# CS 5875 Applied Machine Learning

Homework 2

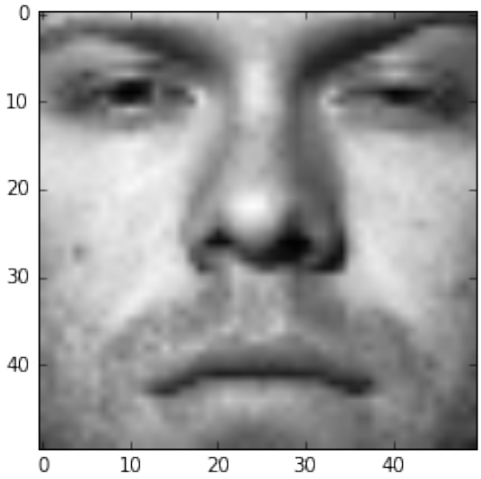
Weiming Zhang, Yan Jiang

## I. Objective

LDA, SVD and Naïve Bayes.

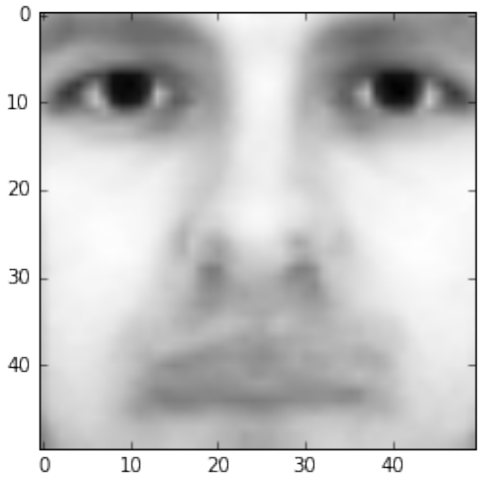
## II. Problem 1: Eigenface

*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.*




See source code function **1-b Load Data**.

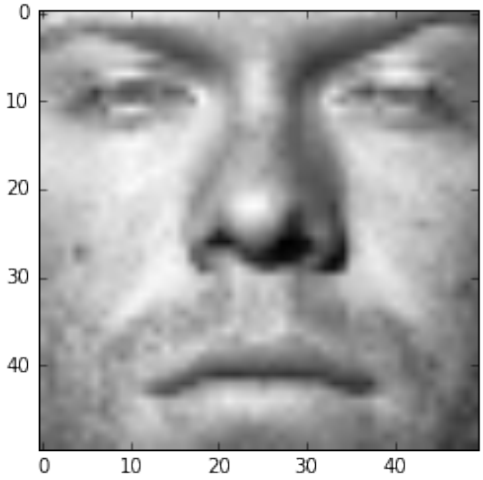
*c) Average Face. Compute the average face μ from the whole training set by summing up every column in X then dividing by the number of faces. Display the average face as a grayscale image.*



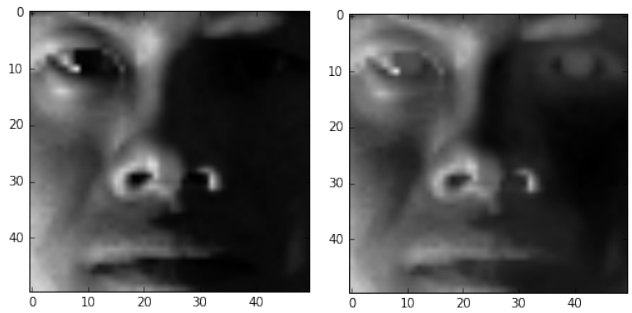

See code for this part in **1-c Average Face ()**.

*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).*

Face #10 in training data after subtraction:

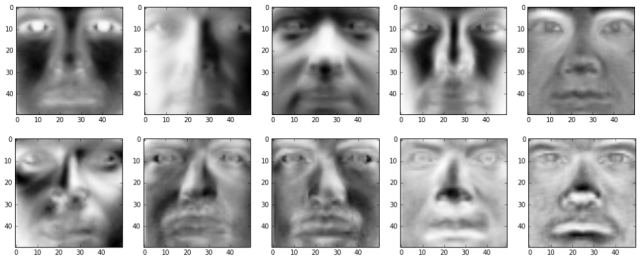



I also have the same for test data in the code.
Face #10 in test data before and after subtraction:



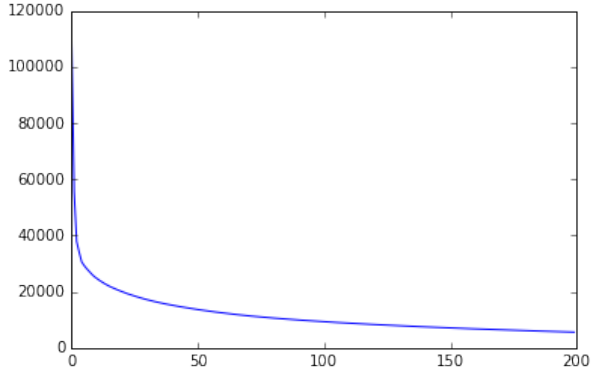
See code for this part in **1-d Mean Subtraction.**

e) *Eigenface. Perform Singular Value Decomposition (SVD) on training set X (X = UßVT ) to get matrix 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.*



See code for this part in **1-e Eigenface**.

f) *Low-rank Approximation. Since ß is a diagonal matrix with non-negative real numbers on the diagonal in non-ascending order, we can use the first r elements in ß together with first r columns in U and first r rows in VT to approximate X. That is, we can approximate X by Xˆ r = U[:,: r ] ß[: r,: r ] VT [: r,:]. The matrix Xˆ r is called rank-r approximation of X. Plot the rank-r approximation error kX°Xˆ rkF 3 as a function of r when r = 1, 2,..., 200.*

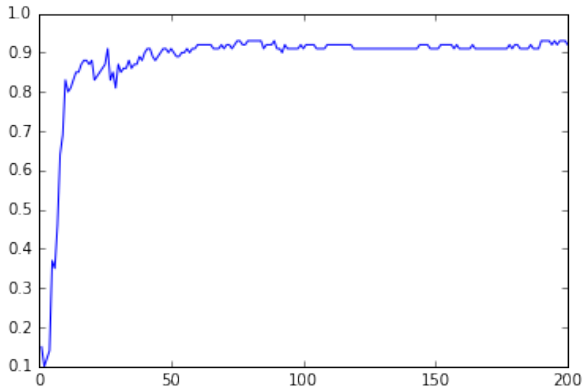



See code for this part in **1-f Low-rank Approximation**.

g) *Eigenface Feature. The top r eigenfaces VT [: r,:] = {v1, v2,..., vr } T span 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 feature vector f: f = VT [: r,:] z = [v1, v2,..., vr ] T z. 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).*

See code for this in **1-g Eigenface Feature.**

h) *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.*




See code at **1-h Face Recognition.**

## III. Problem 2: The Titanic Disaster

b) *Tell us about the data. How many samples (dishes) are there in the training set? How many categories (types of cuisine)? Use a list to keep all the unique ingredients appearing in the training set. How many unique ingredients are there?*

Number of samples in training set: 39774
Number of cuisine: 20
Number of unique ingredients in training set: 6714
See code in **1-b Tell us about the data.**

c) *Represent each dish by a binary ingredient feature vector. Suppose there are d different ingredients in total from the training set, represent each dish by a 1£d binary ingredient vector x, where xi = 1 if the dish contains ingredient i and xi = 0 otherwise. For example, suppose all the ingredients we have in the training set are { beef, chicken, egg, lettuce, tomato, rice } and the dish is made by ingredients { chicken, lettuce, tomato, rice }, then the dish could be represented by a 6£1 binary vector [0, 1, 0, 1, 1, 1] as its feature or attribute. Use n £d feature matrix to represent all the dishes in training set and test set, where n is the number of dishes.*

See code in **1-c Feature Vector Extration.**

*d) Using Naïve Bayes Classifier to perform 3 fold cross-validation on the training set and report your average classification accuracy. Try both Gaussian distribution prior assumption and Bernoulli distribution prior assumption.*

Gaussian: 37.98%
Bernoulli: 68.35%
See code in **1-d Naïve Bayes.**

*e) For Gaussian prior and Bernoulli prior, which performs better in terms of cross-validation accuracy? Why? Please give specific arguments.*

Based on observation, **Bernoulli** is performing better than **Gaussian.** Gaussian is generally good for continuous data where each feature is a real number. Bernoulli only describes whether an event occurred or not. In this case the feature is a vector of ingredients present or not. So Bernoulli would perform better.

*f) Using Logistic Regression Model to perform 3 fold cross-validation on the training set and report your average classification accuracy.*

Logistic Regression: 77.56%
See code in **1-f Logistic Regression.**

*g) Train your best-performed classifier with all of the training data, and generate test labels on test set. Submit your results to Kaggle and report the accuracy.*

Using Logistic Regression, I get accuracy 78.339%
See code in **1-g Kaggle.**