**ML Assignment 1**

**Fishers Linear Discriminant Analysis(FLDA)**

Fishers Linear Discriminant is one of the classification methods used in Machine Learning to find the linear discriminant function that separates 2 or more classes. In this assignment we have 2 classes and 2-D data. FLDA projects data from higher dimensions to 1-D and performs classification. The projection done by FLDA will maximize distance between means of 2 classes and also minimises variance within respective classes.

\*In order to use FLDA the classes should be linearly separable.

Python libraries used: pandas, numpy, matplotlib, math, scipy

**Data Set 1:**

The classes were linearly separable in case of dataset 1. The normalised “w” values are [0.0083803, 0.99996488] and the Threshold value is -0.39301273677. In Fig 1.1 the points are triangles and rectangles. The circles in the centre are the projections of those points on the line. In Fig 1.2 we can see the plot of only projections. In Fig 1.3, we can see the normal graphs of projections and their point of intersection represented in black colour.

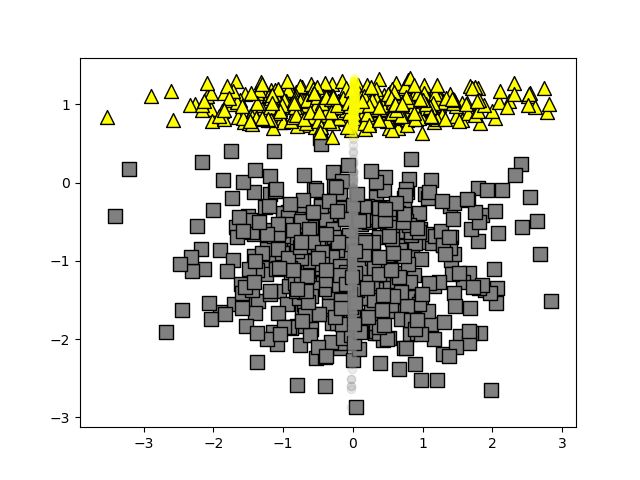


Fig1.1: Points (triangles and squares) and their projections (hued circles)

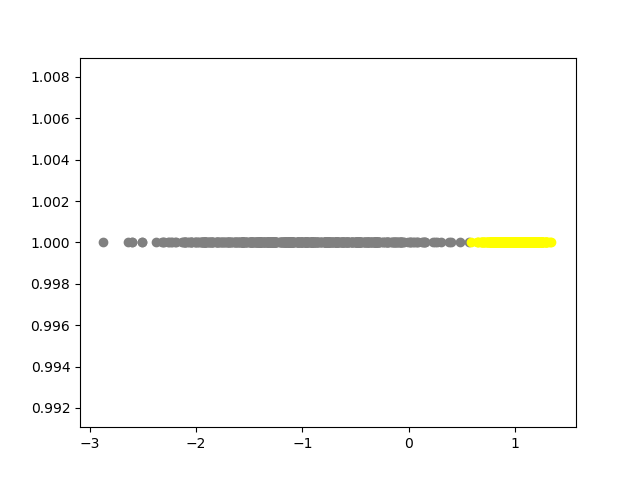


Fig 1.2: Just the projected values

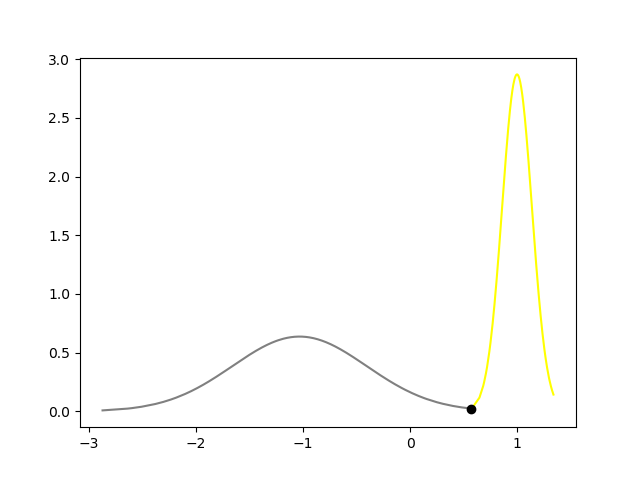


Fig 1.3 : Normal Curves and their point of intersection

**Data Set 2:**

The classes were not clearly linearly separable in case of dataset 2. The normalised “w” values are [0.03304637, 0.99945382] and the Threshold value is -0.15207088285281756. In Fig 2.1 the points are triangles and rectangles. The circles in the centre are the projections of those points on the line. In Fig 2.2 we can see the plot of only projections. In Fig 2.3, we can see the normal graphs of projections and their point of intersection represented in black colour.

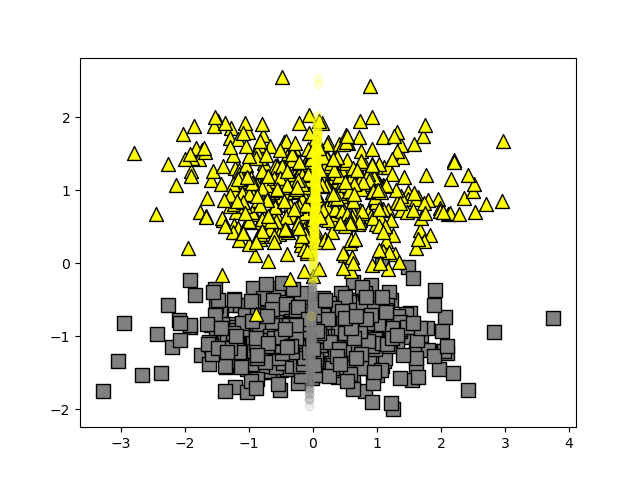


Fig2.1: Points (triangles and squares) and their projections (hued circles)

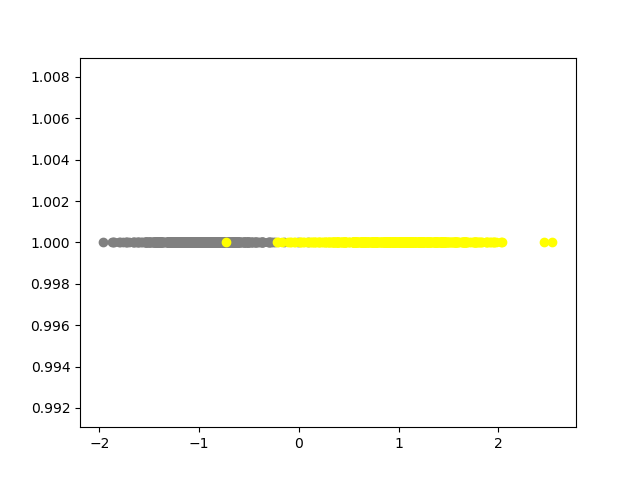


Fig 2.2: Just the projected values

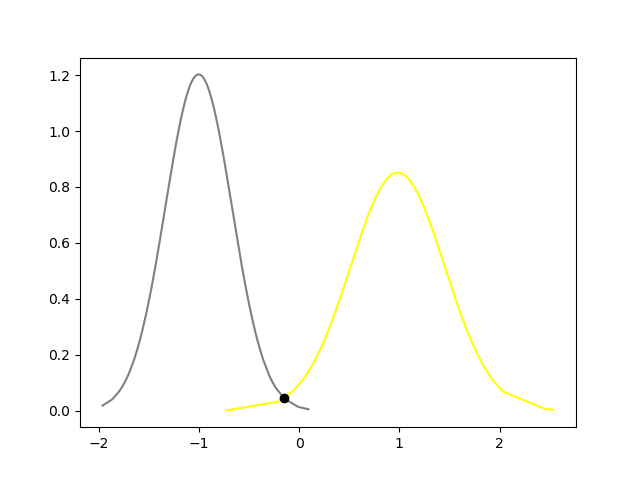


Fig 2.3 : Normal Curves and their point of intersection

**Data Set 3:**

The classes were linearly separable in case of dataset 3. The normalised “w” values are [-0.01760377, 0.99984504] and the Threshold value is-0.39301273677447685. In Fig 3.1 the points are triangles and rectangles. The circles in the centre are the projections of those points on the line. In Fig 3.2 we can see the plot of only projections. In Fig 3.3, we can see the normal graphs of projections and their point of intersection represented in black colour.

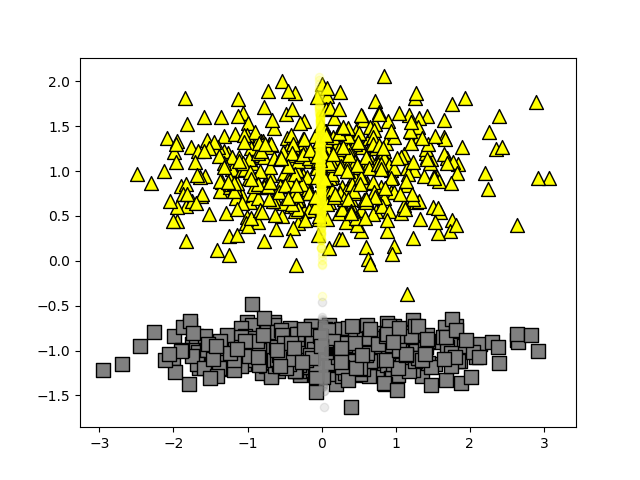


Fig3.1: Points (triangles and squares) and their projections (hued circles)

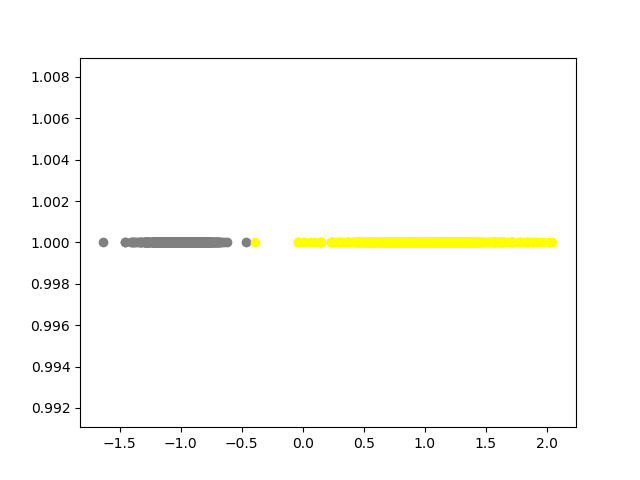


Fig 3.2: Just the projected values

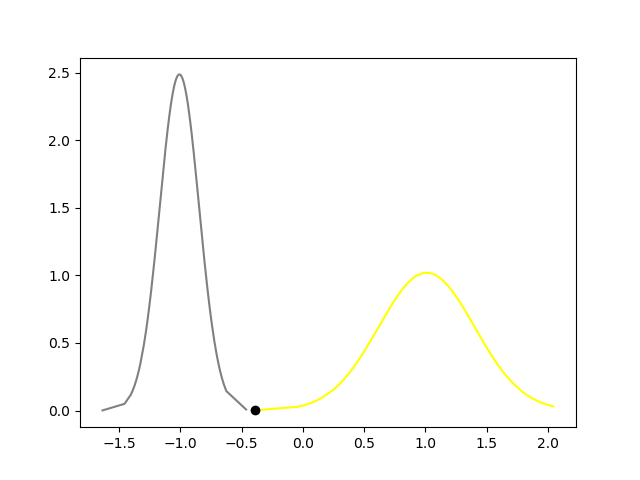


Fig 3.3 : Normal Curves and their point of intersection

**Perceptron**

We cycle through the training patterns in turn, and for each patternwe evaluate the perceptron function activation = sum(weight\_i \* x\_i) + bias. If the pattern is correctly classified, then the weight vector remains unchanged, whereas if it is incorrectly classified, then for class *C*1 we add the vector ***φ***(**x**) onto the current estimate of weight vector **w** while for class *C*2 we subtract the vector ***φ***(**x**) from **w**.

We set the learning rate to 0.01 and the threshold value ( Number of Epochs) to 100 after a bit of experimentation.

The initialisation of the parameter weights dosen’t matter and the weights will be automatically updated after each iteration and will converge as long as the data set if linearly separable. The points belonging to class1 ( had a target value of 1) were colored blue and the points that belonged to the other class were colored red in the plots.

Python libraries used: numpy, matplotlib, pandas, imageio(for making the gif)

**Data Set 1:**

Since the data set was clearly separable, the perceptron algorithm quickly converged and resulted in the weight vector having values [-0.6,0.09,1.13] where the first value is the value of the bias.

**Gif Link:**

<https://drive.google.com/file/d/1rLX2hDbnDQlpbLXBJIk7jLNZpb2frye1/view?usp=sharing>

**Data Set 2:**

For this data set, the points belonging to the two classes were intermixed and so they could not be linearly separable and hence the perceptron algorithm failed to converge for any threshold/ learning rate values.

The weight vector when we stopped the program had a value [0.2,0.09449557,0.8850115 ] where the first value is the value of the bias.

Though the algorithm couldn't converge , the resulting decision boundary successfully classified almost all the points.

**Gif Link:**

<https://drive.google.com/file/d/1CHVRRMosUH-R3eRFK2CEFGNAH27ALNIL/view?usp=sharing>

**Data Set 3:**

The perceptron algorithm converged immediately (within the first iteration) for this set of points as they were visibly separable. The converged weight vector had values [0.4,0.180,1.188] where the first value is the value of the bias.

**Gif Link:**

<https://drive.google.com/file/d/1lPrfCnc9Phd3elM2hZhgoaYn8xgkmlEX/view?usp=sharing>

**Impact of training data order**

As long as the data set is linearly separable, the order of the data set doesn't matter and the converged weights will have the same value.

**Impact of parameter initialisation**

The result on the training set doesn't matter but different initial parameters give different weights when converged.