NCTU Pattern Recognition Homework 2

M093781 趙宇涵

Part 1. Coding (60%)

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

First, separate 2 classes. I use numpy.mean to calculate m1, m2.

```
## Your code HERE
# Seperate two classes
class1 = x_train[np.where(y_train == 0)]
class2 = x_train[np.where(y_train == 1)]

# Calculate average of each class
m1 = np.mean(class1, axis=0)
m2 = np.mean(class2, axis=0)
```

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

Use the formula to calculate S_w.

$$\mathbf{S}_{\mathrm{W}} = \sum_{n \in \mathcal{C}_1} (\mathbf{x}_n - \mathbf{m}_1)(\mathbf{x}_n - \mathbf{m}_1)^{\mathrm{T}} + \sum_{n \in \mathcal{C}_2} (\mathbf{x}_n - \mathbf{m}_2)(\mathbf{x}_n - \mathbf{m}_2)^{\mathrm{T}}$$

```
In [6]: ## Your code HERE
        # Use the formula to calculate sw
        sw1 = (class1 - m1).T.dot(class1 - m1)
        sw2 = (class2 - m2).T.dot(class2 - m2)
        sw = sw1 + sw2
        sw

Out[6]: array([[140.40036447,  -5.30881553],
               [ -5.30881553, 138.14297637]])
```

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

Use the formula to calculate S_B.

$$\mathbf{S}_{\mathrm{B}} = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^{\mathrm{T}}$$

```
In [8]: ## Your code HERE
        # Use the formula to calculate sb
        sb = np.outer((m1 - m2), (m1 - m2).T)  # outer production
        sb

Out[8]: array([[ 0.41895314, -0.68052227],
               [-0.68052227,  1.10539942]])
```

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

W is proportional to S_w^{-1}(m1-m2), use norm to get unit vector.

$$\mathbf{w} \propto \mathbf{S}_{\mathrm{W}}^{-1}(\mathbf{m}_2 - \mathbf{m}_1)$$

```
In [10]: ## Your code HERE
         # w is propotional to (sw^(-1))(m1-m2), norm to get unit vector.
         inv_sw = np.linalg.inv(sw)
         w = (inv_sw.dot(m2 - m1))
         w /= np.linalg.norm(w)
         w

Out[10]: array([-0.50266214,  0.86448295])
```

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)

Step 1. Define the Euclidean distance by $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

Step 2. Define get neighbor function. Sort the all training point by distance, and get the nearest n points' index.

Step 3. Define predict classification. Based on the nearest points index, find their correlated label by y_train. Because the n is an odd number, there must be a label more than another. Choose the major label to be the predicted class.

Step 4. Use a for loop to find all predicted classes.

Step 5. Use accuracy_score function to calculate the accuracy.

Noted. I tried n form 3 to 7, and I can get the highest accuracy 0.912 at n = 5. Thus, I choose n = 5 to be my final answer.

In [12]:
```
# calculate the Euclidean distance between two vectors
from math import sqrt
def euclidean_distance(row1, row2):
    distance = 0.0
    for i in range(len(row1)):
        distance += (row1[i] - row2[i])**2
    return sqrt(distance)

# Locate the most similar neighbors
def get_neighbors(train, test_row, num_neighbors):
    distances = list()
    for i, train_row in enumerate(train):
        dist = euclidean_distance(test_row, train_row)
        distances.append((i, train_row, dist))
        distances.sort(key=lambda tup: tup[2])
    neighbors = list()
    for i in range(num_neighbors):
        neighbors.append(distances[i][0])
    return neighbors

 # Make a classification prediction with neighbors
def predict_classification(train, test_row, num_neighbors):
    neighbors = get_neighbors(train, test_row, num_neighbors)
    output_values = [y_train[row] for row in neighbors]
    prediction = max(set(output_values), key=output_values.count)
    return prediction
```

In [13]:
```
# Test all testing data
y_pred = np.empty(len(x_test))
for i in range(len(x_test)):
    prediction = predict_classification(x_train, x_test[i], 5)
    y_pred[i] = prediction
```

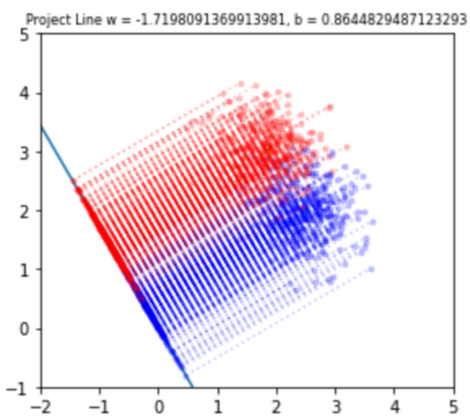
In [14]:
```
from sklearn.metrics import accuracy_score
acc = accuracy_score(y_test, y_pred)
acc
```

Out[14]: 0.912

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)



Date: 2022 / 4 / 6.

M093781 趙孚澍

Part 2. Questions

1. <given> $L(\lambda, w) = w^T(m_2 - m_1) + \lambda(w^T w - 1)$

$w^T w = 1$

< prof > $w \propto (m_2 - m_1)$

<solve> 1. setting the gradient of the function wrt w to 0:

$$\frac{\partial}{\partial w} L(\lambda, w) = \frac{\partial}{\partial w}\left(w^T(m_2 - m_1) + \lambda(1 - w^T w)\right) = m_2 - m_1 + 2\lambda w = 0$$

$$w = \frac{m_2 - m_1}{2\lambda}$$

2. Setting the derivate wrt λ to 0:

$$\frac{\partial}{\partial \lambda} L(\lambda, w) = \frac{\partial}{\partial \lambda}\left(w^T(m_2 - m_1) + \lambda(1 - w^T w)\right) = 1 - w^T w = 0$$

$$1 - w^T w = 1 - \frac{(m_2 - m_1)^T(m_2 - m_1)}{4\lambda^2} = 0$$

$$\lambda = \frac{\sqrt{(m_2 - m_1)^T(m_2 - m_1)}}{2} = \frac{\|m_2 - m_1\|_2}{2}$$

3. Combine 1. & 2.

$$w = \frac{m_2 - m_1}{\|m_2 - m_1\|_2} \propto (m_2 - m_1) \quad \#$$

Reference: https://tvml.github.io/ml1718/slides/linear_classification.pdf

2. Prof (eq6) $J(w) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}$ can be written as (eq7) $J(w) = \frac{w^T S_B w}{w^T S_W w}$.

<sol> From (eq2) $m_2 - m_1 = w^T(m_2 - m_1)$

$$(m_2 - m_1)^2 = w^T(m_2 - m_1) \cdot w(m_2 - m_1)^T$$

$$= w^T(m_2 - m_1)(m_2 - m_1)^T w$$

$$= w^T S_B w \quad \text{from (eq4) } S_B = (m_2 - m_1)(m_2 - m_1)^T$$

From (eq3) $s_k^2 = \sum_{n \in C_k} (y_n - m_k)^2$ and (eq1) $y = w^T x$.

$$s_1^2 = \sum_{n \in C_1} (y_n - m_1)^2 = w^T\left[\sum_{n \in C_1} (x_n - m_1)(x_n - m_1)^T\right] w$$

$$s_2^2 = \sum_{n \in C_2} (y_n - m_2)^2 = w^T\left[\sum_{n \in C_2} (x_n - m_2)(x_n - m_2)^T\right] w$$

$$\Rightarrow s_1^2 + s_2^2 = w^T\left[\sum_{n \in C_1} (x_n - m_1)(x_n - m_1)^T + \sum_{n \in C_2} (x_n - m_2)(x_n - m_2)^T\right] w$$

$$= w^T S_W w \quad \text{from (eq5)}$$

$$\Rightarrow J(w) = \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2} = \frac{w^T S_B w}{w^T S_W w} \quad \#$$

3. show derivative error of (eq 9) $E(w) = -\ln p(t|w) = -\sum_{n=1}^{N} \{ t_n \ln y_n + (1-t_n) \ln (1-y_n) \}$

is given by (eq 10) $\nabla E(w) = \sum_{n=1}^{N} (y_n - t_n)\, \phi_n$

where $y_n = \sigma(a_n)$, $a_n = w^T \phi_n$

<sol> in eq 9. replace y_n as σ:

$$E(w) = -\sum_{n=1}^{N} \{ t_n \ln \sigma + (1-t_n) \ln (1-\sigma) \}$$

$$\nabla E(w) = \frac{E(w)}{dw} = -\sum_{n=1}^{N} \left[\frac{t_n \cdot \sigma \cdot (1-\sigma) \cdot \phi_n}{\sigma} + \frac{(1-t_n)\, \sigma \cdot (1-\sigma) \cdot (-\phi_n)}{(1-\sigma)} \right]$$

$$= -\sum_{n=1}^{N} \left[\frac{t_n \cancel{\sigma} \cdot (1-\sigma)^{\cancel{2}} \cdot \phi_n - (1-t_n)\, \sigma^{\cancel{2}} (\cancel{1-\sigma})\, \phi_n}{\cancel{\sigma}(\cancel{1-\sigma})} \right]$$

$$= -\sum_{n=1}^{N} \left[t_n - t_n \sigma - \sigma + t_n \sigma \right] \phi_n$$

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

Reference : https://cedar.buffalo.edu/~srihari/CSE574/Chap4/4.3.2-LogisticReg.pdf