**Project 2: Bayesian Classifiers and Skin Pixel Detection**

**CS479: Pattern Recognition**

**Terence Henriod**

**April 9, 2014**

**Introduction**

In this project the use of Bayesian Minimum-Error Classifiers and Battacharyya and Chernoff upper error bounds to create and describe a pattern classifier for a Gaussian object. A similar method was used in Project 1, but this time instead of assuming certain parameter values for the distribution of the data, the parameters were actually estimated for the data, then the classifier will was tested on the training data.

It should be noted that I used a different, randomly generated set of data than was provided, but still used the same specified distribution parameters as listed in the assignment document. This same data set was used for both projects.

In addition, skin pixel detection was practiced using two color spaces: the *chromatic* color space, and the *YCbCr* color space. These spaces were collapsed from 3 to 2 dimensional space (discussed later). Using a reference photo, average values and covariances in each collapsed color space were found and used to classify pixels using a Gaussian probability density. Probability thresholds were systematically chosen to either accept pixels as being skin or reject them as not being skin, and the resulting data was used to demonstrate *Receiver Operating Characteristic Curves*.

**Technical Discussion**

*Bayesian Classifiers*

Given that the topics of Bayesian Classifiers, discriminant functions, and the application of Chernoff and Battacharyya bounds were discussed in detail in Project 1, they will not be discussed here.

*Maximum Likelihood Estimation*

It should be mentioned that the result of the Maximum Likelihood Estimations for the mean and covariance parameters of the Gaussian distribution were used for parameter estimation. Maximum Likelihood Estimation simply uses Calculus methods to find the maximum of a likelihood of a given parameter for a given set of data. In short, the well-defined estimates for the Gaussian distribution’s parameters are the mean and covariance that are always presented when discussing such parameters:

The mean µ is estimated by:

|  |  |
| --- | --- |
|  | [1] |

The covariance matrix Σ is estimated by:

|  |  |
| --- | --- |
|  | [2] |

*Multivariate Gaussian Distribution*

It is only fitting that the probability density function (pdf) of the Multivariate Gaussian Distribution is listed:

|  |  |
| --- | --- |
|  | [3] |

This pdf (or variations of it) is used in all parts of the project. In short, we have assumed that all data follows the Gaussian (Normal) Distribution, and use it to compute the likelihood that certain test objects actually belong to a given class.

*Color Spaces*

As it turns out, different color spaces can be used to better encode/represent information. For example, the RGB color space tends to encode a lot of redundant information. The Blue component contains both information that is redundant to Red and Green, but it also contains brightness information, which is unimportant for identifying skin. The 3-dimensional RGB space can be mapped to the 2-dimensional *r*, *g* space by the following:

|  |  |
| --- | --- |
|  | [4] |

|  |  |
| --- | --- |
|  | [5] |

Similarly, the YCrCb space can be used, but since the Y component contains irrelevant luminance information, this 3-dimensonal space can converted from RGB to YCbCr space and also be collapsed to a 2-dimensional space by neglecting the transformation of RGB to Y:

|  |  |
| --- | --- |
|  | [6] |

|  |  |
| --- | --- |
|  | [7] |

These dimensionality reductions both ease the burden of computations as well as get rid of possibly confounding excess information.

*Thresholds and ROC Curves*

In this project, when classifying pixels, arbitrary likelihood thresholds are systematically used. This means that many arbitrary likelihood values were chosen, and if a pixel has a likelihood of belonging to the class of skin pixels lower than the threshold, it was classified as a non-skin pixel; alternately, a pixel was classified as skin only if it had a likelihood (as determined by the Gaussian pdf and estimated parameters) greater than the given threshold.

By computing the rates of false acceptance and false rejection, a Receiver Operating Characteristic curve was generated, which is useful in helping to estimate the ideal threshold to be used for classification.

It should be noted that the term ROC curve is usually applied to tracking the discriminability between multiple classes, but in our case we only test against one class so the curve should be referred to as an *Operating Characteristic curve*.

**Results**

*Problem 1*

The performance of the classifiers after estimating the parameters rather than assuming them was improved compared to the classification performance in Project 1 when the prior probabilities of each distribution were assumed to be equal. Since the data was generated and processed in equal proportions, it was an appropriate assumption, and the results confirmed this. Classification performance suffered when assuming that one class was more likely than the other.

It is also interesting to observe that the error bounds became tighter/lower, as well as more able to properly bound the experimental error.

Part A

As can be seen, improved performance was observed through the use of estimated parameters rather than assumed values, as would be expected intuitively.

|  |  |  |  |
| --- | --- | --- | --- |
| Part 1-A | | Project 1 | Project 2 |
| Mean | Class 1 |  |  |
| Class 2 |  |  |
| Covariance | Class 1 |  |  |
| Class 2 |  |  |
| Prior Probabilities | Class 1 | 0.5 | 0.5 |
| Class 2 | 0.5 | 0.5 |
| Number of Data | | 20000 | 20000 |
| Number of Incorrect Classifications | | 6553 | 6340 |
| Test Error Rate | | 0.3276 | 0.317 |
| Chernoff Error Bound | | 0.3033 | 0.4177 |
| β\* | | 0.5000 | 0.3731 |
| Battacharyya Error Bound | | 0.3033 | 0.4222 |

Table 1: A comparison of the Part 1A results of Project 1 (where parameters were assumed), and Project 2 (where parameters were estimated using the results of MLE).

Part B

In part B the error rate rose dramatically, likely due to the assumption that one class was much more likely than the other, when in reality, both classes were equally likely to appear.

|  |  |  |  |
| --- | --- | --- | --- |
| Part 1-B | | Project 1 | Project 2 |
| Mean | Class 1 |  |  |
| Class 2 |  |  |
| Covariance | Class 1 |  |  |
| Class 2 |  |  |
| Prior Probabilities | Class 1 | 0.3 | 0.3 |
| Class 2 | 0.7 | 0.7 |
| Number of Data | | 20000 | 20000 |
| Number of Incorrect Classifications | | 6743 | 9112 |
| Test Error Rate | | 0.3371 | 0.4556 |
| Chernoff Error Bound | | 0.2542 | 0.3000 |
| β\* | | 0.7000 | 1.0000 |
| Battacharyya Error Bound | | 0.2779 | 0.3870 |

Table 2: A comparison of the Part 1B results of Project 1 (where parameters were assumed), and Project 2 (where parameters were estimated using the results of MLE).

*Problem 2*

As in Problem 1, performance was improved by estimating the parameters and using a correct prior probability assumption.

Part A

Again, classification using estimated parameters performed better than assuming parameter values.

|  |  |  |  |
| --- | --- | --- | --- |
| Part 2-A | | Project 1 | Project 2 |
| Mean | Class 1 |  |  |
| Class 2 |  |  |
| Covariance | Class 1 |  |  |
| Class 2 |  |  |
| Prior Probabilities | Class 1 | 0.5 | 0.5 |
| Class 2 | 0.5 | 0.5 |
| Number of Data | | 20000 | 20000 |
| Number of Incorrect Classifications | | 5288 | 4637 |
| Test Error Rate | | 0.2644 | 0.2319 |
| Chernoff Error Bound | | 0.3198 | 0.3866 |
| β\* | | 0.4250 | 0.3716 |
| Battacharyya Error Bound | | 0.3237 | 0.3926 |

Table 3: A comparison of the Part 2A results of Project 1 (where parameters were assumed), and Project 2 (where parameters were estimated using the results of MLE).

Part B

Once again, an incorrect assumption about prior probabilities resulted in less than ideal classification performance.

|  |  |  |  |
| --- | --- | --- | --- |
| Part 2-A | | Project 1 | Project 2 |
| Mean | Class 1 |  |  |
| Class 2 |  |  |
| Covariance | Class 1 |  |  |
| Class 2 |  |  |
| Prior Probabilities | Class 1 | 0.5 | 0.5 |
| Class 2 | 0.5 | 0.5 |
| Number of Data | | 20000 | 20000 |
| Number of Incorrect Classifications | | 5638 | 6191 |
| Test Error Rate | | 0.2819 | 0.3096 |
| Chernoff Error Bound | | 0.2799 | 0.3000 |
| β\* | | 0.7250 | 1.0000 |
| Battacharyya Error Bound | | 0.2967 | 0.3598 |

Table 4: A comparison of the Part 2B results of Project 1 (where parameters were assumed), and Project 2 (where parameters were estimated using the results of MLE).

*Problem 3*

In this problem, skin colored pixels were identified with varying success. As can be seen by the ROC curves, the choice of threshold value plays a pivotal role in the performance of the classifier. As is intuitive, a higher threshold results in more false rejections, while a lower one results in more false positives.

It appears that the CbCr space performs better than the *r*, *g* space at differentiating skin pixels from others, as it produces lower error rates, and at lower threshold values (see Figures 3 and 4).

In this problem, thresholds from 0.01 to 0.99 in increments of 0.01 were used to give the ROC plots sufficient granularity to appropriately display the trend.

Part A

The following table and graphs illustrate the classification performance demonstrated by simply using an estimated Gaussian distribution for skin pixel color and rejection threshold to identify pixels as being skin or not.

|  |  |
| --- | --- |
| Skin Colored Pixel Parameter Estimates in Collapsed Chromatic Color Space | |
| Skin pixel mean chromatic color  (collapsed to 2 dimensions): |  |
| Skin pixel feature covariance ( and ): |  |

Table 5: The estimated parameters for skin colored pixels in the training photo in the collapsed chromatic color space.

Figure 1: The ROC curve for skin pixel identification in Training\_3.ppm. It appears that a choice of x\* = 0.34 produces a nearly equal error rate.

Figure 2: The ROC curve for skin pixel identification in Training\_3.ppm. It appears that a choice of x\* = 0.31 produces a nearly equal error rate.

Part B

It can be seen by the resulting ROC curves that the CbCr space is much more effective for skin pixel identification purposes than the *r*, *g* space is (although the latter did perform moderately well).

|  |  |
| --- | --- |
| Skin Colored Pixel Parameter Estimates in the CbCr Space | |
| Skin pixel mean CbCr color  (collapsed to 2 dimensions): |  |
| Skin pixel feature covariance ( and ): |  |

Table 6: The estimated parameters for skin colored pixels in the training photo in the collapsed YCbCr space.

Figure 3: The ROC curve for skin pixel identification in Training\_3.ppm. It appears that a choice of x\* = 0.12 produces a nearly equal error rate.

Figure 4: The ROC curve for skin pixel identification in Training\_6.ppm. It appears that a choice of x\* = 0.10 produces a nearly equal error rate.

**Appendix**

*Program Listings*

The student would like to thank the makers of the Eigen linear algebra C++ library for their open-source contribution to the world. Hours were saved in coding and debugging thanks to their contribution. Eigen can be found at <http://eigen.tuxfamily.org/>.

The project was coded in C++. This included the use of the C++ STL, particularly the vector template container.

Project\_2\_driver.cpp: The driver program for the assignment.

bayes\_classifier.h: A class definition for a Bayesian minimum error classifer; provides functionality to classify objects via a dichotomizer/discriminant functions, as well as estimate Chernoff and Battacharyya upper error probability bounds.

bayes\_classifier.cpp: Implementation code for the BayesClassifier class defined in bayes\_classifier.h.

strict\_gaussian\_classifier.h: A class definition for a simple class that keeps track of a mean and covariance (and any other variations of these needed to compute the Gaussian pdf) and a rejection threshold value. The class has the capability to compute the mean and covariance of a data set, compute the Gaussian pdf for a given test vector, and even accept or reject a test vector as being in the class given based on how its likelihood compares with the rejection threshold.

strict\_gaussian\_classifier.cpp: Implementation code for the StrictGaussianClassifier class defined in strict\_gaussian\_classifier.h.

*Complete program listings are attached to this document.*