Fisher’s linear discriminant (HW2)

Student: 張皓雲

Student Id: 310551031

Content

● Part. 1, Coding (60%):

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

```
mean vector of class 1: [2.47107265 1.97913899]
mean vector of class 2: [1.82380675 3.03051876]
```

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

```
Within-class scatter matrix SW: [[140.40036447  -5.30881553]
 [ -5.30881553 138.14297637]]
```

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

```
Between-class scatter matrix SB: [[ 0.41895314 -0.68052227]
 [-0.68052227  1.10539942]]
```

4. (5%) Compute the Fisher's linear discriminantwon training data

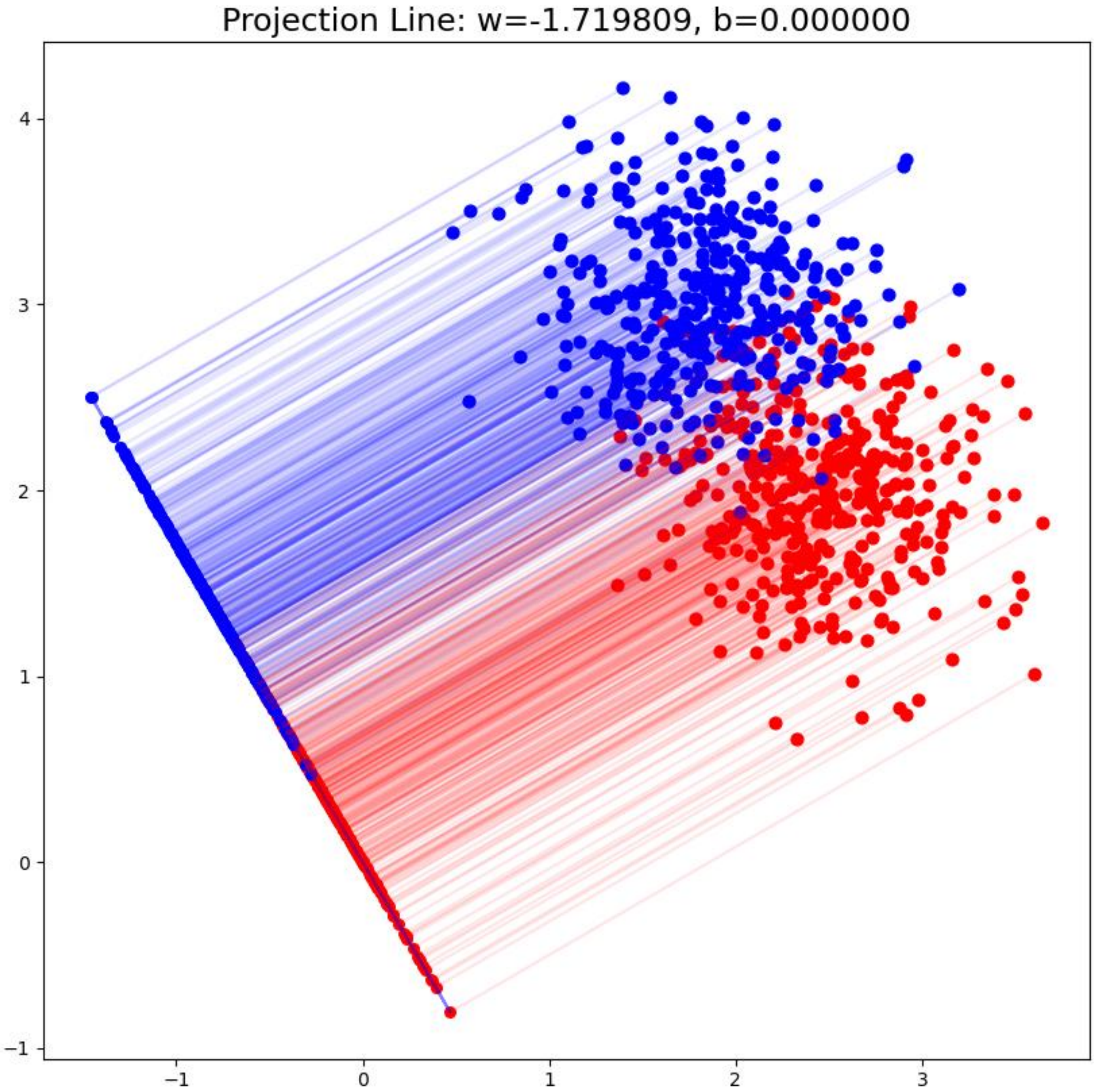
```
Fisher's linear discriminant: [[-0.00432865]
 [ 0.00744446]]
```

5. (20%) Project the testing data by Fisher's linear discriminant to get the class prediction by nearest-neighbor rule and calculate your accuracy score on testing data (you should get accuracy over 0.9)

```
Accuracy of test-set 0.9
```

```python
# ## 5. Project the test data by linear discriminant to get the class prediction by nearest-neighbor rule and calculate the accuracy score
# you can use accuracy_score function from sklearn.metric.accuracy_score

# setting the value of k
k = 3

# project data point to project data
train_project = (x_train @ w) * (w/np.sum(w*w)).reshape(-1,)
test_project = (x_test @ w) * (w/np.sum(w*w)).reshape(-1,)

# find the k-nearst, and the k-nearst training label which occurs most times would be the predicted label.
for test_point in test_project:

    distance_list = []

    for train_point in train_project:
        distance = np.sum((test_point-train_point)**2)
        distance_list.append(distance)

    distance_list = np.array(distance_list).reshape(-1,)
    index = np.argpartition(distance_list, k)[:k]
    exact_value = y_train[index]
    pred_value = np.argmax(np.bincount(exact_value))
    y_pred.append(pred_value)

y_pred = np.array(y_pred)

acc = accuracy(y_pred, y_test)

print(f"Accuracy of test-set {acc}")
```

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. Your result should look like the below image (This image is for reference, not the answer)


Projection Line: w=-1.719809, b=0.000000
4
3
2
1
0
−1
−1
0
1
2
3

- **Part. 2, Questions (40%):**

310551031 張皓雲

1.

$$L(\lambda, W) = W^T(m_2 - m_1) + \lambda(W^T W - 1)$$

微分

$$= (m_2 - m_1) + 2\lambda W$$

$$\Rightarrow m_2 - m_1 = -2\lambda W$$

$$\Rightarrow W \propto (m_2 - m_1) \text{ ※}$$

2.

$$J(W) = \frac{(m_2 - m_1)^2}{s_1^2 + S_2^2} = \frac{(W^T(\vec{m_2} - \vec{m_1}))^2}{s_1^2 + S_2^2}$$

vector

$\vec{m_2} \Rightarrow$ 粗体 m_2

$\vec{m_1} \Rightarrow$ 粗体 m_1

$$= \frac{W^T(\vec{m_2} - \vec{m_1})(\vec{m_2} - \vec{m_1})^T W}{s_1^2 + S_2^2}$$

$$= \frac{W^T S_B W}{s_1^2 + S_2^2} = \frac{W^T S_B W}{\sum_{n \in C_1}(W^T \vec{X_n} - W^T \vec{m_1})^2 + \sum_{n \in C_2}(W^T \vec{X_n} - W^T \vec{m_2})^2}$$

$$= \frac{W^T S_B W}{\sum_{n \in C_1}(W^T(\vec{X_n} - \vec{m_1}))^2 + \sum_{n \in C_2}(W^T(\vec{X_n} - \vec{m_2}))^2}$$

$$= \frac{W^T S_B W}{W^T\left(\sum_{n \in C_1}(\vec{X_n} - \vec{m_1})(\vec{X_n} - \vec{m_1})^T + \sum_{n \in C_2}(\vec{X_n} - \vec{m_2})(\vec{X_n} - \vec{m_2})^T\right)W}$$

$$= \frac{W^T S_B W}{W^T S_W W}$$

3、

$$E(w) = -\ln p(t|w) = -\sum_{n=1}^{N}\{t_n \ln y_n + (1-t_n)\ln(1-y_n)\}$$

對w偏微

$$= -\sum_{n=1}^{N}\left(\frac{t_n}{y_n} - \frac{1-t_n}{1-y_n}\right) \times y_n \times (1-y_n) \cdot \phi_n$$

$$= -\sum_{n=1}^{N}\frac{t_n - y_n}{y_n(1-y_n)} \times y_n \times (1-y_n) \times \phi_n$$

$$= -\sum_{n=1}^{N}(t_n - y_n)\phi_n$$

$$= \sum_{n=1}^{N}(y_n - t_n)\phi_n$$