Zachary Anderson, Andrew Ng

Lab 11

4.1 Creating Training Sets

```
from google.colab import drive
drive.mount("/content/gdrive")

# Load the data
path = '/content/gdrive/MyDrive/att_faces'
folder_names = ['s1', 's2', 's3', 's4', 's5', 's6', 's7', 's8', 's9', 's10',
                's11', 's12', 's13', 's14', 's15', 's16', 's17', 's18', 's19', 's20',
                's21', 's22', 's23', 's24', 's25', 's26', 's27', 's28', 's29', 's30',
                's31', 's32', 's33', 's34', 's35', 's36', 's37', 's38', 's39', 's40']

train_file_names = ['1.pgm', '2.pgm', '3.pgm', '4.pgm', '5.pgm',
                    '6.pgm', '7.pgm', '8.pgm', '9.pgm']

test_file_names = ['10.pgm']


import numpy as np
import scipy as sc
import matplotlib.pyplot as plt
import matplotlib.image as img
```

Created lists for folder and file names, then extracted data, vectorized it, then put it in arrays.

```
[2] #Open Each Folder
    samples = 9
    testSamples = 1
    Xtest = np.zeros((10304,40))
    Xtrain = np.zeros((10304,360))
    Ytrain = np.zeros(40*samples)
    Ytest = np.zeros(40*testSamples)

    for folder in folder_names:
      i = folder_names.index(folder)
      for File in train_file_names:
          j = train_file_names.index(File)
          im_path = path + '/' + folder + '/' + File
          current = img.imread(im_path)
          Xtrain[:, i*samples + j] = np.ravel(current)
          Ytrain[i*samples + j] = int(i + 1)
      for File in test_file_names:
          j = test_file_names.index(File)
          im_path = path + '/' + folder + '/' + File
          current = img.imread(im_path)
          Xtest[:, i*testSamples + j] = np.ravel(current)
          Ytest[i*testSamples + j] = int(i + 1)
```

```
print(Xtest.shape)
print(Xtrain.shape)
print(Ytest.shape)
print(Ytrain.shape)
```

```
(10304, 40)
(10304, 360)
(40,)
(360,)
```

5.1 PCA Transform of Training Points

```
def train(Xtrain, K):
  N, n_faces = Xtrain.shape
  mu = Xtrain.mean(axis=1)
  X = np.zeros((N, n_faces))
  for i in range(n_faces):
    X[:, i] = Xtrain[:, i] - mu

  sigma = np.matmul(X, X.transpose())
  sigma /= n_faces
  _, sigma_eigs = sc.sparse.linalg.eigs(sigma, k= K)
  P_H = np.conj(np.transpose(sigma_eigs))
  X_pca = np.matmul(P_H, X)
  return mu, sigma_eigs, X_pca
```

5.2 PCA Transform of Test Points

```
def test(mean, P, Xtest):
  N, n_faces = Xtest.shape
  X = np.zeros((N, n_faces))

  for i in range(n_faces):
    X[:, i] = Xtest[:, i] - mean

  P_H = np.conj(np.transpose(P))
  X_pca = np.matmul(P_H, X)

  return X_pca
```

6.1 Nearest Neighbor

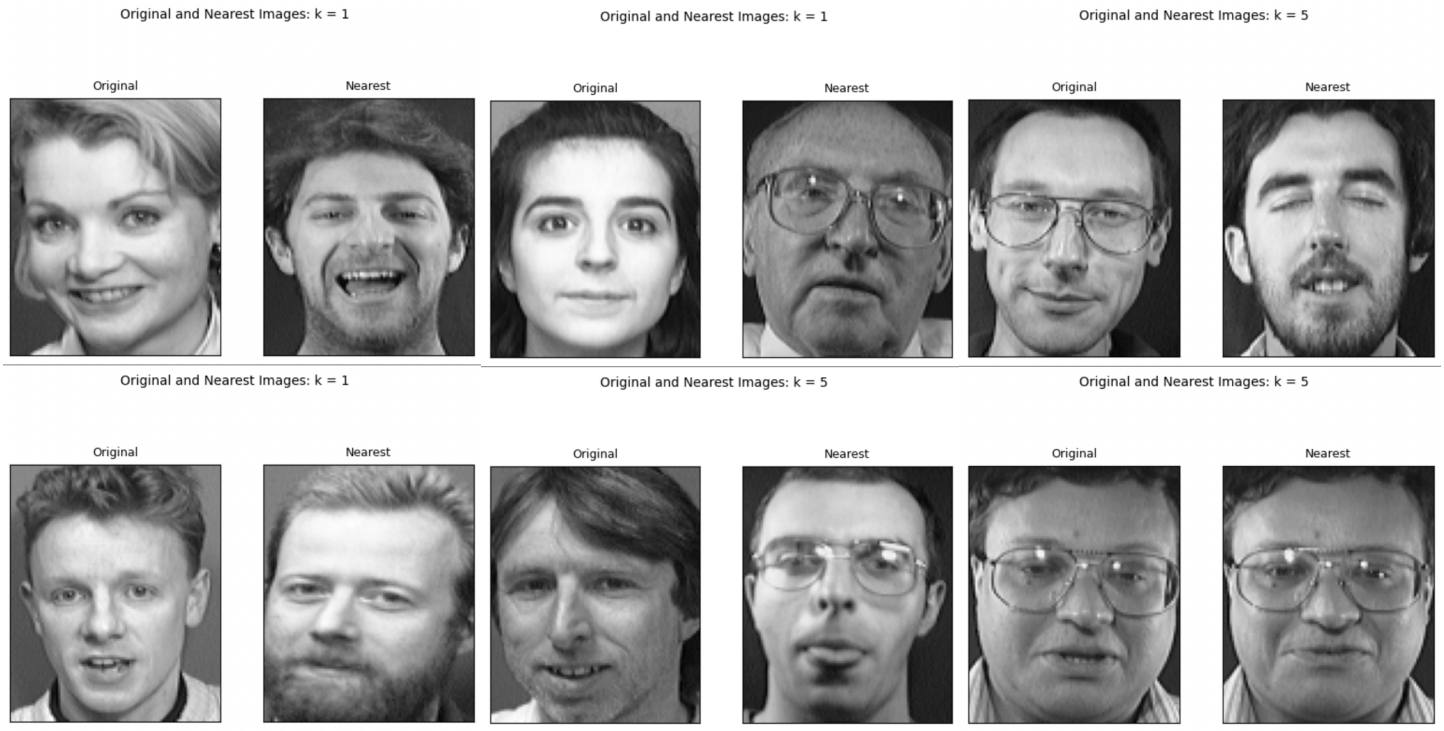
```
mean, eigans, PCA_train = transform_training_points(1, X_train)
PCA_test = transform_test_points(X_test, mean, eigans)
neighbor = nearest_neighbor(PCA_train, PCA_test)
```

```
print(neighbor)
```

```
[109 223 210  18 250  34 125 133 218 299 100 122 255 131 315 309  77 188
 191 295 226  43 124 228 351 224 329 237 343 261 126 232 289 206  55 240
  84 118 343 127]
```

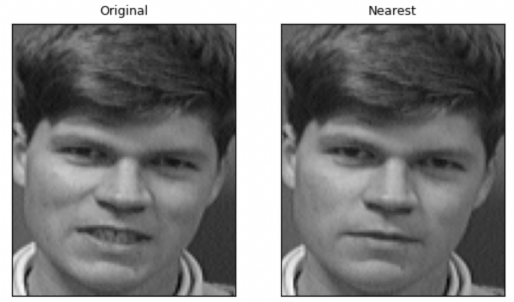
This code shows the nearest neighbor class working, the numbers printed are the column numbers of each of the closest images in the test set. The next step is to reconstruct the columns back into 2D arrays, and then print the image so we can see the similarities.

We have to repeat the PCA transform taking a different amount of eigenvectors, corresponding to the highest eigenvalues, depending on what the value of k is. What we expect to see is that for a greater amount of k, eigenvectors, the nearest neighbor class should work better.



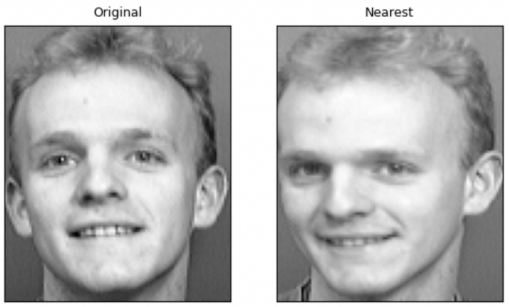
Original and Nearest Images: k = 10

Original and Nearest Images: k = 10

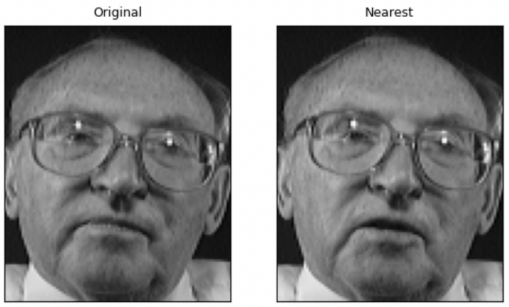


\

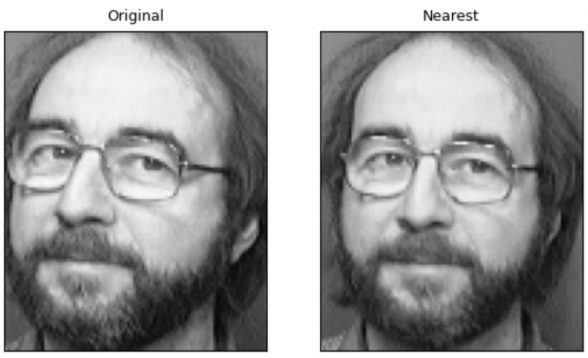
Original and Nearest Images: k = 10



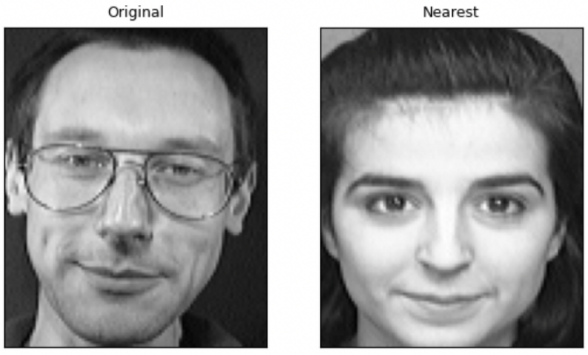
Original and Nearest Images: k = 20



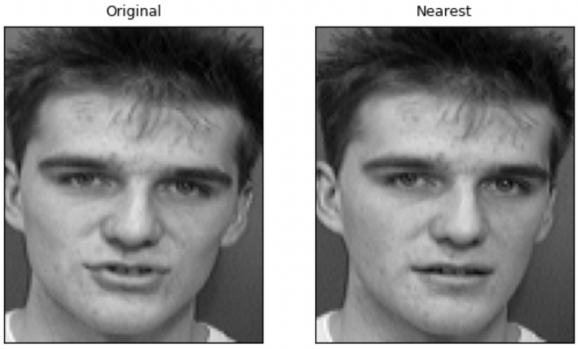
Original and Nearest Images: k = 20



Original and Nearest Images: k = 20



Original and Nearest Images: k = 50



Original and Nearest Images: k = 50

As expected, as the number of eigenvectors increases, we see more of the test images matching the training set counterparts.

This is still not perfect because the nearest neighbor method is not the most effective, especially with a training set as small as nine images per person. But it works well as a preliminary test.