Project Report

Omar Ghaleb
COMP 5107

Problem

The idea of the project is to create a pattern classifier based on different methods and types of classifier on a selected dataset. In the project we created four main different classifiers:

- Quadratic: Maximum Likelihood(ML), Bayesian(BL) and Parzen Window
- K-Nearest Neighbours
- Fisher's Discriminant
- Ho-Kashyap

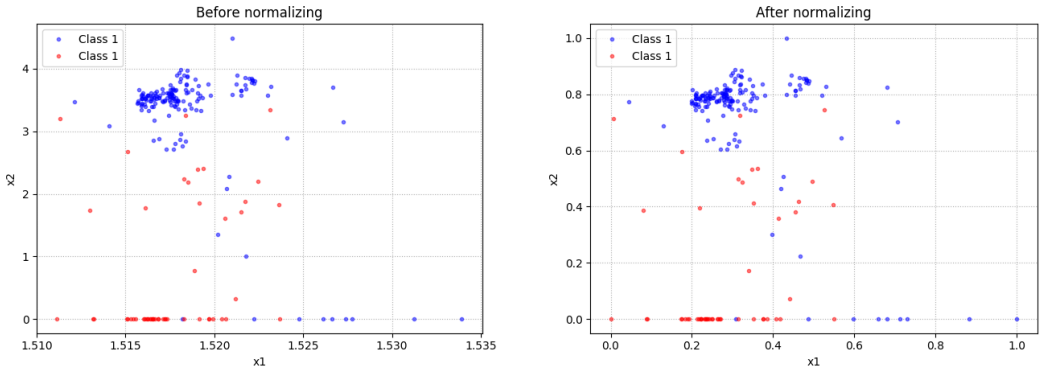
For testing the classifiers, five-cross validation was used. The code is available on the following link: https://github.com/omaramin1992/comp5107

Dataset

The dataset selected is used to classify different types of glass. The reason for this is that identifying the type of glass helps in criminology. The data had no empty values thus it did not need to be cleaned. The data set contained 9 features(6 were selected which are Refractive index, Magnesium, Aluminum, Potassium, Barium and Iron) and two main classes and 9 subclasses. For our project we worked with the whole dataset as two main classes only which are (window or non-window) or class1 and class2 respectively. In the project, the data were projected on the first and second dimensions only (x1-x2).

The data was the normalized since the scales were far off from each other and affected the classification behaviour, hence, it was needed to make the scales similar by normalizing the data between 0 and 1. In Figure 1, we can see the effect of normalizing the data.

One problem with the data, and we will see the effect of that, is how close both classes are to each other as they both have very close means. Another problem is that the first class (window) consists of 3 times the instances of the other class(non-window).



(a) Before Normalizing

(b) After Normalizing

Figure 1: Data Points in x1-x2 dimensions

Classifiers

Below, we are going to go through our different classifiers that we created for this dataset.

a. Quadratic Classifiers

First, we are going to talk about the quadratic classifiers (ML, BL, Parzen).

Maximum Likelihood (ML)

We use the ML to estimate the parameters of the quadratic classifier, M and Σ_{X_1}, for each class by using the equation:

$$\hat{M} = \frac{1}{N}\sum_{i=1}^{N} x_i, \qquad \hat{\Sigma} = \frac{1}{N}\sum_{i=1}^{N}[X_i - \hat{M}][X_i - \hat{M}]^T$$

And the resulting means were the following for each class:

$$\hat{M_1} = \begin{bmatrix} 0.32673694 \\ 0.73383251 \\ 0.30892724 \\ 0.07688964 \\ 0.00946538 \\ 0.13256345 \end{bmatrix}, \quad \hat{M_2} = \begin{bmatrix} 0.28480435 \\ 0.16341325 \\ 0.52232606 \\ 0.09011398 \\ 0.20292561 \\ 0.04536717 \end{bmatrix}$$

and covariance matrices as follows:

$$\Sigma_{X_1} = \begin{bmatrix} 0.0178 & -0.0149 & -0.0061 & -0.00291 & 0.00349 & 0.0028 \\ -0.0149 & 0.0388 & 0.0001 & 0.00149 & -0.00454 & -0.0053 \\ -0.0061 & 0.0001 & 0.0101 & 0.00251 & 0.00161 & 0.0010 \\ -0.0029 & 0.0014 & 0.0025 & 0.00123 & 0.00003 & 0.0005 \\ 0.0034 & -0.0045 & 0.0016 & 0.00003 & 0.00643 & 0.0027 \\ 0.0028 & -0.0053 & 0.0010 & 0.00051 & 0.00270 & 0.0378 \end{bmatrix},$$

$$\Sigma_{X_2} = \begin{bmatrix} 0.0159 & 0.0075 & -0.0086 & -0.0072 & -0.0050 & 0.0033 \\ 0.0075 & 0.0589 & -0.0082 & 0.0017 & -0.0056 & 0.0004 \\ -0.0086 & -0.0082 & 0.0340 & 0.0120 & 0.0125 & 0.0015 \\ 0.0072 & 0.0017 & 0.0120 & 0.0419 & -0.0050 & -0.0014 \\ -0.0050 & -0.0056 & 0.0125 & -0.0050 & 0.0549 & -0.0032 \\ 0.0033 & 0.0004 & 0.0015 & -0.0014 & -0.0032 & 0.0257 \end{bmatrix}$$

These values estimate parameter are gonna be our basis for any actual values needed or calculations that needs the real covariances(such as BL). Once we estimated the parameters, we were able to estimate the discriminant function and plot it in $(x1 - x2)$ dimensions, as in Figure 2, before and after diagonalizing the data.

After that we diagonalize the data and get the corresponding covariance matrices. For class 1 it becomes the identity matrix I and for class 2 is the diagonal matrix of the following:

$$\Sigma_{X_2} = \begin{bmatrix} 82.8012 & 11.2717 & 4.4823 & 3.0525 & 0.4904 & 0.6506 \end{bmatrix}$$

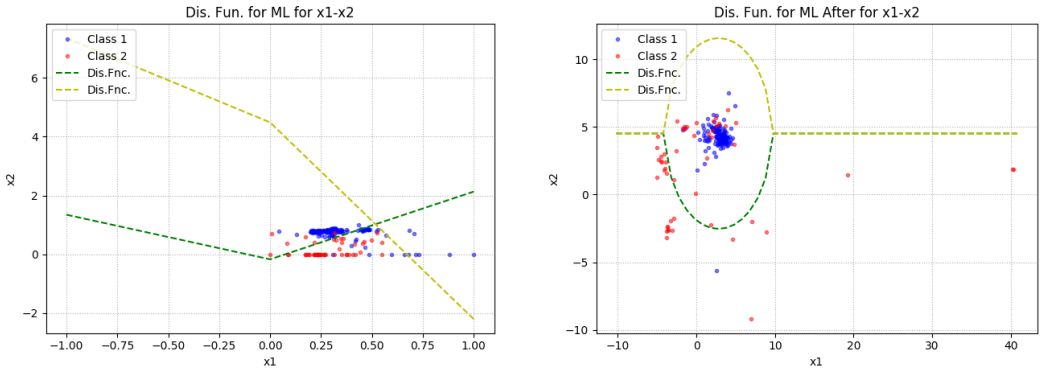
After that, we tested our classifier using five-cross validation and we got an average accuracy of 78.55%. (97.57% for class 1 and 17.81% for class 2) for both before and after diagonalizing the data. We can see that the accuracy is skewed towards the first class more and that i believe due to three main reasons:

- The data means are close to each other
- The number of instance in each each are significantly different as class 1 is three times class 2
- The correlation between the selected features and the classses

Bayesian Learning (BL)

Next, we are going to estimate the means using BL method. In the calculating of the mean we used the following equation:

$$(m)_n = \frac{1}{n}\Sigma \left[\frac{1}{n}\Sigma + \Sigma_0\right]^{-1} m_0 + \Sigma_0 \left[\frac{1}{n}\Sigma + \Sigma_0\right]^{-1} \left(\frac{1}{n}\sum_{j=1}^{n} x_j\right)$$



(a) Before Diagonalizing

(b) After Diagonalizing

Figure 2: ML Disc. Func. in x1-x2

In the above equation, we used the estimated covariance from ML as the actual covariance in calculating the mean. Since we used that covariance, we had the resulting estimate mean pretty close with high precision to ML estimates.

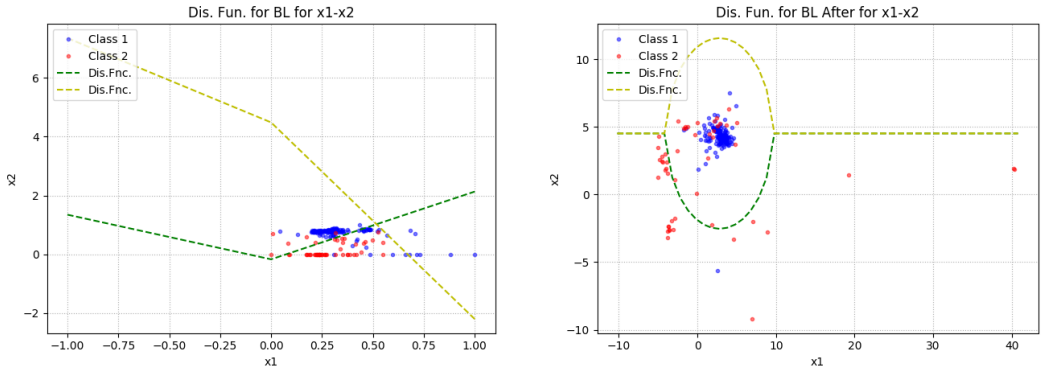
$$\hat{M}_1 = \begin{bmatrix} 0.32677967 \\ 0.73378739 \\ 0.30897436 \\ 0.07690064 \\ 0.00953297 \\ 0.13279159 \end{bmatrix}, \quad \hat{M}_2 = \begin{bmatrix} 0.28492792 \\ 0.16435958 \\ 0.52282895 \\ 0.09079785 \\ 0.20358894 \\ 0.04584223 \end{bmatrix}$$

Then we used these estimates to calculate the discriminant function which resulted in a similar result as the ML as it is shown in Figure 3.

Since we got the same discriminant function(DF), in testing our DF, we got the same results are ML with an accuracy of 78.554% for both before and after diagonalizing the data.

Parzen Window

Next we use the Parzen window method to estimate the one feature only as requested and we used that to calculate the mean (estimate the density function) of the first feature (dimension). In Figure 4 we can see the estimated mean for the first dimension for both



(a) Before Diagonalizing

(b) After Diagonalizing

Figure 3: BL Disc. Func. in x1-x2

classes how they are very close to each other. Parzen estimated means for all dimensions:

$$\hat{M_1} = \begin{bmatrix} 0.32673694 \\ 0.73383251 \\ 0.30892724 \\ 0.07688964 \\ 0.00946538 \\ 0.13256345 \end{bmatrix}, \quad \hat{M_2} = \begin{bmatrix} 0.28480435 \\ 0.16341325 \\ 0.52232606 \\ 0.09011398 \\ 0.20292561 \\ 0.04536717 \end{bmatrix}$$

b. K-Nearest Neighbours Classifier

Next, we used the k-nearest neighbours classifier to classify the data. In our case we used $k = 5$ and the algorithm calculates the distances between the test point and all other training points and takes the closest five points and votes on the majority of the classes of these points. Then the testing point is classified according to the majority class.

Using K-NN, we reached the maximum accuracy we had for the data with a 90.18% accuracy before diagonalizing and and accuracy of 85.89% after diagonalizing the data. (Code is available online on the link provided in the beginning).

c. Fisher's Discriminant Classifier

Then, we used the concept of Fisher's Discriminant to classify the data by finding a hyperplane that the point can be projected on and then find the linear classifier that can classify the data as accurate as possible. We calculated the vector W that represents the

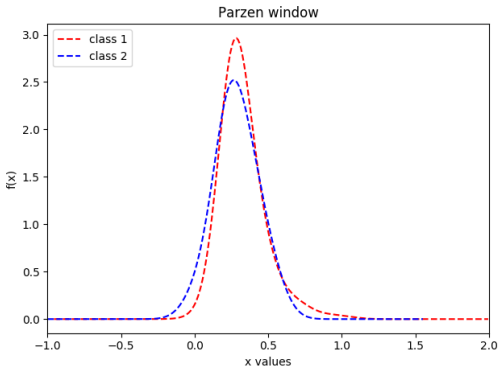


Figure 4: Parzen Window

plane that the points will be projected on:

$$W = \begin{bmatrix} 0.6697 \\ 5.5320 \\ -3.4231 \\ 0.4729 \\ -1.3701 \\ 1.871 \end{bmatrix}$$

And by using this transforming vector we get the points projected as in Figure 5 and we calculated the points of the discriminant function on that plane by solving the equation for one dimension instead of d dimensions. After transforming the data the first and second class have means of 3.49242 and -0.84373 respectively.

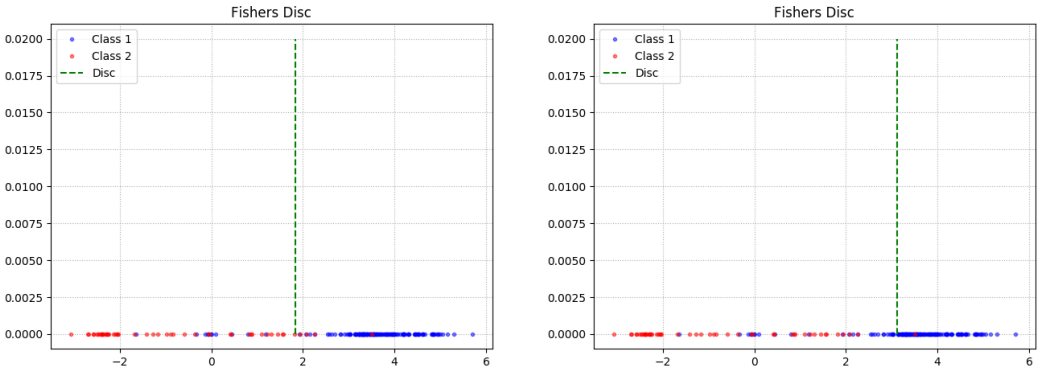
By testing the linear classifier, we got an accuracy of 88.825% before diagonalizing and 88.8251% which is the same accuracy after diagonalizing.

d. Ho-Kashyap Classifier

In ho-kashyap, a linear classifier is created based on the points only by using the concept of gradient decent. The algorithm for it is in Listing 1

Listing 1: Ho-Kashyap Code

```
1 def ho_kashyap(class1_data, class2_data):
2     class1_ones = np.ones(len(class1_data[0]))
3     class2_ones = np.ones(len(class2_data[0]))
4
5     y1 = np.array(class1_data)
```



(a) Before Diagonalizing

(b) After Diagonalizing

Figure 5: Fishers Discriminant

```
    y2 = np.array(class2_data)

    y1 = np.insert(y1, 0, class1_ones, axis=0)
    y2 = np.insert(y2, 0, class2_ones, axis=0)

    y1 = y1.transpose()
    y2 = -1 * y2.transpose()

    y = np.append(y1, y2, axis=0)

    a = np.ones((len(y[0]), 1))
    b = np.ones((len(y), 1))

    lr = 0.9  # learning rate
    i = 0
    e = y @ a - b
    e1 = np.linalg.norm(e)

    for i in range(400):
        e = y @ a - b
        b = b + lr * (e - np.abs(e))
        a = np.linalg.inv(y.transpose() @ y) @ y.transpose() @ b
        e1 = np.linalg.norm(e)
        i = i + 1

    return a, b
```

By using this method we solve for A and B until it converges and since the classes are not linearly separable, it will not find a line that separates them completely. and by updating the weights A, we reach an A and B such that the error is minimized.

The testing of the classifier was tested using same method used before, five-cross validation, and the accuracy before diagonalizing was 85.675% and after diagonalizing 78.6424%.

Improvements

To make this system a full pattern classification system, some of the class handling is meant to deal with only two classes and that could be extended to include multiple classes classification. It can also include a way of cleaning the data by removing outliers and removing the noisy data and that could help create a better classifier.