CS 479/679 Pattern Recognition
Spring 2025 – Prof. Bebis
Programming Assignment 2 – Due on 4/14/2025 at 11:59pm

Experiment 1: In assignment #1, you designed a Bayes classifier assuming the following 2D Gaussian densities:

$$\mu_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \Sigma_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \mu_2 = \begin{bmatrix} 4 \\ 4 \end{bmatrix} \quad \Sigma_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

In this assignment, we will assume that you do **not** know the **true** parameters of the Gaussian densities and that you would need to estimate them from the training data using the **Maximum Likelihood (ML)** approach.

a. Using **exactly** the same 200,000 samples from assignment #1 (i.e., 60,000 samples from $N(\mu_1,\Sigma_1)$ and 140,000 samples from $N(\mu_2,\Sigma_2)$), estimate the parameters of each distribution using the ML approach. Then, classify all 200,000 samples using a Bayes classifier, count the number of misclassifications (for each class and overall), and compare your results with those obtained in assignment #1.

b. Next, you will test how the number of training data affects parameter estimation and consequently, classification accuracy. For this, consider using only **(i)** 0.01%, **(ii)** 0.1%, **(iii)** 1%, and **(vi)** 10% of the samples from each density (**randomly** chosen) to estimate the parameters of the two densities using ML. Then, classify **all** 200,000 samples for each case, count the number of misclassified samples (for each class and overall), and compare your results with those obtained in (1.a).

   For example, in case **(iii)**, you need to estimate the parameters of $N(\mu_1,\Sigma_1)$ using 600 **randomly** chosen samples from the original 60,000 samples of $N(\mu_1,\Sigma_1)$ and the parameters of $N(\mu_2,\Sigma_2)$ using 1,400 randomly chosen samples from the original 140,000 samples of $N(\mu_2,\Sigma_2)$.Then, you need to classify the original 200,000 samples (60,000 samples from $N(\mu_1,\Sigma_1)$ and 140,000 samples from $N(\mu_2,\Sigma_2)$) using the estimated parameters from this case.

   Although the **true** covariance matrix for each class is the identity matrix in this problem, the estimated covariance matrices might **not** be equal or diagonal anymore. This implies that Case 1 might **not** strictly apply. One option is to choose the optimum Case based on the estimated covariance matrices since we do not really know the true covariance matrices in practice. Another option is to choose the optimum Case by explicitly setting the off-diagonal elements of the covariance matrices to zero **assuming that the features are uncorrelated**. Although such an assumption might not always be true in practice, it allows us to **reduce the number of parameters** (i.e., reduce model complexity) which can be beneficial when the number of training data is **small**. Experiment with both cases and analyze your results.

   Hint: Tabulate your results (i.e., estimated parameters and classification errors for each case) in the **same table** for easier comparison.

Experiment 2: Repeat experiment 1 using the 2D Gaussian densities below; for comparison purposes, use **exactly** the same 200,000 samples from assignment #1.

$$\mu_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \Sigma_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \mu_2 = \begin{bmatrix} 4 \\ 4 \end{bmatrix} \quad \Sigma_2 = \begin{bmatrix} 4 & 0 \\ 0 & 8 \end{bmatrix}$$

(Extra Credit 30%) Experiment 3: Face detection using **skin color** is a popular approach. While color images are typically in RGB format, most techniques transform RGB to a different color space (e.g., chromatic, HSV, etc.). This is because RGB values are more sensitive to variations in brightness due to illumination changes.

a. Implement the skin-color methodology of [Yang96 "A Real-time Face Tracker"] which uses the **chromatic color space**:

$$r = R / (R + G + B)$$
$$g = G / (R + G + B)$$

To build the skin color **model**, use *Training_1.ppm* (and *ref1.ppm*), shown in Figure 1, which are available from the course's webpage. To **test** your method, use *Training_3.ppm* (and *ref3.ppm*) and *Training_6.ppm* (and *ref6.ppm*), which are also available from the course's webpage. Note that each reference image provides the correct class (**face/skin** vs **non-face/non-skin**) for each pixel in the corresponding training/test images (e.g., non-black pixels in the reference images correspond to face/skin pixels in the training images). You would need to use the reference images to select the face/skin pixels for parameters estimation purposes but also to see how well your classifier works (i.e., by computing the FP/FN rates (i.e., FPR/FNR) for different thresholds in order to create the ROC curves as discussed below). As discussed in class, FPR=FP/(FP+TN)=FP/N and FNR=FN/(FN+TP)=FN/P.

By modeling the skin-color distribution using a multivariate Gaussian as we discussed in the lecture, you should be able to assign a likelihood g(**x**) to each pixel $\mathbf{x}=[r,g]^T$. If g(**x**)>**t**, where **t** is a threshold, then **x** is assigned to the skin-color class; otherwise; it is assigned to the non-skin color class. Please note that you are **not** supposed to use the discriminants derived in class since we considered modeling two classes while here we are only modeling one class (i.e., skin-color class). To quantitatively evaluate the performance of your method, generate **ROC** plots (i.e., FPR in the x-axis **vs** FNR in the y-axis) by varying the threshold **t**. To generate a reasonably smooth ROC curve, select 20 different thresholds in the interval [0, c] (i.e., uniformly distributed using a step=c/20) where c is the normalizing factor of the Gaussian function (i.e., $c=1/2\pi|\Sigma|^{1/2}$) which is the **max** value achieved by g(**x**) when **μ**=0). Note that **t** must in the interval [0, c] since it is being compared to g(**x**). A FP would be a non-face pixel which was classified as skin-color while a FN would be a face pixel which was classified as non-skin color.

Again, to determine whether a classification is correct or not (i.e., FP or FN), you would need to use the information provided in the reference images as mentioned above. In addition to the ROC plots, show the classification results on the test images using the threshold value that corresponds to the **Equal Error Rate** (ERR) (i.e., when FPR=FNR as shown in the lecture slides).

In showing the classified images, use the same convention as in Fig 5 from Yang96 paper shown below (i.e., use white (255,255,255) for pixels classified as non-skin and the original RGB value for pixels classified as skin). For visual comparison, show the corresponding reference images too next to the classified images.

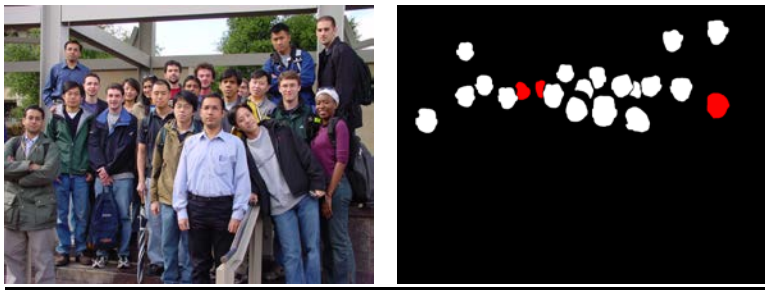
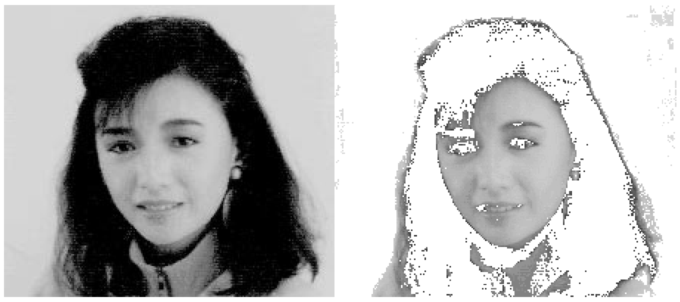


Figure 1. Training_1.ppm and *ref1.ppm* images.


(Fig 5 from Yang96 paper)

b. In this experiment, you will investigate the effect of using different features for classification. For this, you would need to repeat (3.a) using the **YC_bC_r color space.** The RGB components can be converted to the YC_bC_r components using the transformation below.

$$Y = 0.299R + 0.587G + 0.114B$$
$$C_b = -0.169R - 0.332G + 0.500B$$
$$C_r = 0.500R - 0.419G - 0.081B$$

It should be noted that in the YC_bC_r color space, the luminance information is captured in the Y component while the chrominance information is captured in the C_b and C_r components. Therefore, Y should not be used in the skin color model and it is only provided above for completeness. Therefore, you need to use a 2D Gaussian density again to model skin-color in this experiment based on the C_b and C_r components.

How do the chromatic and **YC_bC_r** color spaces compare? Hint: plot the ROC curves (FPR vs FNR) for each color space in the **same graph** for easier comparison.