CS 479/679: Pattern Recognition
Programming Assignment 2
Gunner Stone
Derek Stratton
We did approximately 50% of the coding and 50% of the writeup each.

I. Theory

A. General Theory

Maximum likelihood estimation is a technique to estimate the parameters of a distribution by determining the parameters that were most likely to generate a sample. The equation for a point estimate on the data D = $\{x_1, x_2 ... x_n\}$ is:

$$\hat{\boldsymbol{\theta}} = \arg\max_{\theta} \; p(D / \boldsymbol{\theta})$$

If the samples are assumed to be independent, then finding the maximum values of the log likelihood function is preferable since it is easier to find the critical values.

For Gaussian distributions, the ML estimate for μ is the sample mean and the ML estimate for Σ is the sample covariance. The estimate for μ is an unbiased estimator since the expected value of the estimate is equal to the parameter's true value. However, the estimated value for Σ is biased since the expected value of the estimate is not equal to Σ.

The accuracy of ML's point estimates are directly related to the size and quality of the samples. As the number of samples in a training set increases, it should become more representative of the true distribution, which yields better ML estimates.

B. Experiment 1 and 2 Theory

In the covariance matrix, the off-diagonal terms represent the covariance between the variables. When the covariance is 0, those variables are said to be uncorrelated. Treating variables as uncorrelated can help in simplifying the models used to classify variables. When estimating a covariance matrix from a sample, it's rare that the computed value of this is exactly 0, so considerations can be made of setting the covariance to 0 to simplify the model. This approach makes a lot of sense when the computed covariance is close to 0. While it can still possibly be beneficial when the covariance is not close to 0 since it simplifies the model, it might cause problems because it loses information when the variables are actually correlated.

C. Experiment 3 Theory

Gaussian discriminant functions return score values that vary based on how close the input fits the gaussian distribution. It is up to the designer of the classifier to determine what scores are considered 'in class' and what scores are considered 'out of class.' This determination is essentially a threshold that the score is compared to. If the score exceeds a threshold, it is considered in-class, if it falls below it is not considered in class. This threshold can be changed to allow for more leniency or strictness in your model. This desirability of lenience is completely up to the designer. But this freedom of choice comes at a cost. Too lenient, and your classifier will produce a lot of false positives (FP). Too strict, and your classifier will produce a lot of false negatives (FN).

This is where the ROC curve comes into consideration. ROC curves are strategies to help select desired threshold values. They essentially are plots of FP vs FN. This allows a designer to see how changing a threshold affects the leniency of the model. Experiment 3 asks for our classifiers to have equal error rate (ERR). This term describes selecting a threshold that makes

our classifier have an equal number of FP and FN. Examining the ROC curve of our Experiment 3 classifiers will help determine which threshold value to select to allow for ERR.

II. Implementation

A. Experiment 1 and 2 Implementation

To start, we read the data samples from the previous lab into an Eigen matrix. This matrix held all 200,000 values, and was used for every classification test. We also created a function that took this data matrix and returned a matrix that contained a subset of this matrix of specified size. To get this subset, a randomly chosen set of indices from each distribution was chosen so that the proportion of elements in the first and second distribution was always the same. These subset matrices were used in the different trials for calculating the parameters.

To calculate the parameters, we created functions that computed the sample mean and sample covariance for a given matrix, since those represent the ML parameters for a Gaussian distribution. The matrices containing different amounts of samples were all passed in and the parameters were recorded.

Classification was done using the Bayesian techniques used in the previous experiments, by comparing the g values based on the case. Classification was always done on the original 200,000 samples, and misclassifications were recorded as they were encountered. These values were used to determine the classification accuracy for each class and the total classification accuracy, as was done in the first project.

B. Experiment 3 Implementation

To gather the data required to estimate a gaussian of skin value densities, we utilized opencv and C++. OpenCV is a library that allows for easy image extraction and augmentation. To gather the density of skin samples, the program iterated through every pixel in a test image and the mask image concurrently. If the mask contained a value that wasn't black, that meant that on the test image, it was considered skin. If a pixel was considered skin, its R, G, Cb, and Cr, values were recorded and stored together in a tuple.

Calculating the Thetas is the most important part to estimating a gaussian. Assuming our skin sample data was independent along both its dimensions, and assuming that the data is gaussian in nature, we are able to make shortcuts with Maximum likelihood estimation. Using Maximum likelihood estimation, the sample mean of the gathered data was the estimated value for the mean of our estimated gaussian. Also, once this sample mean is calculated, we can derive the sample covariance matrix. Because of the aforementioned assumptions, we can use this covariance matrix for our estimated gaussian as well. The last theta to approximate using MLE is the prior probability of a pixel being skin. For this value, the simple derived equation is just the ratio of skin pixels to all pixels.

As outlined in the homework criterion, not only do we need to have a working gaussian discriminant function, but we need to figure out what threshold value results in the same amount of false positives to false negatives ratio. This process is known as examining an ROC curve.

Finding a threshold value with an Equal Error Rate between False Positives and False Negatives is needed to determine the optimum classifier. Implementing this process was quite tedious, it took several minutes to comb over an entire image with a case III gaussian discriminant function. Each pixel's score was compared with a threshold value. If the score was above the threshold, it was classified as skin, otherwise it would be considered non-skin. Every pixel's classification would be compared with the ground truth to determine the number of false positives and false negatives. From here, a threshold was determined that would result in the same number of false negatives and false positives.

Once all of the above values are found for both the rg and cb cr models, the optimum model is combed over through a given image. If a pixel is determined to be non-skin, the pixel is turned black. Leaving only the image with detected skin pixels.

III. Results and Discussion

A. Experiment 1 Analysis

In the first project, we built a Bayesian classifier using the true parameters of the distribution. It got a 97.19% accuracy on ω_1, 98.95% accuracy on ω_2, and a 98.42% accuracy overall. The maximum likelihood estimates for the parameters are shown in Table 1, and the resulting accuracies of the models from the corresponding parameters are shown in Table 2. The accuracy of models using ML estimation on 200,000 are very close to these original values, getting a 97.22% accuracy on ω_1, 98.94% accuracy on ω_2 , and 98.43% accuracy overall. The accuracies are close since the ML estimated parameters are very close to the true parameters, because there is a large amount of data samples and the distribution is known.

As the amount of data decreased, the difference between the true distribution parameters and predicted parameters became larger. The accuracy steadily remained above 98.4% until there were only 20 samples. This is most likely because even though the predictions were worse, they were still good enough to discriminate against these 2 distributions for the vast majority of the data points.

Tests were also done to try treating the data as uncorrelated to reduce the parameters in the model. The difference between the resulting accuracies of these approaches was very minor, but tended to favor the uncorrelated models. This may be due to the true covariance between the distributions being 0, and as such this will remove the error of the estimated covariance being some small nonzero value, which it was when the estimated correlations were left in.

Table 1: Maximum Likelihood Estimates of Gaussian Parameters for Varying Samples for Experiment 1.

Count	μ_1 estimate	μ_2 estimate	Σ_1 estimate		Σ_2 estimate	
200000	1.00874	4.00101	1.00504	0.000222914	1.00344	0.00120914
	1.00177	3.99898	0.000222914	1.00149	0.00120914	1.00027
20000	1.0282	4.01244	1.03499	0.0124684	0.995565	-0.0131812
	1.01105	3.99592	0.0124684	1.0074	-0.0131812	1.00292
2000	0.967198	3.98981	0.985396	0.00982014	0.980992	-0.0301212
	0.980253	4.00779	0.00982014	0.968342	-0.0301212	0.975752
200	1.04888	3.96039	1.16907	-0.0555838	1.06264	0.0370447
	0.960799	4.14033	-0.0555838	0.996853	0.0370447	1.15566
20	0.791194	3.92868	0.884904	-0.0452619	1.30849	-0.268106
	0.822325	3.56592	-0.0452619	0.144125	-0.268106	0.900342

Table 2: Prediction Accuracy of Model on Full Dataset Based on the Amount of Samples for Maximum Likelihood Estimates for Experiment 1.

Count	Set covariance to 0?	Class 1 accuracy	Class 2 accuracy	Total accuracy
200000	Y	0.972183	0.989443	0.984265
	N	0.97215	0.989464	0.98427
20000	Y	0.97335	0.988914	0.984245
	N	0.974067	0.988643	0.98427
2000	Y	0.9711	0.989971	0.98431
	N	0.9722	0.989436	0.984265
200	Y	0.971983	0.9893	0.984105
	N	0.969983	0.990271	0.984185
20	Y	0.78205	0.998843	0.933805
	N	0.823183	0.998521	0.94592

B. Experiment 2 Analysis

In the first project, we built a Bayesian classifier using the true parameters of the distribution. It got a 72.54% accuracy on ω_1, 96.49% accuracy on ω_2, and a 91.71% accuracy overall. The maximum likelihood estimates for the parameters are shown in Table 3, and the resulting accuracies of the models from the corresponding parameters are shown in Table 4. The accuracy of models using ML estimation on 200,000 are very close to these original values, getting a 72.56% accuracy on ω_1, 96.48% accuracy on ω_2, and 91.69% accuracy overall. The accuracies are close since the ML estimated parameters are very close to the true parameters, because there is a large amount of data samples and the distribution is known.

The accuracies for each class and overall remained pretty close to the optimal value for 200,000, 20,000, and 2,000 samples. At 200 samples, the accuracy on ω_1 took a major hit and caused the overall accuracy to decline. The most interesting result that we got was for 20 samples. Even though the estimates were very far off, which was expected, the resulting accuracy was remarkable. This was difficult to explain, so we ran the experiment again with a different random seed, and got very different estimates and a total accuracy of 85.39%, which was a more expected accuracy based on the trend. Our takeaway from this is that the results are much less predictable with less data, but are more likely to be inaccurate.

The difference between uncorrelating the variables was minor as it was in the first experiment, and there doesn't appear to be a pattern whether it is better based on the number of samples.

Table 3: Maximum Likelihood Estimates of Gaussian Parameters for Varying Samples for Experiment 2.

Count	μ_1 estimate	μ_2 estimate	Σ_1 estimate		Σ_2 estimate	
200000	1.00911	4.00349	1.00181	0.000981303	4.00746	0.00838954
	1.00391	3.98726	0.000981303	1.0014	0.00838954	7.99425
20000	1.0298	4.02089	1.02498	-0.000825319	3.98864	-0.00977068
	1.02235	3.95512	-0.000825319	0.999053	-0.00977068	7.98526
2000	0.986773	4.101	0.960952	-0.0491651	4.07379	0.0420399
	0.953301	3.96415	-0.0491651	0.961855	0.0420399	8.02384
200	1.12157	3.32476	1.10086	-0.0246179	4.20365	0.0310444
	0.965673	3.09348	-0.0246179	1.0069	0.0310444	8.24453
20	1.01863	4.75947	1.01507	-0.0118318	3.73304	-0.555394
	0.877205	-4.29964	-0.0118318	0.389161	-0.555394	7.12042

Table 4: Prediction Accuracy of Model on Full Dataset Based on the Amount of Samples for Maximum Likelihood Estimates for Experiment 2.

Count	Set covariance to 0?	Class 1 accuracy	Class 2 accuracy	Total accuracy
200000	Y	0.725625	0.9648	0.916965
	N	0.724925	0.9649	0.916905
20000	Y	0.729825	0.964594	0.91764
	N	0.730475	0.964512	0.917705
2000	Y	0.72525	0.964606	0.916735
	N	0.7247	0.965094	0.917015
200	Y	0.463	0.98495	0.88056
	N	0.46075	0.98515	0.88027
20	Y	0.8509	0.948969	0.929355
	N	0.826575	0.953644	0.92823

C. Experiment 3 Analysis

Experiment 3 aims to create a skin classifier using color values. Figure 1 depicts the sampled data for skin in each color space.

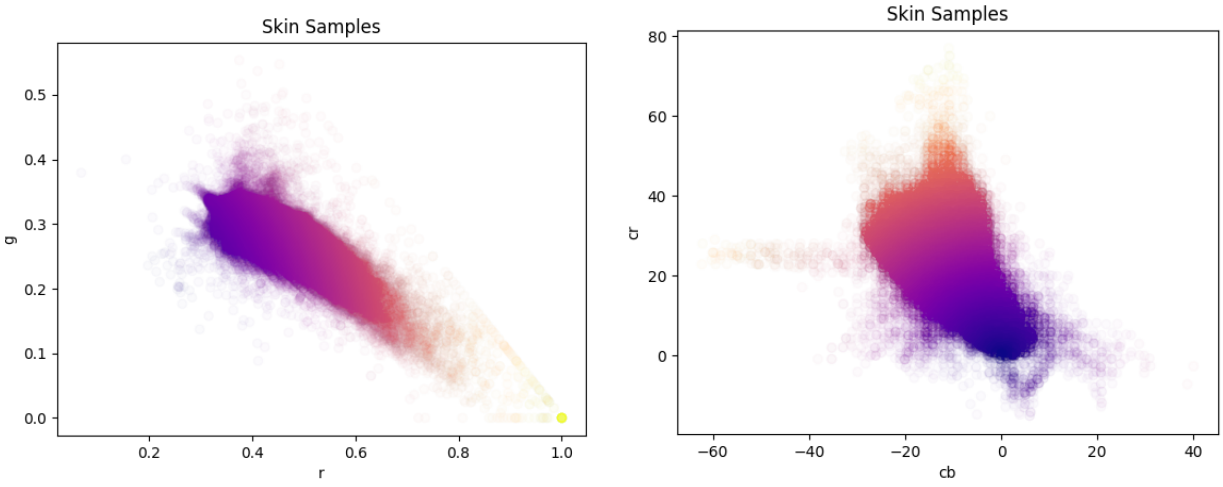

Fig 1. Graphs plotting skin samples on RG and Cb Cr color spaces.

In order for our lives to be easier and use the shortcuts from MLE, these distributions of points need to represent a multivariate Gaussian. The RG color space is the graph on the left, and the

Cb Cr color space is the graph on the right. These distributions do not look like perfect Gaussian distributions, but they will be assumed to be Gaussian for creating the classifier.

The calculated theoretical values for the RG color space are shown in Table 5. It is important to note that the theoretical prior probability of a pixel being skin was 0.0574786 (or around 5%).

Table 5: Theoretical Values for Parameters in RG color space.

Sample Mean	Sample Covariance	
0.425481	0.00222843	-0.001048814
0.297995	-0.001048814	0.00071284

Using those values, scores were calculated for every pixel in the image. We looked at the range of values for scores and empirically determined a range of potential optimum thresholds. For clarification, this implementation did not normalize the scores to fall between [0,1]. Because of this, threshold values might look like arbitrarily chosen values, but these threshold the data at raw and unnormalized scores. The following graph is highlighting the threshold range between -0.4 and -0.2 in increments of 0.1. We found that my optimum threshold value (where FP = FN) was at -0.35. When the threshold value is set to -0.35 this classifier has ERR, FP=FN=~80,000 as can be shown in Figure 2.

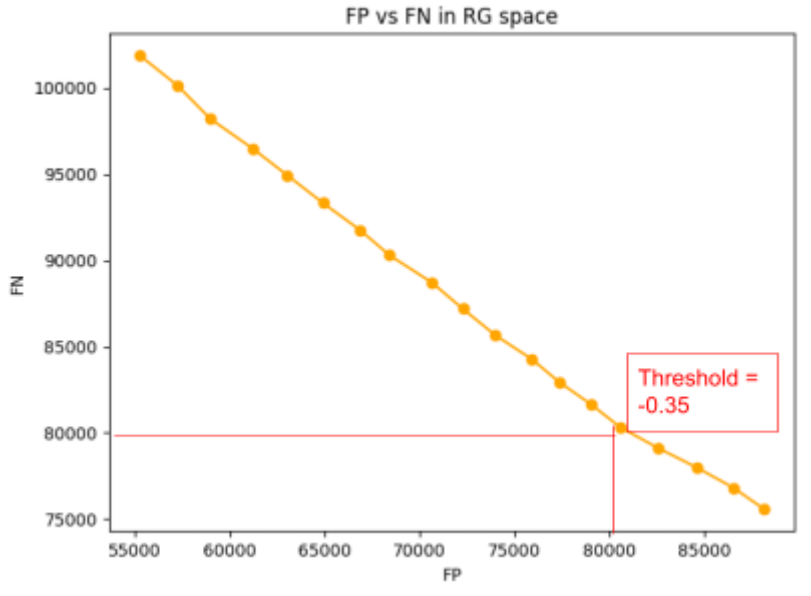

Fig 2. False Positives vs False Negatives for varying gaussian discriminant function thresholds in the RG classifier. The ERR is achieved at threshold =-0.35.

Running the classifier with the threshold at -0.35 results in the image shown in Figure 3. This is pretty impressive considering it picked up arms and hands, which were not explicitly labeled as skin in the labeled data.

Fig 3. Original image (left) shown besides output of RG skin classifier (right).

The calculated theoretical values for the Cb Cr colorspace are shown in Table 6.

Table 6: Theoretical Values for Parameters in Cb Cr Color Space.

Sample Mean	Sample Covariance	
-12.7208	30.242179	-20.192179
23.9883	-20.192179	47.2157386

Using those values, scores were calculated for every pixel in the image. I looked at the range of values for scores and empirically determined a range of potential optimum thresholds. For clarification, this implementation did not normalize the scores to fall between [0,1]. Because of this, threshold values might look like arbitrarily chosen values, but these threshold the data at raw / unnormalized scores. The following graph is highlighting the threshold range between -7.0 and -5.0 in increments of 0.1 for 20 ROC plots total. I found that my optimum threshold value (where FP = FN) was at -5.8. When the threshold value is set to -5.8 this classifier has ERR, FP=FN=~60,000 as shown in Figure 4.

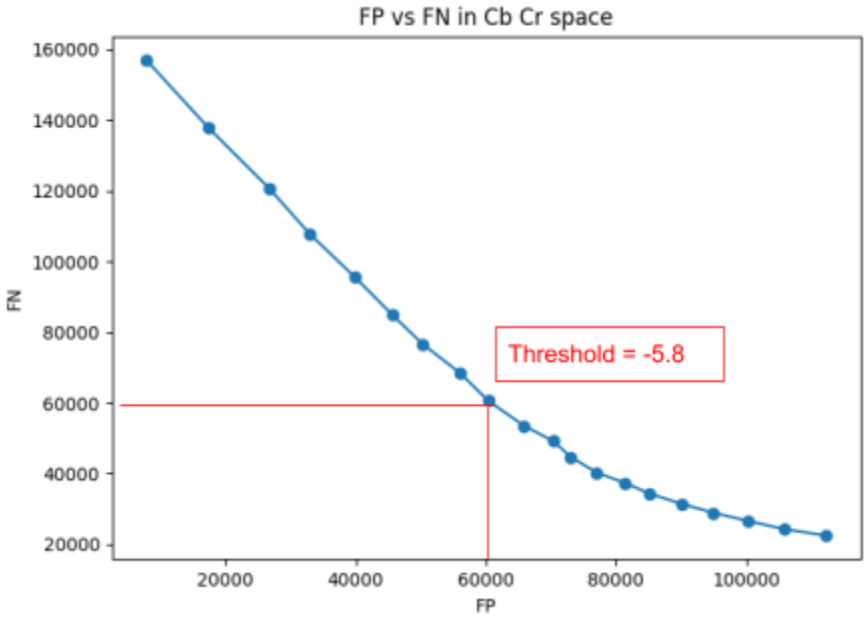

Fig 4. False Positives vs False Negatives for varying gaussian discriminant function thresholds in the Cb Cr classifier. ERR is achieved at threshold = -5.8.

Running the classifier with the threshold at -5.8 results in the image shown in Figure 5. This is pretty impressive considering it picked up arms and hands even better than the RG classifier. The woman on the far right has her entire arm picked up in this classifier while the RG only picked up pieces of it.

Fig 5. Original image (left) shown besides output of Cb Cr skin classifier (right).

So the Cr Cb color space classifier got 20,000 less FN and 20,000 less FP than the RG color space classifier. This means that overall, the Cb Cr classifier is better in general cases. You can see from the pictures slight differences where RG does better than Cb Cr but overall I'm a lot happier with the Cb Cr's performance.

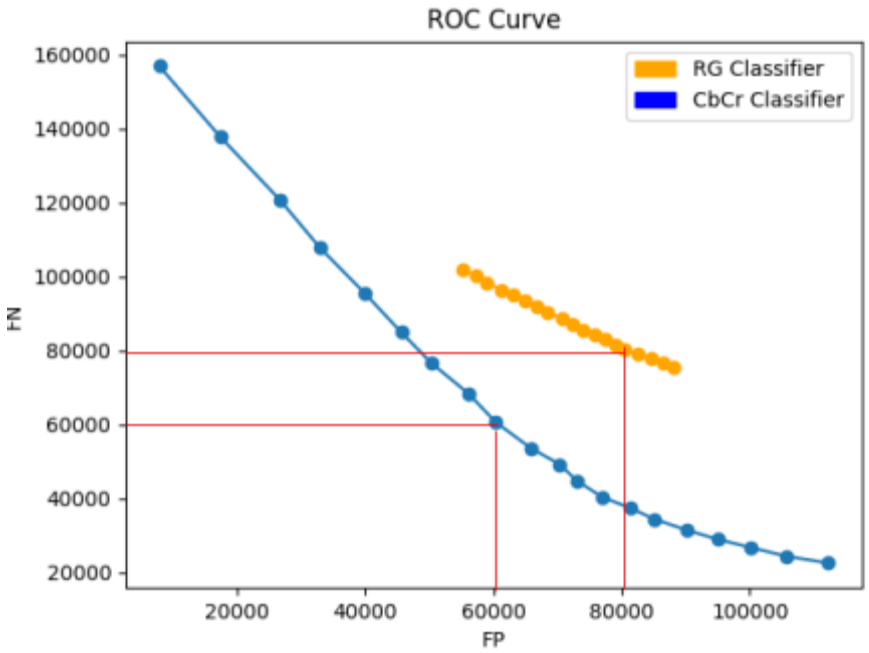

Fig 6. ROC curve of the RG and Cb Cr classifiers highlighting both the ERR of the RG classifier in relation to the ERR of the Cb Cr classifier.

Figure 6 shows that the ERR of the Cb Cr curve is much smaller than the ERR for the RG curve. That indicates that the Cb Cr classifier should be chosen over the RG classifier. Both curves show 20 different threshold values.