

# IT581 Adversarial Machine Learning

## Lab Assignment - 01

### Group 05

2021 17013: Shaikh Faizan Ahmed

2021 17014: Sonam Bharti

### Question 1.

Write a function that plots for a given norm  $p$ . Use this function to plot for  $p = 1, 2, \infty$ .

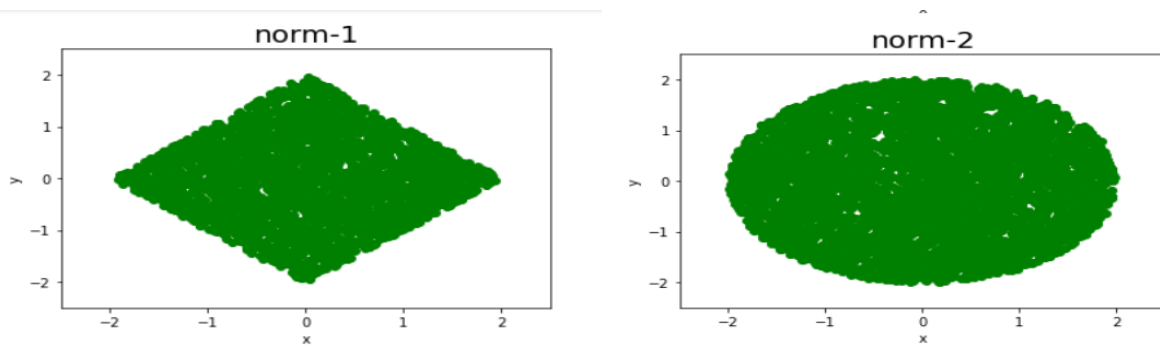
### Answer

#### code

```
1 #import required libraries
2 import numpy as np
3 import pylab
4
5 #define function to find norm
6 def plot_p_norm(p):
7     for i in range(6000):
8         x = np.array([np.random.rand()*4-2, np.random.rand()*4-2])
9
10        if np.linalg.norm(x,p) < 2:
11            pylab.plot(x[0], x[1], 'go')
12    pylab.title(label=f"norm-%f"%p,fontsize=20)
13    pylab.xlabel('x')
14    pylab.ylabel('y')
15    pylab.axis([-2.5, 2.5, -2.5, 2.5])
16    pylab.show()
17
18
19
20 p = np.array([1,2,3,np.inf])
21 for i in p:
22     plot_p_norm(i)
```

Listing 1: Question 1

### Output



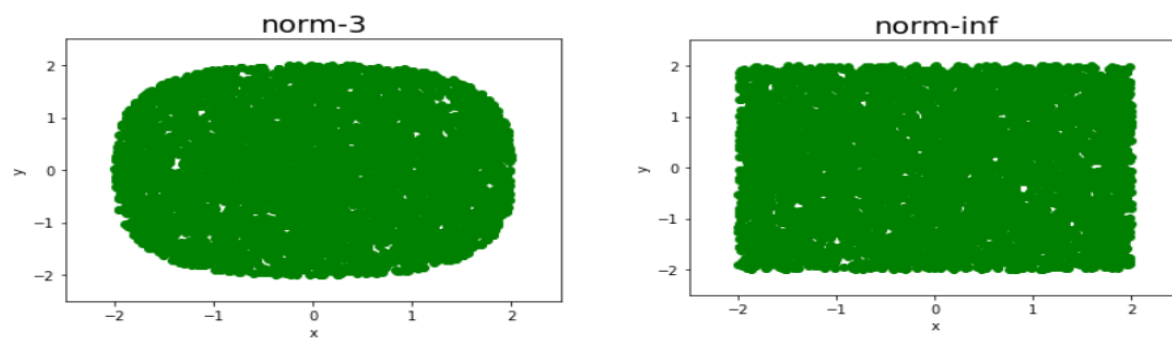


Figure 1: norm Plots

### Observation/ Justification

Norms are a way of measuring the "length" of vectors, matrices, etc. Mathematically, a norm is any function  $f$  that satisfies....

- $f(x) = 0 \Rightarrow x = 0$
- $f(x + y) \leq f(x) + f(y)$  (*TriangleInequality*)
- $\forall a \in \mathbb{R}, f(ax) = |a|f(x)$  (*Linearity*)

The most commonly used norms are clubbed under p-norms (or l-norms) family where  $p$  is any number greater or equal to 1.

Formula to calculate the p-form of vector  $x$ ...

$$\|x\|_p = (x_1^p + x_2^p + \dots + x_n^p)^{1/p}$$

It can be also written as.....

$$\|x\|_p = \left( \sum_{i=1}^n x_i^p \right)^{1/p}$$

where,

$p = 1$  for  $L^1$  norm

$p = 2$  for  $L^2$  norm

$p = 3$  for  $L^3$  norm

$p = \infty$  for  $L^\infty$  norm

Conclusion: The function we have written in the above code follows this norms equation and returns a norm value for each vector and plot a graph upto 6000 vectors for each required value of  $p$ .

### Question 2.

You will work with a widely used Iris dataset. The Iris Dataset contains four features (sepal length, sepal width, petal length, and petal width) of 50 samples of three species of Iris (Iris setosa, Iris virginica, and Iris versicolor). For this lab we'll use only two species. Visualize the data in the Iris Dataset by considering maximum combinations of two features in a 2D plot. Use red and green colors for labeling the two classes: Iris setosa and Iris versicolor, respectively. Comment on whether any two classes among the three can be separated by a line? Plot its linear discriminant function as line using two features. And also plot hyperplane using 3 features....

## Answer

### code

```
1 import numpy as np
2 import pandas as pd
3 import plotly.express as px
4 import plotly.graph_objects as go
5 from sklearn.preprocessing import OrdinalEncoder
6 from sklearn.preprocessing import LabelEncoder as le
7 from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
8
9
10 iris_dat = pd.read_csv("Iris.csv")
11
12 new_iris = iris_dat[(iris_dat.Species == 'Iris-setosa') | (iris_dat.Species == 'Iris-
    versicolor')]
13 iris_df = new_iris.iloc[:,1:6]
14 iris_data = new_iris.iloc[:,1:5]
15
16 fig1 = px.scatter(iris_df, x=iris_df['SepalLengthCm'],
17                    y=iris_df['SepalWidthCm'],
18                    labels={"x": "Sepal Length", "y": "Sepal Width"},
19                    opacity=1,
20                    color=iris_df['Species'],
21                    color_discrete_sequence=["red", "green"])
22 fig1.update_layout(title_text="Sepal Length VS Sepal Width",
23                    xaxis_title="Sepal Length in cm",
24                    yaxis_title="Sepal Width in cm")
25 fig1.show()
26
27 fig2 = px.scatter(iris_df, x=iris_df['PetalLengthCm'],
28                    y=iris_df['PetalWidthCm'],
29                    labels={"x": "Petal Length", "y": "Petal Width"},
30                    opacity=1,
31                    color=iris_df['Species'],
32                    color_discrete_sequence=["red", "green"])
33 fig2.update_layout(title_text="Petal Length VS Petal Width",
34                    xaxis_title="Petal Length in cm",
35                    yaxis_title="Petal Width in cm")
36 fig2.show()
37
38
39 fig3 = px.scatter(iris_df, x=iris_df['SepalLengthCm'],
40                    y=iris_df['PetalLengthCm'],
41                    labels={"x": "Sepal Length", "y": "Petal Length"},
42                    opacity=1,
43                    color=iris_df['Species'],
44                    color_discrete_sequence=["red", "green"])
45 fig3.update_layout(title_text="Sepal Length VS Petal Length",
46                    xaxis_title="Sepal Length in cm",
47                    yaxis_title="Petal Length in cm")
48 fig3.show()
49
50
51 fig4 = px.scatter(iris_df, x=iris_df['SepalLengthCm'],
52                    y=iris_df['PetalWidthCm'],
53                    labels={"x": "Sepal Length", "y": "Petal Width"},
54                    opacity=1,
55                    color=iris_df['Species'],
56                    color_discrete_sequence=["red", "green"])
57 fig4.update_layout(title_text="Sepal Length VS Petal Width",
58                    xaxis_title="Sepal Length in cm",
59                    yaxis_title="Petal Width in cm")
60 fig4.show()
61
62
63 fig5 = px.scatter(iris_df, x=iris_df['SepalWidthCm'],
64                    y=iris_df['PetalLengthCm'],
65                    labels={"x": "Sepal Width", "y": "Petal Length"},
66                    opacity=1,
67                    color=iris_df['Species'],
68                    color_discrete_sequence=["red", "green"])
69 fig5.update_layout(title_text="Sepal Width VS Petal Length",
70                    xaxis_title="Sepal Width in cm",
```

```

71         yaxis_title="Petal Length in cm")
72 fig5.show()
73
74 fig6 = px.scatter(iris_df, x=iris_df['SepalWidthCm'],
75                    y=iris_df['PetalWidthCm'],
76                    labels={"x": "Sepal Width", "y": "Petal Width"},
77                    opacity=1,
78                    color=iris_df['Species'],
79                    color_discrete_sequence=["red", "green"])
80
81 fig6.update_layout(title_text="Sepal Width VS Petal Width",
82                   xaxis_title="Sepal Width in cm",
83                   yaxis_title="Petal Width in cm")
84 fig6.show()
85
86
87 A = iris_data.to_numpy()
88 X = A[:, :2]
89 y = iris_df.iloc[:, -1]
90 y = le().fit_transform(y)
91
92 lda = LinearDiscriminantAnalysis()
93 lda.fit_transform(X, y)
94
95 i, w0, w1 = lda.intercept_[0], lda.coef_[0][0], lda.coef_[0][1]
96 x0 = np.array([X[:, 0].min(), X[:, 0].max()])
97 y0 = -(i + x0 * w0) / w1
98
99 fig7 = px.scatter(iris_df, x=iris_df['SepalLengthCm'], y=iris_df['SepalWidthCm'],
100                  labels={"x": "Sepal Length", "y": "Sepal Width"},
101                  opacity=1,
102                  color=iris_df['Species'],
103                  color_discrete_sequence=["red", "green"])
104 fig7.add_traces(go.Scatter(x=x0, y=y0, name="LDA"))
105 fig7.update_layout(title_text="Linear Discriminant Function as Line using Two Features",
106                   xaxis_title="Sepal Length in cm",
107                   yaxis_title="Sepal Width in cm")
108 fig7.show()
109
110
111 X1 = A[:, :3]
112 lda1 = LinearDiscriminantAnalysis()
113 lda1.fit_transform(X1, y)
114
115
116 i, w0, w1, w2 = lda1.intercept_[0], lda1.coef_[0][0], lda1.coef_[0][1], lda1.coef_[0][2]
117 x0 = np.array([X[:, 0].min(), X[:, 0].max()])
118 x1 = np.array([X[:, 1].min(), X[:, 1].max()])
119
120 x0, x1 = np.meshgrid(x0, x1)
121 y0 = -(i + x0 * w0 + x1 * w1) / w2
122
123 fig8 = px.scatter_3d(iris_df,
124                     x='SepalLengthCm', y='SepalWidthCm', z='PetalLengthCm',
125                     color='Species',
126                     color_discrete_sequence=["red", "green"],
127                     height=900, width=900)
128 fig8.add_traces(go.Surface(x=x0, y=x1, z=y0, name="LDA", colorscale='blues'))
129 fig8.update_layout(title_text="Linear Discriminant Function as Line using Three Features",
130                   showlegend=True,
131                   legend=dict(orientation="h", yanchor="top", y=0, xanchor="center", x=0.5))
132 fig8.show()

```

Listing 2: Question 2

## Output

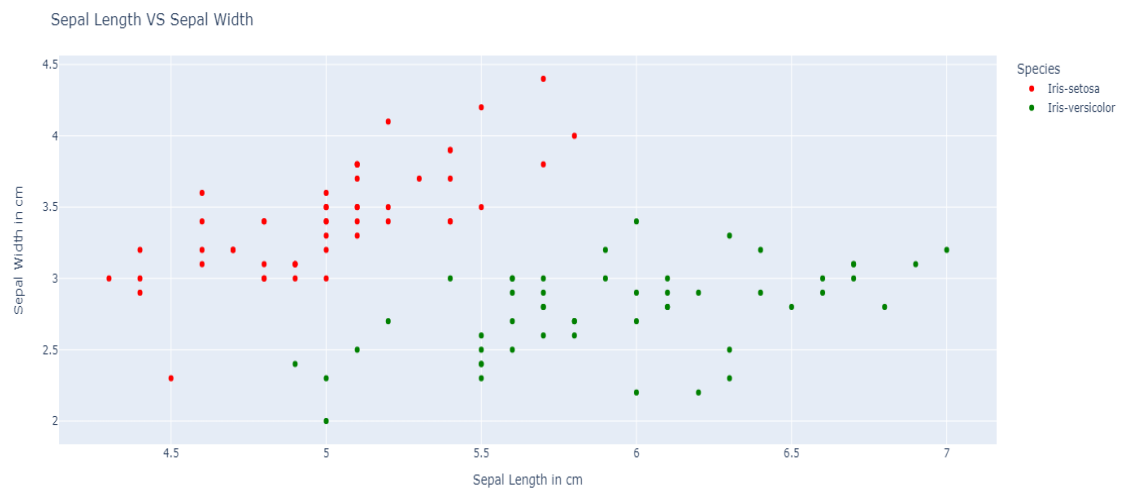


Figure 2: Sepal-Sepal-features

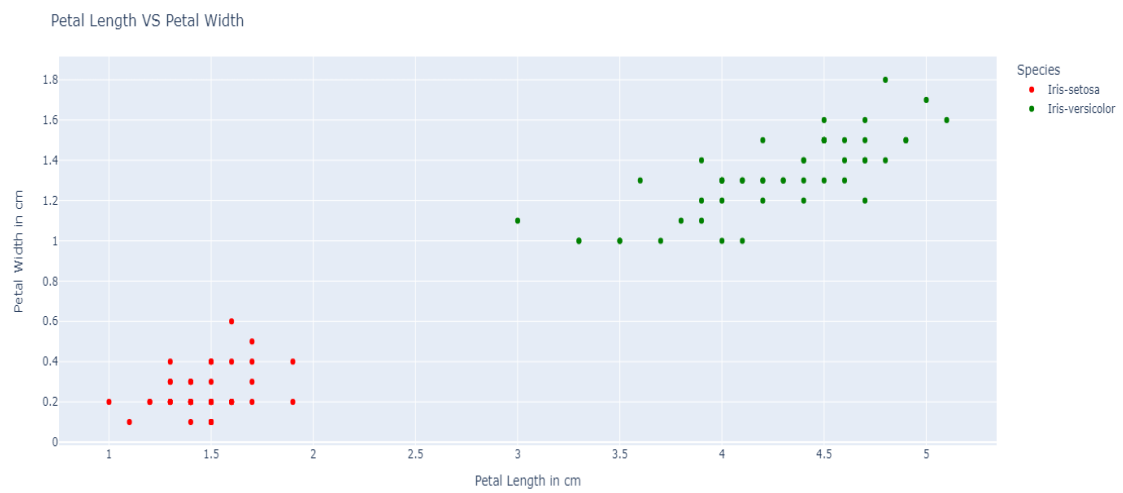


Figure 3: Petal-Petal-features

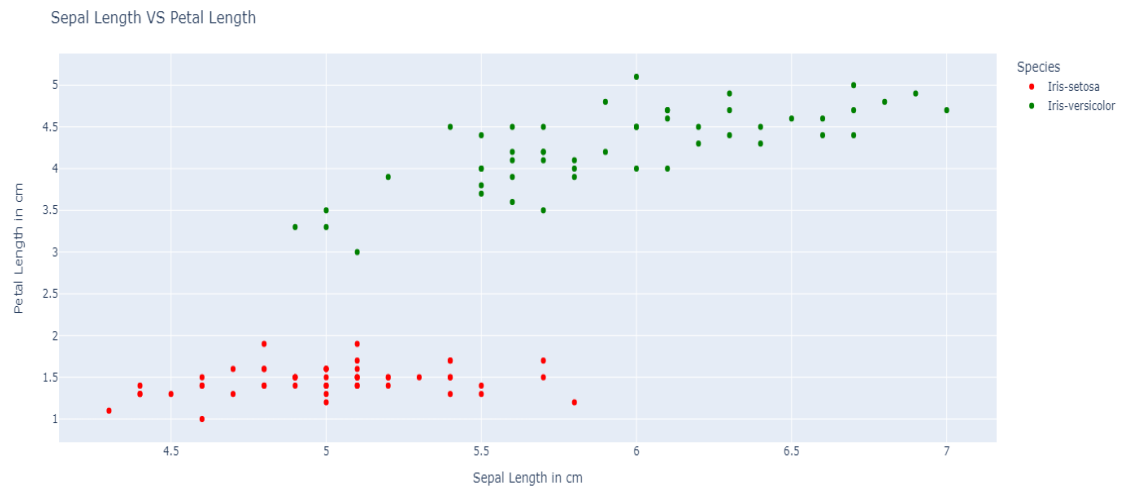


Figure 4: Petal-Sepal-length

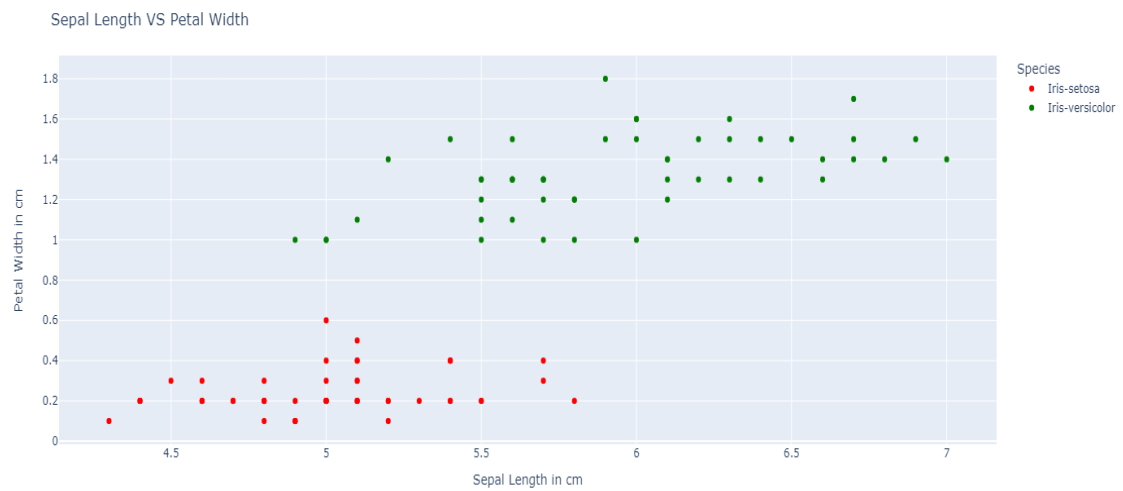


Figure 5: SepalLen-PetalWid-features

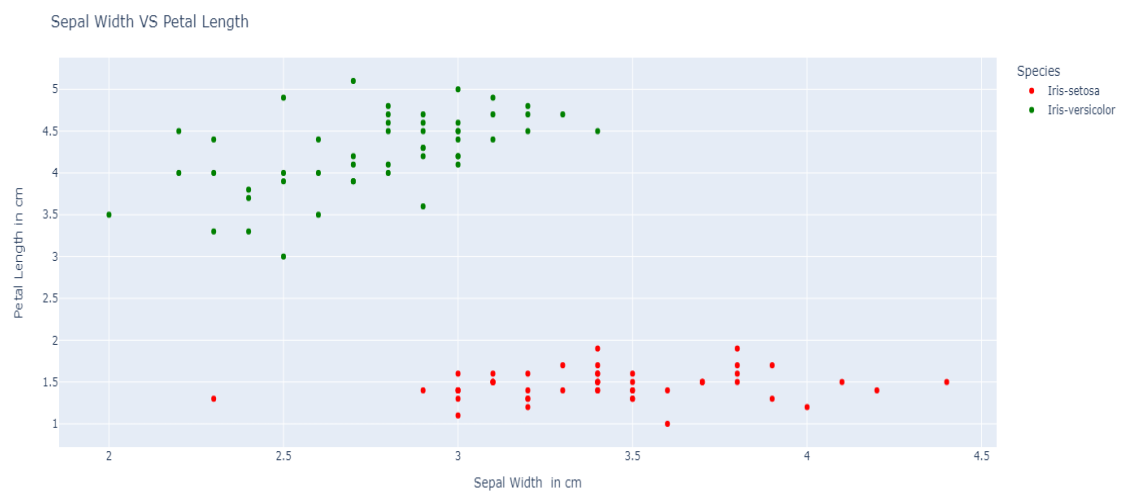


Figure 6: PetalLen-SepalWid-length

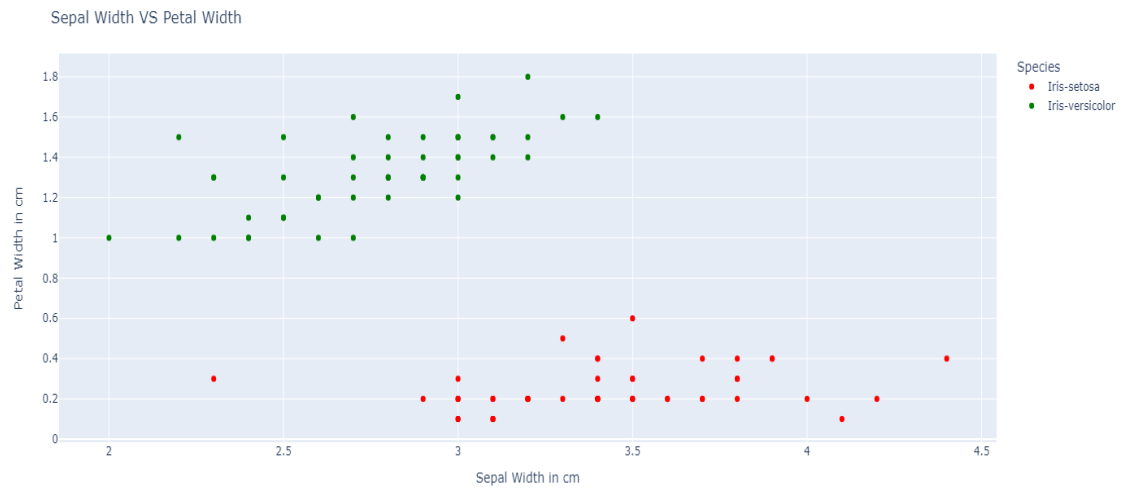


Figure 7: SepalWid-PetalWid-features



Figure 8: Scatter plot: Sepal-features

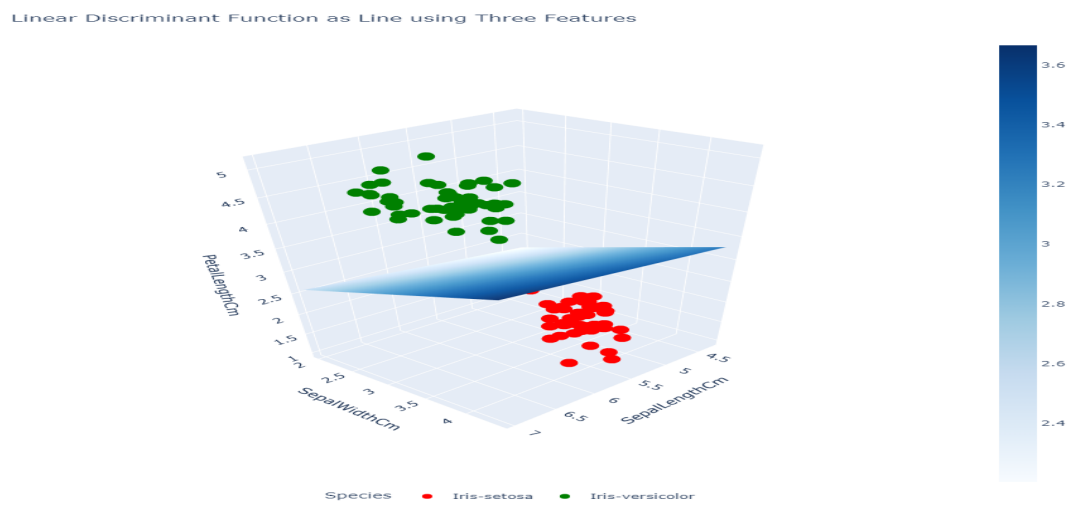


Figure 9: 3D-Plot:90

Linear Discriminant Function as Line using Three Features

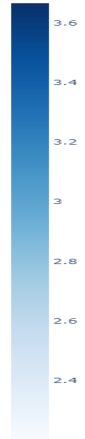
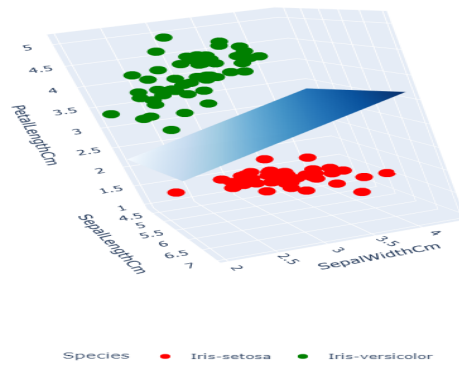


Figure 10: 3D-Plot:25

Linear Discriminant Function as Line using Three Features

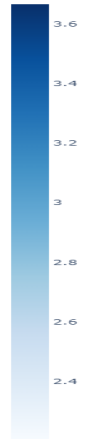
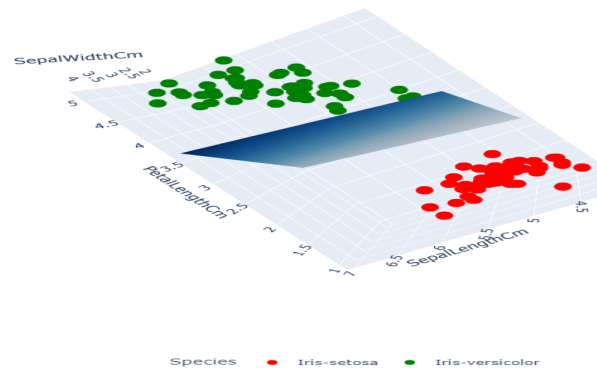


Figure 11: 3D-Plot:135

Linear Discriminant Function as Line using Three Features

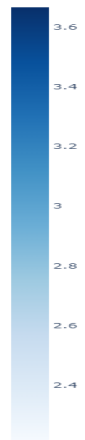
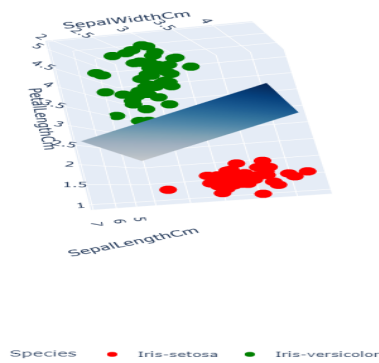


Figure 12: 3D-Plot:285



## Definition

A discriminant function that is a linear combination of the components of  $x$  can be written as

$$g(x) = w^T x + w_0$$

where  $w$  is the weight vector and  $w_0$  the bias or threshold weight.

A two-category classifier with a discriminant function of the form uses following rule:

- decide  $w_1$  if  $g(x) > 0$  and  $w_2$  if  $g(x) < 0$ .
- If  $g(x) = 0 \Rightarrow x$  is assigned to either class.
- The equation  $g(x) = 0$  defines the decision surface that separates points assigned to the category  $w_1$  from points assigned to the category  $w_2$ .
- When  $g(x)$  is linear, the decision surface is a hyperplane.
- If  $x_1$  and  $x_2$  are both on the decision surface, then:

$$w^T x_1 + w_0 = w^T x_2 + w_0$$

$$w^T (x_1 - x_2) = 0$$

- $w$  is normal to any vector lying on the hyperplane.

## Observation/ Justification

After visualizing the data by considering the maximum combination of two-features in a 2D-plot we can clearly see that it can be divided into 2-clusters using LDA (Ref. fig.2-fig.7). Also we can generate a line using LDF to clearly visualise the separation.(Ref. fig.8) The 3D-plot shows the hyperplane using three features.