Yilong Wang 483728
Linda Li 458029

Task 1

1. Given a set of two random variables $X = \{X_1, X_2\}$

Suppose the sample values of X_1, X_2 are
$$X_1 = [x_{11}, x_{12}, \cdots x_{1n}]$$
$$X_2 = [x_{21}, x_{22}, \cdots x_{2n}]$$

We form a $2 \times n$ Matrix as: $x = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \end{bmatrix}$

$$\Sigma_X = \frac{1}{n}(X-\bar{X})(X-\bar{X})^T = \frac{1}{n} X X^T = \frac{1}{n} \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \end{bmatrix}_{2\times n} \begin{bmatrix} x_{11} & x_{21} \\ x_{12} & x_{22} \\ \vdots & \vdots \\ x_{1n} & x_{2n} \end{bmatrix}_{n\times 2}$$

$$= \frac{1}{n} \begin{bmatrix} x_{11}^2 + x_{12}^2 + \cdots + x_{1n}^2 & x_{11}x_{21} + x_{12}x_{22} + \cdots + x_{1n} \cdot x_{2n} \\ x_{11}x_{21} + x_{12}x_{22} + \cdots + x_{1n} \cdot x_{2n} & x_{21}^2 + x_{22}^2 + \cdots + x_{2n}^2 \end{bmatrix}$$

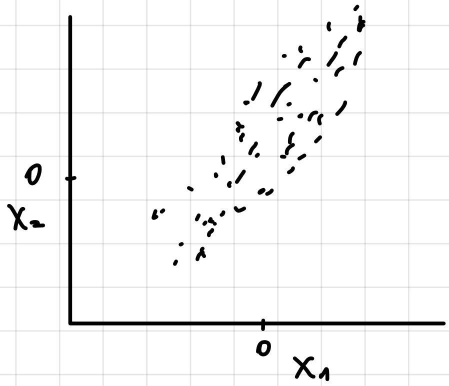
$$= \frac{1}{n} \begin{bmatrix} X_1^2 & X_1 X_2 \\ X_1 X_2 & X_2^2 \end{bmatrix}_{2\times 2} = \begin{bmatrix} Var(X_1) & Cov(X_1, X_2) \\ Cov(X_2, X_1) & Var(X_2) \end{bmatrix}_{2\times 2}$$

The dimensionality of the Σ_X is 2×2

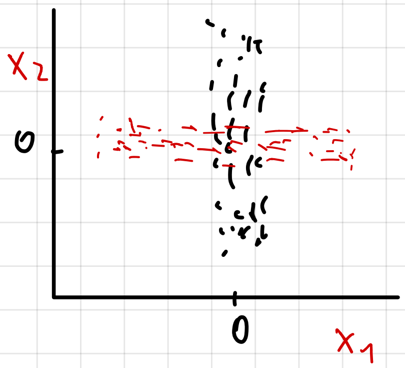
2. $Var(X_1) > Var(X_2)$

for a: $\begin{bmatrix} + & + \\ + & + \end{bmatrix}$ for b: $\begin{bmatrix} + & 0 \\ 0 & + \end{bmatrix}$ for c: $\begin{bmatrix} + & - \\ - & + \end{bmatrix}$

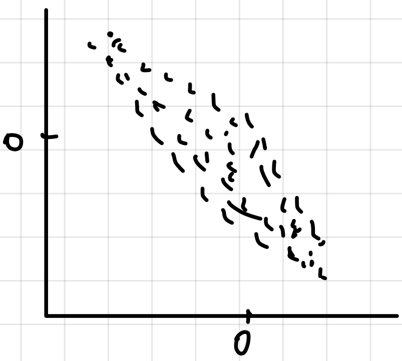
3. for $Cov(X_1, X_2) > 0$:



for $Cov(X_1, X_2) = 0$



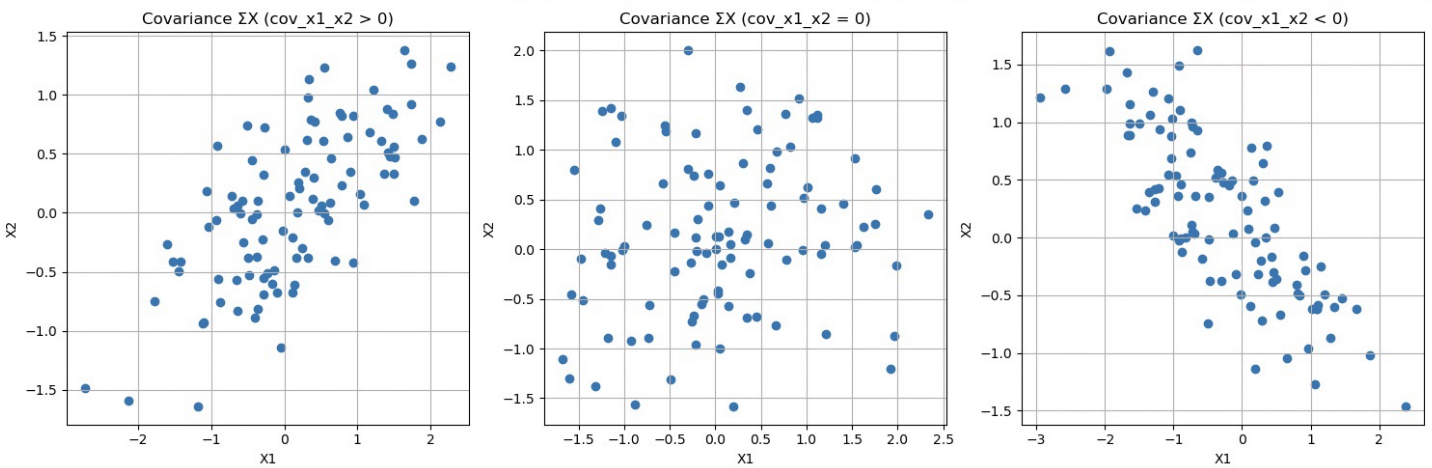
for $Cov(X_1, X_2) < 0$



because $Var[X_1] > Var[X_2]$

3.

```python
import numpy as np
import matplotlib.pyplot as plt

# Define the variances and covariances for the three cases
var_x1 = 1.0
var_x2 = 0.5

# Case 1: Covariance (cov_x1_x2) >0
cov_x1_x2_pos = 0.5

# Case 2: Covariance (cov_x1_x2) =0
cov_x1_x2_zero = 0.0

# Case 3: Covariance (cov_x1_x2) <0
cov_x1_x2_neg = -0.5


mean = [0, 0]

data_pos = np.random.multivariate_normal(mean, [[var_x1, cov_x1_x2_pos], [cov_x1_x2_pos, var_x2]], 100)
data_zero = np.random.multivariate_normal(mean, [[var_x1, cov_x1_x2_zero], [cov_x1_x2_zero, var_x2]], 100)
data_neg = np.random.multivariate_normal(mean, [[var_x1, cov_x1_x2_neg], [cov_x1_x2_neg, var_x2]], 100)

# Extract X1 and X2 values for each case
X1_pos, X2_pos = data_pos[:, 0], data_pos[:, 1]
X1_zero, X2_zero = data_zero[:, 0], data_zero[:, 1]
X1_neg, X2_neg = data_neg[:, 0], data_neg[:, 1]

# Create subplots to display all three cases
plt.figure(figsize=(15, 5))

plt.subplot(131)
plt.scatter(X1_pos, X2_pos)
plt.xlabel('X1')
plt.ylabel('X2')
plt.title('Covariance ΣX (cov_x1_x2 > 0)')
plt.grid(True)

plt.subplot(132)
plt.scatter(X1_zero, X2_zero)
plt.xlabel('X1')
plt.ylabel('X2')
plt.title('Covariance ΣX (cov_x1_x2 = 0)')
plt.grid(True)

plt.subplot(133)
plt.scatter(X1_neg, X2_neg)
plt.xlabel('X1')
plt.ylabel('X2')
plt.title('Covariance ΣX (cov_x1_x2 < 0)')
plt.grid(True)

plt.tight_layout()
plt.show()
```



4. The diagonal elements of all covariance matrices represent the variance of each random variable in the same row

$$\begin{bmatrix} \mathrm{Var}(X_1) & \cdots & \mathrm{Cov}(X_1, X_d) \\ \vdots & \ddots & \vdots \\ \mathrm{Cov}(X_d, X_1) & \cdots & \mathrm{Var}(X_d) \end{bmatrix}$$

Task 2

1. a) $X \in \mathbb{R}^{d \times n}$

Suppose $x_k \in \mathbb{R}^{d \times 1}$, $x_k^T \in \mathbb{R}^{1 \times d}$, then we have $x_k x_k^T \in \mathbb{R}^{d \times d}$

$\Rightarrow$ the dimension of $x_k x_k^T$ is $d \times d$

b) $$[\hat{\Sigma}]_{0,0} = \frac{1}{n}\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{12} \\ \vdots \\ x_{1n} \end{bmatrix} = \frac{1}{n}(x_{11}^2 + x_{12}^2 + \dots + x_{1n}^2) = \frac{1}{n}\sum_{k=1}^{n} x_{1k} \cdot x_{1k}$$

this sum represent the variance of the first component (first row and first colum of $X \in \mathbb{R}^{d \times n}$)

c) The centering step is performed to ensure that the resulting covariance matrix reflects the variability between variables, rather than influenced by the absolute scales of the variables.

Given $x_1, x_2, \dots x_n \in \mathbb{R}^d$ is n observations of d random variables

the empirical covariance matrix: $$\Sigma_X = \frac{1}{n}\sum_{k=1}^{n}(x_k - \bar{x})(x_k - \bar{x})^T$$ (for non-centered data)

for centered data we have: $x_k - \bar{x} = x_k$

$$\Rightarrow \Sigma_X = \frac{1}{n}\sum_{k=1}^{n} x_k x_k^T = \frac{1}{n} X X^T$$

2. $$\bar{x} = \begin{bmatrix} \frac{1}{4}(-1-1+1+1) \\ \frac{1}{4}(-1+0+0+1) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\Sigma_X = \frac{1}{n} X X^T = \frac{1}{4}\begin{bmatrix} -1 & -1 & 1 & 1 \\ -1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & -1 \\ -1 & 0 \\ 1 & 0 \\ 1 & 1 \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 4 & 2 \\ 2 & 2 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}$$

3. a) Given $v \in \mathbb{R}^d \setminus \{0\}$, $x_k \in \mathbb{R}^{d \times n}$, $\bar{x} \in \mathbb{R}^{d \times n}$, $\Sigma_X \in \mathbb{R}^{d \times d}$

here: $v^T A v = v^T \Sigma_X v$

two way to prove:

1. $v^T \hat{\Sigma} v \geq 0 \quad \forall v$

2. eigenvalue$(\hat{\Sigma}) \geq 0$

$$v^T \Sigma_X v = v^T\left(\frac{1}{n}\sum_{k=1}^{n}(x_k - \bar{x})(x_k - \bar{x})^T\right)v$$
$$= \frac{1}{n}\sum_{k=1}^{n}\left(v^T(x_k - \bar{x})(x_k - \bar{x})^T v\right)$$
$B^T A^T = (AB)^T$
$$= \frac{1}{n}\sum_{k=1}^{n}\left(v^T(x_k - \bar{x})\left(v^T(x_k - \bar{x})\right)^T\right) = \frac{1}{n}\sum \left[v^T(x_k - \bar{x})\right]^2 \geq 0$$

we set $t = v^T(x_k - \bar{x}) \in \mathbb{R}$, because of $v \in \mathbb{R}^d \setminus \{0\}$, $x_k \in \mathbb{R}^d$, $\bar{x} \in \mathbb{R}^d$

now we have $v^T \Sigma_x v = \frac{1}{n} \sum_{k=1}^{n} t \cdot t^T = \frac{1}{n} \sum_{k=1}^{n} t^2 \geq 0$

$\Rightarrow$ positive semi-definite

b) $(\Sigma_x)^T = \left(\frac{1}{n} X X^T\right)^T = \frac{1}{n} (X^T)^T X^T = \frac{1}{n} X X^T = \Sigma_x$

$\Rightarrow$ Σ_x is always symmetric

task 4

1. This value helps adjust the classification boundary to better accommodate data imbalances. If there is a significant difference in the number of samples between classes, $\left[+\log\left(\frac{n_-}{n_+}\right)\right]$ reflects this difference and helps correct the classification boundary to better account for class imbalances.

i) if n_- much larger than n_+ : $\log \frac{n_-}{n_+} > 0$

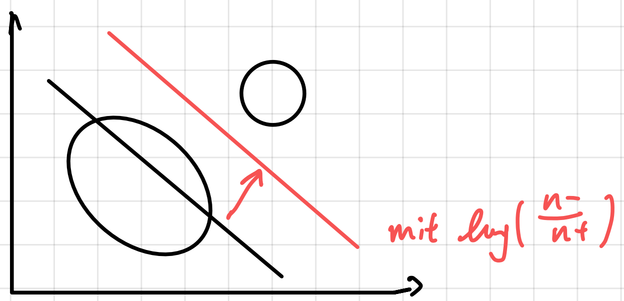
In this case, this value makes the classification boundary more towards the class +

ii) if n_- much smaller than n_+ : $\log \frac{n_-}{n_+} < 0$

Similarly, this value makes the boundary more towards the class $-$.

iii) if $n_- = n_+$: $\log \frac{n_-}{n_+} = 0$

In this case this value has no effect on the boundary



2. if $S^{-1} = \overset{I}{1}$ and class -, class + are equal, LDA and NCC are equal

dem: $w = S^{-1}(\bar{x}_+ - \bar{x}_-) = \bar{x}_+ - \bar{x}_-$

$$\beta = \frac{1}{2} w^T(\bar{x}_+ + \bar{x}_-) + \log\left(\frac{n_-}{n_+}\right) = \frac{1}{2} w^T(\bar{x}_+ + \bar{x}_-)$$

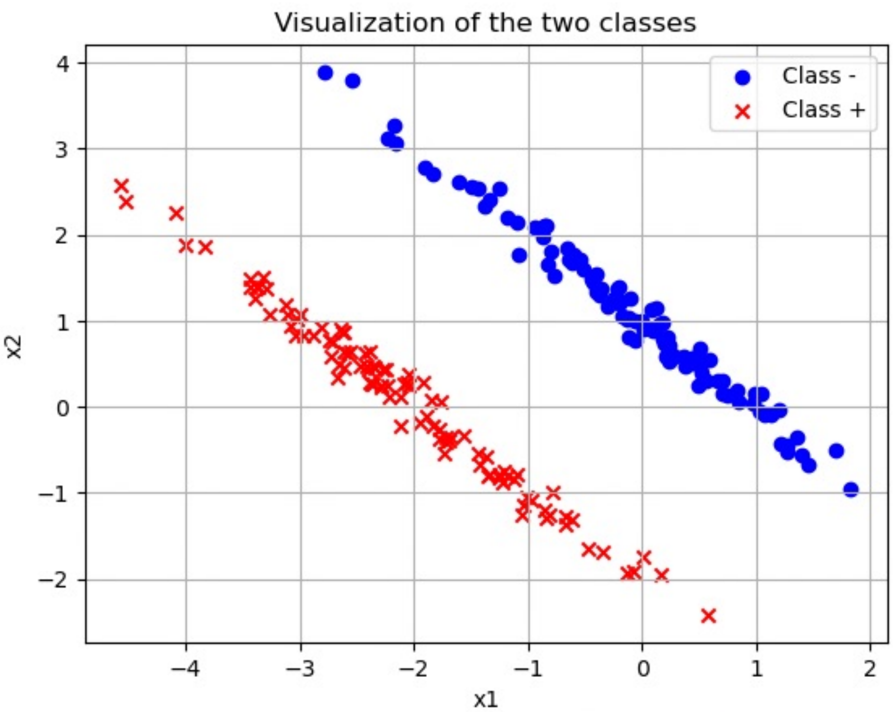
LDA在经过 Whitening后得到

$$S = \begin{bmatrix} 1 & & \\ & 1 & \\ & & \ddots \end{bmatrix} = I$$

此时 LDA和NCC相等

3. a)

```
import numpy as np
import matplotlib.pyplot as plt

# Given means and covariance matrices
mean_minus = np.array([0, 1])
mean_plus = np.array([-2, 0])
cov_minus = np.array([[1, -0.99], [-0.99, 1]])
cov_plus = np.array([[1, -0.99], [-0.99, 1]])

# Generate samples for each class
num_samples = 100
samples_minus = np.random.multivariate_normal(mean_minus, cov_minus, num_samples)
samples_plus = np.random.multivariate_normal(mean_plus, cov_plus, num_samples)

# Plot the samples
plt.scatter(samples_minus[:, 0], samples_minus[:, 1], color='blue', marker='o', label='Class -
plt.scatter(samples_plus[:, 0], samples_plus[:, 1], color='red', marker='x', label='Class +')
plt.xlabel('x1')
plt.ylabel('x2')
plt.legend()
plt.title('Visualization of the two classes')
plt.grid(True)
plt.show()
```



b) for NCC:

$$w = \bar{x}_+ - \bar{x}_- = \begin{pmatrix} -2 \\ 0 \end{pmatrix} - \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} -2 \\ -1 \end{pmatrix}$$

$$\beta = \frac{1}{2} w^T (\bar{x}_+ + \bar{x}_-) = \frac{1}{2} \begin{pmatrix} -2 & -1 \end{pmatrix} \begin{pmatrix} -2 \\ 1 \end{pmatrix} = \frac{1}{2} (4 - 1) = \frac{3}{2}$$

for LDA:

because of $S_- = S_+$, $S = \begin{pmatrix} 1 & -0.99 \\ -0.99 & 1 \end{pmatrix}$

$$S^{-1} = \frac{1}{ad - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix} = \frac{1}{1 - 0.99^2} \begin{pmatrix} 1 & 0.99 \\ 0.99 & 1 \end{pmatrix} = \frac{1}{0.0199} \begin{pmatrix} 1 & 0.99 \\ 0.99 & 1 \end{pmatrix}$$

$$w = S^{-1}(X_+ - X_-) = \frac{1}{0.0199} \begin{pmatrix} 1 & 0.99 \\ 0.99 & 1 \end{pmatrix} \cdot \begin{pmatrix} -2 \\ -1 \end{pmatrix} = \frac{1}{0.0199} \begin{pmatrix} -2 - 0.99 \\ -1.98 - 1 \end{pmatrix} = \frac{1}{0.0199} \begin{pmatrix} -2.99 \\ -2.98 \end{pmatrix}$$

$$= \begin{pmatrix} -150.251 \\ -149.748 \end{pmatrix}$$

$$\beta = \frac{1}{2} w^T (\bar{x}_+ + \bar{x}_-) + \log\left(\frac{n_-}{n_+}\right)$$

$$= \frac{1}{2} \cdot \frac{1}{0.0199} \cdot \begin{pmatrix} -2.99 & -2.98 \end{pmatrix} \begin{pmatrix} -2 \\ 1 \end{pmatrix} + 0$$

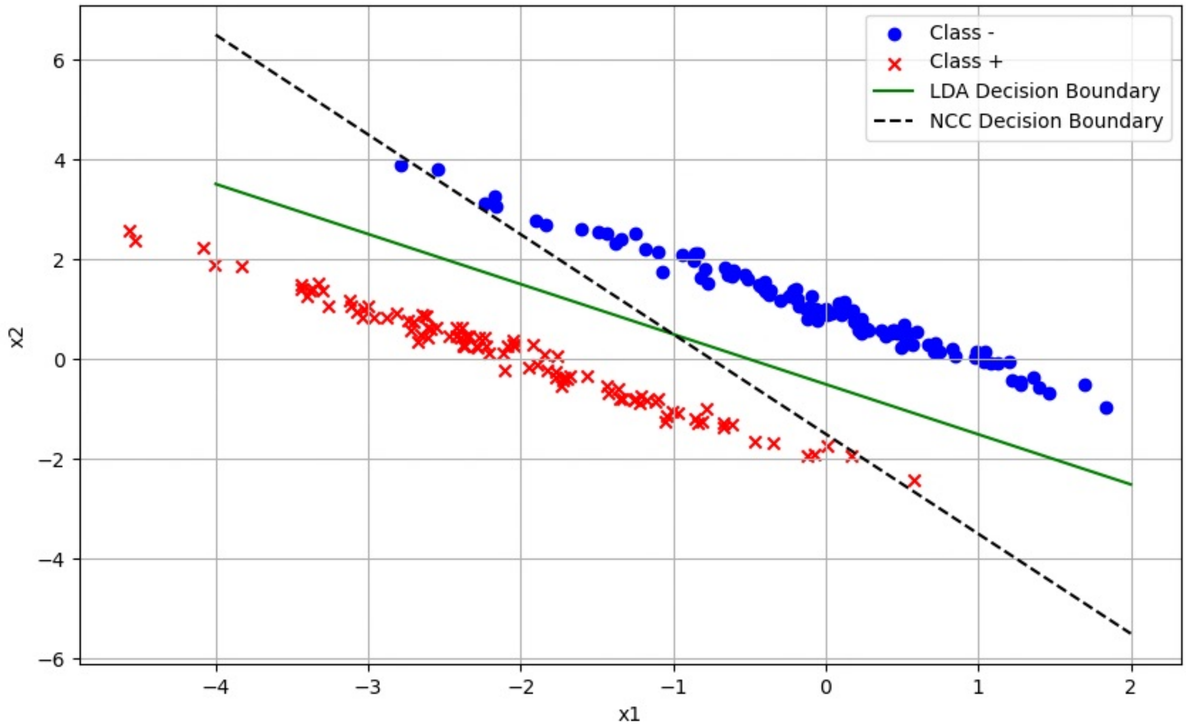
$$= \frac{1}{0.0398} \cdot \left(5.98 - 2.98 \right) = \frac{3}{0.0398} \approx 75.38$$

```python
# LDA
S = (cov_minus + cov_plus) / 2
S_inv = np.linalg.inv(S)
w_LDA = np.dot(S_inv, mean_plus - mean_minus)
beta_LDA = 0.5 * np.dot(w_LDA.T, mean_plus + mean_minus)

# NCC
w_NCC = mean_plus - mean_minus
beta_NCC = 0.5 * np.dot(w_NCC.T, mean_plus + mean_minus)

print("w_LDA:", w_LDA)
print("beta_LDA:", beta_LDA)
print("w_NCC:", w_NCC)
print("beta_NCC:", beta_NCC)
```

```
w_LDA: [-150.25125628 -149.74874372]
beta_LDA: 75.37688442211046
w_NCC: [-2 -1]
beta_NCC: 1.5
```

c)

```python
# Generating a range of x1 values for our line
x1_vals = np.linspace(-4, 2, 400)

# Calculate x2 values for the decision boundaries
x2_vals_LDA = (beta_LDA - w_LDA[0] * x1_vals) / w_LDA[1]
x2_vals_NCC = (beta_NCC - w_NCC[0] * x1_vals) / w_NCC[1]

plt.figure(figsize=(10,6))
plt.scatter(samples_minus[:, 0], samples_minus[:, 1], color='blue', marker='o', label='Class -')
plt.scatter(samples_plus[:, 0], samples_plus[:, 1], color='red', marker='x', label='Class +')
plt.plot(x1_vals, x2_vals_LDA, label='LDA Decision Boundary', color='green')
plt.plot(x1_vals, x2_vals_NCC, label='NCC Decision Boundary', color='black', linestyle='--')
plt.xlabel('x1')
plt.ylabel('x2')
plt.legend()
plt.grid(True)
plt.show()
```



d) In this case LDA performed better because the covariance matrices S_+, S_- are equal, which makes LDA especially suited, because it uses covariance to determine the decision boundary