Part. 1, Coding (60%):

In this coding assignment, you are required to implement Fisher's linear discriminant by using only NumPy, then train your model on the provided dataset, and evaluate the performance on testing data. Find the sample code and data on the GitHub page https://github.com/NCTU-VRDL/CS_CS20024/tree/main/HW2

Please note that only NumPy can be used to implement your model, you will get 0 point by calling sklearn.discriminant_analysis.LinearDiscriminantAnalysis.

1. (5%) Compute the mean vectors mi (i=1, 2) of each 2 classes on **training data**

```
print(f"mean vector of class 1: {m1}", f"mean vector of class 2: {m2}")
```
✓ 0.3s

```
mean vector of class 1: [ 0.99253136 -0.99115481] mean vector of class 2: [-0.9888012   1.00522778]
```

2. (5%) Compute the within-class scatter matrix SWon **training data**

```
print(f"Within-class scatter matrix SW: {sw}")
```
[478] ✓ 0.3s Python

```
Within-class scatter matrix SW: [[ 4337.38546493 -1795.55656547]
 [-1795.55656547  2834.75834886]]
```

3. (5%) Compute the between-class scatter matrix SB on **training data**

```
print(f"Between-class scatter matrix SB: {sb}")
```
[480] ✓ 0.1s Python

```
Between-class scatter matrix SB: [[ 3.92567873 -3.95549783]
 [-3.95549783  3.98554344]]
```

4. (5%) Compute the Fisher's linear discriminant on **training data**

```
print(f" Fisher's linear discriminant: {w}")
```
Python

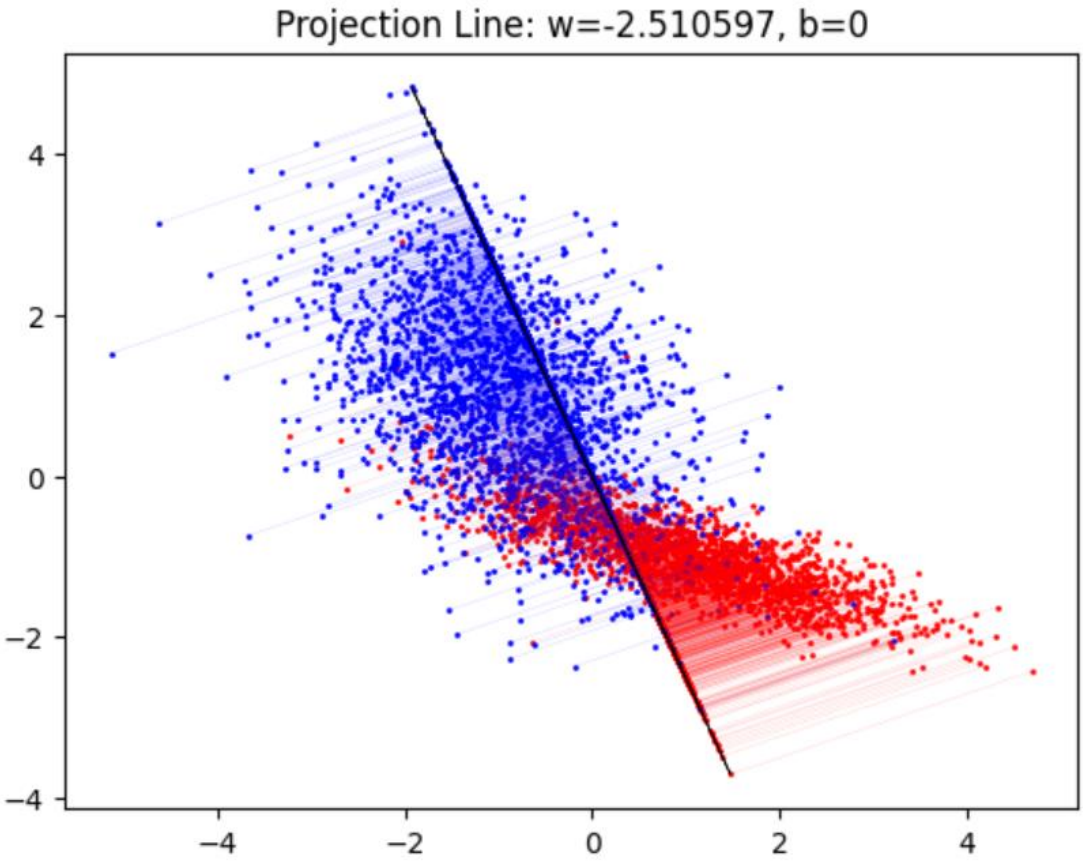
```
Fisher's linear discriminant: [-0.000224    0.00056237]
```

5. (20%) Project the **testing data** by Fisher's linear discriminant to get the class prediction by K-Nearest-Neighbor rule and report the accuracy score on **testing data** with K values from 1 to 5 (you should get accuracy over **0.88**)

```
print(f"Accuracy of test-set where k = 1 ~ 5 {acc_lst}")
```
✓ 0.3s Python

```
Accuracy of test-set where k = 1 ~ 5 [0.8488, 0.8704, 0.8792, 0.8824, 0.8912]
```

6. (20%) Plot the **1) best projection line** on the **training data** and show the slope and intercept on the title *(you can choose any value of **intercept** for better visualization)*
   2) colorize the data with each class **3)** project all data points on your projection line.



Part. 2, Questions (40%):

(10%) 1. What's the difference between the Principle Component Analysis and Fisher's Linear Discriminant?

Both of them can serve as dimension reduction tools. However, only Fisher's Linear Discriminant can be utilized as classifiers. It maximizes the "between-class variance" and minimizes the "within-class variance" of the projected data. On the other hand, PCA is a pure dimension reduction tool.

(10%) 2. Please explain in detail how to extend the 2-class FLD into multi-class FLD (the number of classes is greater than two).

For multi-class FLD, we simply compute the "within-class covariance matrix" by summing the covariance matrix of every class, just like binary class FLD. For "between-class covariance matrix", instead of subtracting the mean of each other, compute the covariance matrix by subtracting the mean of data from all classes with a weight N_k.

$$\mathbf{S}_{\mathrm{B}} = \sum_{k=1}^{K} N_k (\mathbf{m}_k - \mathbf{m})(\mathbf{m}_k - \mathbf{m})^{\mathrm{T}} \qquad \mathbf{m} = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n$$

Let K be the number of classes. Note that the rank of the between-class covariance matrix is at most $K - 1$. Therefore, the dimension of the projected space by FLD is at most $K - 1$.

(6%) 3. By making use of Eq (1) ~ Eq (5), show that the Fisher criterion Eq (6) can be written in the form Eq (7).

$$y = \mathbf{w}^{\mathrm{T}}\mathbf{x} \qquad \text{Eq (1)}$$

$$\mathbf{m}_1 = \frac{1}{N_1}\sum_{n \in \mathcal{C}_1} \mathbf{x}_n \qquad \mathbf{m}_2 = \frac{1}{N_2}\sum_{n \in \mathcal{C}_2} \mathbf{x}_n \qquad \text{Eq (2)}$$

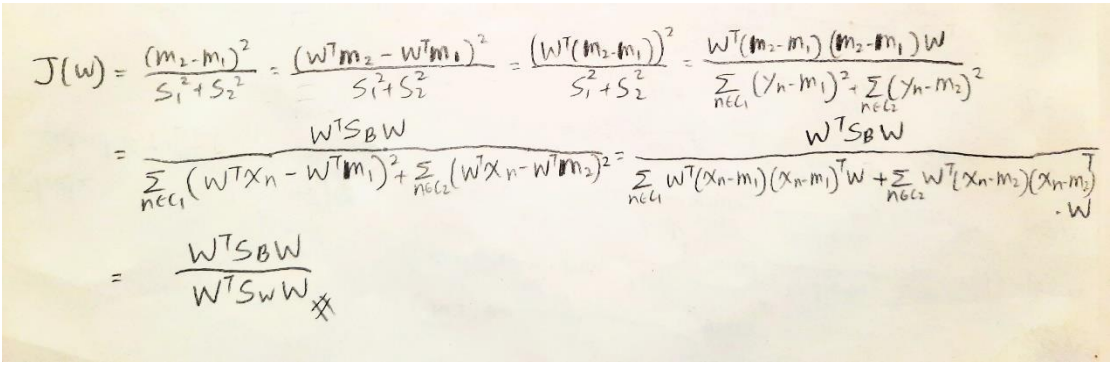
$$m_2 - m_1 = \mathbf{w}^{\mathrm{T}}(\mathbf{m}_2 - \mathbf{m}_1) \qquad \text{Eq (3)}$$

$$m_k = \mathbf{w}^{\mathrm{T}}\mathbf{m}_k \qquad \text{Eq (4)}$$

$$s_k^2 = \sum_{n \in \mathcal{C}_k} (y_n - m_k)^2 \qquad \text{Eq (5)}$$

$$J(\mathbf{w}) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} \qquad \text{Eq (6)}$$

$$J(\mathbf{w}) = \frac{\mathbf{w}^{\mathrm{T}}\mathbf{S}_{\mathrm{B}}\mathbf{w}}{\mathbf{w}^{\mathrm{T}}\mathbf{S}_{\mathrm{W}}\mathbf{w}} \qquad \text{Eq (7)}$$



(7%) 4. Show the derivative of the error function Eq (8) with respect to the activation ak for an output unit having a logistic sigmoid activation function satisfies Eq (9).

$$E(\mathbf{w}) = -\sum_{n=1}^{N} \{t_n \ln y_n + (1 - t_n)\ln(1 - y_n)\} \qquad \text{Eq (8)}$$

$$\frac{\partial E}{\partial a_k} = y_k - t_k \qquad \text{Eq (9)}$$

$$\frac{\partial E}{\partial a_k} = \frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial a_k} = -\left(\frac{t_k}{y_k} + \frac{-(1-t_k)}{1-y_k}\right)(1-y_k)y_k = -\left[t_k(1-y_k) - (1-t_k)y_k\right]$$

$$= -\left[t_k - t_k y_k - y_k + t_k y_k\right] = y_k - t_k \quad \#$$

(7%) 5. Show that maximizing likelihood for a multiclass neural network model in which the network outputs have the interpretation yk(x, w)=p(tk=1 | x) is equivalent to the minimization of the cross-entropy error function Eq (10).

$$E(\mathbf{w}) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{kn} \ln y_k(\mathbf{x}_n, \mathbf{w}) \qquad \text{Eq (10)}$$

$y_k(x,w) = p(t_k=1|x) = \prod_{n=1}^{N}\prod_{k=1}^{K} y_k(x_n,w)^{t_{nk}}$

$\Rightarrow \ln y_k(x,w) = \sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} \ln y_k(x_n,w)$

Since $E(w) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} \ln y_k(x_n,w)$, $E(w) = -\ln y_k(x,w)$.

Therefore, maximizing likelihood is equivalent to minimizing the cross entropy.

(因為差一個負號)