**Project 1: Bayesian Classifiers**

**CS479: Pattern Recognition**

**Terence Henriod**

**March 10, 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.

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 doucument.

**Technical Discussion**

*Discriminant Functions*

When classifying data, minimum error rate classification can be achieved with the discriminant function:

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

When applied to data that can be assumed to be multivariate-Gaussian, this function takes the form:

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

One should consider that since discriminant functions must be compared to one another when used, the resulting discriminant function, or *dichotomizer*, will not have terms that are constant with respect to the classes being chosen between. That is, the final *g(x)* that is the result of the difference of various discriminant functions will only contain terms that are dependent on the input feature vector and the parameters of the class represented by the discriminant function.

Thus, a two category Gaussian dichotomizer would appear as:

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

It should be noted that if , then that term too can be eliminated to simplify computations.

Sometimes, different cases of assumptions are made to ease computations. In equation [3], no assumptions are made, and this is in fact Case Three of the three cases associated with Gaussian discriminant functions. Case One is where the two classes are (or are assumed to be) statistically independent, and the covariance matrices between the two classes are identical and strictly diagonal. This results in a discriminant function of the following form:

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

once “would be” redundant constants are removed. The Second Case is when the covariance matrices of the two classes are identical but non-diagonal. This case will not be discussed further in this report since it was not applicable in the assigned computations.

*Error Bounds*

While Bayesian classifiers offer minimal average-error classification, it would be beneficial to be able to predict that error. Computing the actual error can be difficult, but in the two-class case there are two upper bounds to error that are often considered: the Chernoff bound and the Battacharyya bound; the Battacharyya bound is easier to compute, but the Chernoff bound is a tighter bound. Both are best suited for the Gaussian distribution and lose their effectiveness when applied to non-Gaussian distributions.

The Chernoff bound is computed by finding , either analytically or numerically, for which the following expression is minimized, and this probability is the Chernoff bound:

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

where is the approximation of the classification error integral for the Gaussian distribution and is found by:

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

The Battacharyya bound is simpler in that is not found, is simply used instead in equation [5], making the computations somewhat easier and eliminate the need for a minimization solution.

**Results**

In general, I found that the Bayesian Classifier had a much higher than desirable error rate. In fact, in some cases, my classifier exceeded the supposed upper bounds for classification error. I am tempted to think that this is because Matlab generated random data that was not particularly suited to the distributions specified. Decision boundaries were computed by hand.

*Problem 1*

The following are the resulting data tables from the classification of randomly generated data using the following parameters:



Part A

In this part, all theory is satisfied, with the error result falling within the error bounds.

|  |  |
| --- | --- |
| Part 1 | A |
| Decision Boundary |  |
| Number of Data | 20000 |
| Number of Incorrect Classifications | 6553 |
| Test Error Rate | 0.3276 |
| Chernoff Error Bound | 0.3033 |
| B\* | 0.5000 |
| Battacharyya Error Bound | 0.3033 |

Table 1: Problem 1, Part A Results

Figure 1: Data and Decision Boundary for Problem 1, Part A

Part B

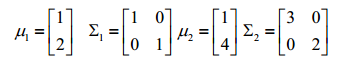
As seen in Table 2, the error classification rate was extremely higher than expected. I have no explanation for this, except possibly the inaccuracy of the C++ double.

|  |  |
| --- | --- |
| Part 1 | B |
| Decision Boundary |  |
| Number of Data | 20000 |
| Number of Incorrect Classifications | 6743 |
| Test Error Rate | 0.3371 |
| Chernoff Error Bound | 0.2542 |
| B\* | 0.7000 |
| Battacharyya Error Bound | 0.2779 |

Table 2: Problem 1, Part B Results

*Problem 2*

The following are the resulting data tables from the classification of randomly generated data using the following parameters:



Part A

Results were as expected and were in accordance with Bayesian decision making theory.

|  |  |
| --- | --- |
| Part 2 | A |
| Decision Boundary |  |
| Number of Data | 20000 |
| Number of Incorrect Classifications | 5288 |
| Test Error Rate | 0.2644 |
| Chernoff Error Bound | 0.3198 |
| B\* | 0.4250 |
| Battacharyya Error Bound | 0.3237 |

Table 3: Problem 2, Part A Results

Part B

As seen in Table 4, the error rate was strangely higher than the computed error bounds. It is difficult to say why this occurred.

|  |  |
| --- | --- |
| Part 2 | b |
| Decision Boundary |  |
| Number of Data | 20000 |
| Number of Incorrect Classifications | 5638 |
| Test Error Rate | 0.2819 |
| Chernoff Error Bound | 0.2799 |
| B\* | 0.7250 |
| Battacharyya Error Bound | 0.2967 |

Table 4: Problem 2, Part B Results

**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/>.

For this assignment, only a single driver program and BayesClassifier class were used. The source code for these is attached. The project was coded in C++.

Project\_1.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.cpp.