# CS 383 - Machine Learning

Assignment 3 - Dimensionality Reduction

## Introduction

In this assignment you'll work on visualizing data, reducing its dimensionality and clustering it.

You may not use any functions from machine learning library in your code, however you may use statistical functions. For example, if available you **MAY NOT** use functions like

- pca
- k-nearest neighbors functions

Unless explicitly told to do so. But you **MAY** use basic statistical functions like:

- std
- mean
- cov
- eig
- svd

## Grading

| | |
|---|---|
| Part 1 (Theory) | 10pts |
| Part 2 (PCA) | 40pts |
| Part 3 (Eigenfaces) | 40pts |
| Report | 10pts |
| **TOTAL** | 100pts |

Table 1: Grading Rubric

# DataSets

**Labeled Faces in the Wild Datasaet** This dataset consists of celebrities download from the Internet from the early 2000s. We use the grayscale version from sklearn.datasets.

we will download the images in a specific way as shown below. You will have 3,023 images, each 87x65 pixels large, belonging to 62 different people.

```
from sklearn.datasets import fetch_lfw_people
import matplotlib.pyplot as plt
import matplotlib.cm as cm

people = fetch_lfw_people(min_faces_per_person=20, resize=0.7)
image_shape = people.images[0].shape

fig, axes = plt.subplots(2, 5, figsize=(15, 8),
                         subplot_kw={'xticks': (), 'yticks': ()})
for target, image, ax in zip(people.target, people.images, axes.ravel()):
    ax.imshow(image, cmap=cm.gray)
    ax.set_title(people.target_names[target])
```

# 1 Theory Questions

1. Consider the following data:

$$\begin{bmatrix} -2 & 1 \\ -5 & -4 \\ -3 & 1 \\ 0 & 3 \\ -8 & 11 \\ -2 & 5 \\ 1 & 0 \\ 5 & -1 \\ -1 & -3 \\ 6 & 1 \end{bmatrix}$$

(a) Find the principle components of the data (you must show the math, including how you compute the eigenvectors and eigenvalues). Make sure you standardize the data first and that your principle components are normalized to be unit length. As for the amount of detail needed in your work imagine that you were working on paper with a basic calculator. Show me whatever you would be writing on that paper. (7pts).

$$\mu_1 = \frac{-2-5-3-8-2+1+5-1+6}{10} = -0.9$$

$$\mu_2 = \frac{1-4+1+3+11+5-1-3+1}{10} = 1.4$$

$$\sigma_1 = \sqrt{\frac{\sum(X_{i,1}-\mu_1)^2}{N-1}} = 4.23$$

$$\sigma_2 = \sqrt{\frac{\sum(X_{i,2}-\mu_2)^2}{N-1}} = 4.27$$

$$C = \frac{X_s^T X_s}{N-1} = \frac{1}{9}\begin{bmatrix} 9 & -3.67 \\ -3.67 & 9 \end{bmatrix} = \begin{bmatrix} 1 & -0.41 \\ -0.41 & 1 \end{bmatrix}$$

$$|C-\lambda I| = 0 \implies \begin{vmatrix} 1-\lambda & -0.41 \\ -0.41 & 1-\lambda \end{vmatrix} = 0$$

$$\implies (1-\lambda)^2 = (-.41)^2 \implies 1-\lambda = \pm .41 \implies \lambda = 1 \pm .41 = \{.59, 1.41\}$$

$$\lambda_1 = .59 \implies \begin{bmatrix} .41 & -.41 \\ -.41 & .41 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \implies \begin{bmatrix} .41 & -.41 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \implies x_1 = x_2$$

$$\lambda_2 = 1.41 \implies \begin{bmatrix} -.41 & -.41 \\ -.41 & -.41 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \implies \begin{bmatrix} -.41 & -.41 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \implies x_1 = -x_2$$

$$\implies v_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix} = \begin{bmatrix} 0.707 \\ 0.707 \end{bmatrix}, v_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix} = \begin{bmatrix} -0.707 \\ 0.707 \end{bmatrix}$$

(b) Project the data onto the principal component corresponding to the largest eigenvalue found in the previous part (3pts).

$$Z = X_s W = \begin{bmatrix} -0.26 & -0.09 \\ -0.97 & -1.26 \\ -0.5 & -0.09 \\ 0.21 & 0.37 \\ -1.68 & 2.25 \\ -0.26 & 0.84 \\ 0.45 & -0.33 \\ 1.40 & -0.56 \\ -0.02 & -1.03 \\ 1.63 & -0.09 \end{bmatrix} \begin{bmatrix} -0.707 \\ 0.707 \end{bmatrix} = \begin{bmatrix} 0.12 \\ -0.21 \\ 0.29 \\ 0.11 \\ 2.78 \\ 0.78 \\ -0.55 \\ -1.38 \\ -0.71 \\ -1.22 \end{bmatrix}$$

## 2 Dimensionality Reduction via PCA

Import the data as shown above. This the labeled faces in the wild dataset.
Verify that you have the correct number of people and classes

```
print("people.images.shape: {}".format(people.images.shape))
print("Number of classes: {}".format(len(people.target_names)))
```

```
people.images.shape: (3023, 87, 65)
Number of classes: 62
```

This dataset is skewed toward George W. Bush and Colin Powell as you can verify here

```
# count how often each target appears
counts = np.bincount(people.target)
# print counts next to target names
for i, (count, name) in enumerate(zip(counts, people.target_names)):
    print("{0:25} {1:3}".format(name, count), end='   ')
    if (i + 1) % 3 == 0:
        print()
```

To make the data less skewed, we will only take up to 50 images of each person (otherwise, the feature extraction would be overwhelmed by the likelihood of George W. Bush):

```
mask = np.zeros(people.target.shape, dtype=np.bool)
for target in np.unique(people.target):
    mask[np.where(people.target == target)[0][:50]] = 1

X_people = people.data[mask]
y_people = people.target[mask]

# scale the grayscale values to be between 0 and 1
# instead of 0 and 255 for better numeric stability
X_people = X_people / 255.
```

We are now going to compute how well a KNN classifier does using just the pixels alone.

```
from sklearn.neighbors import KNeighborsClassifier
# split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(
    X_people, y_people, stratify=y_people, random_state=0)
# build a KNeighborsClassifier using one neighbor
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train, y_train)
print("Test set score of 1-nn: {:.2f}".format(knn.score(X_test, y_test)))
```

You should have an accuracy around 23% - 27%.

Once you have your setup complete, write a script to do the following:

1. Write your own version of KNN (k=1) where you use the SSD (sum of squared differences) to compute similarity
2. Verify that your KNN has a similar accuracy as sklearn's version
3. Standardize your data (zero mean, divide by standard deviation)
4. Reduces the data to 100D using PCA
5. Compute the KNN again where K=1 with the 100D data. Report the accuracy
6. Compute the KNN again where K=1 with the 100D Whitened data. Report the accuracy
7. Reduces the data to 2D using PCA
8. Graphs the data for visualization

Recall that although you may not use any package ML functions like *pca*, you **may** use statistical functions like *eig* or *svd*.

Answers:

- My KNN Accuracy: 0.23255813953488372
- Sklearn KNN Accuracy: 0.23
- 100D KNN Accuracy: 0.25387596899224807
- Whitened 100D KNN Accuracy: 0.3313953488372093

# 3 Eigenfaces

Import the data as shown above. This the labeled faces in the wild dataset.
Use the X_train data from above. Let's analyze the first and second principal components.

**Write a script that:**

1. Imports the data as mentioned above.
2. Standardizes the data.
3. Performs PCA on the data (again, although you may not use any package ML functions like *pca*, you **may** use statistical functions like *eig*). No need to whiten here.
4. Find the max and min image on PC1's axis. Find the max and min of PC2. Plot and report which faces these points correspond to, what variation do these components capture?
5. Visualizes the most important principle component as a 87x65 image.
6. Reconstructs the *X_train[0,:]* image using the primary principle component. To best see the full re-construction, "unstandardize" the reconstruction by multiplying it by the original standard deviation and adding back in the original mean.
7. Determines the number of principle components necessary to encode at least 95% of the information, $k$.
8. Reconstructs the *X_train[0,:]* image using the $k$ most significant eigen-vectors. For the fun of it maybe even look to see if you can perfectly reconstruct the face if you use all the eigen-vectors! Again, to best see the full re-construction, "unstandardize" the reconstruction by multiplying it by the original standard deviation and adding back in the original mean.

Answers:

- In PC1, the eyes are wider and the lips/mouth are wider. It is similar to the surprised face.
- In PC2, the eyes are squinted and the lips/mouth perturb outwards. It is as if the picture was taken mid-speech.
- Number of principle components to reach 95%: 189

# Submission

For your submission, upload to Blackboard a single zip file containing:

1. A LaTeX typeset PDF or Jupyter Notebook PDF containing:
   (a) Part 1: Your answers to the theory questions.
   (b) Part 2: The visualization of the PCA result, KNN accuracies
   (c) Part 3:
      i. Visualization of primary principle component
      ii. Number of principle components needed to represent 95% of information, $k$.
      iii. Visualization of the reconstruction of the first person using
         A. Original image
         B. Single principle component
         C. $k$ principle components.
   (d) Source Code - python notebook