3) Let us assume our training data D, to have N samples. The dimensions of which sample is d

$$\vec{x}_i = \begin{bmatrix} x_1^{(i)} \\ x_2^{(i)} \\ x_3^{(i)} \\ \vdots \\ x_d^{(i)} \end{bmatrix}$$

Let the mean of the sample be

$$\mu = \frac{1}{N} \sum_{i=1}^{N} \vec{x}_i$$

Let ~~Σ~~ σ be the covariance matrix.

$$\sigma\ \cancel{\Sigma} = \frac{1}{N} \sum_{i=1}^{N} (\vec{x}_i - \mu)(\vec{x}_i - \mu)^T$$

Let $\vec{p}$ be a non zero vector s.t $\vec{p} \in \mathbb{R}^d$

$$p^T \sigma p = p^T \left[\frac{1}{N} \sum_{i=1}^{N} (\vec{x}_i - \mu)(\vec{x}_i - \mu)^T \right] \vec{p}$$

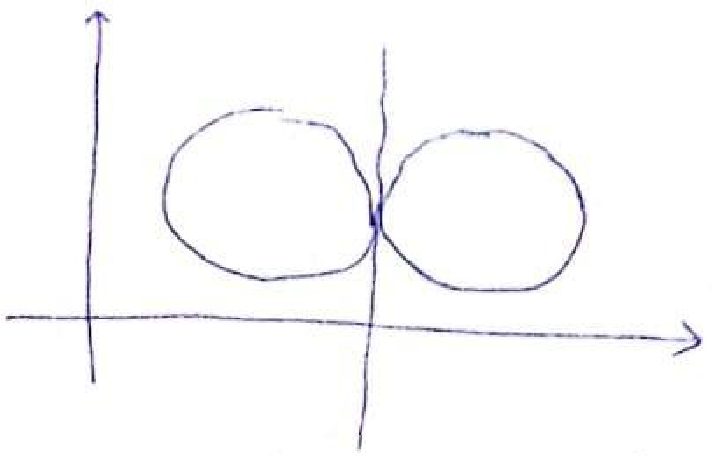
$$= \frac{1}{N} \left[\sum_{i=1}^{N} p^T (\vec{x}_i - \mu)(\vec{x}_i - \mu)^T \vec{p} \right]$$

$$= \frac{1}{N} \sum_{i=1}^{N} \left[(\vec{x}_i - \mu)^T \vec{p} \right]^2$$

It is square of a real values ∴ +ve. and $\vec{p}$ is a non zero vector

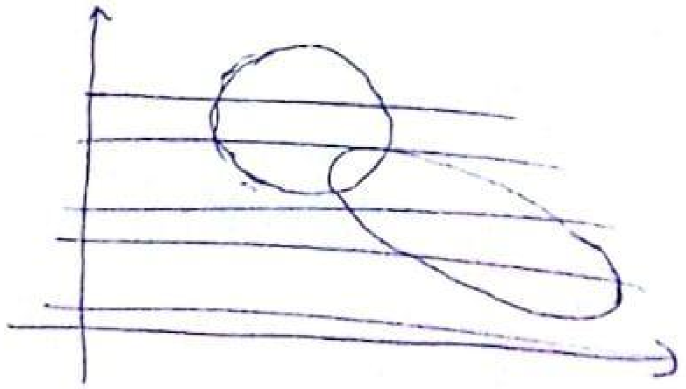
$\therefore$ σ is PSD.

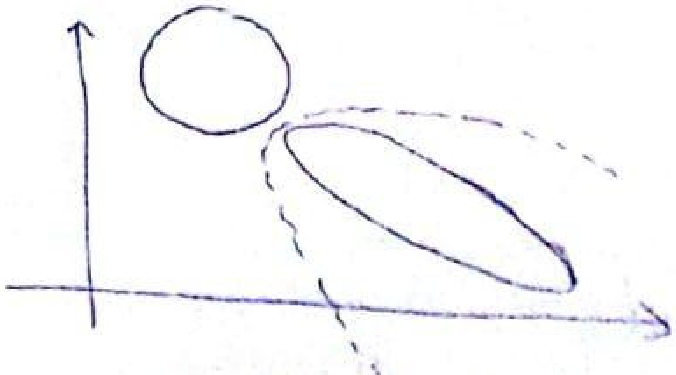
1) b) i) Since, covariance matrices are same, corresponds to a visualisation as below in 2D.



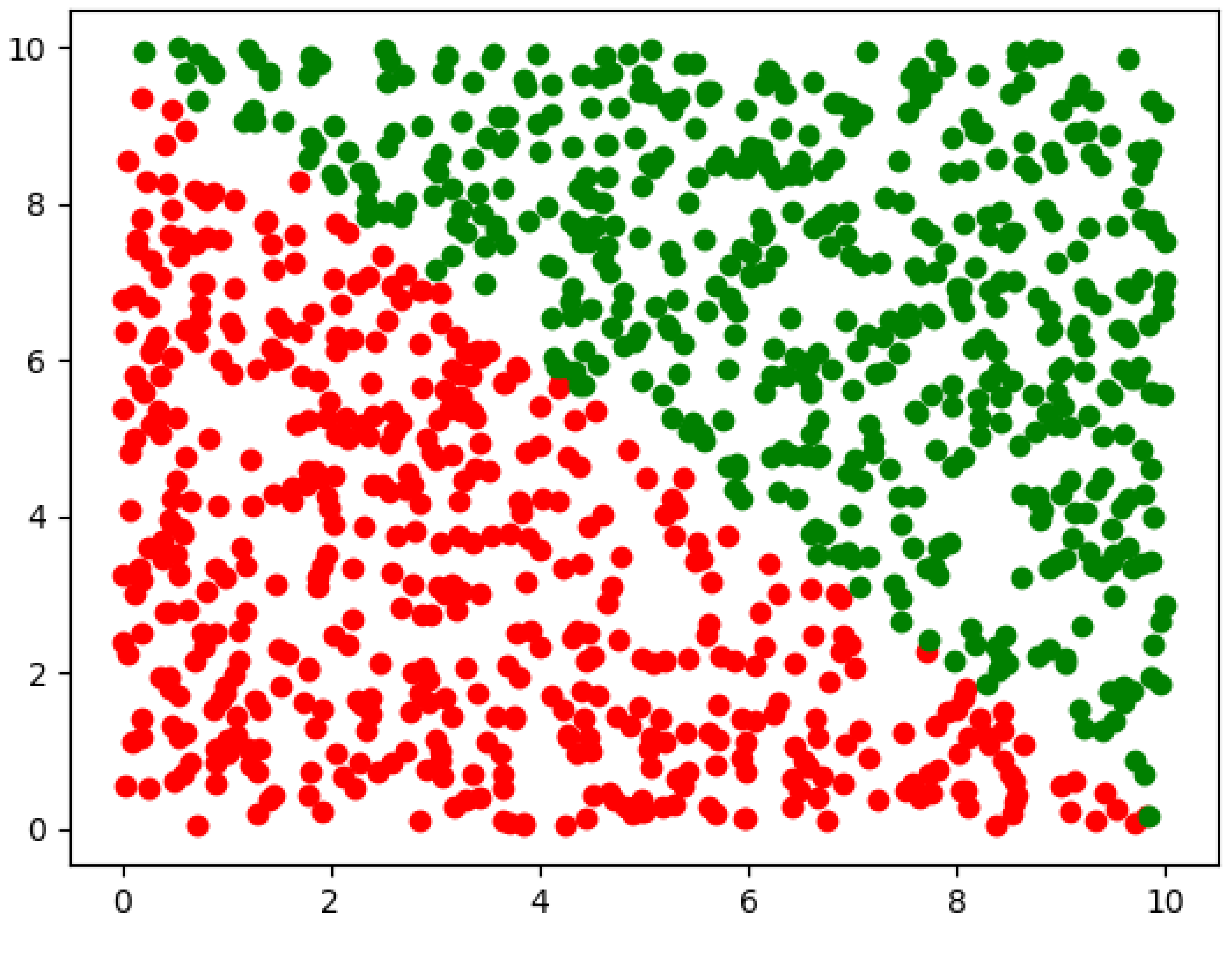
$\therefore$ We expect a ~~strai~~ linear boundary to separate the two classes.

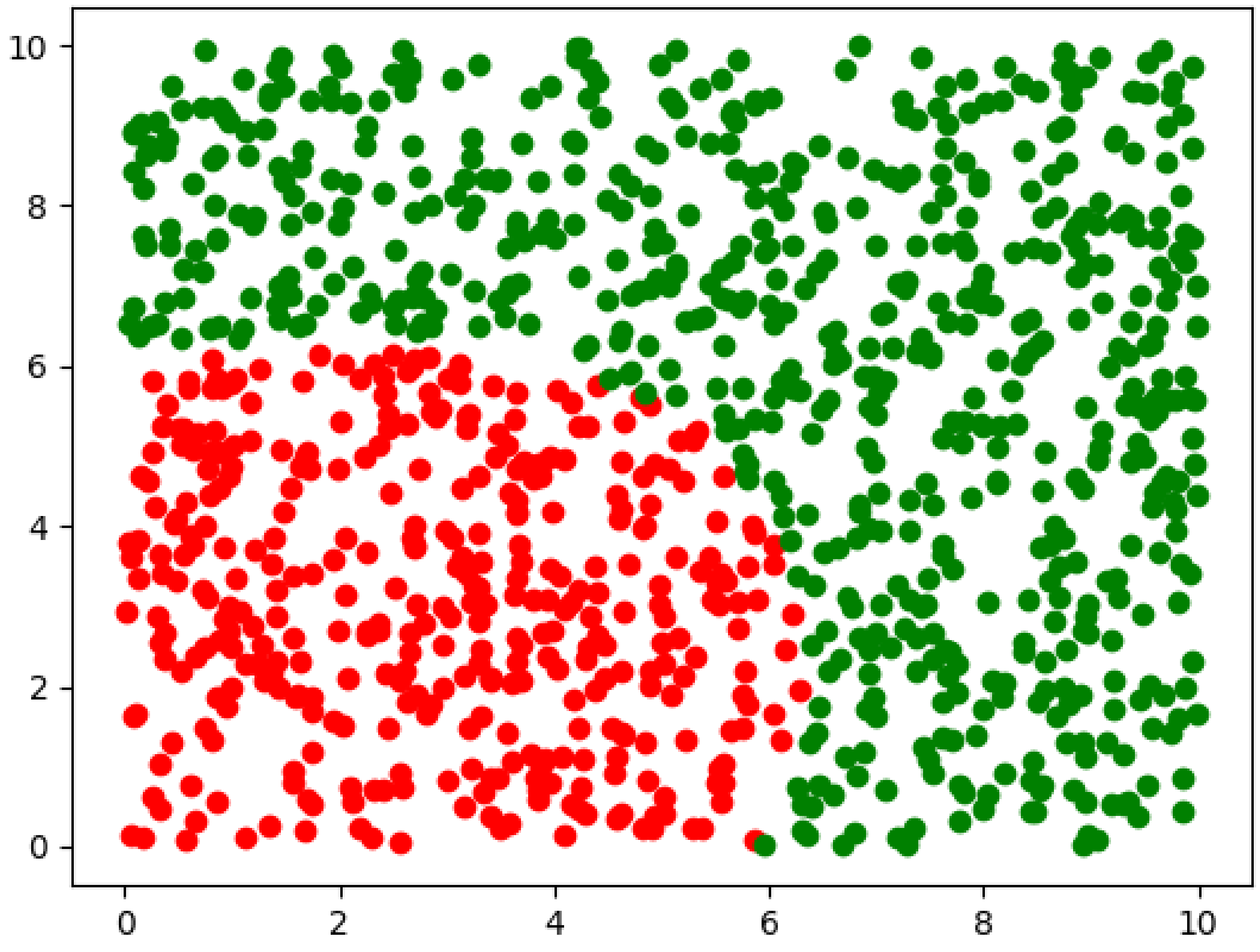
ii) $\because$ Covariances are different, corresponds to a situation shown below:



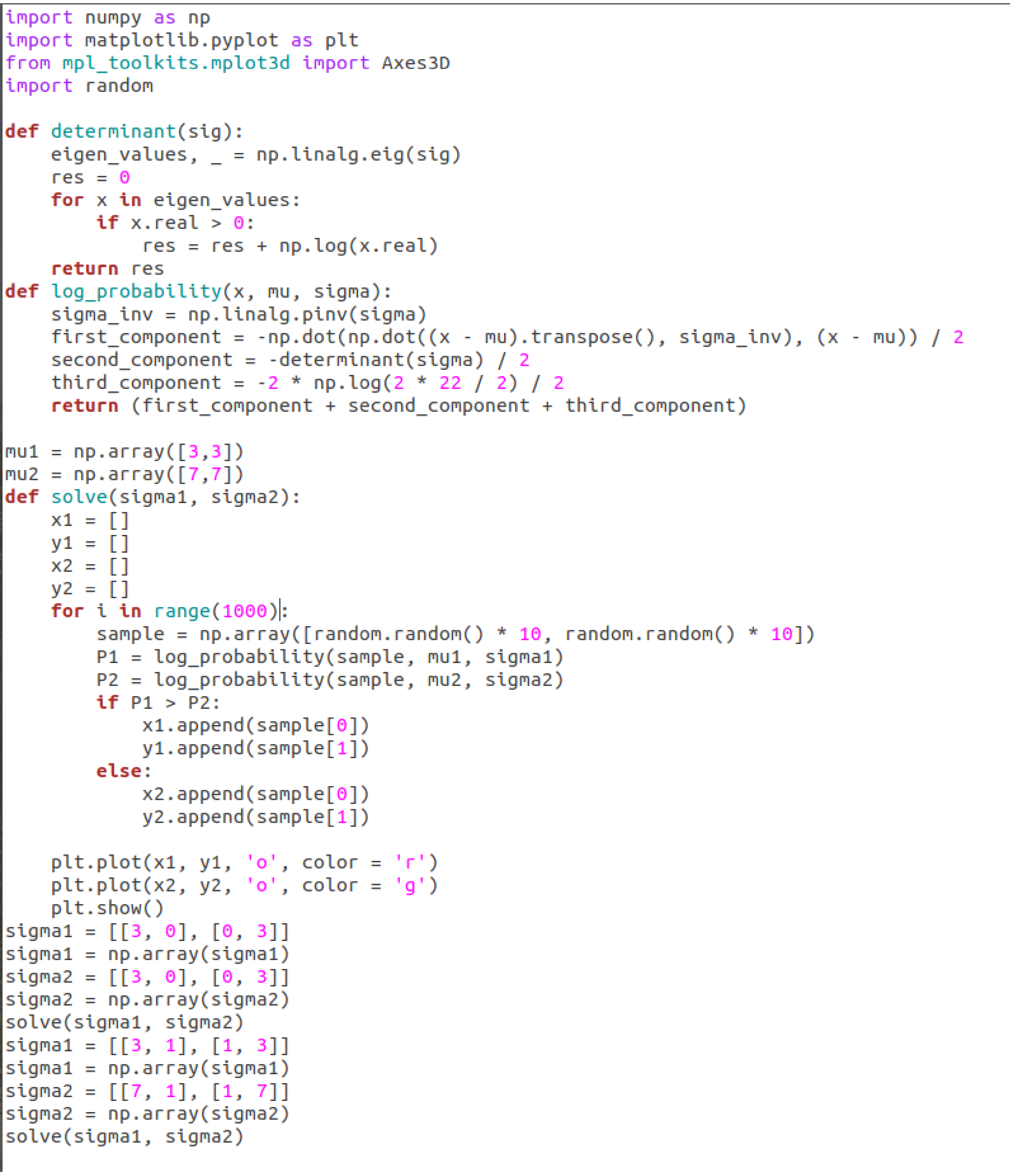


$\therefore$ We expect a quadratic boundary.


0
2
4
6
8
10
0
2
4
6
8
10



0
2
4
6
8
10
0
2
4
6
8
10



```python
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import random

def determinant(sig):
    eigen_values, _ = np.linalg.eig(sig)
    res = 0
    for x in eigen_values:
        if x.real > 0:
            res = res + np.log(x.real)
    return res
def log_probability(x, mu, sigma):
    sigma_inv = np.linalg.pinv(sigma)
    first_component = -np.dot(np.dot((x - mu).transpose(), sigma_inv), (x - mu)) / 2
    second_component = -determinant(sigma) / 2
    third_component = -2 * np.log(2 * 22 / 2) / 2
    return (first_component + second_component + third_component)

mu1 = np.array([3,3])
mu2 = np.array([7,7])
def solve(sigma1, sigma2):
    x1 = []
    y1 = []
    x2 = []
    y2 = []
    for i in range(1000):
        sample = np.array([random.random() * 10, random.random() * 10])
        P1 = log_probability(sample, mu1, sigma1)
        P2 = log_probability(sample, mu2, sigma2)
        if P1 > P2:
            x1.append(sample[0])
            y1.append(sample[1])
        else:
            x2.append(sample[0])
            y2.append(sample[1])

    plt.plot(x1, y1, 'o', color = 'r')
    plt.plot(x2, y2, 'o', color = 'g')
    plt.show()
sigma1 = [[3, 0], [0, 3]]
sigma1 = np.array(sigma1)
sigma2 = [[3, 0], [0, 3]]
sigma2 = np.array(sigma2)
solve(sigma1, sigma2)
sigma1 = [[3, 1], [1, 3]]
sigma1 = np.array(sigma1)
sigma2 = [[7, 1], [1, 7]]
sigma2 = np.array(sigma2)
solve(sigma1, sigma2)
```