EE559-Homework#8-Lei Lei

1. **Universal Function Approximation**

a) $\mathcal{G}_M = \{ x_i = i/(M-1)\}_{\mathrm{i}=0}^{\mathrm{M-1}}$

How many parameters are in the hidden layer? How many total parameters in the ANN? Specify how each of the weights and biases in both layers should be chosen to obtain the approximation.

For the hidden layer, there would be 2(M-1) weights.

For the second layer, there would be M weights. So, for the ANN there would be 3M-2 parameters.

The approximation function is:

$$\hat{f}(x) = w_{10}^{(2)} v_0(x) + \sum_{m=1}^{M-1} w_{1m}^{(2)} v_m(x)$$

In the problem's case, $v_m(x) = h_{ReLU}(w_{m1}^{(1)} x + w_{m0}^{(1)})$

Let ReLU parameter $b=1$, let $w_{m1}^{(1)} = 1, \forall m \Rightarrow v_m(x) = \max\{0, x + w_{m0}^{(1)}\}$

$w_{m0}^{(1)}$ is the bias in layer 1.

Order the grid, so that $x_0 < x_1 < \cdots x_{n-1}$

Let $\hat{f}_{n-1}(x)$ be the function approximation based on data points $x_0, x_1 \cdots x_{n-1}$

Set $\hat{f}_0(x) = y_0$, $w_{10}^{(2)} = y_0$, $v_0(x) = 1$.

For $\hat{f}_1(x)$, use $\hat{f}_0(x)$ and (x_1, y_1)

$\therefore \hat{f}_1(x) = \hat{f}_0(x) + w_{11}^{(2)} \max\{0, x + w_{10}^{(1)}\}$

Choose $w_{10}^{(1)} = -x_0$

$\therefore \hat{f}_1(x) = f_0(x) + w_{11}^{(2)} v_1(x)$

$\hookrightarrow \max\{0, x_1 - x_0\} = x_1 - x_0$

To make $f_1(x) = y_1$, $w_{11}^{(2)} = \dfrac{y_1 - y_0}{x_1 - x_0}$

Iterate until all the grid point, and get the final approximation

$$\hat{f}_n(x) = \hat{f}_{n-1}(x) + w_{1n}^{(2)} v_n(x), \quad v_n(x) = \{0, x - x_{n-1}\}$$

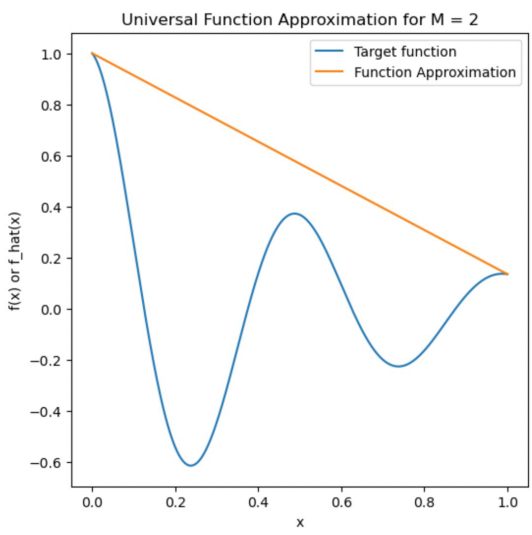
To make $\hat{f}_n(x) = y_n$, $w_{1n}^{(2)} = \dfrac{y_n - \hat{f}_{n-1}(x_n)}{x_n - x_{n-1}}$

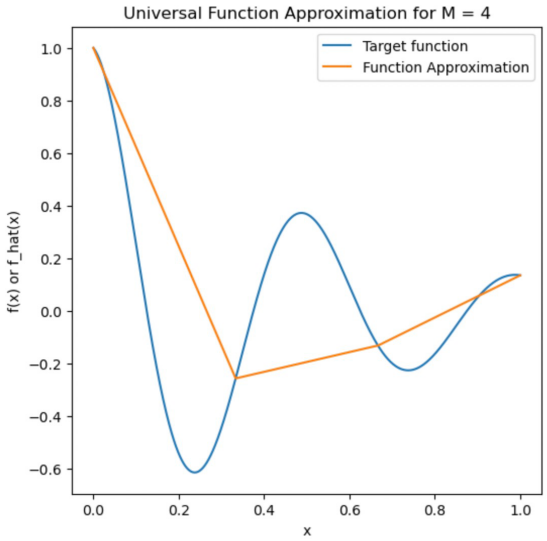
To conclude:

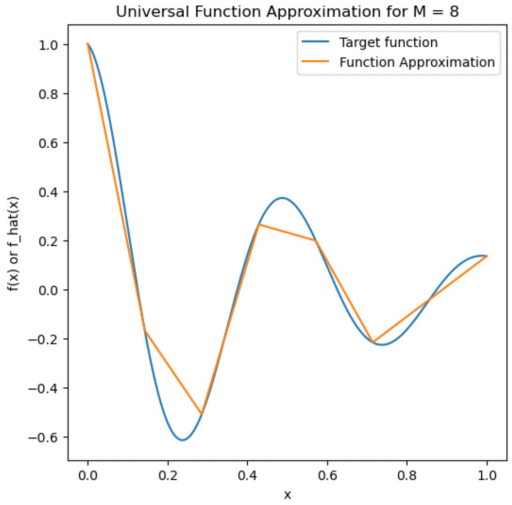
Layer 1 weights = 1, bias $w_{m0}^{(1)} = -x_{m-1} \quad \forall m > 1$

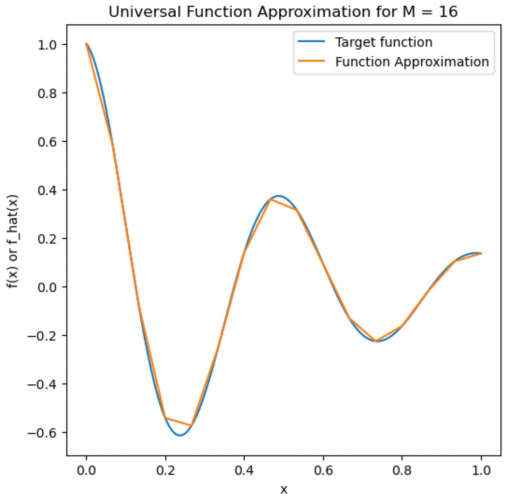
Output layer weights: $w_{1n}^{(2)} = \dfrac{y_n - \hat{f}_{n-1}(x_n)}{x_n - x_{n-1}}$, $n \geq 2$, bias $w_{10}^{(2)} = y_0$

(b)

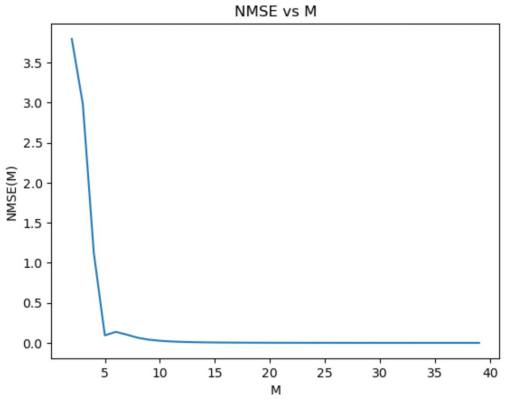

Universal Function Approximation for M = 2
Target function
Function Approximation
f(x) or f_hat(x)
x

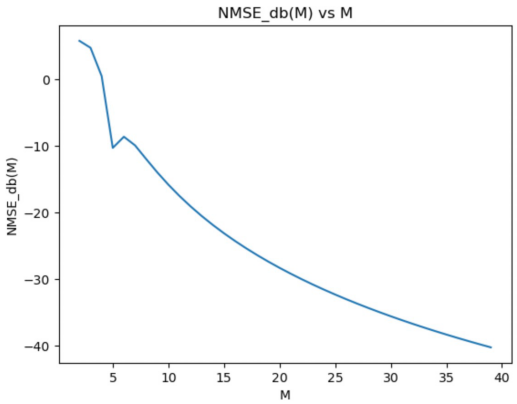

Universal Function Approximation for M = 4
Target function
Function Approximation
f(x) or f_hat(x)
x


Universal Function Approximation for M = 8
Target function
Function Approximation
f(x) or f_hat(x)
x


Universal Function Approximation for M = 16
Target function
Function Approximation
f(x) or f_hat(x)
x

(c)

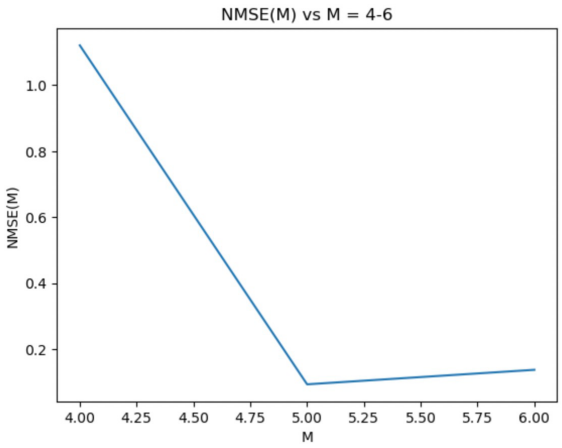

NMSE vs M
NMSE(M)
M

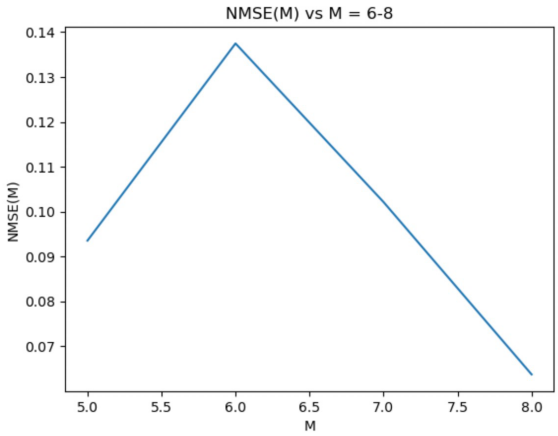

NMSE_db(M) vs M
NMSE_db(M)
M

(d) Interpretation of the NMSE results:

(i) Yes. This means that the approximation function is higher or lower than the original function. This is caused by an approximation function with a constant output regardless the input value.

(ii) No.





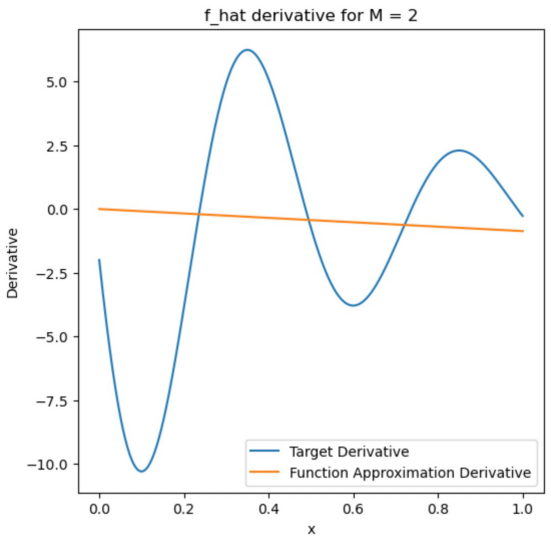
This is because the approximation function fit the original function too well but there exits a bump in the original function. When the approximation function increases after training the left hand side points of the bump, the original function decreases immediately and increases the NMSE. We are using the mean squared error (MSE) and normalized mean squared error (NMSE).

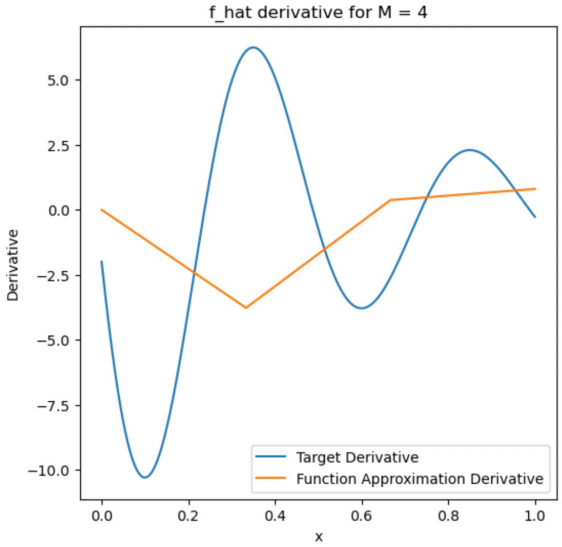
(iii)

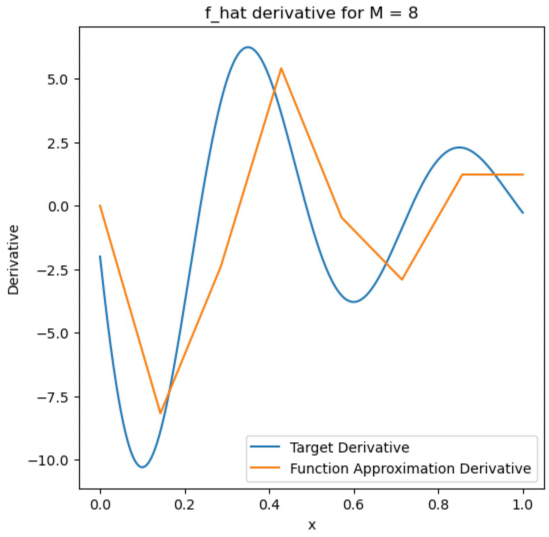
M = 39

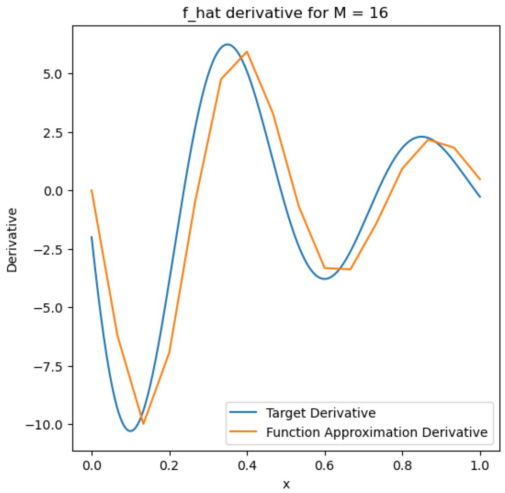
Total = 3 * M – 2 = 72 parameters.

(e)

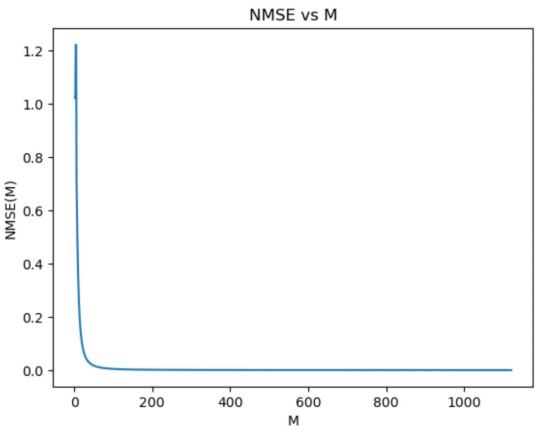

f_hat derivative for M = 2
Derivative
x
Target Derivative
Function Approximation Derivative

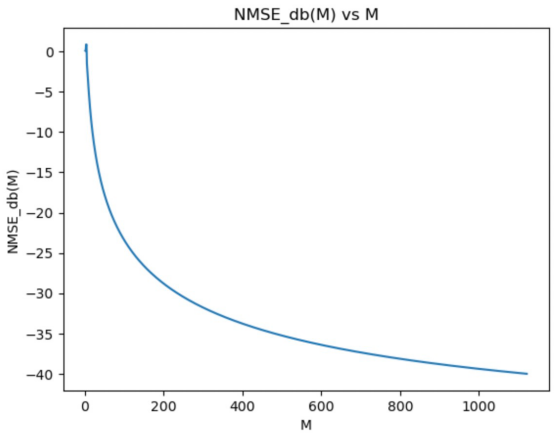

f_hat derivative for M = 4
Derivative
x
Target Derivative
Function Approximation Derivative


f_hat derivative for M = 8
Derivative
x
Target Derivative
Function Approximation Derivative


f_hat derivative for M = 16
Derivative
x
Target Derivative
Function Approximation Derivative

(f)

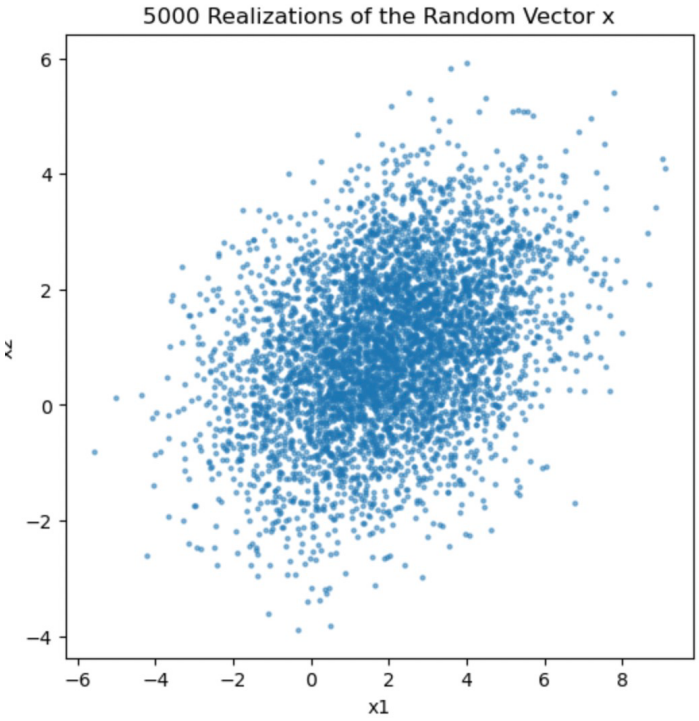

NMSE vs M
NMSE(M)
M


NMSE_db(M) vs M
NMSE_db(M)
M

(g) It’s not. Because the two points in the middle are on the two sides of a bump. The left one should increase but it doesn’t because it’s lower than the initial value. And the right hand side point should decrease. However, it is lower than the last points, the derivative increased instead.

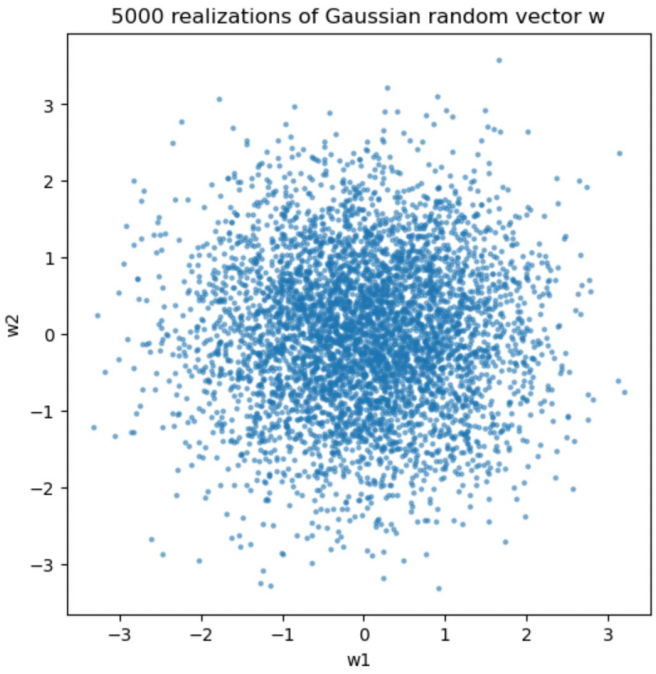
2. **Whitening and Simulation of Gaussian Random Vectors**

(a)

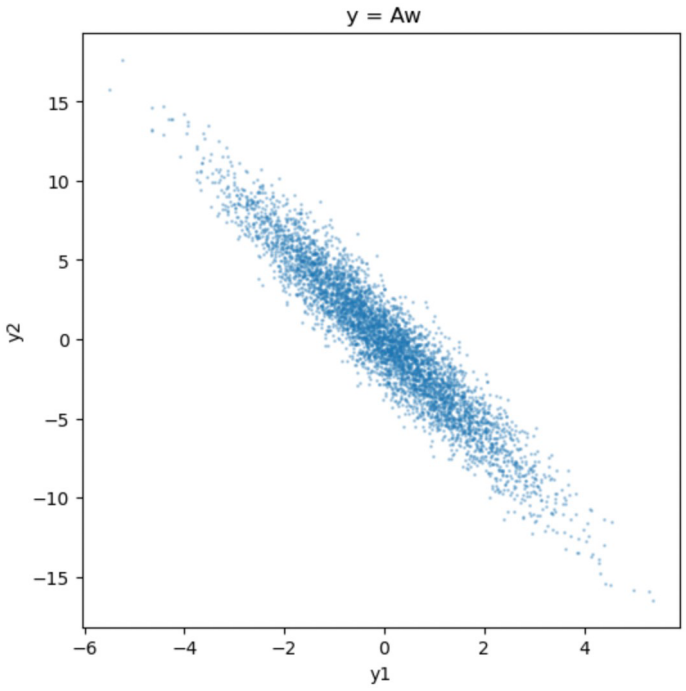


Use np.random.normal(0,1,size=(5000, 2)) to generate 5000 points, and then calculate eigvalues and eigvectors of the covariance matrix. Transform the realizations and add the mean to each realization to get the final points.

(b) (i)


5000 realizations of Gaussian random vector w
w1
w2

(ii)


y = Aw
y1
y2

2. $\underline{y} = \underline{\underline{A}}\,\underline{w}$

$E[\underline{y}] = E[\underline{\underline{A}}\,\underline{w}] = \underline{\underline{A}}\,E[\underline{w}]$

As we already know $E[\underline{w}] = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

$\therefore E[\underline{y}] = \underline{\underline{A}}\,E[\underline{w}] = \begin{bmatrix} -1 & -1 \\ 2 & 4 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

$Var[\underline{y}] = E[(\underline{y} - E[\underline{y}])(\underline{y} - E[\underline{y}])^T]$

$= E[\underline{\underline{A}}\,\underline{w}\,\underline{w}^T\,\underline{\underline{A}}^T]$

$= \underline{\underline{A}}\,E[\underline{w}\,\underline{w}^T]\,\underline{\underline{A}}^T$

$= \underline{\underline{A}}\,\underline{\underline{I}}\,\underline{\underline{A}}^T$

$= \begin{bmatrix} -1 & -1 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} -1 & 2 \\ -1 & 4 \end{bmatrix} = \begin{bmatrix} 2 & -6 \\ -6 & 20 \end{bmatrix}$

$\therefore Var[y_1] = 2 \quad Var[y_2] = 20 \quad \sigma_{y_1 y_2} = -6$

$r_{y_1 y_2} = \dfrac{\sigma_{y_1 y_2}}{\sigma_{y_1}\sigma_{y_2}} = \dfrac{-6}{\sqrt{2 \times 20}} = -0.94868$

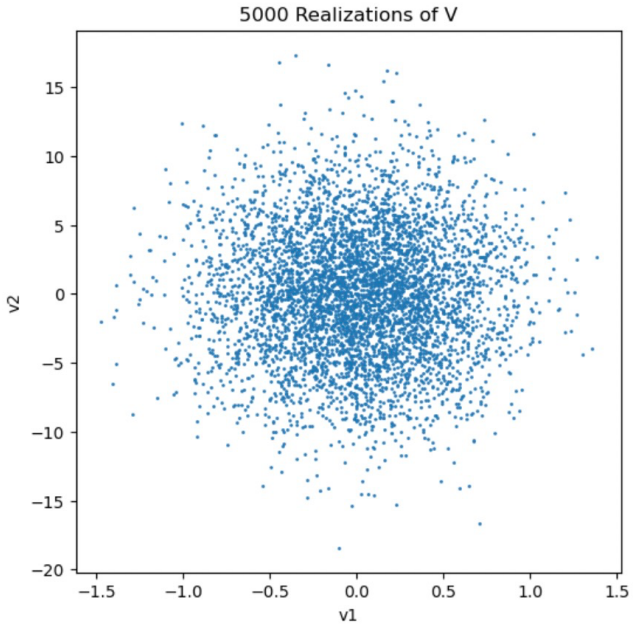
$\therefore y_1$ and y_2 are negatively correlated.

The sample covariance matrix is [[2.00033385 -5.99667272]
[-5.99667272 19.96204433]]

Comparing this to the covariance matrix of *y* as derived : [[2 -6],[-6 20]], we can found that they have very approximate value.

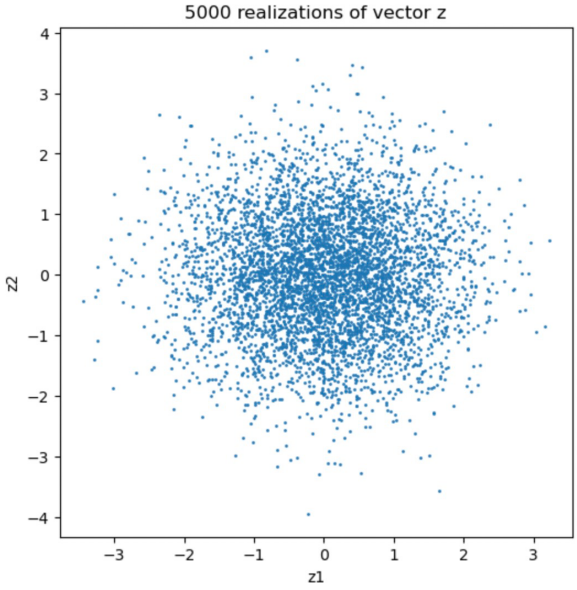
(iii)

The covariance of v is [[1.82310308e-01 -7.85113202e-03]
[-7.85113202e-03 2.17800679e+01]]



(iv)

The covariance of z is [[0.99435022 -0.00392557]
 [-0.00392557 0.99832303]]



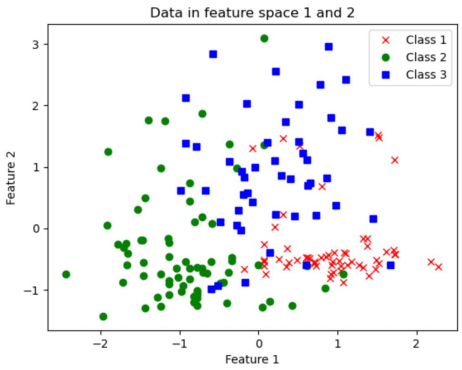
(v) What is the pdf of w and the pdf of z? Is $z = w$ -- i.e., since we colored w to get y, then we whitened y to get z? If not explain why.

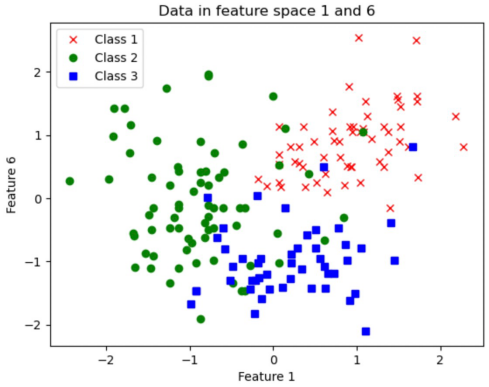
The pdf of w is a Gaussian distribution and same as the pdf of z. As w and z have the same pdf, z = w because we colored w to get y, the mean of x equals to zero and the covariance of z is also an identity matrix.

3. **Comparison of PCA and MDA on wine dataset.**

(a) Baseline for comparison.

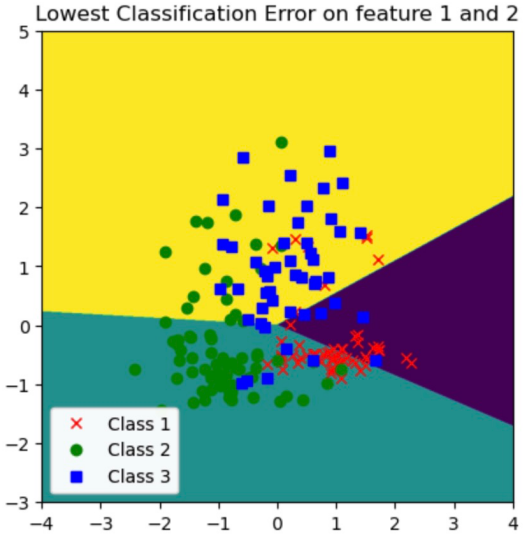
(i)

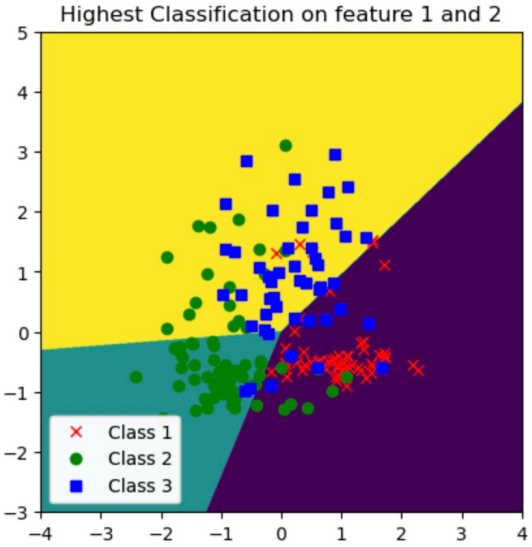




(ii) Feature 1 and 2:

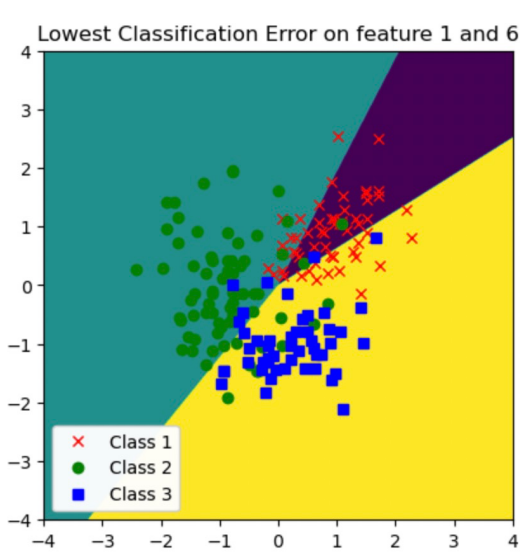
```
The mean classification error rate in run 1 = 0.28125000000000006
The mean classification error rate in run 2 = 0.25972222222222224
The mean classification error rate in run 3 = 0.23680555555555557
The mean classification error rate in run 4 = 0.225
The mean classification error rate in run 5 = 0.3152777777777778
The average and standard deviation of the mean classification error over the 5 runs is 0.26361111111111113 and 0.03
226469901407941
```

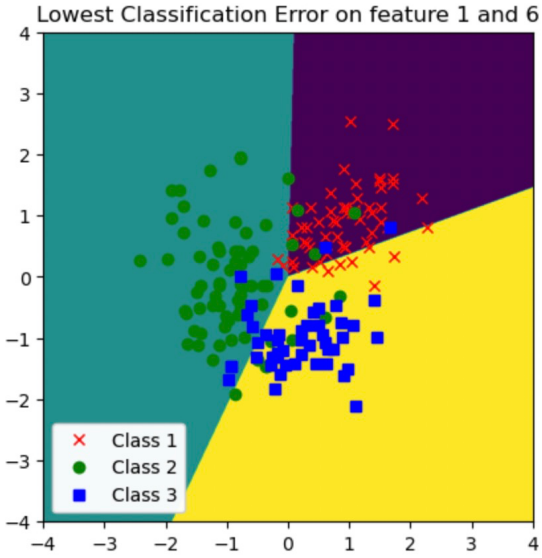




(iii)Feature 1 and 6:

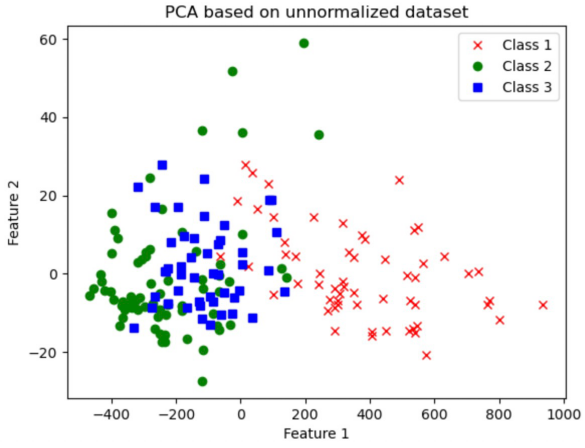
```
The mean classification error rate in run 1 = 0.24930555555555559
The mean classification error rate in run 2 = 0.23819444444444446
The mean classification error rate in run 3 = 0.22083333333333335
The mean classification error rate in run 4 = 0.2534722222222222
The mean classification error rate in run 5 = 0.22916666666666666
The average and standard deviation of the mean classification error over the 5 runs is 0.23819444444444446 and 0.01
2163685581006285
```





(b) PCA based on unnormalized dataset.

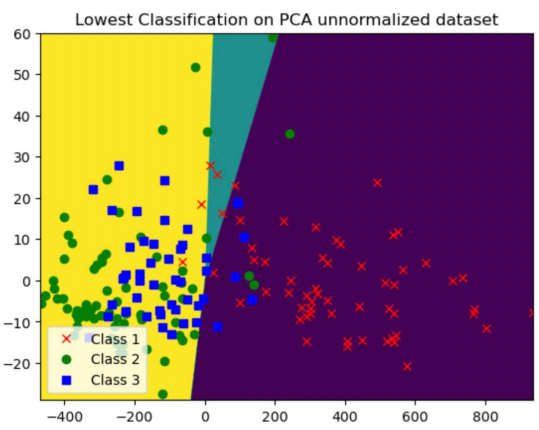
(i)

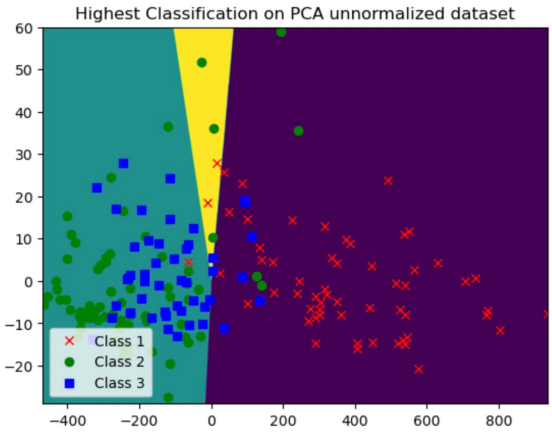


Class 2 and Class 3 almost overlap, I don't think there would be a better classification result with PCA.

(ii)

```
The mean classification error rate in run 1 = 0.34375000000000006
The mean classification error rate in run 2 = 0.36944444444444446
The mean classification error rate in run 3 = 0.46944444444444444
The mean classification error rate in run 4 = 0.4583333333333333
The mean classification error rate in run 5 = 0.45138888888888895
The average and standard deviation of the mean classification error ove
r the 5 runs is 0.41847222222222225 and 0.051493137502744396
```



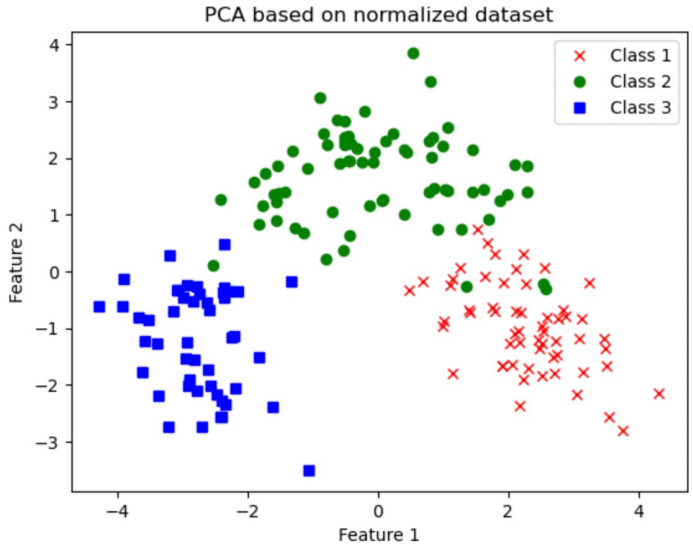


(iii) How does it compare with the baselines in (a)(ii)?

The shape of decision regions seem to like the lowest error result from baselines.

(c) PCA based on standardized dataset.

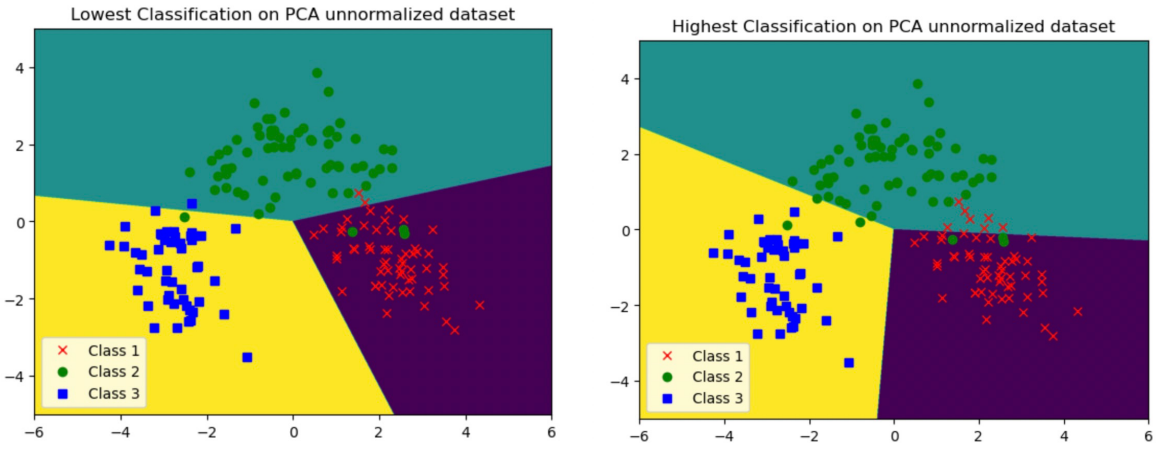
(i)



It will have a better classification result.

(ii)

```
The mean classification error rate in run 1 = 0.05208333333333335
The mean classification error rate in run 2 = 0.052083333333333336
The mean classification error rate in run 3 = 0.0388888888888889
The mean classification error rate in run 4 = 0.04444444444444445
The mean classification error rate in run 5 = 0.10277777777777779
The average and standard deviation of the mean classification error ove
r the 5 runs is 0.05805555555555557 and 0.022908668637994484
```

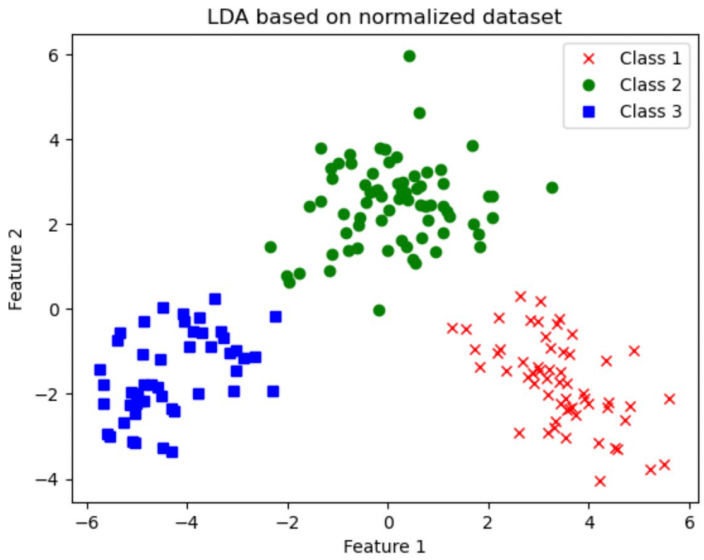


(iii)

The PCA on normalized data make a better performance on classification.

(d) MDA (using LDA as an approximation to MDA).

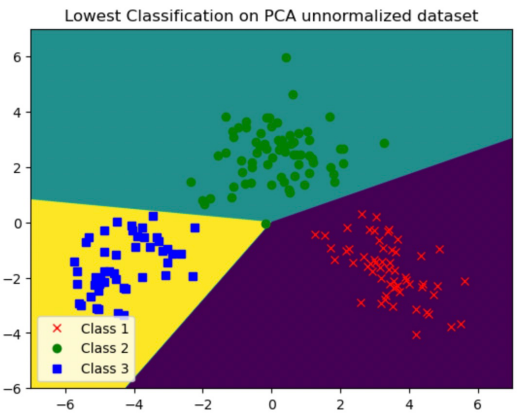
(i)

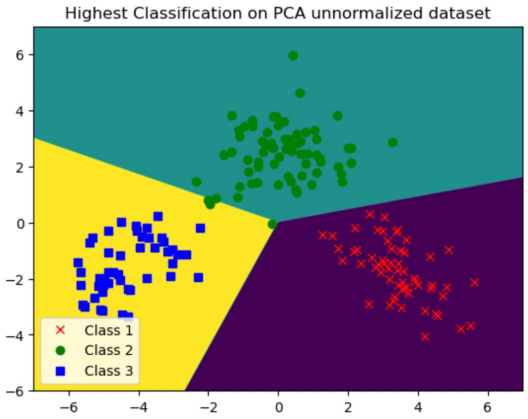


I expect that the classification result from those data would have low error rate.

(ii)

```
The mean classification error rate in run 1 = 0.01666666666666667
The mean classification error rate in run 2 = 0.011111111111111117
The mean classification error rate in run 3 = 0.005555555555555558
The mean classification error rate in run 4 = 0.016666666666666673
The mean classification error rate in run 5 = 0.011111111111111117
The average and standard deviation of the mean classification error ove
r the 5 runs is 0.012222222222222226 and 0.004157397096415491
```

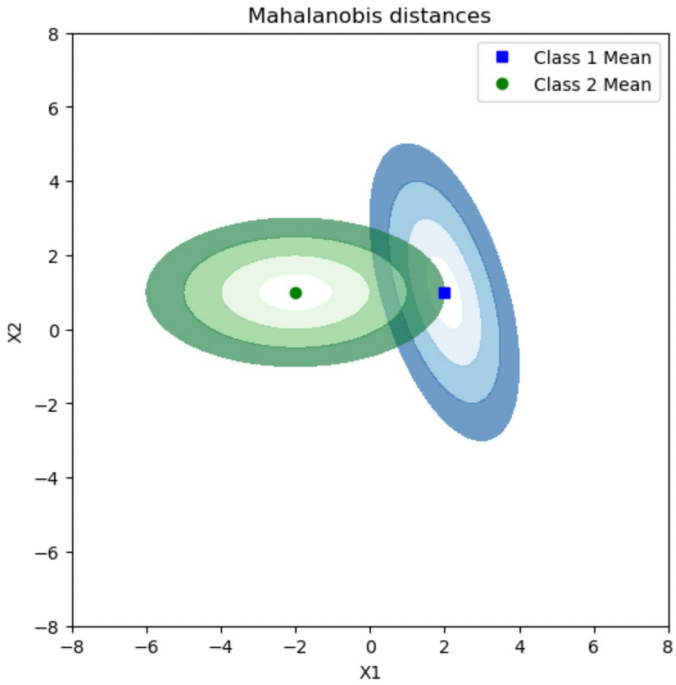




(iii) According to the mean classification error, the LDA perfom best compare to baseline and PCA because the centers of each classes much farer after LDA.

4. **Mahalanobis distance and Bayes classification for minimum error**

(a)



(b)

(b) $P(X=x|S_i)P(S_i) \Rightarrow P(\underline{x}|S_i) = N(\underline{x}|\underline{m}_i, \underline{\Sigma}_i)$, $i = 1, 2$

$$\ln \frac{P(X=x|S_1)P(S_1)}{P(X=x|S_2)P(S_2)} = \ln \frac{N(\underline{x}, \underline{m}_1, \underline{\Sigma}_1) P(S_1)}{N(\underline{x}, \underline{m}_2, \underline{\Sigma}_2) P(S_2)} = 0$$

$$= \ln \frac{\exp\{-\frac{1}{2}[(\underline{x}-\underline{m}_1)^T \underline{\Sigma}_1^{-1} (\underline{x}-\underline{m}_1)]\} P(S_1)}{(2\pi)\,|\underline{\Sigma}_1|^{1/2}} - \ln \frac{\exp\{-\frac{1}{2}[(\underline{x}-\underline{m}_2)^T \underline{\Sigma}_2^{-1} (\underline{x}-\underline{m}_2)]\} P(S_2)}{(2\pi)\,|\underline{\Sigma}_2|^{1/2}}$$

$$= \frac{1}{2}\ln|\underline{\Sigma}_1| - \frac{1}{2}(\underline{x}-\underline{m}_1)^T \underline{\Sigma}_1^{-1} (\underline{x}-\underline{m}_1) + \ln P(S_1) - \frac{1}{2}\ln|\underline{\Sigma}_2| + \frac{1}{2}(\underline{x}-\underline{m}_2)^T \underline{\Sigma}_2^{-1} (\underline{x}-\underline{m}_2) - \ln P(S_2)$$

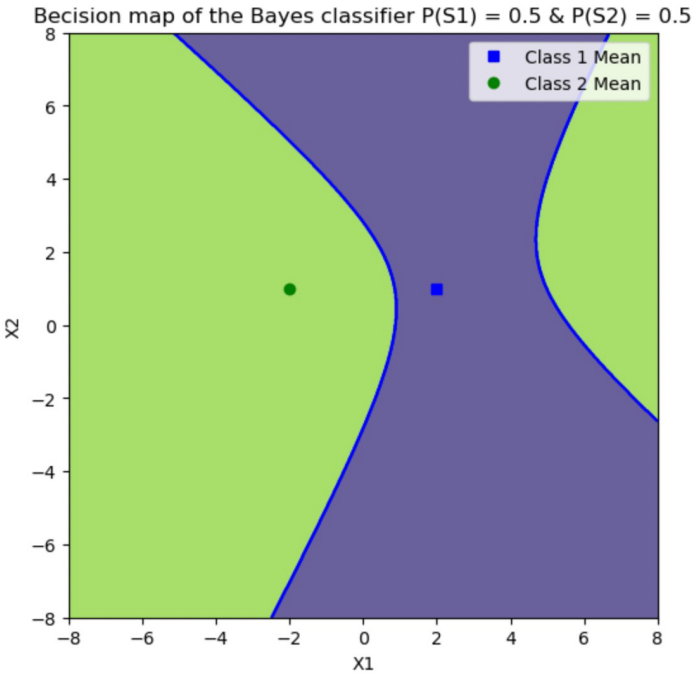
$$= \frac{1}{2}\ln|\underline{\Sigma}_1| - \frac{1}{2} d_m^2(\underline{x}, \underline{m}_1) + \ln P(S_1) - \frac{1}{2}\ln|\underline{\Sigma}_2| + \frac{1}{2} d_m^2(\underline{x}, \underline{m}_2) - \ln P(S_2)$$

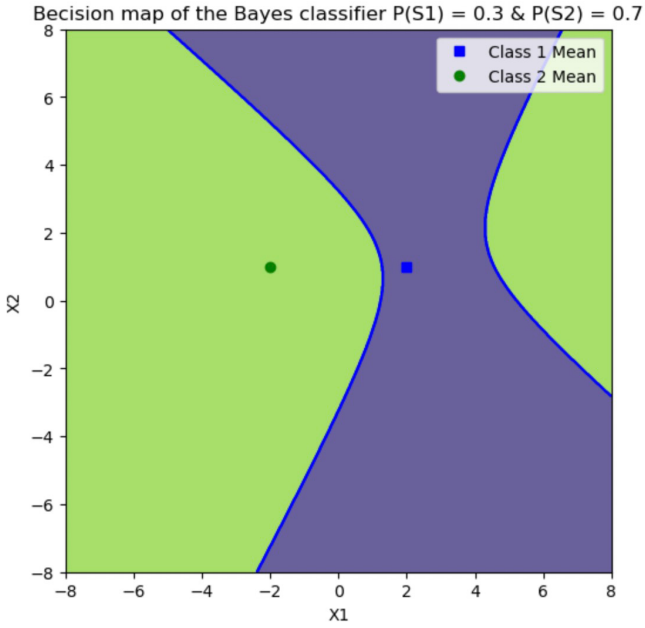
$$\ln P(S_1) - \ln P(S_2) + \frac{1}{2}\ln|\underline{\Sigma}_1| - \frac{1}{2}\ln|\underline{\Sigma}_2| = \frac{1}{2} d_m^2(\underline{x}, \underline{m}_1) - \frac{1}{2} d_m^2(\underline{x}, \underline{m}_2)$$

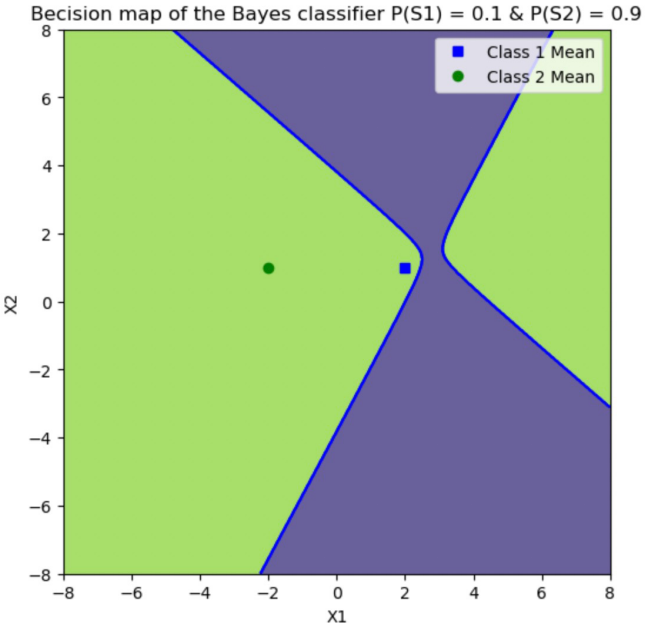
∴ The decision boundary is:

$$d_m^2(\underline{x}, \underline{m}_1) - d_m^2(\underline{x}, \underline{m}_2) = 2\ln \frac{P(S_1)}{P(S_2)} + \ln \frac{|\underline{\Sigma}_1|}{|\underline{\Sigma}_2|}$$

(c)


Becision map of the Bayes classifier P(S1) = 0.5 & P(S2) = 0.5
Class 1 Mean
Class 2 Mean
X1
X2


Becision map of the Bayes classifier P(S1) = 0.3 & P(S2) = 0.7
Class 1 Mean
Class 2 Mean
X1
X2


Becision map of the Bayes classifier P(S1) = 0.1 & P(S2) = 0.9
Class 1 Mean
Class 2 Mean
X1
X2
8
6
4
2
0
−2
−4
−6
−8
−8
−6
−4
−2
0
2
4
6
8