CS 479 - Pattern Recognition

Programming Assignment #1

Winston Yan (Formerly Team 10)

Spring 2023

All work was done by Winston Yan

**Introduction:**

Even with only a shallow level of consideration, a statistical approach to pattern recognition and classification seems highly appealing for both its shared themes and mathematical applicability. Specifically, Bayes’ Theorem allows for a relatively simple application of statistics in pattern recognition, which provides the foundation for the Bayes Classifier—the focus of this study. In this study, the development of a functioning Bayes Classifier will be used to illustrate the applications statistics has in pattern recognition and to analyze its effectivity and accuracy given varied data sets.

**Theory:**

*Introduction to the Bayesian Application*

Statistical classifiers are used to classify an object as the class which has the highest probability to be the correct one. In order to quantitatively express this, assume a set of features, x, and classes, ω. To express the probability that a set of features belongs to a certain class (titled the posterior probability) the expression, P(ωi/x), is used. Now to calculate this value, Bayes’ Theorem is used, thus finding that the previous expression is equivalent to p(x/ωi)P(ωi)/p(x).

*Risk and the Loss Function*

However, this expression is not exactly what is used to classify feature sets. Instead, minimization of the risk of taking the wrong action is used—consider, for example, how the risk of incorrectly granting access to a criminal into a bank vault is much higher than incorrectly refuting access to an executive. Because erroneous decisions are not necessarily equally weighted, a classifier seeks to take the action, α, with the minimum risk, as given by R(αi/x). In a two class situation, this risk is defined by the expression, λ(αi/ωj)P(ωj/x), where i is the class chosen, j is the class not chosen, and λ(αi/ωj) is the loss function that returns the cost of taking action αi given the that the class is an incorrect class. Furthermore, note how risk is a posterior probability that is weighted by the loss function. In the special case where the loss function is impartial to the different types of errors, a Zero-One loss function can be used. This Zero-One loss function returns one for taking the correct decision, and zero for any wrong decision, thus indiscriminately punishing all errors equally.

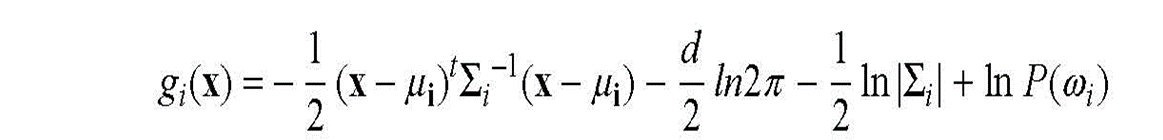
*The Discriminant*

To implement risk as a quantifiable and computable value, the discriminant is used.

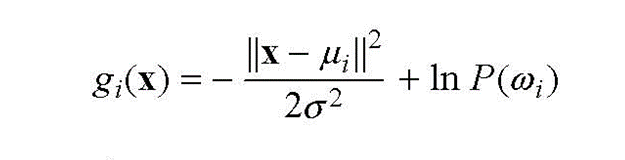
For simplicity, assume a Zero-One loss function. The discriminant is thus now equivalent to the posterity, and to calculate it, assume that it is a normal distribution. In almost all cases, there will be numerous features, and taking this into consideration gives us the multivariate gaussian density function (assuming d dimensions),



and thus the discriminant,



which can be simplified to varying degrees based on the characteristics of the covariance matrix, Σ. For example, a situation in which all classes share the same covariance matrix results in the following Euclidean Distance Classifier:



Letting the discriminant for adjacent classes equal each other offers the decision boundary, which is the hypersurface on which points are equally likely (or possess an equal risk) to belong to more than one class.

*Examining Error*

Because a classifier is never going to be perfectly accurate, it is important to understand the error dynamics of the Bayesian classifier. Assuming that p(x/ωi) is Gaussian (if it is not, the error bounds will be uninformative), the error bounds of our discriminant function become



To precisely find the error bound—called the Chernoff Error Bound—minimize the error bound above. Given that the Chernoff Error Bound is somewhat computationally intensive, a faster solution is to compute the Bhattacharyya Error Bound, which is defined as the error bound above with ß=0.5, which gives a rough estimate of the error bounds.

**Results and Analysis:**

In the study, the prediction that using case 1 for data set A and case 3 for data set B would be the most accurate was made, and the results support this claim (see table 1). Furthermore, the prior probabilities of the classes were calculated as 0.3 and 0.7 for P(ω1) and P(ω2) respectively by dividing the number of a specific class by the total amount of data.

For the Euclidean Distance Classifier, it resulted in comparable accuracies to Case I and Case III with data set A, which was more defined, but with data set B, it fared much worse, finding the worst overall accuracy by a wide margin. This is expected because both major conditions for the Euclidean Distance Classifier were not met: the data had different covariance matrices and also did not have equal priors. Note: the Euclidean Classifier ran noticeably quicker than both Case I and Case III because of its simplicity.

Finding the Bhattacharyya Error Bound (±0.0966 for data set A and ±0.4516 for data set B), led to the conclusion that they seem remarkably accurate for all classifiers and all datasets except for the case of the Euclidean Classifier with data set B.

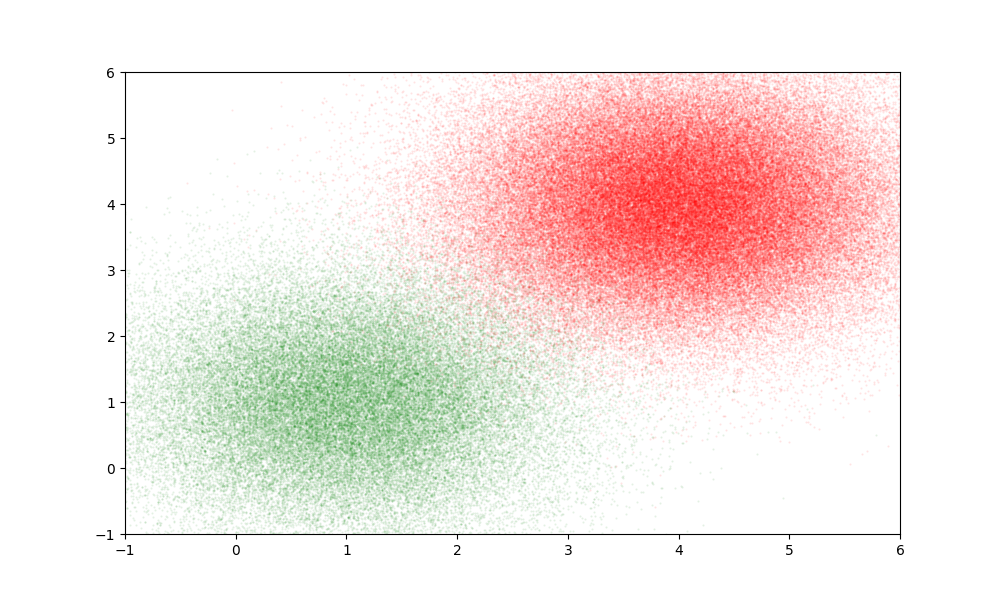
For classification rates, also see table 1 and 2. For data scatter plots of data, see Figure 1 and 2.

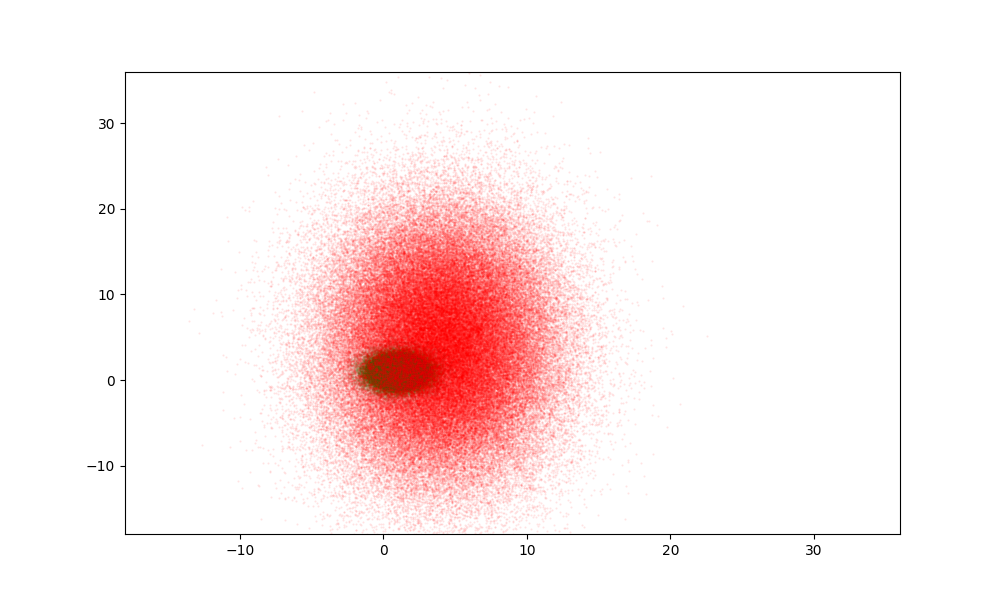
**Table 1 (average accuracy in 10 trials)**

|  | Data Set A | Data Set B |
| --- | --- | --- |
| Case I | 97.892% | 70.431% |
| Case III | 96.083% | 86.994% |
| Euclidean | 95.612% | 47.765% |

**Table 2 (error rates averaged over 10 trials)**

|  | Classified 1 as 2 | Classified 2 as 1 | No classifications |
| --- | --- | --- | --- |
| Case I - A | 0.362% | 1.747% | 0 |
| Case III - A | 0.146% | 3.771% | 0 |
| Euclidean - A | 0.122% | 4.266% | 0 |
| Case I - B | 7.676% | 21.893% | 0 |
| Case III - B | 0.095% | 12.911% | 0 |
| Euclidean - B | 0.122% | 52.114% | 0 |

**Figure 1 (plot of data set A)**

**Figure 2 (plot of data set B)**

**Source Code:**

Github Repository: <https://github.com/Yan-Winston/cs479-hw1>

Packages Required: Eigen, iostream, fstream, cstdlib, cmath, algorithm, and python/matplotlib (optional for data visualization)

**Sources:**

1. BayesianDecisionTheory.ppt by Dr. George Bebis, from https://www.cse.unr.edu/~bebis/CS479/, Spring 2023.