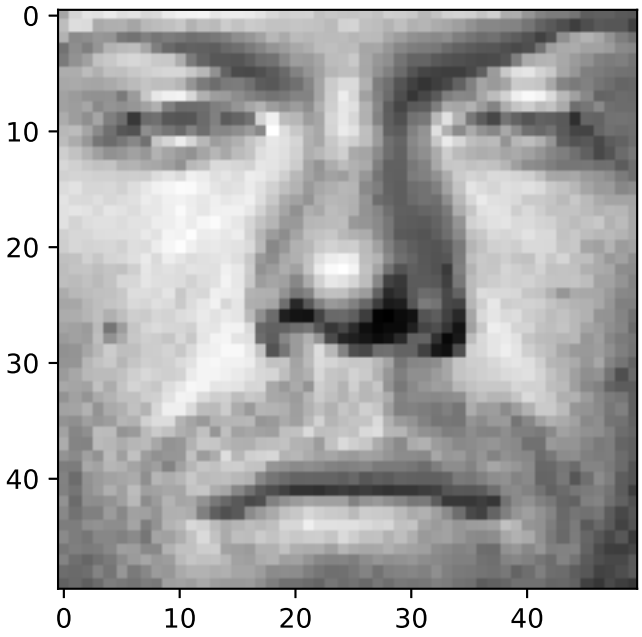
(c) Average Face. Compute the average face μ from the whole training set by summing up every column in X then dividing by the number of faces. Display the average face as a grayscale image.

In [16]:
```
avg_face = train_data.mean(axis=0)

plt.imshow(avg_face.reshape(50,50), cmap = cm.Greys_r)
plt.show()
```

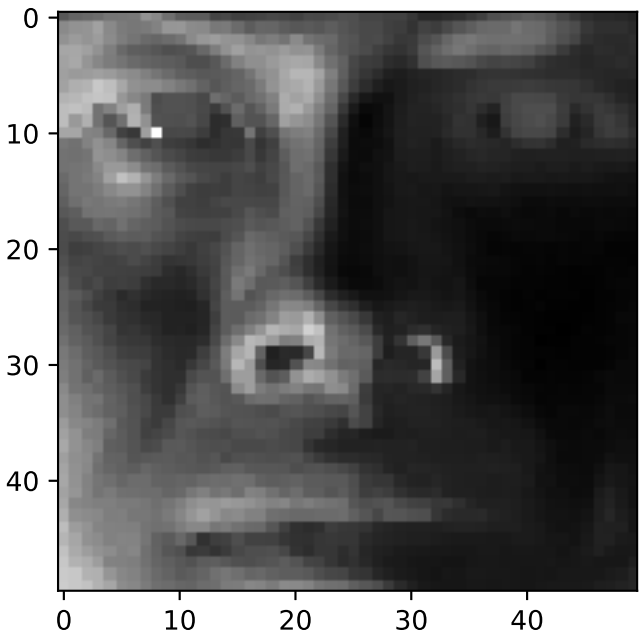


(d) Mean Subtraction. Subtract average face μ from every column in X. That is, xi := xi − μ, where xi is the i-th column of X. Pick a face image after mean subtraction from the new X and display that image in grayscale. Do the same thing for the test set Xtest using the precomputed average face μ in (c).

In [9]:
```
new_mtx = []

for i in train_data:
    new_mtx.append((i - avg_face))

new_mtx = np.array(new_mtx)
plt.imshow(new_mtx[10, :].reshape(50,50), cmap = cm.Greys_r)
plt.show()
```



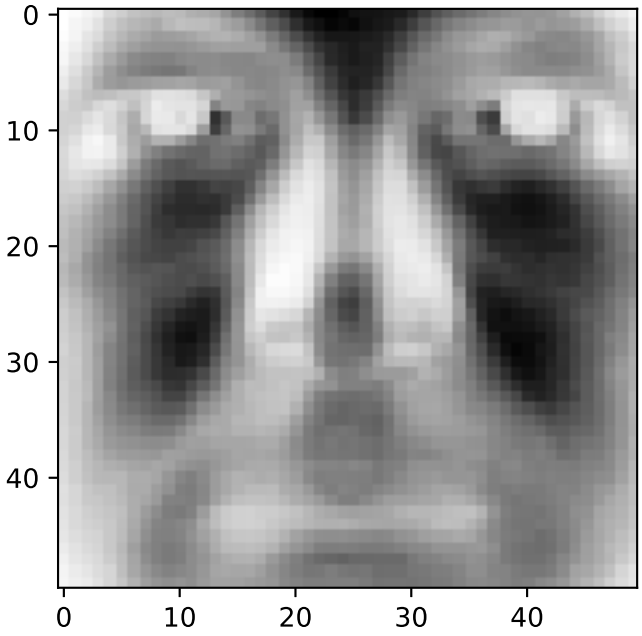
In [17]:
```
test_mtx = []

for i in test_data:
    test_mtx.append((i - avg_face))

test_mtx = np.array(test_mtx)

plt.imshow(test_mtx[10, :].reshape(50,50), cmap = cm.Greys_r)
plt.show()
```

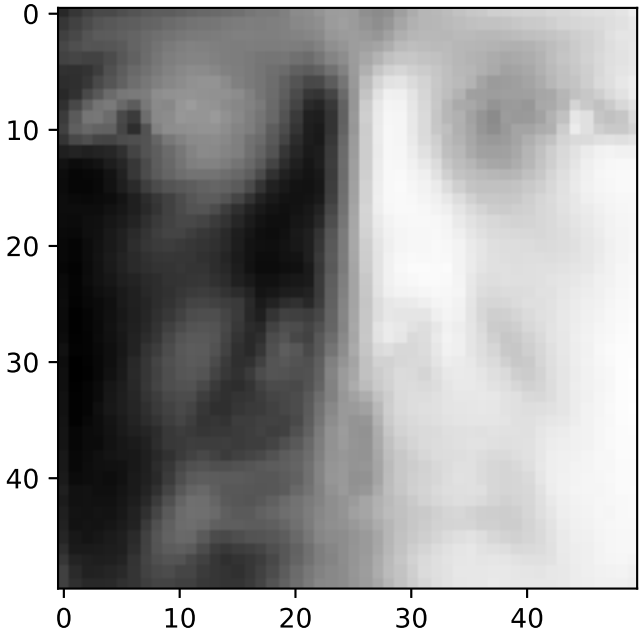


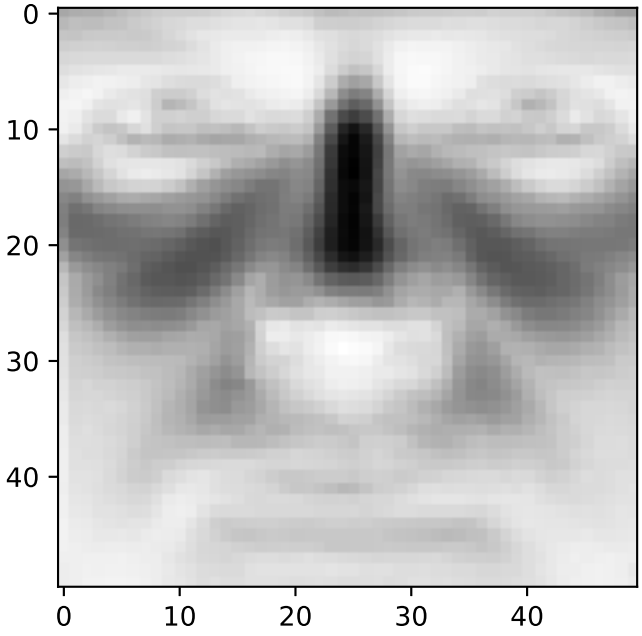
(e) Eigenface. Perform eigendecomposition on XT X = VΛVTto get eigenvectors VT, where each row of VT has the same dimension as the face image. We refer to vi, the i-th row of VT, as i-th eigenface. Display the first 10 eigenfaces as 10 images in grayscale.

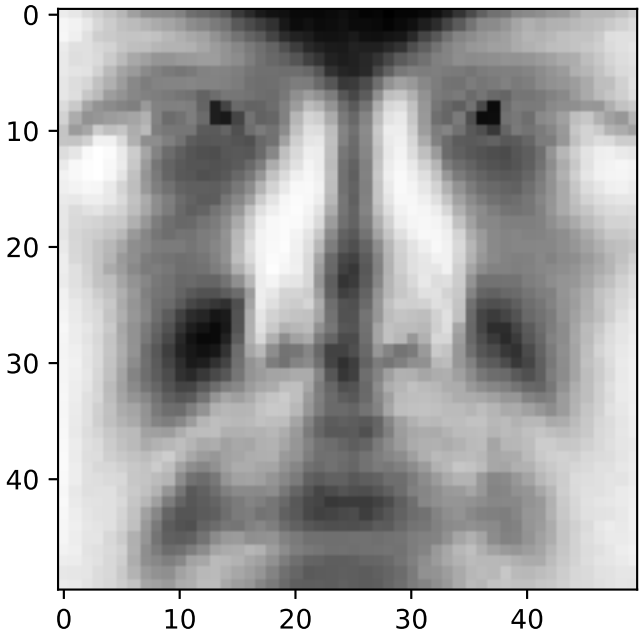
In [10]:
```
U, s, V = np.linalg.svd(new_mtx, full_matrices=False)
S = np.diag(s)
print(V.shape)

for i in range(10):
    plt.imshow(V[i, :].reshape(50,50), cmap = cm.Greys_r)
    plt.show()
```

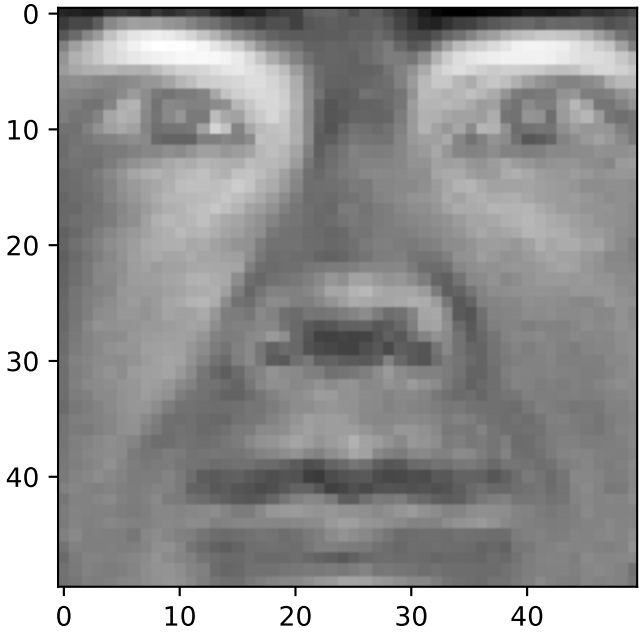
```
(540, 2500)
```

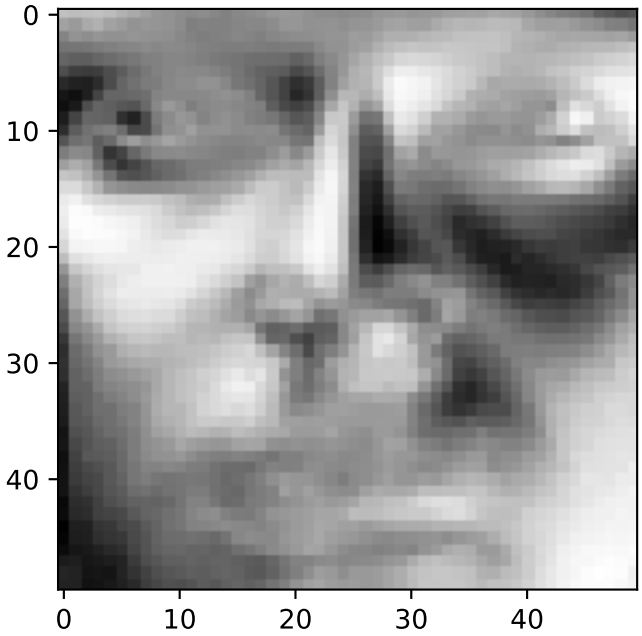


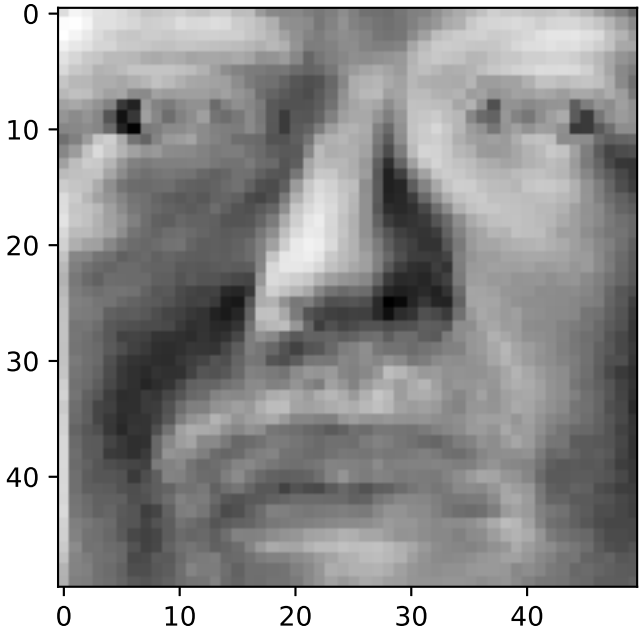


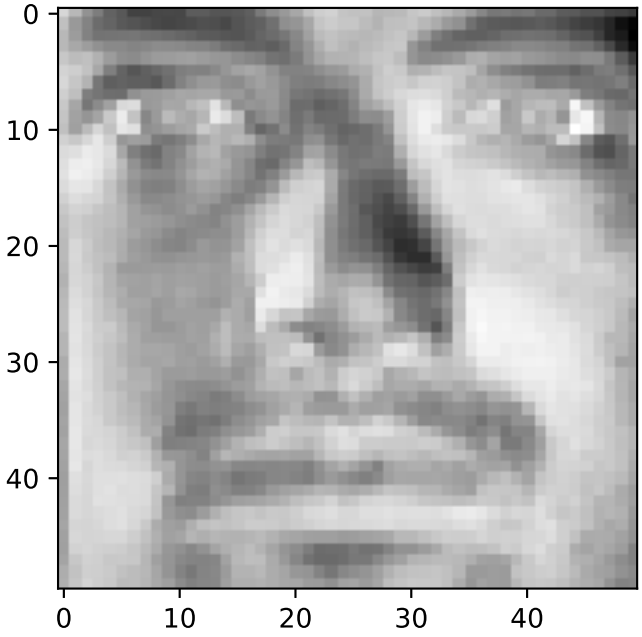

0
10
20
30
40
0
10
20
30
40

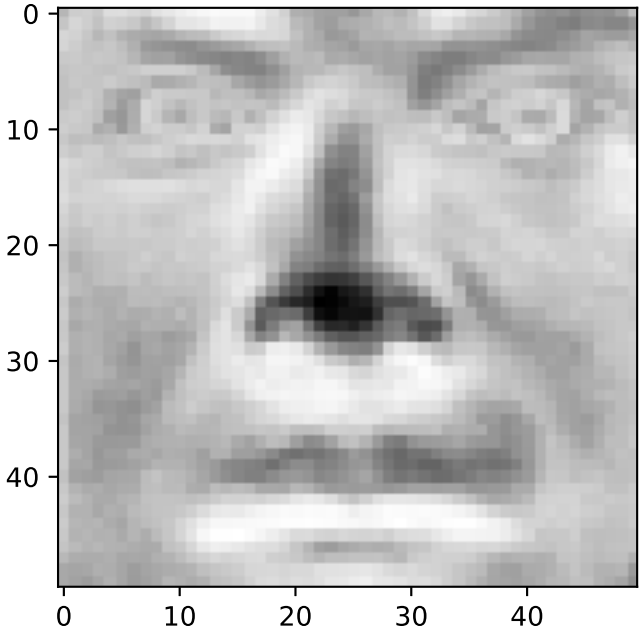

0
10
20
30
40
0
10
20
30
40

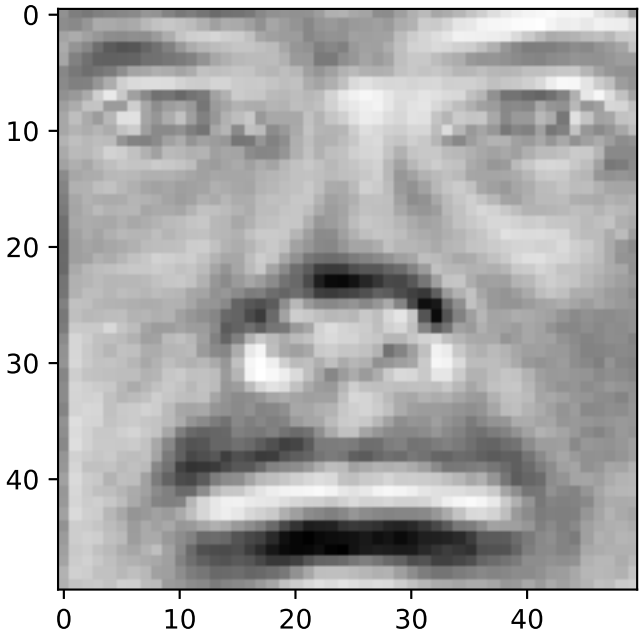

0
10
20
30
40
0
10
20
30
40


0
10
20
30
40
0
10
20
30
40

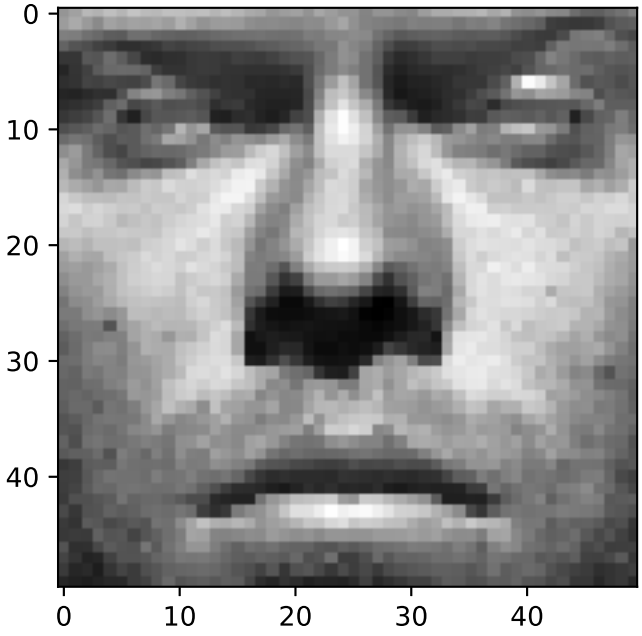

0
10
20
30
40
0
10
20
30
40

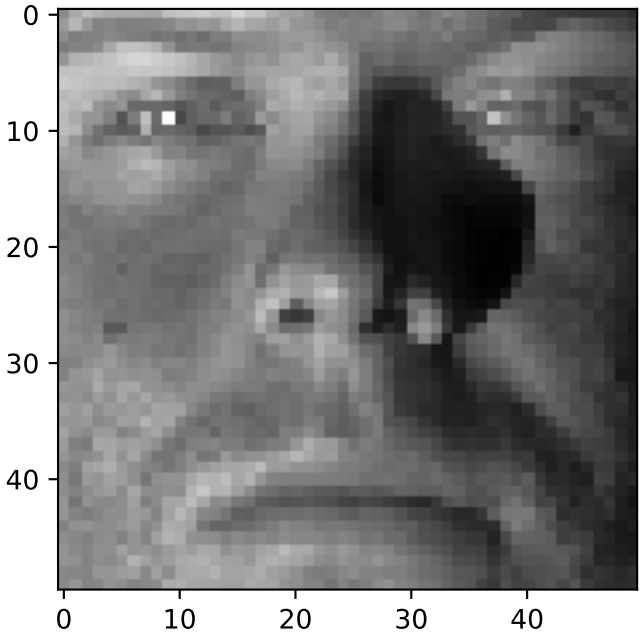

0
10
20
30
40
0
10
20
30
40

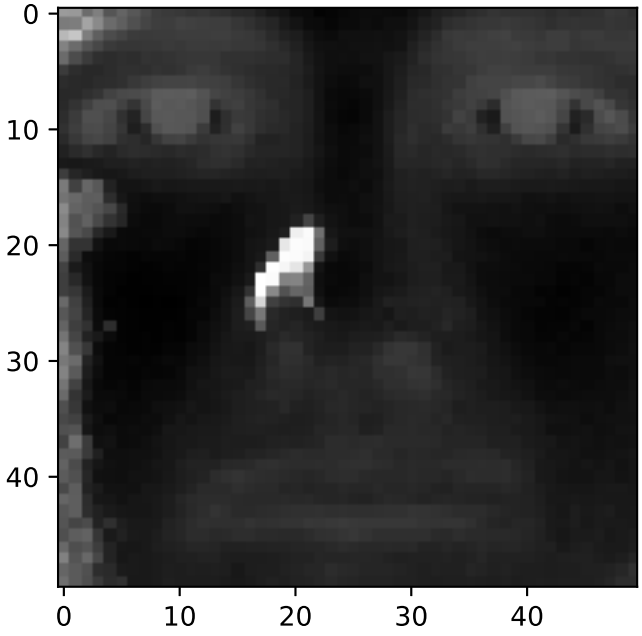

0
10
20
30
40
0
10
20
30
40

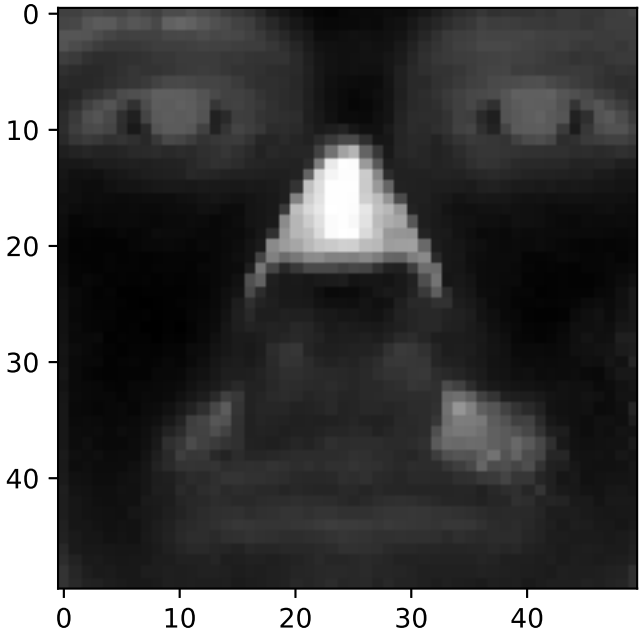

0
10
20
30
40
0
10
20
30
40

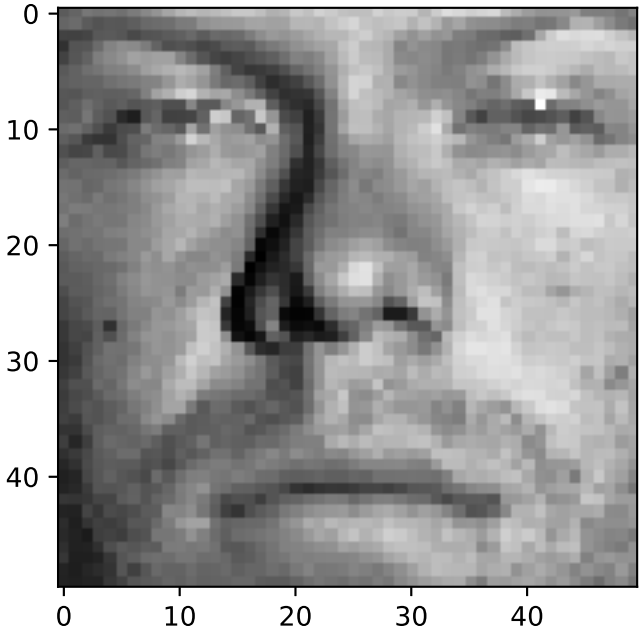
In [11]:
```
for i in range(10):
    plt.imshow(test_mtx[i, :].reshape(50,50), cmap = cm.Greys_r)
    plt.show()
```

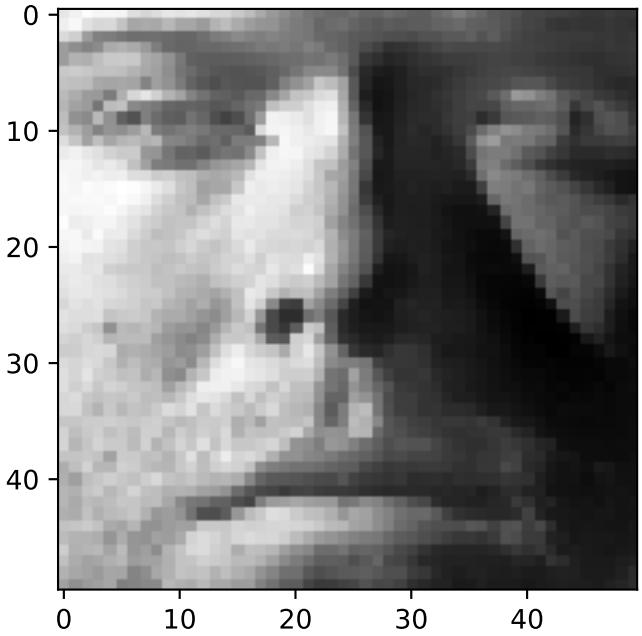

0
10
20
30
40
0
10
20
30
40

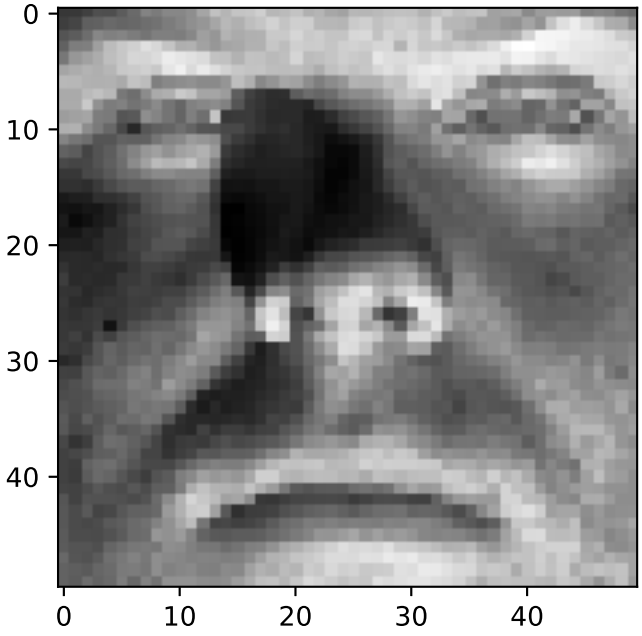

0
10
20
30
40
0
10
20
30
40

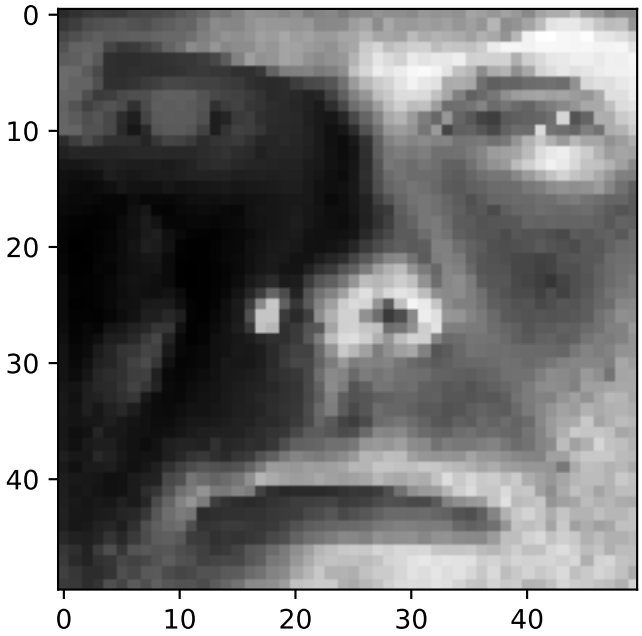

0
10
20
30
40
0
10
20
30
40

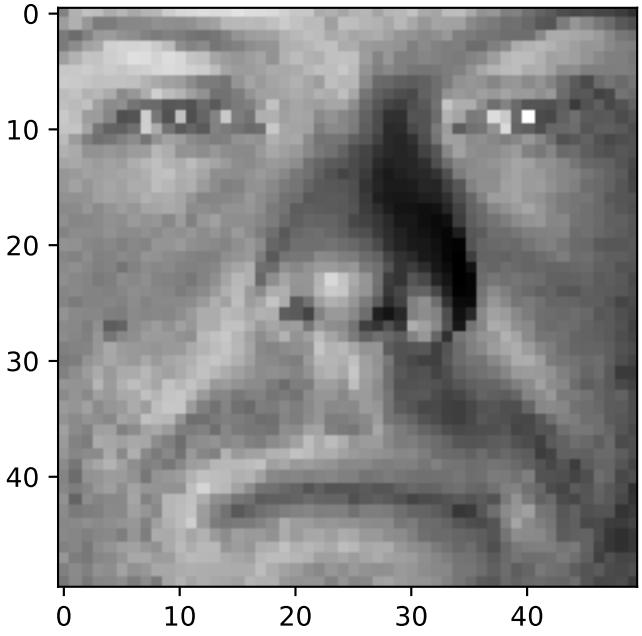

0
10
20
30
40
0
10
20
30
40

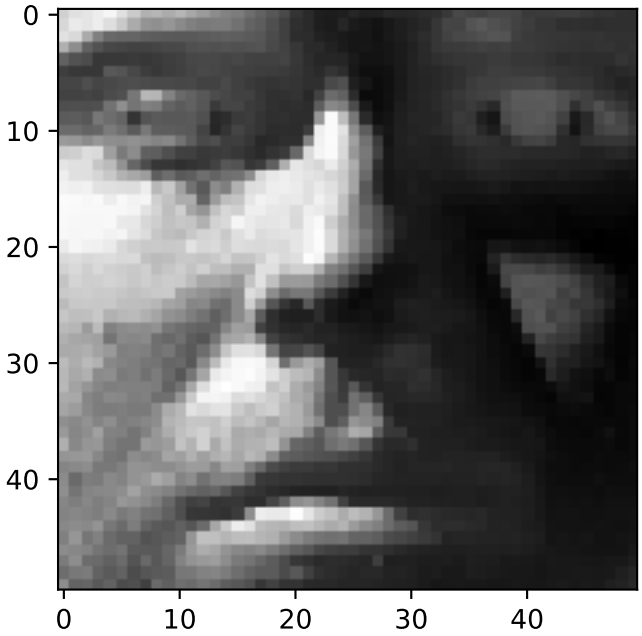

0
10
20
30
40
0
10
20
30
40


0
10
20
30
40
0
10
20
30
40


0
10
20
30
40
0
10
20
30
40


0
10
20
30
40
0
10
20
30
40





(f) Eigenface Feature. The top r eigenfaces VT[: r,:] = {v1, v2,..., vr }Tspan an r -dimensional linear subspace of the original image space called face space, whose origin is the average face μ, and whose axes are the eigenfaces {v1, v2,..., vr }. Therefore, using the top r eigenfaces {v1, v2,..., vr }, we can represent a 2500-dimensional face image z as an r -dimensional fea ture vector f: f = VT[: r,:] z = [v1, v2,..., vr]Tz. Write a function to generate r -dimensional feature matrix F and Ftest for training images X and test images Xtest, respectively (to get F, multiply X to the transpose of first r rows of VT, F should have same number of rows as X and r columns; similarly for Xtest).

In [12]:
```
effs = []
def FeatureMTX():
    for r in range(1, 201, 1):
        VT = V[:r,:].T
        F = np.dot(new_mtx, VT)
        effs.append(F)
    return effs

FeatureMTX()
```

```
Out[12]: 62e-01, -1.37696648e+01,  5.01241488e+01],
                [-1.36371277e+03, -2.93060527e+03, -1.28714465e+03, ...,
                 -1.74803787e+01, -2.16519798e+01, -1.09339222e+01]]),
         array([[ 3071.98503336,   440.55436121,   127.39265075, ...,
                    27.9222525 ,    -3.59133794,     8.21628574],
                [-2197.05744965,   760.60052365,   562.46174352, ...,
                    58.15905942,   -14.31764059,     4.59280369],
                [-2473.33262159,  -163.06716256,  -608.22806399, ...,
                   -17.33902919,   -25.98698221,    12.5154881 ],
                ...,
                [-1841.10091761, -1907.14587095,   226.44323   , ...,
                    12.98925812,    14.27669217,    -8.53233842],
                [ 2586.30249394,  -163.15381029,   151.68973865, ...,
                   -13.76966477,    50.12414884,    27.00015009],
                [-1363.71276667, -2930.60526623, -1287.14464964, ...,
                   -21.65197984,   -10.93392222,   100.97622128]]),
         array([[ 3071.98503336,   440.55436121,   127.39265075, ...,
                    -3.59133794,     8.21628574,   -18.85740953],
                [-2197.05744965,   760.60052365,   562.46174352, ...,
                   -14.31764059,     4.59280369,   -12.19727397],
                [-2473.33262159,  -163.06716256,  -608.22806399, ...,
                   -25.98698221,    12.5154881 ,    20.89093396],
                ...,
                [-1841.10091761, -1907.14587095,   226.44323   , ...,
                    14.27669217,    -8.53233842,   -11.55980756],
                [ 2586.30249394,  -163.15381029,   151.68973865, ...,
                    50.12414884,    27.00015009,   -56.41615204],
                [-1363.71276667, -2930.60526623, -1287.14464964, ...,
                   -10.93392222,   100.97622128,    16.95289823]]),
         array([[ 3071.98503336,   440.55436121,   127.39265075, ...,
                     8.21628574,   -18.85740953,     8.52809353],
                [-2197.05744965,   760.60052365,   562.46174352, ...,
                     4.59280369,   -12.19727397,    18.44524904],
                [-2473.33262159,  -163.06716256,  -608.22806399, ...,
                    12.5154881 ,    20.89093396,   -26.96658268],
                ...,
                [-1841.10091761, -1907.14587095,   226.44323   , ...,
                    -8.53233842,   -11.55980756,    48.52827404],
                [ 2586.30249394,  -163.15381029,   151.68973865, ...,
                    27.00015009,   -56.41615204,    50.30375943],
                [-1363.71276667, -2930.60526623, -1287.14464964, ...,
                   100.97622128,    16.95289823,   -21.77087566]]),
         array([[ 3.07198503e+03,  4.40554361e+02,  1.27392651e+02, ...,
                 -1.88574095e+01,  8.52809353e+00,  3.56022060e+00],
                [-2.19705745e+03,  7.60600524e+02,  5.62461744e+02, ...,
                 -1.21972740e+01,  1.84452490e+01, -3.06520623e+00],
                [-2.47333262e+03, -1.63067163e+02, -6.08228064e+02, ...,
                  2.08909340e+01, -2.69665827e+01, -1.98412405e+01],
                ...,
                [-1.84110092e+03, -1.90714587e+03,  2.26443230e+02, ...,
                 -1.15598076e+01,  4.85282740e+01, -1.60903827e+01],
                [ 2.58630249e+03, -1.63153810e+02,  1.51689739e+02, ...,
                 -5.64161520e+01,  5.03037594e+01, -7.25615474e+00],
                [-1.36371277e+03, -2.93060527e+03, -1.28714465e+03, ...,
                  1.69528982e+01, -2.17708757e+01,  2.58425779e+01]]),
         array([[ 3.07198503e+03,  4.40554361e+02,  1.27392651e+02, ...,
                  8.52809353e+00,  3.56022060e+00, -5.94199701e+01],
```

```
       [-2.19705745e+03,  7.60600524e+02,  5.62461744e+02, ...,
         1.84452490e+01, -3.06520623e+00, -1.00013909e+01],
       [-2.47333262e+03, -1.63067163e+02, -6.08228064e+02, ...,
        -2.69665827e+01, -1.98412405e+01,  1.69812618e+01],
       ...,
       [-1.84110092e+03, -1.90714587e+03,  2.26443230e+02, ...,
         4.85282740e+01, -1.60903827e+01, -3.58605223e-01],
       [ 2.58630249e+03, -1.63153810e+02,  1.51689739e+02, ...,
         5.03037594e+01, -7.25615474e+00, -1.37045344e+01],
       [-1.36371277e+03, -2.93060527e+03, -1.28714465e+03, ...,
        -2.17708757e+01,  2.58425779e+01,  4.53710442e+01]]),
array([[ 3.07198503e+03,  4.40554361e+02,  1.27392651e+02, ...,
         3.56022060e+00, -5.94199701e+01,  4.44060124e+01],
       [-2.19705745e+03,  7.60600524e+02,  5.62461744e+02, ...,
        -3.06520623e+00, -1.00013909e+01,  8.84050773e+00],
       [-2.47333262e+03, -1.63067163e+02, -6.08228064e+02, ...,
        -1.98412405e+01,  1.69812618e+01,  2.39720472e+01],
       ...,
       [-1.84110092e+03, -1.90714587e+03,  2.26443230e+02, ...,
        -1.60903827e+01, -3.58605223e-01,  1.02869770e+01],
       [ 2.58630249e+03, -1.63153810e+02,  1.51689739e+02, ...,
        -7.25615474e+00, -1.37045344e+01, -2.27039300e+01],
       [-1.36371277e+03, -2.93060527e+03, -1.28714465e+03, ...,
         2.58425779e+01,  4.53710442e+01,  6.44548982e+01]]),
array([[ 3.07198503e+03,  4.40554361e+02,  1.27392651e+02, ...,
        -5.94199701e+01,  4.44060124e+01, -6.22505778e+00],
       [-2.19705745e+03,  7.60600524e+02,  5.62461744e+02, ...,
        -1.00013909e+01,  8.84050773e+00,  8.57256013e+00],
       [-2.47333262e+03, -1.63067163e+02, -6.08228064e+02, ...,
         1.69812618e+01,  2.39720472e+01, -5.14412453e+00],
       ...,
       [-1.84110092e+03, -1.90714587e+03,  2.26443230e+02, ...,
        -3.58605223e-01,  1.02869770e+01,  7.56341731e-01],
       [ 2.58630249e+03, -1.63153810e+02,  1.51689739e+02, ...,
        -1.37045344e+01, -2.27039300e+01, -9.32318963e+00],
       [-1.36371277e+03, -2.93060527e+03, -1.28714465e+03, ...,
         4.53710442e+01,  6.44548982e+01,  8.91588981e+00]]),
array([[ 3.07198503e+03,  4.40554361e+02,  1.27392651e+02, ...,
         4.44060124e+01, -6.22505778e+00, -1.25230363e+01],
       [-2.19705745e+03,  7.60600524e+02,  5.62461744e+02, ...,
         8.84050773e+00,  8.57256013e+00,  1.66064471e+00],
       [-2.47333262e+03, -1.63067163e+02, -6.08228064e+02, ...,
         2.39720472e+01, -5.14412453e+00, -1.02697815e+01],
       ...,
       [-1.84110092e+03, -1.90714587e+03,  2.26443230e+02, ...,
         1.02869770e+01,  7.56341731e-01, -1.61090109e+01],
       [ 2.58630249e+03, -1.63153810e+02,  1.51689739e+02, ...,
        -2.27039300e+01, -9.32318963e+00,  2.03452944e+01],
       [-1.36371277e+03, -2.93060527e+03, -1.28714465e+03, ...,
         6.44548982e+01,  8.91588981e+00,  2.08140110e+00]]),
array([[ 3.07198503e+03,  4.40554361e+02,  1.27392651e+02, ...,
        -6.22505778e+00, -1.25230363e+01, -2.95581201e+01],
       [-2.19705745e+03,  7.60600524e+02,  5.62461744e+02, ...,
         8.57256013e+00,  1.66064471e+00,  6.22280948e+00],
       [-2.47333262e+03, -1.63067163e+02, -6.08228064e+02, ...,
        -5.14412453e+00, -1.02697815e+01,  1.99272910e+01],
       ...,
```

```
       [-1.84110092e+03, -1.90714587e+03,  2.26443230e+02, ...,
         7.56341731e-01, -1.61090109e+01, -4.30980260e+01],
       [ 2.58630249e+03, -1.63153810e+02,  1.51689739e+02, ...,
        -9.32318963e+00,  2.03452944e+01,  6.23875211e+01],
       [-1.36371277e+03, -2.93060527e+03, -1.28714465e+03, ...,
         8.91588981e+00,  2.08140110e+00, -1.48150757e+01]]),
array([[ 3.07198503e+03,  4.40554361e+02,  1.27392651e+02, ...,
        -1.25230363e+01, -2.95581201e+01,  2.78116170e+01],
       [-2.19705745e+03,  7.60600524e+02,  5.62461744e+02, ...,
         1.66064471e+00,  6.22280948e+00, -1.70005435e+01],
       [-2.47333262e+03, -1.63067163e+02, -6.08228064e+02, ...,
        -1.02697815e+01,  1.99272910e+01,  9.58950709e+00],
       ...,
       [-1.84110092e+03, -1.90714587e+03,  2.26443230e+02, ...,
        -1.61090109e+01, -4.30980260e+01, -2.58305254e+01],
       [ 2.58630249e+03, -1.63153810e+02,  1.51689739e+02, ...,
         2.03452944e+01,  6.23875211e+01,  5.58235933e+01],
       [-1.36371277e+03, -2.93060527e+03, -1.28714465e+03, ...,
         2.08140110e+00, -1.48150757e+01,  7.35634427e+01]]),
array([[ 3071.98503336,   440.55436121,   127.39265075, ...,
          -29.55812009,    27.81161698,   -18.6195898 ],
       [-2197.05744965,   760.60052365,   562.46174352, ...,
            6.22280948,   -17.0005435 ,   -20.47506887],
       [-2473.33262159,  -163.06716256,  -608.22806399, ...,
           19.92729097,     9.58950709,     9.94677901],
       ...,
       [-1841.10091761, -1907.14587095,   226.44323   , ...,
          -43.09802596,   -25.83052536,    27.32096804],
       [ 2586.30249394,  -163.15381029,   151.68973865, ...,
           62.3875211 ,    55.82359334,   -15.84151713],
       [-1363.71276667, -2930.60526623, -1287.14464964, ...,
          -14.81507571,    73.56344266,    -4.2401282 ]]),
array([[ 3071.98503336,   440.55436121,   127.39265075, ...,
           27.81161698,   -18.6195898 ,   -12.58943999],
       [-2197.05744965,   760.60052365,   562.46174352, ...,
          -17.0005435 ,   -20.47506887,    24.98418955],
       [-2473.33262159,  -163.06716256,  -608.22806399, ...,
            9.58950709,     9.94677901,     6.2611374 ],
       ...,
       [-1841.10091761, -1907.14587095,   226.44323   , ...,
          -25.83052536,    27.32096804,   -17.26756781],
       [ 2586.30249394,  -163.15381029,   151.68973865, ...,
           55.82359334,   -15.84151713,    43.90844743],
       [-1363.71276667, -2930.60526623, -1287.14464964, ...,
           73.56344266,    -4.2401282 ,     5.50547467]]),
array([[ 3071.98503336,   440.55436121,   127.39265075, ...,
          -18.6195898 ,   -12.58943999,    24.00966508],
       [-2197.05744965,   760.60052365,   562.46174352, ...,
          -20.47506887,    24.98418955,   -31.53146018],
       [-2473.33262159,  -163.06716256,  -608.22806399, ...,
            9.94677901,     6.2611374 ,   -40.16539   ],
       ...,
       [-1841.10091761, -1907.14587095,   226.44323   , ...,
           27.32096804,   -17.26756781,   -23.78416849],
       [ 2586.30249394,  -163.15381029,   151.68973865, ...,
          -15.84151713,    43.90844743,    58.35025485],
       [-1363.71276667, -2930.60526623, -1287.14464964, ...,
```

```
          -4.2401282 ,     5.50547467,    16.64796433]]),
       array([[ 3.07198503e+03,  4.40554361e+02,  1.27392651e+02, ...,
               -1.25894400e+01,  2.40096651e+01, -1.40521126e+01],
              [-2.19705745e+03,  7.60600524e+02,  5.62461744e+02, ...,
                2.49841896e+01, -3.15314602e+01,  2.86929352e+00],
              [-2.47333262e+03, -1.63067163e+02, -6.08228064e+02, ...,
                6.26113740e+00, -4.01653900e+01, -1.92679581e+01],
              ...,
              [-1.84110092e+03, -1.90714587e+03,  2.26443230e+02, ...,
               -1.72675678e+01, -2.37841685e+01, -2.70986862e+01],
              [ 2.58630249e+03, -1.63153810e+02,  1.51689739e+02, ...,
                4.39084474e+01,  5.83502548e+01, -2.97545781e+01],
              [-1.36371277e+03, -2.93060527e+03, -1.28714465e+03, ...,
                5.50547467e+00,  1.66479643e+01,  3.21323914e+01]]),
       array([[ 3.07198503e+03,  4.40554361e+02,  1.27392651e+02, ...,
                2.40096651e+01, -1.40521126e+01,  3.36176963e+01],
              [-2.19705745e+03,  7.60600524e+02,  5.62461744e+02, ...,
               -3.15314602e+01,  2.86929352e+00,  1.67213364e+00],
              [-2.47333262e+03, -1.63067163e+02, -6.08228064e+02, ...,
               -4.01653900e+01, -1.92679581e+01,  8.73205619e+00],
              ...,
              [-1.84110092e+03, -1.90714587e+03,  2.26443230e+02, ...,
               -2.37841685e+01, -2.70986862e+01, -1.57007652e+00],
              [ 2.58630249e+03, -1.63153810e+02,  1.51689739e+02, ...,
                5.83502548e+01, -2.97545781e+01, -5.33368636e+00],
              [-1.36371277e+03, -2.93060527e+03, -1.28714465e+03, ...,
                1.66479643e+01,  3.21323914e+01,  3.57178955e+00]]),
       array([[ 3.07198503e+03,  4.40554361e+02,  1.27392651e+02, ...,
               -1.40521126e+01,  3.36176963e+01,  1.19236237e+00],
              [-2.19705745e+03,  7.60600524e+02,  5.62461744e+02, ...,
                2.86929352e+00,  1.67213364e+00, -4.37330598e+00],
              [-2.47333262e+03, -1.63067163e+02, -6.08228064e+02, ...,
               -1.92679581e+01,  8.73205619e+00,  1.28341296e+01],
              ...,
              [-1.84110092e+03, -1.90714587e+03,  2.26443230e+02, ...,
               -2.70986862e+01, -1.57007652e+00, -1.27337385e+01],
              [ 2.58630249e+03, -1.63153810e+02,  1.51689739e+02, ...,
               -2.97545781e+01, -5.33368636e+00, -1.78324199e+01],
              [-1.36371277e+03, -2.93060527e+03, -1.28714465e+03, ...,
                3.21323914e+01,  3.57178955e+00, -1.33025856e+01]]),
       array([[ 3.07198503e+03,  4.40554361e+02,  1.27392651e+02, ...,
                3.36176963e+01,  1.19236237e+00,  1.35681866e+01],
              [-2.19705745e+03,  7.60600524e+02,  5.62461744e+02, ...,
                1.67213364e+00, -4.37330598e+00, -1.07401808e+01],
              [-2.47333262e+03, -1.63067163e+02, -6.08228064e+02, ...,
                8.73205619e+00,  1.28341296e+01,  5.29597269e+00],
              ...,
              [-1.84110092e+03, -1.90714587e+03,  2.26443230e+02, ...,
               -1.57007652e+00, -1.27337385e+01, -4.37606111e+00],
              [ 2.58630249e+03, -1.63153810e+02,  1.51689739e+02, ...,
               -5.33368636e+00, -1.78324199e+01, -3.89563847e+01],
              [-1.36371277e+03, -2.93060527e+03, -1.28714465e+03, ...,
                3.57178955e+00, -1.33025856e+01, -2.63743713e+01]]),
       array([[ 3.07198503e+03,  4.40554361e+02,  1.27392651e+02, ...,
                1.19236237e+00,  1.35681866e+01, -4.83572594e+01],
              [-2.19705745e+03,  7.60600524e+02,  5.62461744e+02, ...,
               -4.37330598e+00, -1.07401808e+01,  1.64917782e+01],
```

```
       [-2.47333262e+03, -1.63067163e+02, -6.08228064e+02, ...,
         1.28341296e+01,  5.29597269e+00,  1.52361810e+01],
       ...,
       [-1.84110092e+03, -1.90714587e+03,  2.26443230e+02, ...,
        -1.27337385e+01, -4.37606111e+00, -1.59208591e+00],
       [ 2.58630249e+03, -1.63153810e+02,  1.51689739e+02, ...,
        -1.78324199e+01, -3.89563847e+01,  2.69018735e-01],
       [-1.36371277e+03, -2.93060527e+03, -1.28714465e+03, ...,
        -1.33025856e+01, -2.63743713e+01, -1.23187433e+01]]),
array([[ 3.07198503e+03,  4.40554361e+02,  1.27392651e+02, ...,
         1.35681866e+01, -4.83572594e+01,  2.47949818e+01],
       [-2.19705745e+03,  7.60600524e+02,  5.62461744e+02, ...,
        -1.07401808e+01,  1.64917782e+01,  2.11870220e+01],
       [-2.47333262e+03, -1.63067163e+02, -6.08228064e+02, ...,
         5.29597269e+00,  1.52361810e+01, -1.55131254e+00],
       ...,
       [-1.84110092e+03, -1.90714587e+03,  2.26443230e+02, ...,
        -4.37606111e+00, -1.59208591e+00, -1.14155834e+01],
       [ 2.58630249e+03, -1.63153810e+02,  1.51689739e+02, ...,
        -3.89563847e+01,  2.69018735e-01, -9.22733590e+00],
       [-1.36371277e+03, -2.93060527e+03, -1.28714465e+03, ...,
        -2.63743713e+01, -1.23187433e+01, -3.60956519e+01]]),
array([[ 3.07198503e+03,  4.40554361e+02,  1.27392651e+02, ...,
        -4.83572594e+01,  2.47949818e+01,  4.50601098e+00],
       [-2.19705745e+03,  7.60600524e+02,  5.62461744e+02, ...,
         1.64917782e+01,  2.11870220e+01, -4.13576882e-01],
       [-2.47333262e+03, -1.63067163e+02, -6.08228064e+02, ...,
         1.52361810e+01, -1.55131254e+00,  1.73623530e+01],
       ...,
       [-1.84110092e+03, -1.90714587e+03,  2.26443230e+02, ...,
        -1.59208591e+00, -1.14155834e+01,  5.15453943e+00],
       [ 2.58630249e+03, -1.63153810e+02,  1.51689739e+02, ...,
         2.69018735e-01, -9.22733590e+00, -2.29284446e+01],
       [-1.36371277e+03, -2.93060527e+03, -1.28714465e+03, ...,
        -1.23187433e+01, -3.60956519e+01, -2.73145859e+01]]),
array([[ 3.07198503e+03,  4.40554361e+02,  1.27392651e+02, ...,
         2.47949818e+01,  4.50601098e+00, -7.31018845e-01],
       [-2.19705745e+03,  7.60600524e+02,  5.62461744e+02, ...,
         2.11870220e+01, -4.13576882e-01, -1.00772655e+01],
       [-2.47333262e+03, -1.63067163e+02, -6.08228064e+02, ...,
        -1.55131254e+00,  1.73623530e+01, -1.64667425e+01],
       ...,
       [-1.84110092e+03, -1.90714587e+03,  2.26443230e+02, ...,
        -1.14155834e+01,  5.15453943e+00,  2.62871917e+01],
       [ 2.58630249e+03, -1.63153810e+02,  1.51689739e+02, ...,
        -9.22733590e+00, -2.29284446e+01,  4.79632745e+01],
       [-1.36371277e+03, -2.93060527e+03, -1.28714465e+03, ...,
        -3.60956519e+01, -2.73145859e+01,  4.10586100e+01]]),
array([[ 3.07198503e+03,  4.40554361e+02,  1.27392651e+02, ...,
         4.50601098e+00, -7.31018845e-01, -3.98187865e+01],
       [-2.19705745e+03,  7.60600524e+02,  5.62461744e+02, ...,
        -4.13576882e-01, -1.00772655e+01, -1.48787771e+01],
       [-2.47333262e+03, -1.63067163e+02, -6.08228064e+02, ...,
         1.73623530e+01, -1.64667425e+01, -6.69379451e+00],
       ...,
       [-1.84110092e+03, -1.90714587e+03,  2.26443230e+02, ...,
         5.15453943e+00,  2.62871917e+01,  7.04315765e+00],
```

```
       [ 2.58630249e+03, -1.63153810e+02,  1.51689739e+02, ...,
        -2.29284446e+01,  4.79632745e+01,  1.21710178e+01],
       [-1.36371277e+03, -2.93060527e+03, -1.28714465e+03, ...,
        -2.73145859e+01,  4.10586100e+01, -2.25835051e+01]]),
array([[ 3.07198503e+03,  4.40554361e+02,  1.27392651e+02, ...,
        -7.31018845e-01, -3.98187865e+01,  2.44632788e+00],
       [-2.19705745e+03,  7.60600524e+02,  5.62461744e+02, ...,
        -1.00772655e+01, -1.48787771e+01, -9.79116563e+00],
       [-2.47333262e+03, -1.63067163e+02, -6.08228064e+02, ...,
        -1.64667425e+01, -6.69379451e+00,  1.29139019e+01],
       ...,
       [-1.84110092e+03, -1.90714587e+03,  2.26443230e+02, ...,
         2.62871917e+01,  7.04315765e+00,  8.14062242e+00],
       [ 2.58630249e+03, -1.63153810e+02,  1.51689739e+02, ...,
         4.79632745e+01,  1.21710178e+01, -1.43978500e+01],
       [-1.36371277e+03, -2.93060527e+03, -1.28714465e+03, ...,
         4.10586100e+01, -2.25835051e+01, -2.39876994e+01]]),
array([[ 3.07198503e+03,  4.40554361e+02,  1.27392651e+02, ...,
        -3.98187865e+01,  2.44632788e+00,  2.46716383e+01],
       [-2.19705745e+03,  7.60600524e+02,  5.62461744e+02, ...,
        -1.48787771e+01, -9.79116563e+00,  1.65083698e+00],
       [-2.47333262e+03, -1.63067163e+02, -6.08228064e+02, ...,
        -6.69379451e+00,  1.29139019e+01, -3.14884100e+01],
       ...,
       [-1.84110092e+03, -1.90714587e+03,  2.26443230e+02, ...,
         7.04315765e+00,  8.14062242e+00,  1.69455973e+01],
       [ 2.58630249e+03, -1.63153810e+02,  1.51689739e+02, ...,
         1.21710178e+01, -1.43978500e+01, -2.12742512e+01],
       [-1.36371277e+03, -2.93060527e+03, -1.28714465e+03, ...,
        -2.25835051e+01, -2.39876994e+01,  5.26665165e+00]]),
array([[ 3.07198503e+03,  4.40554361e+02,  1.27392651e+02, ...,
         2.44632788e+00,  2.46716383e+01, -8.94223103e+00],
       [-2.19705745e+03,  7.60600524e+02,  5.62461744e+02, ...,
        -9.79116563e+00,  1.65083698e+00, -3.08676027e+00],
       [-2.47333262e+03, -1.63067163e+02, -6.08228064e+02, ...,
         1.29139019e+01, -3.14884100e+01, -2.58060999e+01],
       ...,
       [-1.84110092e+03, -1.90714587e+03,  2.26443230e+02, ...,
         8.14062242e+00,  1.69455973e+01,  2.45259981e+01],
       [ 2.58630249e+03, -1.63153810e+02,  1.51689739e+02, ...,
        -1.43978500e+01, -2.12742512e+01, -4.85636620e+01],
       [-1.36371277e+03, -2.93060527e+03, -1.28714465e+03, ...,
        -2.39876994e+01,  5.26665165e+00, -5.05927496e-01]]),
array([[ 3.07198503e+03,  4.40554361e+02,  1.27392651e+02, ...,
         2.46716383e+01, -8.94223103e+00, -1.06298103e+01],
       [-2.19705745e+03,  7.60600524e+02,  5.62461744e+02, ...,
         1.65083698e+00, -3.08676027e+00, -2.96110608e+00],
       [-2.47333262e+03, -1.63067163e+02, -6.08228064e+02, ...,
        -3.14884100e+01, -2.58060999e+01,  5.65605281e-01],
       ...,
       [-1.84110092e+03, -1.90714587e+03,  2.26443230e+02, ...,
         1.69455973e+01,  2.45259981e+01,  1.89214333e+01],
       [ 2.58630249e+03, -1.63153810e+02,  1.51689739e+02, ...,
        -2.12742512e+01, -4.85636620e+01, -2.16683115e+01],
       [-1.36371277e+03, -2.93060527e+03, -1.28714465e+03, ...,
         5.26665165e+00, -5.05927496e-01, -3.72699047e+01]])]
```

In [14]:
```
ftest = []
def FeatureTestMTX():
    for r in range(1, 201, 1):
        VT = V[:r,:].T
        F_test = np.dot(test_mtx,VT)
        ftest.append(F_test)
    return ftest

FeatureTestMTX()
```

Out[14]:
```
-1.97547669e+03, -5.30835552e+01, -1.56924329e+03, ...,
        -5.34438399e+01,  3.46174547e+01,  1.98848124e+01],
       [-6.99288660e+02, -1.78444521e+03,  8.83082635e+02, ...,
         7.60044605e+00, -3.16169853e+01, -5.63915986e+01],
       [ 2.89306329e+03, -3.42858431e+01,  1.17466461e+02, ...,
        -1.30132428e+01,  1.85456235e+01, -1.26722605e+01],
       ...,
       [-1.34300326e+02,  3.55148415e+03, -5.34447049e+02, ...,
         1.80246758e+01, -1.01672366e+00,  5.92753948e+01],
       [ 3.73568104e+02, -2.78559446e+03,  1.19699164e+03, ...,
         5.52528532e+01,  3.31970879e+01,  4.98024578e+01],
       [-3.57633319e+03,  1.29170341e+03,  4.64224199e+02, ...,
         2.39370761e+01, -1.26433883e+01,  7.58787734e+00]]),
array([[-1.97547669e+03, -5.30835552e+01, -1.56924329e+03, ...,
         3.46174547e+01,  1.98848124e+01,  5.45390583e+01],
       [-6.99288660e+02, -1.78444521e+03,  8.83082635e+02, ...,
        -3.16169853e+01, -5.63915986e+01, -1.18884939e+01],
       [ 2.89306329e+03, -3.42858431e+01,  1.17466461e+02, ...,
         1.85456235e+01, -1.26722605e+01,  1.28936321e+01],
       ...,
       [-1.34300326e+02,  3.55148415e+03, -5.34447049e+02, ...,
        -1.01672366e+00,  5.92753948e+01, -2.57590694e+01],
       [ 3.73568104e+02, -2.78559446e+03,  1.19699164e+03, ...,
         3.31970879e+01,  4.98024578e+01,  4.54835509e+01],
       [-3.57633319e+03,  1.29170341e+03,  4.64224199e+02, ...,
        -1.26433883e+01,  7.58787734e+00, -1.02779450e+01]]),
array([[-1975.47669007,   -53.08355521, -1569.24328777, ...,
          19.88481238,    54.53905827,     8.98920271],
       [ -699.28866044, -1784.44521438,   883.08263538, ...,
          -56.39159861,   -11.88849394,   -12.25784131],
       [ 2893.06328875,   -34.28584308,   117.46646102, ...,
          -12.67226048,    12.89363213,   -62.60910811],
       ...,
       [ -134.30032587,  3551.48415313,  -534.44704866, ...,
           59.27539481,   -25.75906944,     5.56509999],
       [  373.56810371, -2785.59446256,  1196.99163523, ...,
           49.8024578 ,    45.48355093,   -28.56604387],
       [-3576.33318671,  1291.70341032,   464.22419948, ...,
            7.58787734,   -10.27794496,   -10.29876143]]),
array([[-1.97547669e+03, -5.30835552e+01, -1.56924329e+03, ...,
         5.45390583e+01,  8.98920271e+00, -3.59192215e+01],
       [-6.99288660e+02, -1.78444521e+03,  8.83082635e+02, ...,
        -1.18884939e+01, -1.22578413e+01,  3.78867659e+01],
       [ 2.89306329e+03, -3.42858431e+01,  1.17466461e+02, ...,
         1.28936321e+01, -6.26091081e+01, -3.69945825e+00],
       ...,
       [-1.34300326e+02,  3.55148415e+03, -5.34447049e+02, ...,
        -2.57590694e+01,  5.56509999e+00,  2.11736557e+01],
       [ 3.73568104e+02, -2.78559446e+03,  1.19699164e+03, ...,
         4.54835509e+01, -2.85660439e+01,  1.81478606e+01],
       [-3.57633319e+03,  1.29170341e+03,  4.64224199e+02, ...,
        -1.02779450e+01, -1.02987614e+01,  2.72862280e+00]]),
array([[-1.97547669e+03, -5.30835552e+01, -1.56924329e+03, ...,
         8.98920271e+00, -3.59192215e+01,  2.30724850e+00],
       [-6.99288660e+02, -1.78444521e+03,  8.83082635e+02, ...,
        -1.22578413e+01,  3.78867659e+01, -2.11404746e+01],
       [ 2.89306329e+03, -3.42858431e+01,  1.17466461e+02, ...,
```

```
         -6.26091081e+01, -3.69945825e+00, -2.23053958e+01],
        ...,
        [-1.34300326e+02,  3.55148415e+03, -5.34447049e+02, ...,
          5.56509999e+00,  2.11736557e+01, -5.36614423e+01],
        [ 3.73568104e+02, -2.78559446e+03,  1.19699164e+03, ...,
         -2.85660439e+01,  1.81478606e+01,  4.62627910e+01],
        [-3.57633319e+03,  1.29170341e+03,  4.64224199e+02, ...,
         -1.02987614e+01,  2.72862280e+00, -1.72147793e+00]]),
 array([[-1.97547669e+03, -5.30835552e+01, -1.56924329e+03, ...,
         -3.59192215e+01,  2.30724850e+00, -7.68670969e+00],
        [-6.99288660e+02, -1.78444521e+03,  8.83082635e+02, ...,
          3.78867659e+01, -2.11404746e+01, -1.82946130e+01],
        [ 2.89306329e+03, -3.42858431e+01,  1.17466461e+02, ...,
         -3.69945825e+00, -2.23053958e+01, -2.49922350e+01],
        ...,
        [-1.34300326e+02,  3.55148415e+03, -5.34447049e+02, ...,
          2.11736557e+01, -5.36614423e+01,  2.26079055e+01],
        [ 3.73568104e+02, -2.78559446e+03,  1.19699164e+03, ...,
          1.81478606e+01,  4.62627910e+01,  7.01511182e+00],
        [-3.57633319e+03,  1.29170341e+03,  4.64224199e+02, ...,
          2.72862280e+00, -1.72147793e+00,  1.79922048e+01]]),
 array([[-1.97547669e+03, -5.30835552e+01, -1.56924329e+03, ...,
          2.30724850e+00, -7.68670969e+00,  2.75288627e+01],
        [-6.99288660e+02, -1.78444521e+03,  8.83082635e+02, ...,
         -2.11404746e+01, -1.82946130e+01, -2.10357625e+01],
        [ 2.89306329e+03, -3.42858431e+01,  1.17466461e+02, ...,
         -2.23053958e+01, -2.49922350e+01,  1.13062780e+01],
        ...,
        [-1.34300326e+02,  3.55148415e+03, -5.34447049e+02, ...,
         -5.36614423e+01,  2.26079055e+01,  2.44463385e+00],
        [ 3.73568104e+02, -2.78559446e+03,  1.19699164e+03, ...,
          4.62627910e+01,  7.01511182e+00, -3.27103852e+01],
        [-3.57633319e+03,  1.29170341e+03,  4.64224199e+02, ...,
         -1.72147793e+00,  1.79922048e+01,  2.28783422e+01]]),
 array([[-1.97547669e+03, -5.30835552e+01, -1.56924329e+03, ...,
         -7.68670969e+00,  2.75288627e+01,  1.31911027e+01],
        [-6.99288660e+02, -1.78444521e+03,  8.83082635e+02, ...,
         -1.82946130e+01, -2.10357625e+01,  3.44682818e+01],
        [ 2.89306329e+03, -3.42858431e+01,  1.17466461e+02, ...,
         -2.49922350e+01,  1.13062780e+01,  2.13576017e+01],
        ...,
        [-1.34300326e+02,  3.55148415e+03, -5.34447049e+02, ...,
          2.26079055e+01,  2.44463385e+00, -1.19743576e+01],
        [ 3.73568104e+02, -2.78559446e+03,  1.19699164e+03, ...,
          7.01511182e+00, -3.27103852e+01,  1.98091001e+01],
        [-3.57633319e+03,  1.29170341e+03,  4.64224199e+02, ...,
          1.79922048e+01,  2.28783422e+01, -2.37464359e+01]]),
 array([[-1.97547669e+03, -5.30835552e+01, -1.56924329e+03, ...,
          2.75288627e+01,  1.31911027e+01, -3.05945453e+00],
        [-6.99288660e+02, -1.78444521e+03,  8.83082635e+02, ...,
         -2.10357625e+01,  3.44682818e+01,  1.90418533e+01],
        [ 2.89306329e+03, -3.42858431e+01,  1.17466461e+02, ...,
          1.13062780e+01,  2.13576017e+01, -2.34890150e+00],
        ...,
        [-1.34300326e+02,  3.55148415e+03, -5.34447049e+02, ...,
          2.44463385e+00, -1.19743576e+01,  3.68969688e+01],
        [ 3.73568104e+02, -2.78559446e+03,  1.19699164e+03, ...,
```

```
         -3.27103852e+01,  1.98091001e+01, -2.50707913e+01],
        [-3.57633319e+03,  1.29170341e+03,  4.64224199e+02, ...,
         2.28783422e+01, -2.37464359e+01,  3.86428236e-01]]),
 array([[-1.97547669e+03, -5.30835552e+01, -1.56924329e+03, ...,
         1.31911027e+01, -3.05945453e+00,  2.63607813e+01],
        [-6.99288660e+02, -1.78444521e+03,  8.83082635e+02, ...,
         3.44682818e+01,  1.90418533e+01, -1.56238866e+01],
        [ 2.89306329e+03, -3.42858431e+01,  1.17466461e+02, ...,
         2.13576017e+01, -2.34890150e+00,  6.85125859e+00],
        ...,
        [-1.34300326e+02,  3.55148415e+03, -5.34447049e+02, ...,
         -1.19743576e+01,  3.68969688e+01,  2.80035928e+01],
        [ 3.73568104e+02, -2.78559446e+03,  1.19699164e+03, ...,
         1.98091001e+01, -2.50707913e+01,  4.53501689e+01],
        [-3.57633319e+03,  1.29170341e+03,  4.64224199e+02, ...,
         -2.37464359e+01,  3.86428236e-01, -1.45355634e+00]]),
 array([[-1.97547669e+03, -5.30835552e+01, -1.56924329e+03, ...,
         -3.05945453e+00,  2.63607813e+01,  3.18810979e+01],
        [-6.99288660e+02, -1.78444521e+03,  8.83082635e+02, ...,
         1.90418533e+01, -1.56238866e+01,  9.63825071e+00],
        [ 2.89306329e+03, -3.42858431e+01,  1.17466461e+02, ...,
         -2.34890150e+00,  6.85125859e+00,  4.80304373e+00],
        ...,
        [-1.34300326e+02,  3.55148415e+03, -5.34447049e+02, ...,
         3.68969688e+01,  2.80035928e+01,  1.12310033e+00],
        [ 3.73568104e+02, -2.78559446e+03,  1.19699164e+03, ...,
         -2.50707913e+01,  4.53501689e+01,  1.86077906e+01],
        [-3.57633319e+03,  1.29170341e+03,  4.64224199e+02, ...,
         3.86428236e-01, -1.45355634e+00, -1.47953585e+01]]),
 array([[-1.97547669e+03, -5.30835552e+01, -1.56924329e+03, ...,
         2.63607813e+01,  3.18810979e+01,  4.57017853e+01],
        [-6.99288660e+02, -1.78444521e+03,  8.83082635e+02, ...,
         -1.56238866e+01,  9.63825071e+00,  1.66423792e+01],
        [ 2.89306329e+03, -3.42858431e+01,  1.17466461e+02, ...,
         6.85125859e+00,  4.80304373e+00, -3.75749460e+01],
        ...,
        [-1.34300326e+02,  3.55148415e+03, -5.34447049e+02, ...,
         2.80035928e+01,  1.12310033e+00,  2.81249968e+01],
        [ 3.73568104e+02, -2.78559446e+03,  1.19699164e+03, ...,
         4.53501689e+01,  1.86077906e+01,  5.42387740e+01],
        [-3.57633319e+03,  1.29170341e+03,  4.64224199e+02, ...,
         -1.45355634e+00, -1.47953585e+01,  1.88779259e+00]]),
 array([[-1.97547669e+03, -5.30835552e+01, -1.56924329e+03, ...,
         3.18810979e+01,  4.57017853e+01,  5.32333735e+00],
        [-6.99288660e+02, -1.78444521e+03,  8.83082635e+02, ...,
         9.63825071e+00,  1.66423792e+01, -1.77091928e+00],
        [ 2.89306329e+03, -3.42858431e+01,  1.17466461e+02, ...,
         4.80304373e+00, -3.75749460e+01, -4.17984904e+01],
        ...,
        [-1.34300326e+02,  3.55148415e+03, -5.34447049e+02, ...,
         1.12310033e+00,  2.81249968e+01,  3.91580693e+00],
        [ 3.73568104e+02, -2.78559446e+03,  1.19699164e+03, ...,
         1.86077906e+01,  5.42387740e+01,  1.57509393e+01],
        [-3.57633319e+03,  1.29170341e+03,  4.64224199e+02, ...,
         -1.47953585e+01,  1.88779259e+00, -2.46103644e+01]]),
 array([[-1.97547669e+03, -5.30835552e+01, -1.56924329e+03, ...,
         4.57017853e+01,  5.32333735e+00, -3.42644521e+01],
```

```
       [-6.99288660e+02, -1.78444521e+03,  8.83082635e+02, ...,
         1.66423792e+01, -1.77091928e+00,  1.99974260e+01],
       [ 2.89306329e+03, -3.42858431e+01,  1.17466461e+02, ...,
        -3.75749460e+01, -4.17984904e+01, -1.28415515e+01],
       ...,
       [-1.34300326e+02,  3.55148415e+03, -5.34447049e+02, ...,
         2.81249968e+01,  3.91580693e+00, -3.55386356e+01],
       [ 3.73568104e+02, -2.78559446e+03,  1.19699164e+03, ...,
         5.42387740e+01,  1.57509393e+01,  1.41566869e+01],
       [-3.57633319e+03,  1.29170341e+03,  4.64224199e+02, ...,
         1.88779259e+00, -2.46103644e+01, -2.00678937e+01]]),
array([[-1.97547669e+03, -5.30835552e+01, -1.56924329e+03, ...,
         5.32333735e+00, -3.42644521e+01, -8.64215401e+00],
       [-6.99288660e+02, -1.78444521e+03,  8.83082635e+02, ...,
        -1.77091928e+00,  1.99974260e+01,  1.31743399e+01],
       [ 2.89306329e+03, -3.42858431e+01,  1.17466461e+02, ...,
        -4.17984904e+01, -1.28415515e+01,  1.18591113e+01],
       ...,
       [-1.34300326e+02,  3.55148415e+03, -5.34447049e+02, ...,
         3.91580693e+00, -3.55386356e+01, -1.22780142e+01],
       [ 3.73568104e+02, -2.78559446e+03,  1.19699164e+03, ...,
         1.57509393e+01,  1.41566869e+01,  2.14984200e+01],
       [-3.57633319e+03,  1.29170341e+03,  4.64224199e+02, ...,
        -2.46103644e+01, -2.00678937e+01, -5.56183317e+00]]),
array([[-1.97547669e+03, -5.30835552e+01, -1.56924329e+03, ...,
        -3.42644521e+01, -8.64215401e+00, -3.74304420e+01],
       [-6.99288660e+02, -1.78444521e+03,  8.83082635e+02, ...,
         1.99974260e+01,  1.31743399e+01,  5.77184208e+01],
       [ 2.89306329e+03, -3.42858431e+01,  1.17466461e+02, ...,
        -1.28415515e+01,  1.18591113e+01, -4.82312245e+00],
       ...,
       [-1.34300326e+02,  3.55148415e+03, -5.34447049e+02, ...,
        -3.55386356e+01, -1.22780142e+01,  3.81684852e+01],
       [ 3.73568104e+02, -2.78559446e+03,  1.19699164e+03, ...,
         1.41566869e+01,  2.14984200e+01,  4.25385270e+01],
       [-3.57633319e+03,  1.29170341e+03,  4.64224199e+02, ...,
        -2.00678937e+01, -5.56183317e+00, -2.65968230e+00]]),
array([[-1.97547669e+03, -5.30835552e+01, -1.56924329e+03, ...,
        -8.64215401e+00, -3.74304420e+01, -2.55054644e+01],
       [-6.99288660e+02, -1.78444521e+03,  8.83082635e+02, ...,
         1.31743399e+01,  5.77184208e+01,  1.96022936e+01],
       [ 2.89306329e+03, -3.42858431e+01,  1.17466461e+02, ...,
         1.18591113e+01, -4.82312245e+00,  1.85825583e+01],
       ...,
       [-1.34300326e+02,  3.55148415e+03, -5.34447049e+02, ...,
        -1.22780142e+01,  3.81684852e+01, -5.53222661e-01],
       [ 3.73568104e+02, -2.78559446e+03,  1.19699164e+03, ...,
         2.14984200e+01,  4.25385270e+01, -2.10250905e+00],
       [-3.57633319e+03,  1.29170341e+03,  4.64224199e+02, ...,
        -5.56183317e+00, -2.65968230e+00,  2.56776053e+01]]),
array([[-1.97547669e+03, -5.30835552e+01, -1.56924329e+03, ...,
        -3.74304420e+01, -2.55054644e+01, -1.78185765e+01],
       [-6.99288660e+02, -1.78444521e+03,  8.83082635e+02, ...,
         5.77184208e+01,  1.96022936e+01,  1.32416719e+01],
       [ 2.89306329e+03, -3.42858431e+01,  1.17466461e+02, ...,
        -4.82312245e+00,  1.85825583e+01,  2.59440623e+01],
       ...,
```

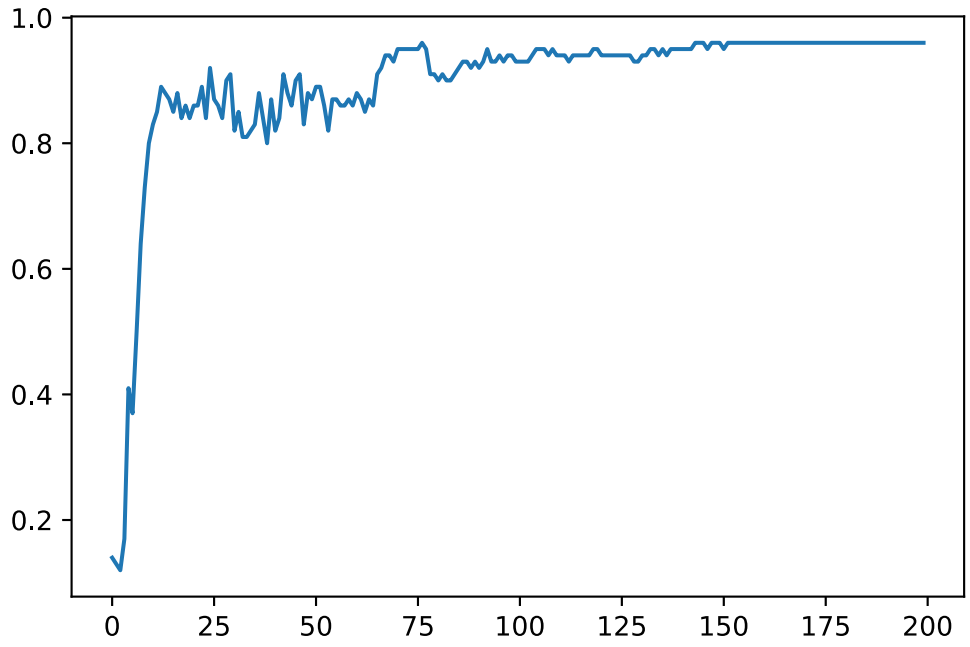
```
       [-1.34300326e+02,  3.55148415e+03, -5.34447049e+02, ...,
         3.81684852e+01, -5.53222661e-01,  5.88466333e+01],
       [ 3.73568104e+02, -2.78559446e+03,  1.19699164e+03, ...,
         4.25385270e+01, -2.10250905e+00,  5.43831255e+00],
       [-3.57633319e+03,  1.29170341e+03,  4.64224199e+02, ...,
        -2.65968230e+00,  2.56776053e+01, -4.53644300e+00]]),
array([[-1.97547669e+03, -5.30835552e+01, -1.56924329e+03, ...,
        -2.55054644e+01, -1.78185765e+01, -5.74941741e+01],
       [-6.99288660e+02, -1.78444521e+03,  8.83082635e+02, ...,
         1.96022936e+01,  1.32416719e+01,  1.26162935e+01],
       [ 2.89306329e+03, -3.42858431e+01,  1.17466461e+02, ...,
         1.85825583e+01,  2.59440623e+01, -1.38852135e+01],
       ...,
       [-1.34300326e+02,  3.55148415e+03, -5.34447049e+02, ...,
        -5.53222661e-01,  5.88466333e+01,  1.81584427e+01],
       [ 3.73568104e+02, -2.78559446e+03,  1.19699164e+03, ...,
        -2.10250905e+00,  5.43831255e+00,  1.80014178e+01],
       [-3.57633319e+03,  1.29170341e+03,  4.64224199e+02, ...,
         2.56776053e+01, -4.53644300e+00, -1.01100255e+01]]),
array([[-1.97547669e+03, -5.30835552e+01, -1.56924329e+03, ...,
        -1.78185765e+01, -5.74941741e+01, -1.59145407e+00],
       [-6.99288660e+02, -1.78444521e+03,  8.83082635e+02, ...,
         1.32416719e+01,  1.26162935e+01, -7.48237680e+00],
       [ 2.89306329e+03, -3.42858431e+01,  1.17466461e+02, ...,
         2.59440623e+01, -1.38852135e+01, -1.16894921e+01],
       ...,
       [-1.34300326e+02,  3.55148415e+03, -5.34447049e+02, ...,
         5.88466333e+01,  1.81584427e+01,  5.16386732e+01],
       [ 3.73568104e+02, -2.78559446e+03,  1.19699164e+03, ...,
         5.43831255e+00,  1.80014178e+01,  9.25385862e+00],
       [-3.57633319e+03,  1.29170341e+03,  4.64224199e+02, ...,
        -4.53644300e+00, -1.01100255e+01,  7.70409578e+00]]),
array([[-1.97547669e+03, -5.30835552e+01, -1.56924329e+03, ...,
        -5.74941741e+01, -1.59145407e+00, -2.77530015e+00],
       [-6.99288660e+02, -1.78444521e+03,  8.83082635e+02, ...,
         1.26162935e+01, -7.48237680e+00, -1.71572097e+01],
       [ 2.89306329e+03, -3.42858431e+01,  1.17466461e+02, ...,
        -1.38852135e+01, -1.16894921e+01,  2.42454076e+01],
       ...,
       [-1.34300326e+02,  3.55148415e+03, -5.34447049e+02, ...,
         1.81584427e+01,  5.16386732e+01,  2.77852810e+01],
       [ 3.73568104e+02, -2.78559446e+03,  1.19699164e+03, ...,
         1.80014178e+01,  9.25385862e+00,  1.42578927e+01],
       [-3.57633319e+03,  1.29170341e+03,  4.64224199e+02, ...,
        -1.01100255e+01,  7.70409578e+00, -6.08178790e+00]]),
array([[-1.97547669e+03, -5.30835552e+01, -1.56924329e+03, ...,
        -1.59145407e+00, -2.77530015e+00, -8.89411762e+00],
       [-6.99288660e+02, -1.78444521e+03,  8.83082635e+02, ...,
        -7.48237680e+00, -1.71572097e+01,  1.49558778e+00],
       [ 2.89306329e+03, -3.42858431e+01,  1.17466461e+02, ...,
        -1.16894921e+01,  2.42454076e+01,  4.68028026e+01],
       ...,
       [-1.34300326e+02,  3.55148415e+03, -5.34447049e+02, ...,
         5.16386732e+01,  2.77852810e+01, -1.69586620e+01],
       [ 3.73568104e+02, -2.78559446e+03,  1.19699164e+03, ...,
         9.25385862e+00,  1.42578927e+01,  2.50244325e+01],
       [-3.57633319e+03,  1.29170341e+03,  4.64224199e+02, ...,
```

```
         7.70409578e+00, -6.08178790e+00, -1.86538906e+01]]),
 array([[-1.97547669e+03, -5.30835552e+01, -1.56924329e+03, ...,
         -2.77530015e+00, -8.89411762e+00,  6.64265043e+01],
        [-6.99288660e+02, -1.78444521e+03,  8.83082635e+02, ...,
         -1.71572097e+01,  1.49558778e+00, -1.86847910e+01],
        [ 2.89306329e+03, -3.42858431e+01,  1.17466461e+02, ...,
          2.42454076e+01,  4.68028026e+01,  1.14156848e+01],
        ...,
        [-1.34300326e+02,  3.55148415e+03, -5.34447049e+02, ...,
          2.77852810e+01, -1.69586620e+01,  1.70743433e+01],
        [ 3.73568104e+02, -2.78559446e+03,  1.19699164e+03, ...,
          1.42578927e+01,  2.50244325e+01, -1.03708977e+01],
        [-3.57633319e+03,  1.29170341e+03,  4.64224199e+02, ...,
         -6.08178790e+00, -1.86538906e+01,  1.95060760e+01]]),
 array([[-1.97547669e+03, -5.30835552e+01, -1.56924329e+03, ...,
         -8.89411762e+00,  6.64265043e+01, -2.72465289e+01],
        [-6.99288660e+02, -1.78444521e+03,  8.83082635e+02, ...,
          1.49558778e+00, -1.86847910e+01,  8.12572779e+00],
        [ 2.89306329e+03, -3.42858431e+01,  1.17466461e+02, ...,
          4.68028026e+01,  1.14156848e+01, -1.46682865e+01],
        ...,
        [-1.34300326e+02,  3.55148415e+03, -5.34447049e+02, ...,
         -1.69586620e+01,  1.70743433e+01,  3.99398658e+01],
        [ 3.73568104e+02, -2.78559446e+03,  1.19699164e+03, ...,
          2.50244325e+01, -1.03708977e+01,  3.28145851e+01],
        [-3.57633319e+03,  1.29170341e+03,  4.64224199e+02, ...,
         -1.86538906e+01,  1.95060760e+01, -2.21772875e+00]]),
 array([[-1.97547669e+03, -5.30835552e+01, -1.56924329e+03, ...,
          6.64265043e+01, -2.72465289e+01,  2.86139069e+01],
        [-6.99288660e+02, -1.78444521e+03,  8.83082635e+02, ...,
         -1.86847910e+01,  8.12572779e+00,  1.39060709e+01],
        [ 2.89306329e+03, -3.42858431e+01,  1.17466461e+02, ...,
          1.14156848e+01, -1.46682865e+01, -1.18988271e+01],
        ...,
        [-1.34300326e+02,  3.55148415e+03, -5.34447049e+02, ...,
          1.70743433e+01,  3.99398658e+01, -2.88494327e+01],
        [ 3.73568104e+02, -2.78559446e+03,  1.19699164e+03, ...,
         -1.03708977e+01,  3.28145851e+01, -3.85305625e+01],
        [-3.57633319e+03,  1.29170341e+03,  4.64224199e+02, ...,
          1.95060760e+01, -2.21772875e+00,  3.82862162e+00]]),
 array([[-1.97547669e+03, -5.30835552e+01, -1.56924329e+03, ...,
         -2.72465289e+01,  2.86139069e+01, -1.87877326e+01],
        [-6.99288660e+02, -1.78444521e+03,  8.83082635e+02, ...,
          8.12572779e+00,  1.39060709e+01, -5.41175533e+00],
        [ 2.89306329e+03, -3.42858431e+01,  1.17466461e+02, ...,
         -1.46682865e+01, -1.18988271e+01,  2.09892096e+00],
        ...,
        [-1.34300326e+02,  3.55148415e+03, -5.34447049e+02, ...,
          3.99398658e+01, -2.88494327e+01,  5.86519724e+00],
        [ 3.73568104e+02, -2.78559446e+03,  1.19699164e+03, ...,
          3.28145851e+01, -3.85305625e+01,  1.34925461e+00],
        [-3.57633319e+03,  1.29170341e+03,  4.64224199e+02, ...,
         -2.21772875e+00,  3.82862162e+00, -2.90411512e+01]])]
```

(g) Face Recognition. Extract training and test features for r = 10. Train a Logistic Regression model using F and test on Ftest. Report the classification accuracy on the test set. Plot the classification accuracy on the test set as a function of r when r = 1, 2,..., 200. Use “one-vs-rest” logistic regression, where a classifier is trained for each possible output label. Each classifier is trained on faces with that label as positive data and all faces with other labels as negative data. sklearn calls this “ovr” mode.

In [15]:
```
f_r10 = effs[9]
test_ft10 = ftest[9]

OVR = OneVsRestClassifier(LogisticRegression()).fit(f_r10,train_labels)

print(OVR.score(test_ft10,test_labels))

accuracy = []

def classifier(r):
    OVR = OneVsRestClassifier(LogisticRegression()).fit(effs[r - 1],train_labe
ls)
    accuracy.append(OVR.score(ftest[r - 1],test_labels))

for r in range(1, 201, 1):
    classifier(r)

plt.plot(accuracy)
plt.show()
```

```
0.8
```



In [ ]:
```

```