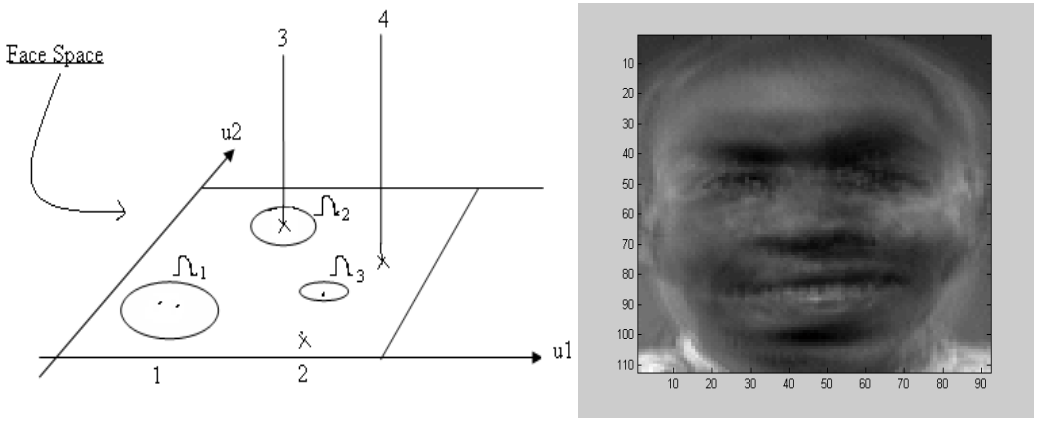
ECE 661: Homework 10
Fall 2014

This homework consists of the following two parts: (1) Face recognition with PCA and LDA for dimensionality reduction and the nearest-neighborhood rule for classification; and (2) Object detection with the cascaded AdaBoost classifier.

Part 1: Face Recognition using PCA/LDA "face space" and NN classification

1. Introduction

For the images that we are dealing with, the dimensionality tends to be very high. Let a face image I(x,y) be a two-dimensional *N* by *N* array of intensity values. An image may also be considered as a vector of dimension N^2, so that a typical image of size 128 by 128 becomes a vector of dimension 2^{28}, or equivalently, a point in 2^{28}-dimensional space. An ensemble of images, then, maps to a collection of points in this huge space. Images of faces, being similar in overall configuration, will not be randomly distributed in this huge image space and thus can be described by a relatively low dimensional subspace. The main idea of the PCA/LDA analysis is to find the vector that best account for the distribution of face images within the entire image space. These vectors define the subspace of face images, which we call "face space" (see Figure 1). Each vector is of length N^2, describes an *N* by *N* image, and is a linear combination of the original face images. Because these vectors are the eigenvectors of the covariance matrix corresponding to the original face images, and because they are face-like in appearance, they are referred to as "Eigen Faces" for PCA and "Fisher Faces" for LDA.



(a) (b)

Figure 1. Illustration of reduced dimensional face space
(a) The face space and the three projected images on it. Here u1 and u2 are the eigenfaces. (b) The projected face from the training database.

The goal of this part is to classify an unknown face image given a database of labeled face images using two different approaches, i.e. "Principle Component Analysis (PCA)" and "Linear Discriminant Analysis (LDA)", to form a data subspace with reduced dimensions.

2. Method of Solution

2.1 PCA Eigen Faces Recognition

In PCA, one would use the eigenvectors corresponding to the p largest eigenvalues of the covariance matrix to span the subspace. One needs to calculate the covariance matrix:

$$C = \frac{1}{N}\sum_{i=1}^{N}(x_i - m)(x_i - m)^T$$

where m is the mean vector of all training images $x_i$and N is the number of training samples. The next step is to do the eigen-decomposition to get the eigenvectors and eigenvalues of the covariance matrix. If we want to construct a *p*-dimension subspace, we should use the eigenvectors corresponding to the *p* largest eigenvalues.

However, since the dimension of covariance is very high, we need an algebraic trick to compute the eigenvalues and eigenvectors. Let

$$X = [x_1 - m \quad x_2 - m \quad \cdots \quad x_N - m]$$

$$C = \frac{1}{N}XX^T$$

What we need to compute is $XX^T u = \lambda u$, however XX^T is huge. Instead we compute $X^T X v = \lambda' v$ where $X^T X$ is N by N. Then $\lambda = \lambda'$ and $u = Xv$, the p largest u are used as the new bases that span the subspace.

Figure 2 shows the results of the mean face and top 8 eigenfaces using PLA and NN Neighbor classification. It can be found that instead of converting all images to gray level images, we keep the rgb color value which gives us more feature information than gray images only.

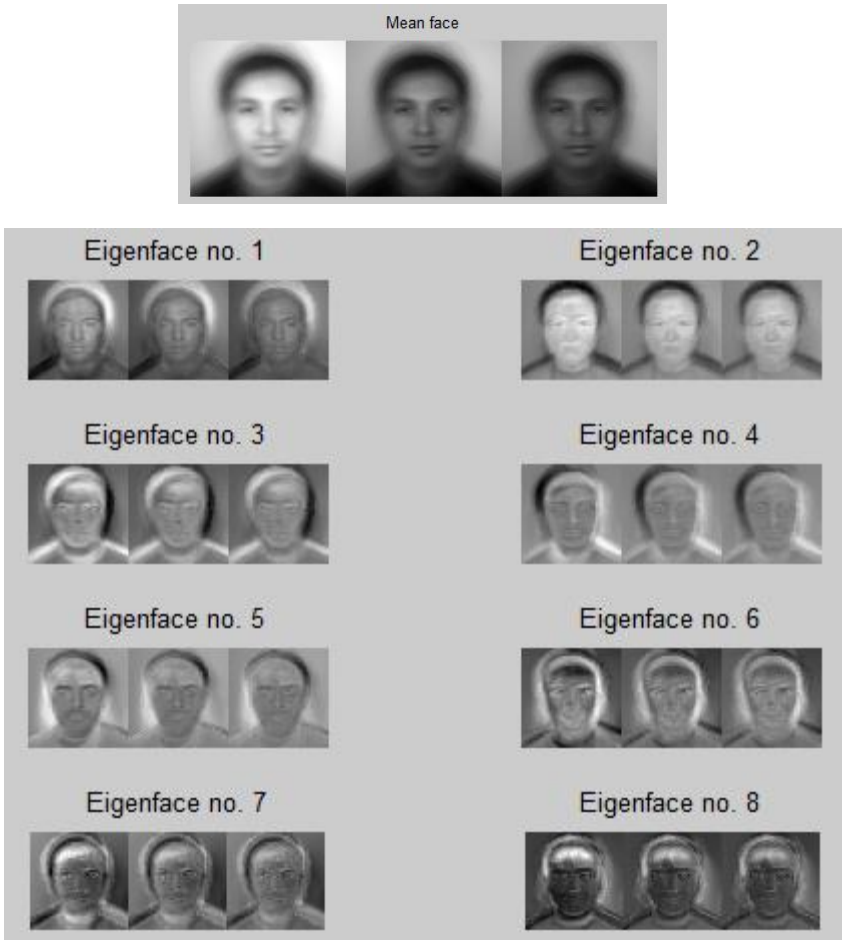


Figure 2. Mean face and top 8 PCA eigenfaces

2.2 LDA Fisher Faces Recognition

For LDA, the main idea is to find the p-dimension subspace that maximize the distance between the sets of classes and minimize the distance within each set of the classes. The between class covariance matrix, S_B and within class covariance matrix, S_W, are defined as the following

$$S_B = \frac{1}{N_C}\sum_{j=1}^{N_C}(m_j - m)(m_j - m)^T$$

where m_j is the mean of class j, and m is the mean of all training data, N_C is the number of classes.

$$S_W = \frac{1}{N_C}\sum_{j=1}^{N_C}(\frac{1}{N_j}\sum_{i=1}^{N_j}(x_{ij} - m_j)\,(x_{ij} - m_j)^T)$$

where N_j is the number of elements of class j, x_{ij} is the ith element of class j. In order

to find the p-dimensional subspace on which the images projects results in maximizing the between-class variance and minimizing the within-class variance, we want to maximize the fisher discriminant function defined as the following.

$$J(w) = \frac{w^T S_B w}{w^T S_W w}$$

which can be converted to the equivalent problem of solving the following equation.

$$S_B w = \lambda S_W w$$

I tried to use the method that professor told in class, which is finding eigenvectors $S_B w' = \lambda w'$, then compute $w = S_W^{-1} w'$. However, we know that S_W is huge, and computing its inverse is nearly impossible. So there is no way but to use Yu and Yang's algorithm to directly solve the eigen-decomposition problem.

i) Using the same trick to get the eigenvectors , eigenvalues of S_B

$$S_B = Y^T D_B Y$$

ii) Discard the smallest eigenvector and eigenvalue of from Y and D_B

iii) Construct Z as

$$Z = Y D_B^{-\frac{1}{2}}$$

iv) Define matrix

$$S_{BW} = Z^T S_W Z$$

v) Using the same trick to get the eigenvectors, eigenvalues of S_{BW}

$$S_{BW} = U^T D_W U$$

vi) Sort the eigenvectors in the increasing order.

vii) Define matrix $W = ZU$

viii) The bases are the first p vectors of W

Figure 3 shows the results of the mean face and top 8 eigenfaces using LDA and NN Neighbor classification. It can be found that instead of converting all images to gray level images, we keep the rgb color value which gives us more feature information than gray images only.

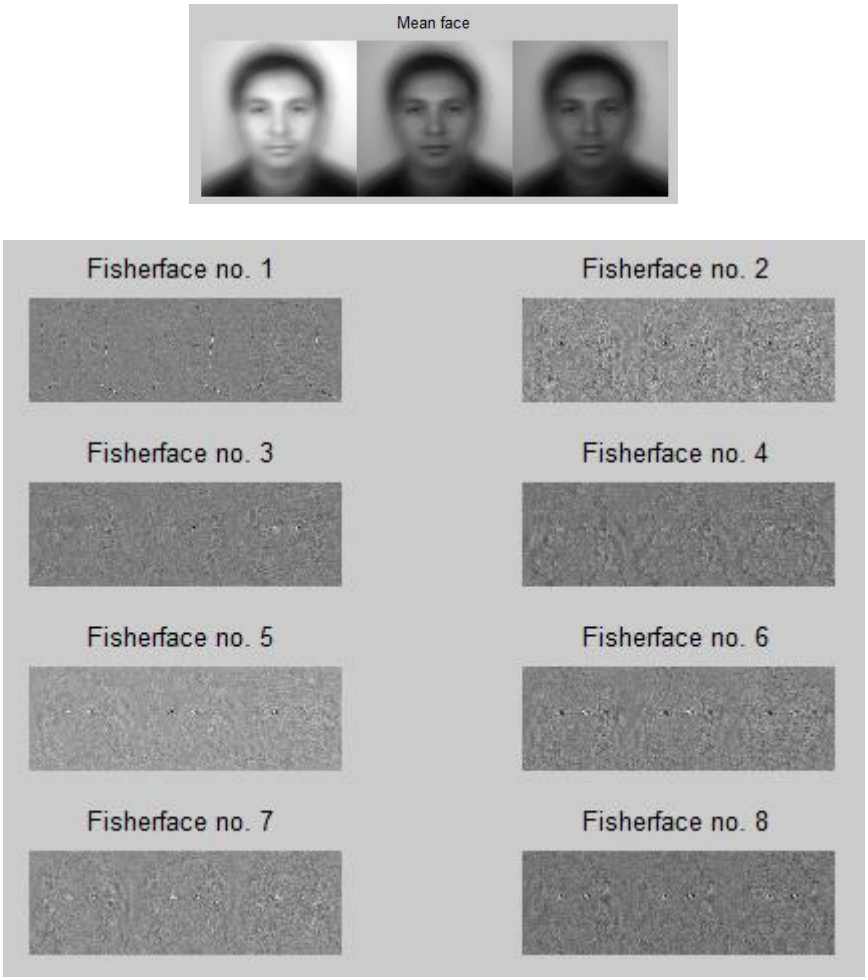


Figure 3. Mean face and top 8 LDA fisherfaces

2.3 Nearest Neighbor Classification

After getting the p bases of the subspace, all training images and testing images are projected onto the subspace. The similarity score based on cosine mahalanobis distance is calculated between an input face image and each of the training images.

Defining the accuracy of classification as:

$$accuracy = \frac{\#\ of\ corrected\ classified\ testing\ samples}{\#\ of\ testing\ samples}$$

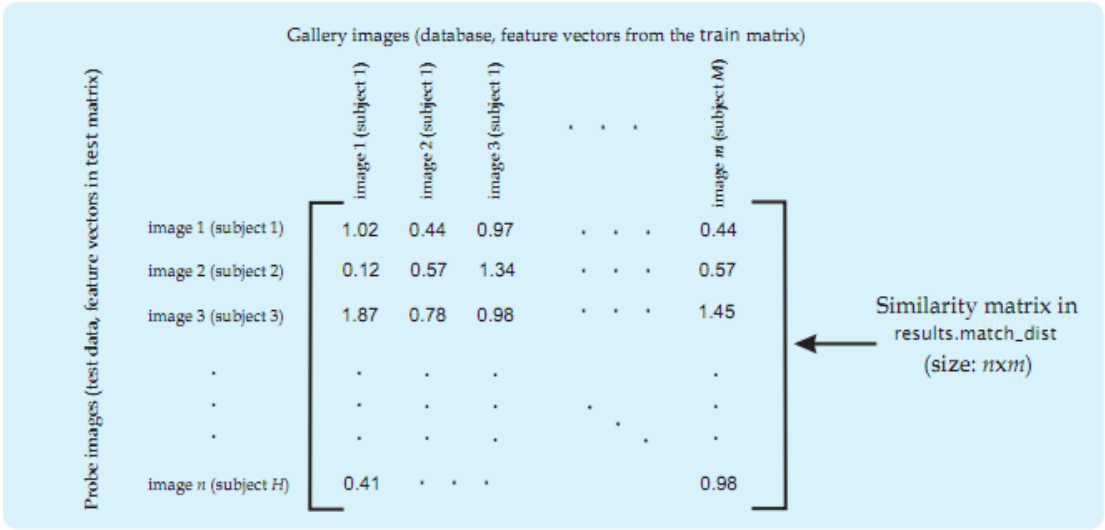


Figure 4. An example of the similarity matrix

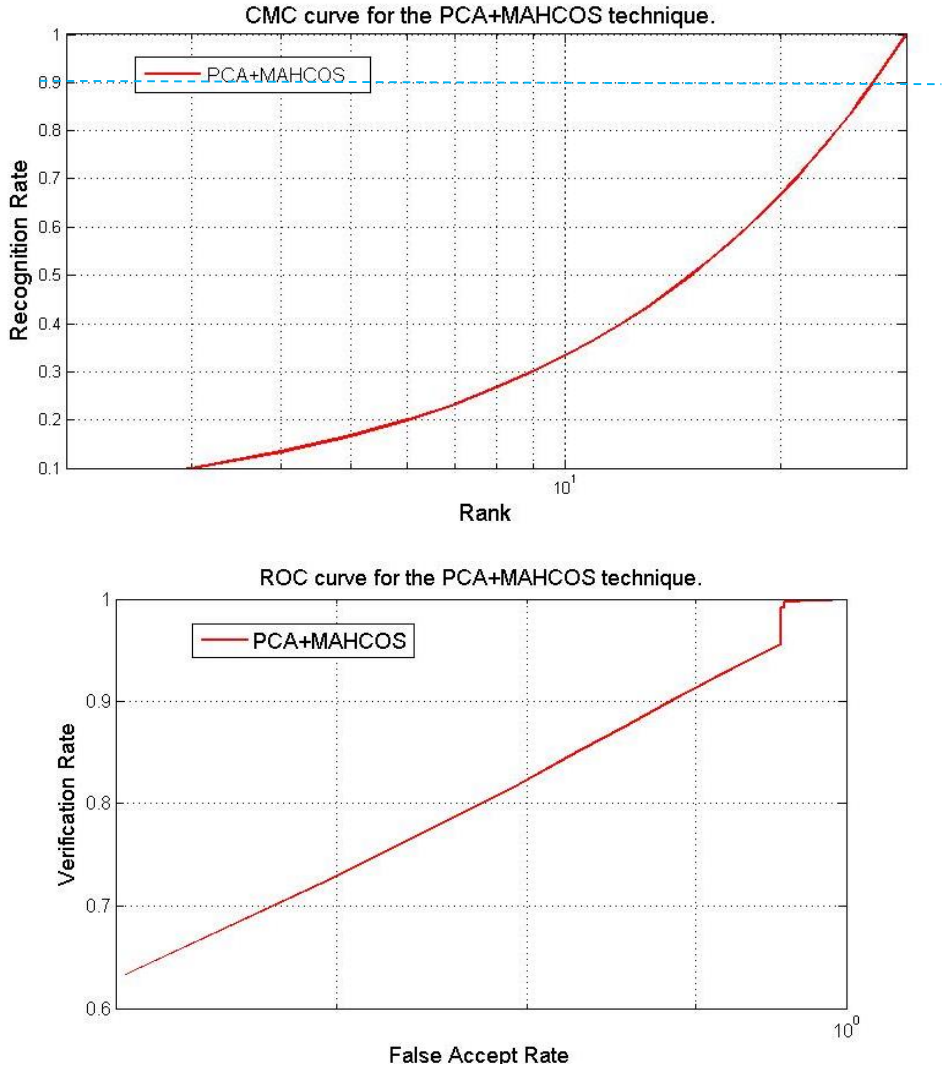


Figure 5. ROC and CMC curves for PCA+MAHCOS

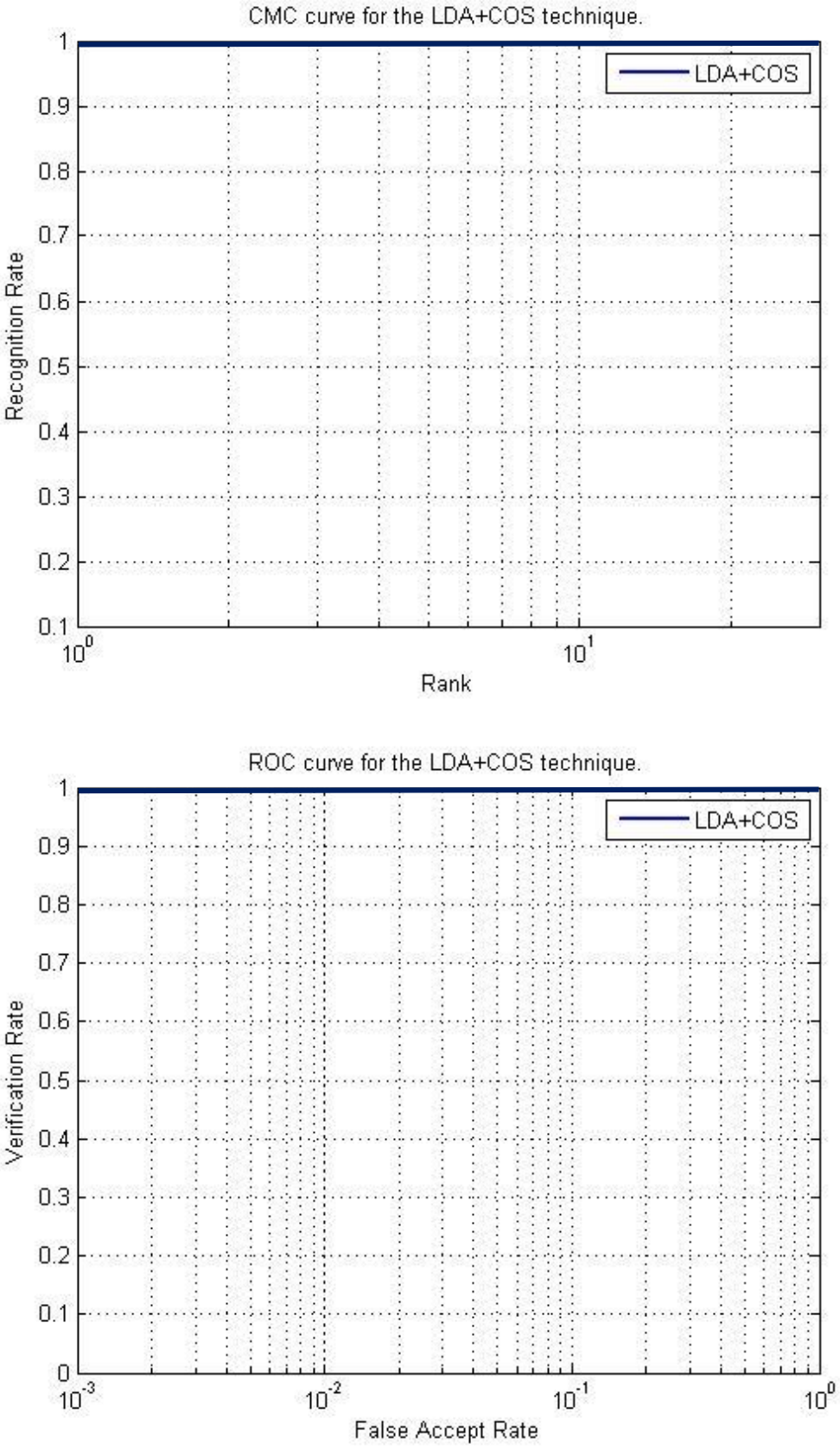


Figure 6. ROC and CMC curves for LDA+MAHCOS

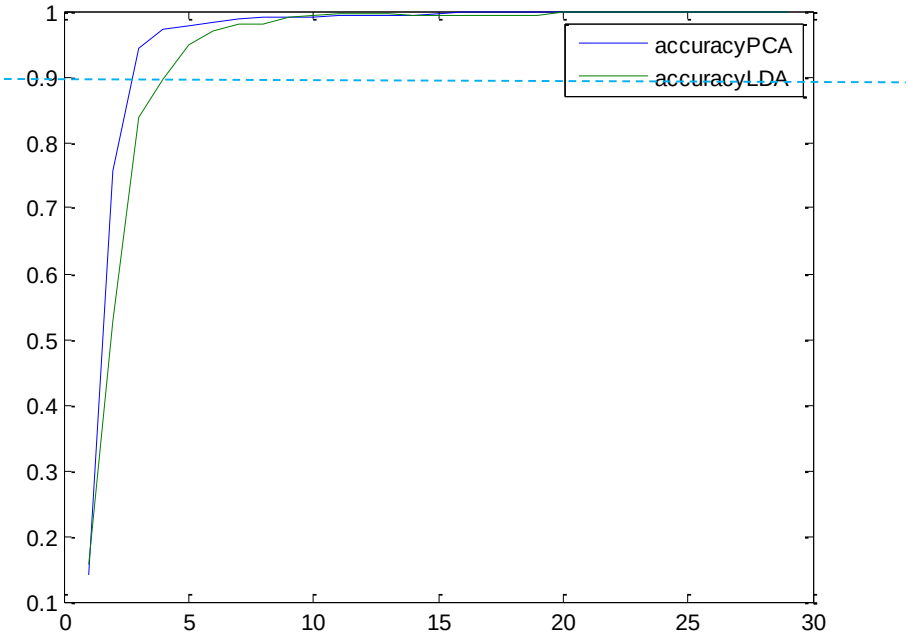


Figure 7. PCA/LDA accuracy versus subspace dimensions using gray images only

Table 1. Some performance metrics

	rank one recognition rate	equal error rate	minimal half total error rate	verification rate at 1% FAR
PCA	3.33%	48.54%	48.04%	63.33%
LDA	100%	0%	0%	100%

2.4 Discussion

(1) In this homework, we use R,G,B information instead of gray image only. The merit is more useful information are kept to help improve the accuracy of face detection. By comparing my results with the results in previous homework which used gray image only (see Figure 5,6,7), I found that it is more efficient and accurate using more useful information for classification, however, the cost is as the features are tripled, the time efficiency is relatively lower. This challenge can be solved by using parallel computing, such as MPI or OpenMP in the future.

(2) During feature matching process, Cosine Mahalanobis Distance, rather than Euclidean Distance, was adopted for Nearest neighbor classification. It is more intrinsic to measure similarity using Mahalanobis Distance, especially in abstract projected subspace, as the Euclidean Distance at this moment cannot represent the real differences

between two datasets.

(3) PCA v.s. LDA
By comparing the ROC and CMC curves (see Figure 5, 6 and Table 1), it can be found that LDA is much more efficient than PCA. The reason could be PCA does not separate the classes as well as LDA.

2.5 Source Codes (Matlab)
See attached zip file.

Part 2: Object Detection with Cascaded AdaBoost Classification

1. Introduction

This goal of this part is to design a car detector using the Viola and Jones approach. It is a two-class classification for each sliding window that scans through the image, i.e., cars or non-cars. The merit of using such approach is it can cascade multiple strong AdaBoost classifiers, each of which comes from several weak classifiers.

2. Method of Solution

Figure 8 illustrates the general outline for cascaded AdaBoost classification. It consists of four main modules, which will be explained step by step in the following.

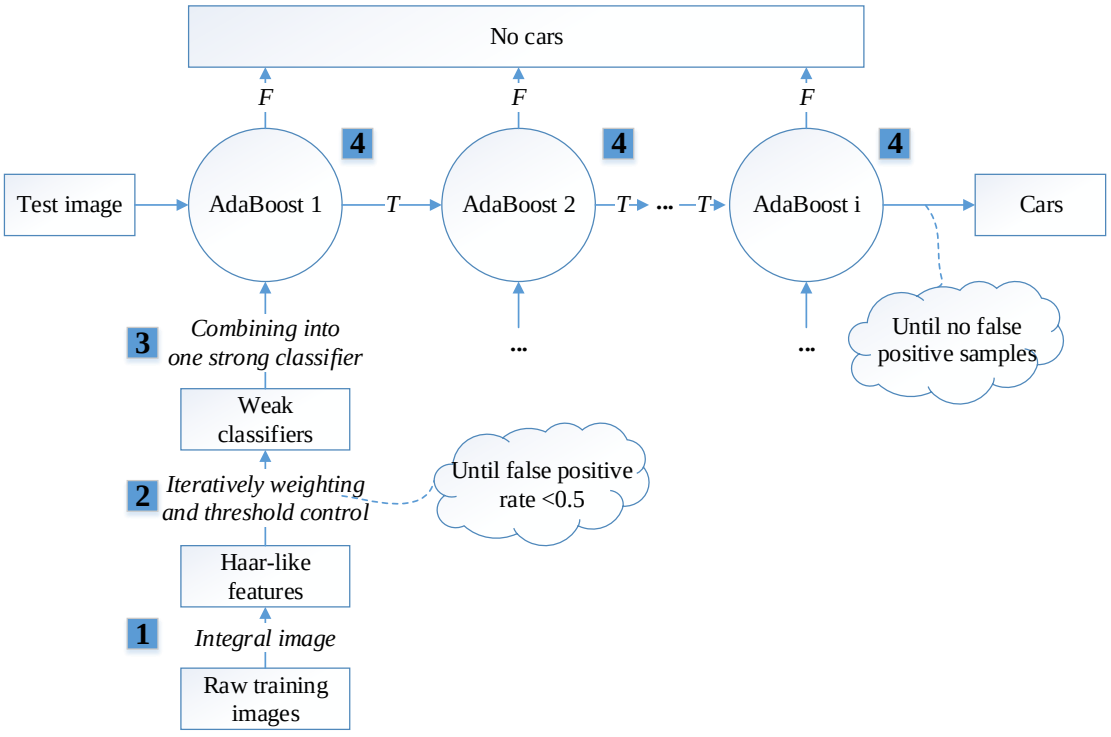


Figure 8. General process for Cascaded AdaBoost Classification

2.1. Computing the integral image (summed area table) for Haar-like features

In this project, we use Haar-like features consisting of two types, vertical ones and horizontal ones. In order to calculate feature values efficiently, integral image (or summed area table) is employed (see both equation and Figure 9 below).

$$S(i,j) = \sum_{x \le i} \sum_{y \le j} I(x,y)$$

where $I(x, y)$ is the grayscale image pixel value at position (x, y).

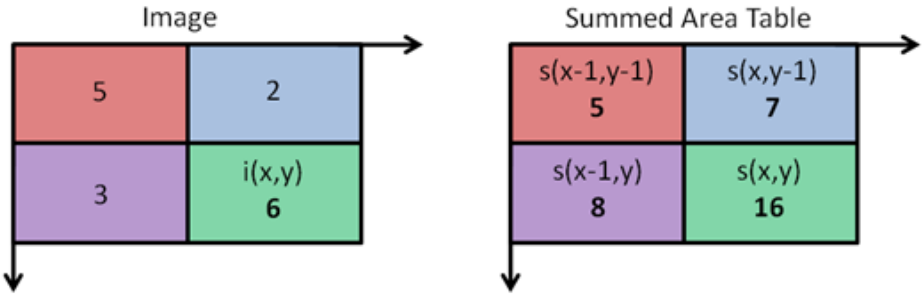


Figure 9. Illustration of how integral image works

Then the Haar-like features can be computed efficiently using a few values from summed area table.

$$f = -S(x_1, y_1) + S(x_2, y_2) + 2S(x_3, y_3) - 2S(x_4, y_4) - S(x_5, y_5) + S(x_6, y_6)$$

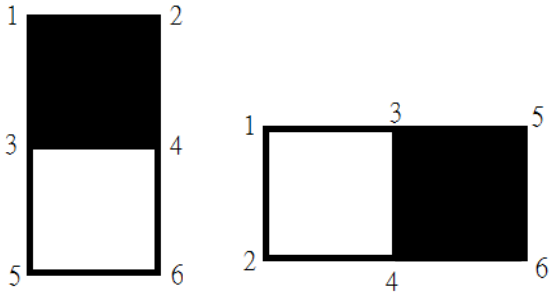


Figure 10. Haar-like edge features

2.2 Constituting weak classifiers by iteratively weighting and threshold control

For each feature, the images are sorted according to their feature values in ascending order. Computing the following number for each threshold:

$$e = \min(S^+ + (T^- - S^-), S^- + (T^+ - S^+))$$

where S^+ and S^- are the sums of the weights of positive and negative samples from the smallest one to current threshold respectively. T^+ and T^- are the sums of all positive and negative samples respectively. If the former is less than the later, the decision rule is the following "if feature value is less than the threshold, the data point is classified as positive" and we say the polarity is 1.

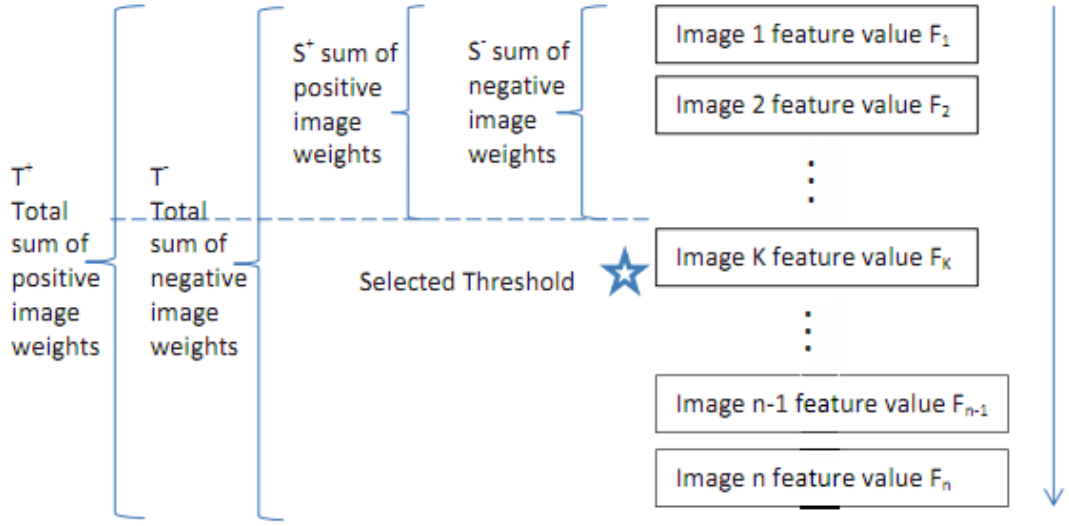


Figure 11. Weighting linear combination of T hypotheses and threshold control

The algorithm shown in Figure 11 can be described as following:
T hypotheses are constructed each using a single feature. The final hypothesis is a weighted linear combination of the T hypotheses where the weights are inversely proportional to the training errors.

- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0,1$ for negative and positive examples respectively. n are the total training set number.
- Initialize weights $w_{1,j}=\frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
- For t=1,…,T:
  1) Normalize the weight, $\varpi_{t,i}= \frac{\varpi_{t,i}}{\sum_{j=1}^{n} \varpi_{t,i}}$
  2) Select the best weak classifier with respect to the weighted error
  $$\varepsilon_i = min_{f,p,\theta} \sum_i \varpi_i |h(x_i, f, p, \theta) - y_i|$$
  3) Define $h_t(x)= h(x, f_i, p_t, \theta_t)$, where f_t, p_t and θ_t are the minimizers of ε_t
  4) Update the weights:
  $$\mathcal{W}_{t+1,i} \mathcal{W}_{t,i} \beta_t^{1-e_i}$$ Where $e_i=0$ if example x_i is classified correctly, $e_i=1$ otherwise, and
  $$\beta_t = \frac{\varepsilon_i}{1-\varepsilon_i}$$

2.3 Combining weak classifiers to a single AdaBoost strong classifier

The decision of whether a sample is positive or negative is determined by the following:

$$C(x) = \begin{cases} 1 \; if \sum_{t=1}^{T} \alpha_t h_t(x) \geq threshold \\ 0 \qquad\qquad otherwise \end{cases}$$

Note that the threshold is chosen to be the minimum of $\sum_{t=1}^{T} \alpha_t h_t(x)$ for all positive training data being classified as positive. The procedure of finding best features continues until the false positive rate is less than 0.5 (see Figure 12).

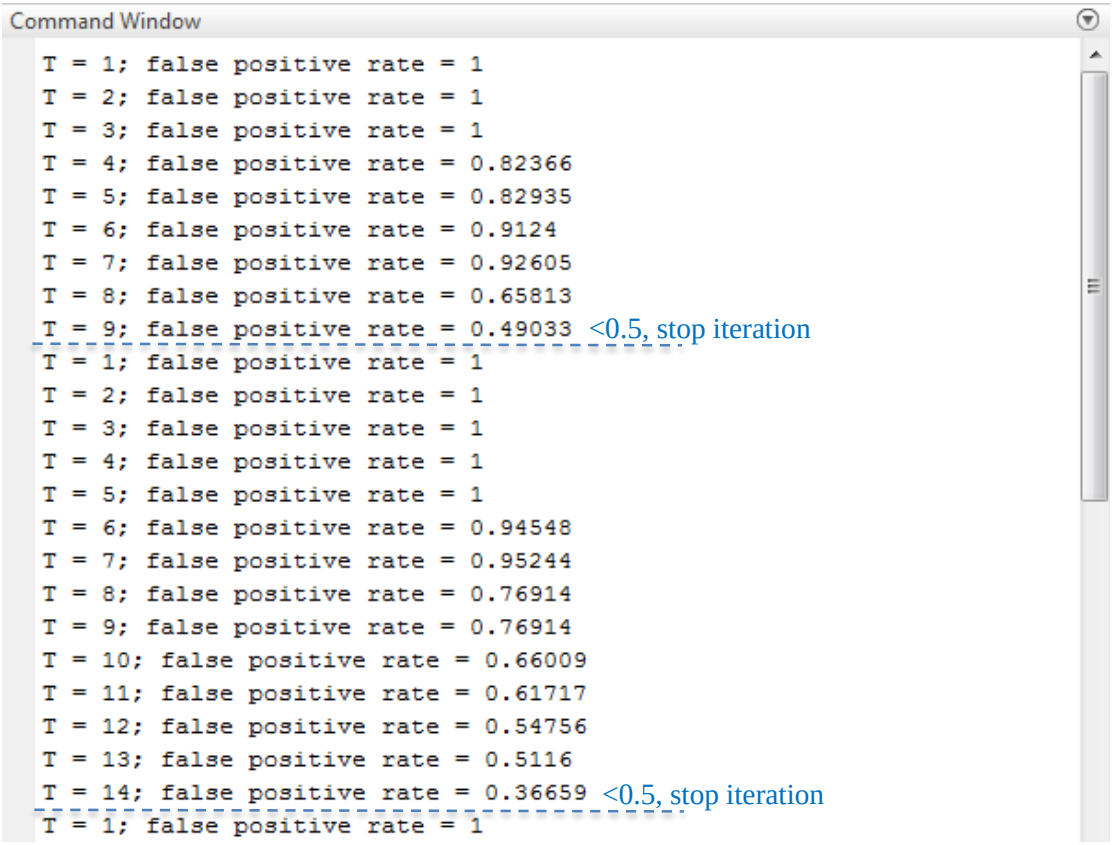


Figure 12. Output of the iteratively features selection process

2.4 Cascading AdaBoost classifiers for car detection

The purpose of cascading classifiers is to lower the false positive rate. We feed all samples that have been classified as positive to the next classifier. By adding more stages, there will be less negative sample being classified as positive. We stop adding stages when all negative samples are correctly classified (see Figure 8 module 4).

3. Testing Procedure

From the training process, we acquired all the parameters of the cascaded classifiers. Feed the query sample to the first classifier, if it is classified as negative, then it is done; if it is classified as positive, we feed this sample into the next classifier.

Table 2. Log of training process

Stage	Negative Sample # for next iteration	Feature # week classifier	False Positive Rate
1	862	9	0.4903
2	316	14	0.3666
3	141	7	0.4462
4	65	9	0.4610
5	26	10	0.4000
6	6	5	0.2308
7	1	3	0.3333
8	0	1	0

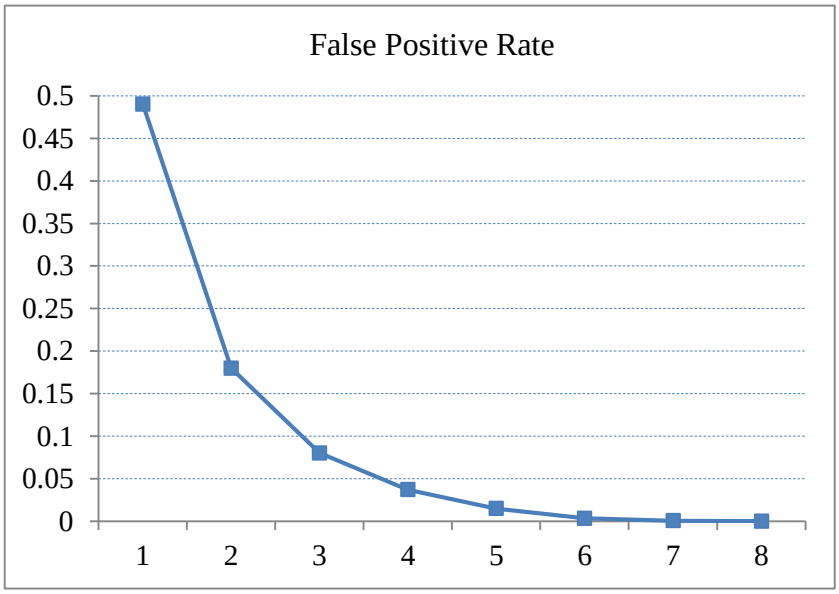


Figure 13. Accumulative false-positive rate for training process

Table 3. Log of testing process

Stage	False Positive #	False Negative #	Cumulative False Positive Rate	Cumulative False Negative Rate
1	209	4	0.4750	0.0225
2	87	11	0.1977	0.0843
3	83	1	0.1886	0.0899
4	74	4	0.1682	0.1124
5	52	20	0.1182	0.2247
6	35	12	0.0796	0.2921
7	35	0	0.0796	0.2921
8	209	0	0.0796	0.2921

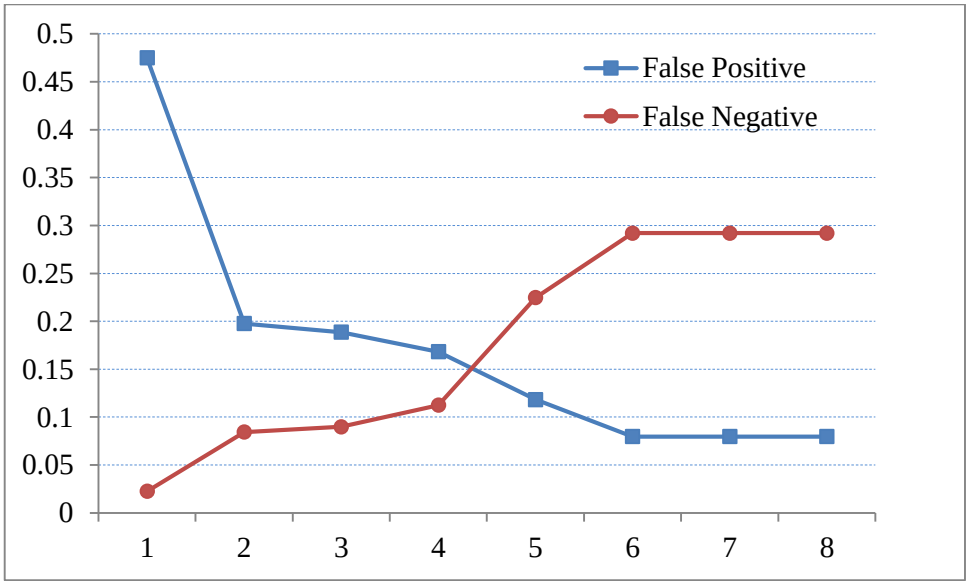


Figure 14. Accumulative false-positive and false-negative rate for test process

4. Source Codes (Matlab)

See attached zip code, it was adapted based on the previous codes of student Sirui Hu and Chyuan-Tyng Wu.