CS5785 HomeWork 2

Yihan Chen
Praveen Kumar Govindaraj

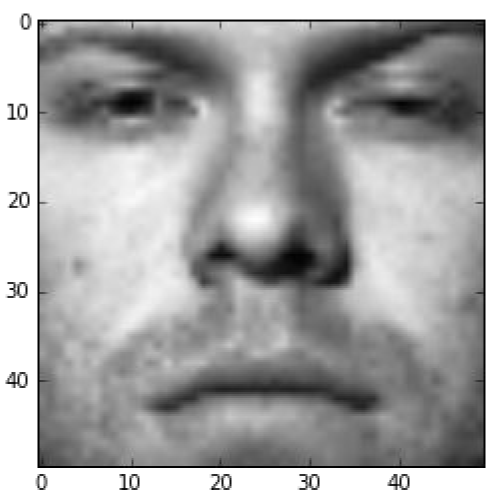
1. Eigenface for face recognition

(a) Download The Face Dataset. After you unzip faces.zip, you will find a folder called images 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.

Downloaded the training set and test set.

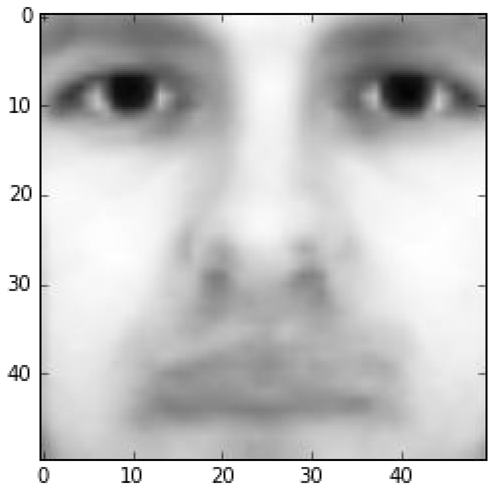
*(b) Load the training set into a matrix X: there are 540 training images in total, each has 50*50 pixels that need to be concatenated into a 2500-dimensional vector. So the size of X should be 540*2500, where each row is a flattened face image. Pick a face image from X and display that image in grayscale.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

We picked up a face image from training set and displayed it.



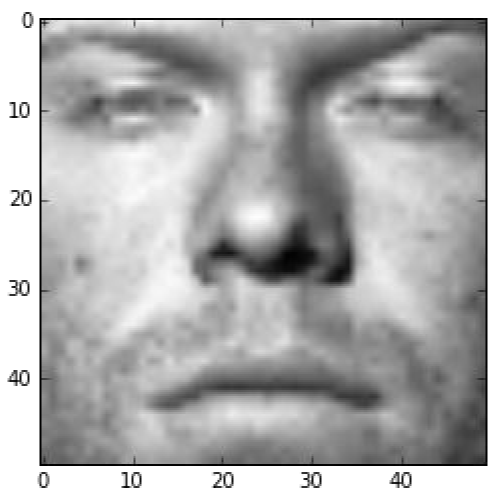
(c) Average Face. Compute the average face μ from the whole training set by summing up every column in X then dividing by the number of faces. Display the average face as a grayscale
image.

This is the average face we got.

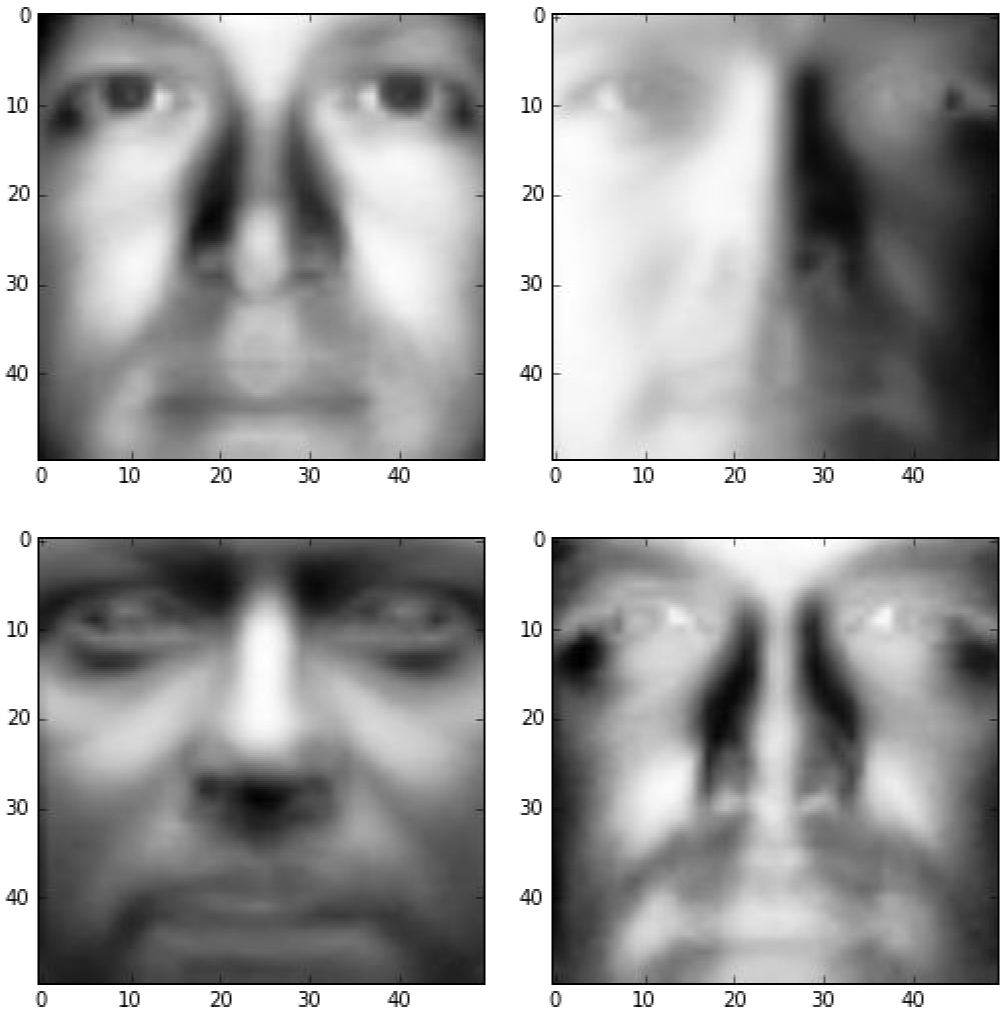


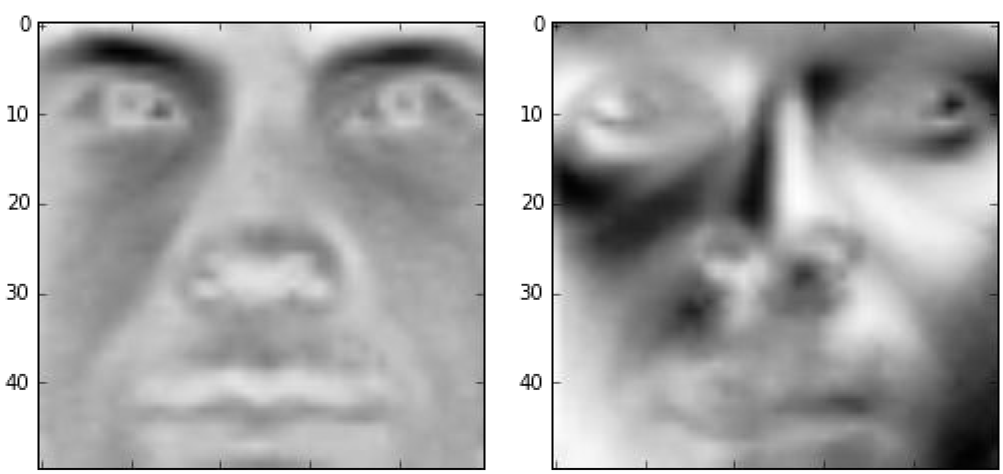
(d) Mean Subtraction. Subtract average face μ *from every column in X. That is* $x_i = x_i - \mu$*, where xi is the i -th column of X. Pick a face image after mean subtraction from the new X and display that image in grayscale. Do the same thing for the test set* X_{test} *using the precomputed average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

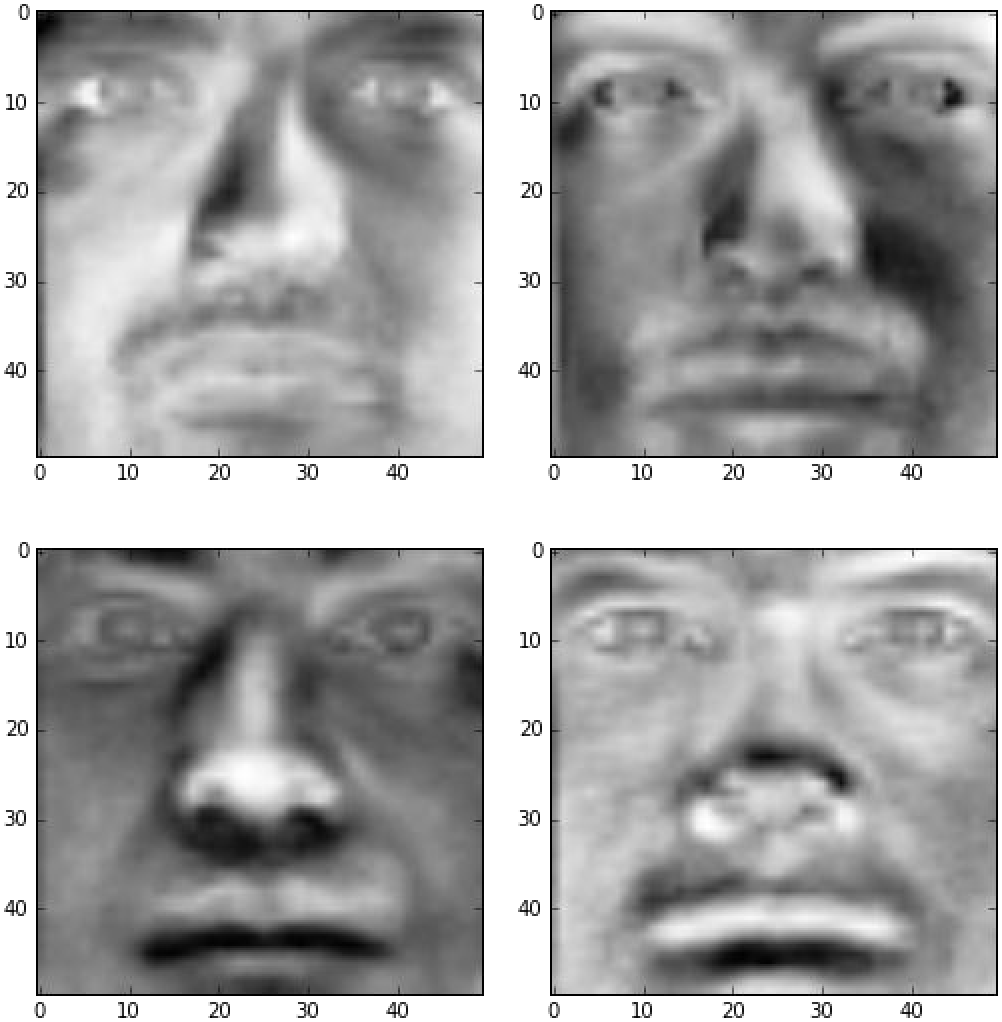
This is one of the face image we got after mean subtraction.



(e) Eigenface. Perform Singular Value Decomposition (SVD) on training set X to get matrix V^T*, where each row of* V^T *has the same dimension as the face image. We refer to* v_i*, the i -th row of* V^T*, as i -th eigenface. Display the first 10 eigenfaces as 10 images in grayscale*

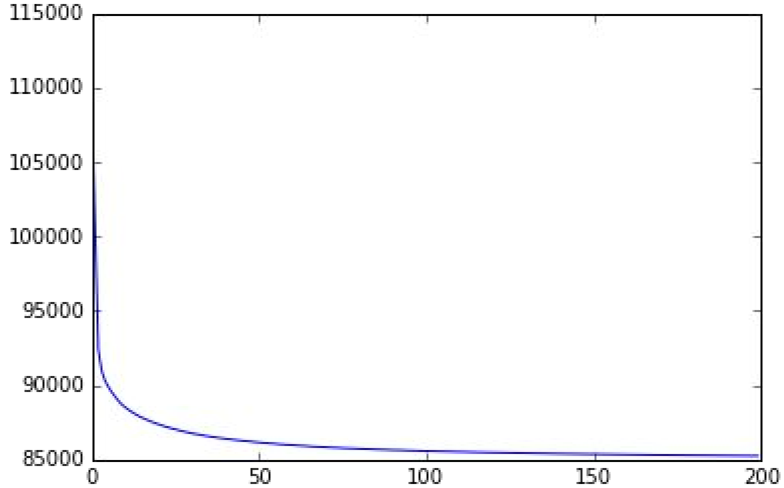

0
10
20
30
40
0 10 20 30 40
0
10
20
30
40
0 10 20 30 40
0
10
20
30
40
0 10 20 30 40
0
10
20
30
40
0 10 20 30 40


0
10
20
30
40
0 10 20 30 40
0
10
20
30
40
0 10 20 30 40



(f) Low-rank Approximation. Since Σ is a diagonal matrix with non-negative real numbers on the diagonal in non-ascending order, we can use the first r elements in Σ together with first r columns in U and first r rows in V^T to approximate X. That is, we can approximate X by Plot the rank-r approximation error as a function of r when r = 1, 2, . . . ,200.

We used the svd function in numpy to generated the svd matrix. And reconstruct the matrix with different value of R. Based on this, we got a graph as below which indicates that as the r increases, the rank-r approximation error become lower and lower.

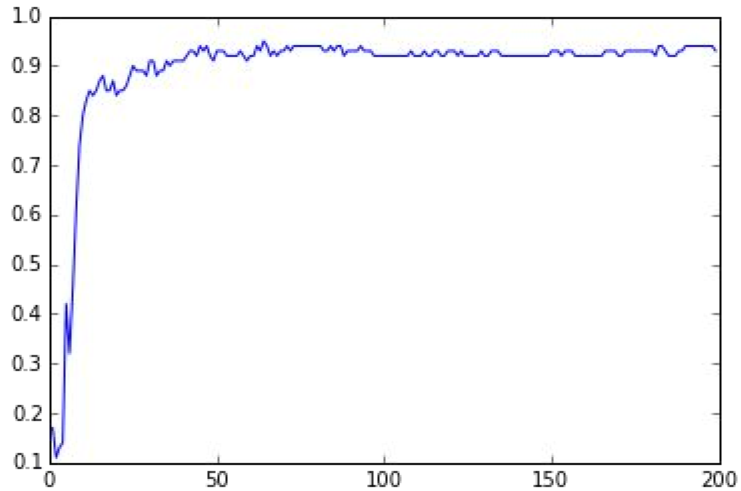


(g) Eigenface Feature. The top r eigenfaces span an r-dimensional linear subspace of the original image space called face space, whose origin is the average face μ, and whose axes are the eigenfaces {v1, v2, . . . , vr }. Therefore, using the top r eigenfaces {v1, v2, . . . , vr }, we can represent a 2500-dimensional face image z as an r-dimensional feature vector f. 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).

The function can be found in HW2_PART2.ipynb.

(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.

This is the classification accuracy based on r. We can see that the accuracy become stable at 95% when r reach 50 and later.



2.What’s cooking?

(a) Join the What’s Cooking competition on Kaggle. Download the training and test data (in .json). The competition page describes how these files are formatted.

downloaded the training set and test set.

(b) Tell us about the data. How many samples 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?

There are 39774 samples in training set. It includes 20 types of different cuisines and these cuisines are made of 6714 kinds of unique ingredients.

*(c) Represent each cuisine by a binary ingredient feature vector. Suppose there are d different ingredients in total, represent each cuisine by a 1*d binary ingredient vector x, where* x_i *= 1 if the cuisine contains ingredient i and xi = 0 otherwise.Use n*d feature matrix to represent all the dishes in training set and test set, where n is the number of cuisines.*

We used in1d function in numpy to construct the feature matrix. This function will help to test whether each element of a 1-D array is also present in a second array and return a bool matrix. We found this function work faster than implemented the function by ourself. Finally, we got a 39774*6714 matrix for training set. And 9944*6714 matrix for test set.

(d) Using Naïve Bayes Classifier to perform3 fold cross-validation on the training set and report your average classification accuracy. Try both Gaussian distribution prior assumption and Bernoulli distribution prior assumption.

We used sklearn to implement naive bayes classification. Through 3 fold cross-validation, we get an average accuracy for gaussian naive bayes as 38% and 68% for bernoulli.

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

In our experiment, Bernoulli prior perform much better than Gaussian. In Gaussian prior, we have an assumption that the value of feature x is continuous and distributed in Gaussian model in each class. However, in this case, the features, which says ingredients is binary. So it doesn't fit to Gaussian distribution. That's why Gaussian prior perform poor in this case. On the other hand, bernoulli distribution is binary which suits to this case.

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

The average of 3-fold-cross-validation of logistic regression model is 77.5%.

(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.

By submitting our result to Kaggle. We got an accuracy of 78.319%. Our team name on kaggle is Yeehan.

3.Written Exercise

1.HTF Exercise 4.1 (From Generalized to Standard Eigenvalue Problem)

$$\max_{a} \left(\frac{a^T B a}{a^T W a} \right)$$

$$\lambda = \frac{a^T B a}{a^T W a}$$

$$\lambda' = \frac{2Ba - 2\lambda W a}{a^T W a} = 0$$

$$Ba - \lambda W a = 0$$

$$Ba = \lambda W a$$

$$\boxed{W^{-1} B a = \lambda a}$$

standard eigen value

HTF Exercise 4.2

(a) Show that the LDA rule classifies to class 2 if

HTF Exercise 4.2

Show that the LDA rule classifies to class ....

In the two class case, we classify to 2 if

$$\delta_1(x) < \delta_2(x)$$

Linear discriminant function:

$$\delta_j(x) = x^T \Sigma^{-1} \mu_j - 1/2\, \mu_j^T \Sigma^{-1} \mu_j + \log \pi_j$$

$$\delta_1(x) = x^T \Sigma^{-1} \mu_1 - 1/2\, \mu_1^T \Sigma^{-1} \mu_1 + \log \pi N_1$$

$$\delta_2(x) = x^T \Sigma^{-1} \mu_2 - 1/2\, \mu_2^T \Sigma^{-1} \mu_2 + \log \pi N_2 .$$

On substitution we have

$$x^T \Sigma^{-1} \mu_1 - 1/2\, \mu_1^T \Sigma^{-1} \mu_1 + \log \pi N_1$$

$$> x^T \Sigma^{-1} \mu_2 - 1/2\, \mu_2^T \Sigma^{-1} \mu_2 + \log \pi N_2$$

$$x^T \Sigma^{-1} (\mu_2 - \mu_1) > \frac{1}{2} \mu_2^T \Sigma^{-1} \mu_2 - 1/2\, \mu_1^T \Sigma^{-1} \mu_1 + \log \frac{N_1}{N} - \log\left(\frac{N_2}{N}\right)$$

target coded

and class 1 otherwise.

(b) Consider minimization of the least squares criterion

HTF Exercise 4.2

b) Consider minimization of the least squares criterion

Let u_i be the n element vector with j^{th} element 1 if the j^{th} observation is class i else zero

$$Y = t_1 u_1 + t_2 u_2$$

$$1 = u_1 + u_2$$

$$X^T u_i = N_i \mu_i \qquad X^T Y = t_1 N_1 \mu_1 + t_2 N_2 \mu_2$$

By LS criterion

$$RSS = \sum_{i=1}^{N} (y_i - \beta_0 - \beta^T x)^2$$

$$= (Y - \beta_0 1 - X\beta)^T (Y - \beta_0 1 - X\beta)$$

On minimizing with respect to β and β_0

$$2X^T X\beta - 2X^T Y + 2\beta_0 X^T 1 = 0$$

$$2N\beta_0 - 2 1^T (Y - X\beta) = 0$$

Solving for β_0 and β by substitution

$$\beta_0 = \frac{1}{N} 1^T (Y - X\beta)$$

$$\left(X^T X - \frac{1}{N} X^T 1 1^T X\right)\beta = X^T Y - \frac{1}{N} X^T 1 1^T Y$$

Right hand can be written as

$$X^T Y - \frac{1}{N} X^T 1 1^T Y = t_1 N_1 \mu_1 + t_2 N_2 \mu_2 - \frac{1}{N}(N_1\mu_1 + N_2\mu_2)(t_1 N_1 + t_2 N_2)$$

$$= \frac{N_1 N_2}{N}(t_1 - t_2)(\mu_1 - \mu_2)$$

by using $1 = u_1 + u_2$

Now for LHS

term for β

$$X^T X - \frac{1}{N} X^T 1 1^T X$$

can be rewritten as

$$X^T X = (N-2)\Sigma + N_1 \mu_1 \mu_1^T + N_2 \mu_2 \mu_2^T$$

Σ_B definition can be written as

$$X^T X - \frac{1}{N} X^T 1 1^T X = (N-2)\Sigma + \frac{N_1 N_2}{N} \Sigma_B$$

putting altogether

$$\left((N-2)\Sigma + \frac{N_1 N_2}{N} \Sigma_B\right)\beta = \frac{N_1 N_2}{N}(t_1 - t_2)(\mu_1 - \mu_2)$$

substituting $t_1 = -N/N_1$, $t_2 = N/N_2$

$$\left((N-2)\Sigma + \frac{N_1 N_2}{N} \Sigma_B\right)\beta = N(\mu_2 - \mu_1)$$

(c) Hence show that ˆ§Bβ is in the direction (ˆµ2 − ˆµ1) ………..

$$\hat{\beta} \propto \Sigma^{-1}(\mu_2 - \mu_1) .$$

Let $\alpha = (\mu_2 - \mu_1)^T \beta \in \mathbb{R}$

$$\Sigma_B \beta = (\mu_2 - \mu_1)(\mu_2 - \mu_1)\beta$$

$$= (\mu_2 - \mu_1)\alpha$$

is in the direction of $(\mu_2 - \mu_1)$

By using

$$\left((N-2)\Sigma + \frac{N_1 N_2}{N}\Sigma_B\right)\beta = N(\mu_2 - \mu_1) .$$

$$(N-2)\Sigma\beta = N(\mu_2 - \mu_1) - \frac{N_1 N_2}{N}\Sigma_B \beta$$

$$= \mu_2 - \mu_1 \left[N - \frac{N_1 N_2}{N}\alpha\right]$$

Such that $\beta \propto \Sigma^{-1}(\mu_2 - \mu_1)$.

(d) Show that this result holds for any (distinct) coding of the two classes.

if the vector Y is shifted by a constant regression fit would likewise shift by shifting B_0 by that same constant, while leaving the rest of β unchanged.

$\bar{y} = 0$ and y is proportional to the y vector.

$$\sum_{i=1}^{N} x_i y_i = \sum_{k=1}^{2} \sum_{g_i=k} x_i y_i$$

$$= \sum_{k=1}^{2} \sum_{g_i=k} x_i (-1)^k N/N_k \alpha \propto \frac{N}{N_2} \sum_{g_i=2} x_i - \frac{N}{N_1} \sum_{g_i=1} x_i$$

$$= N(\mu_2 - \mu_1)$$

such that equation

$$\left[(N-2)\Sigma + \frac{N_1 N_2}{N} \Sigma_B\right] \beta = N(\mu_2 - \mu_1)$$

still holds (up to a scalar). and the result is still true.

(e) Find the solution ˆ β0, and hence the predicted values ˆ f = ˆ β0 + ˆ βT x.
Consider the following rule:...

(e)

Let $\hat{f} = \beta_0 + \beta^T x$

where

$$\beta_0 = -\bar{x}^T \beta$$

$$\hat{\beta} = y \Sigma^{-1} (\mu_2 - \mu_1)$$

$$f = -1/N \, (N_1 \mu_1 + N_2 \mu_2)^T y \Sigma^{-1} (\mu_2 - \mu_1) + y (\mu_2 - \mu_1)^T \Sigma^{-1} x$$

which classifies to class 2 if $f > 0$

$$y (\mu_2 - \mu_1)^T \Sigma^{-1} x > \frac{1}{N} (N_1 \mu_1 + N_2 \mu_2)^T y \Sigma^{-1} (\mu_2 - \mu_1)$$

$$= \frac{y}{N} \left[N_1 \mu_1^T \Sigma^{-1} \mu_2 + N_2 \mu_2^T \Sigma^{-1} \mu_2 - N_1 \mu_1^T \Sigma^{-1} \mu_1 - N_2 \mu_2^T \Sigma^{-1} \mu_1 \right]$$

if $N_1 = N_2$ then

$$(\mu_2 - \mu_1)^T \Sigma^{-1} x > \frac{N_1}{N} \left[\mu_2^T \Sigma^{-1} \mu_2 - \mu_1^T \Sigma^{-1} \mu_1 \right]$$

This is not the same as LDA rule unless the classes have equal number of observations.

..

3. RLU Exercise 11.3.1 (SVD of Rank DeficientMatrix)
(a) Compute the matrices MTM and MMT .
MM^T =[[14 26 22 16 22]
[26 50 46 28 40]

[22 46 50 20 32]
[16 28 20 20 26]
[22 40 32 26 35]]

$M^T M =$ [[36 37 38]
[37 49 61]
[38 61 84]]

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

the eigenvalues for MM^T is [0 0 0 15 154]

and the eigenvalues for $M^T M$ is [0 15 154]

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

U= [[-0. -1. -1. -0. -0.]
[0. -0. -1. 1. 0.]
[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0.]]

V= [[-0. -1. -1.]
[-1. -0. 1.]
[0. 0. 0.]]

(d) Find the SVD for the original matrix M from parts (b) and (c). Note that there are only two nonzero eigenvalues, so your matrix Σ should have only two singular values, while U and V have only two columns.

D = [[12. -0. 0.]
[-0. 4. 0.]
[0. 0. 0.]
[0. 0. 0.]
[0. 0. 0.]]

U= [[-0. -1. -1. -0. -0.]
[0. -0. -1. 1. 0.]
[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0.]
[0. 0. 0. 0. 0.]]

V= [[-0. -1. -1.]
[-1. -0. 1.]

[0. 0. 0.]]

(e) Set your smaller singular value to 0 and compute the one-dimensional approximation to the matrix M from Fig. 11.11.

We can use Frobenius Norm to compute the approximation.

Reference :

1)Standard class notes.

2)Elements of statistical learning book

3) MIT open source matrix lecture.