NCTU Pattern Recognition, Homework 2

Deadline: May 1, 23:59

Part. 1, Coding (60%):

In this coding assignment, you need to implement Fisher's linear discriminant by using only NumPy, then train your implemented model by the provided dataset and test the performance with testing data. Find the sample code and data on the GitHub page
https://github.com/NCTU-VRDL/CS_DCP3121/tree/master/HW2

Please note that only NumPy can be used to implement your model, you will get no points by simply calling sklearn.discriminant_analysis.LinearDiscriminantAnalysis. Find the FLD algorithm from page 16 in the course slides

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

```
m1, m2 = np.mean(x_train, axis=0)
class_1, class_2 = [], []
for i in range(y_train.size):
    if y_train[i] == 0:
        class_1.append(x_train[i])
    else:
        class_2.append(x_train[i])
c1 = np.array(class_1)
c2 = np.array(class_2)
m1 = np.mean(c1, axis=0)
m2 = np.mean(c2, axis=0)
```

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 S_W on training data

```
sw = np.dot((c1 - m1).transpose(), c1 - m1)
```

S_W : [[69.47006077 -4.05640247]
[-4.05640247 73.63343183]]

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

```
m21 = (m2 - m1)[:, None]
sb = np.dot(m21, m21.transpose())
```

S_B : [[0.41895314 -0.68052227]
[-0.68052227 1.10539942]]

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

```
w = np.dot(np.linalg.inv(sw), m21)
w = w / np.linalg.norm(w)
```

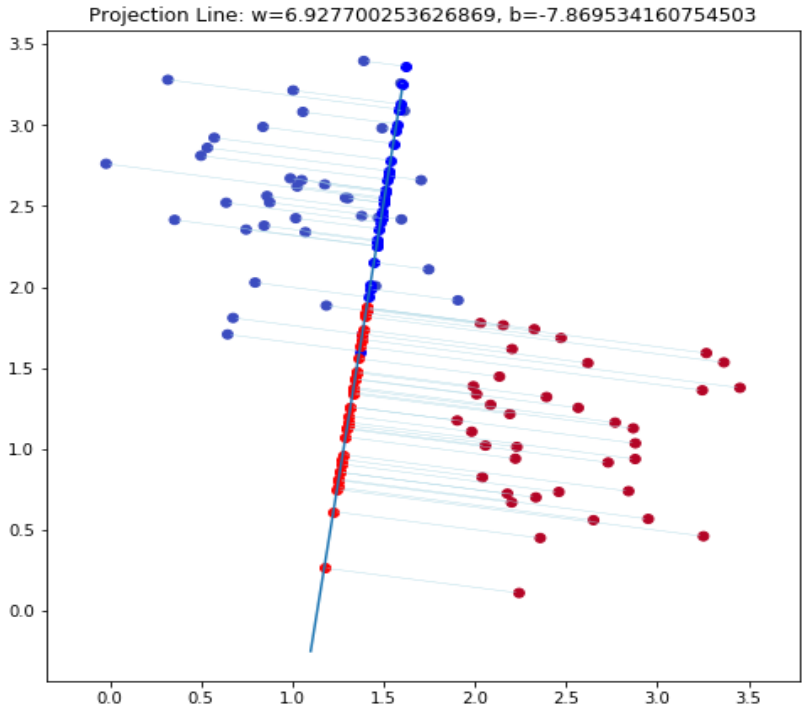
W : [[-0.52465848] [0.8513128]]

5. (20%) Project the **testing data** by linear discriminant to get the class prediction by nearest-neighbor rule and calculate your accuracy score on **testing data** (you should get accuracy over 0.8)

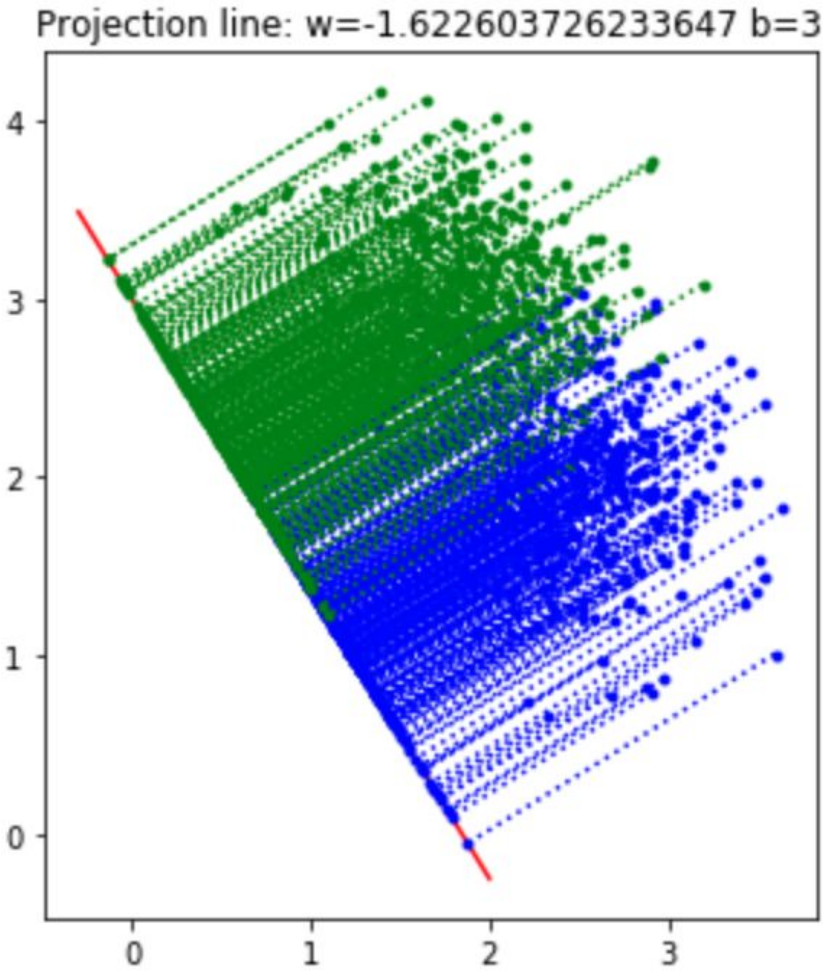
```
z1 = c1.dot(w)
z2 = c2.dot(w)
z_test = x_test.dot(w)
y_pred = np.empty([250, 1])
z_train = x_train.dot(w)
for i in range(z_test.size):
    minimum = 100
    min_index = 0
    for j in range(z_train.size):
        if abs(z_test[i] - z_train[j]) < minimum:
            minimum = abs(z_test[i] - z_train[j])
            min_index = j
    y_pred[i] = y_train[min_index]
from sklearn.metrics import accuracy_score
acc = accuracy_score(y_test, y_pred)
```

Accuracy of test-set: 0.888

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)



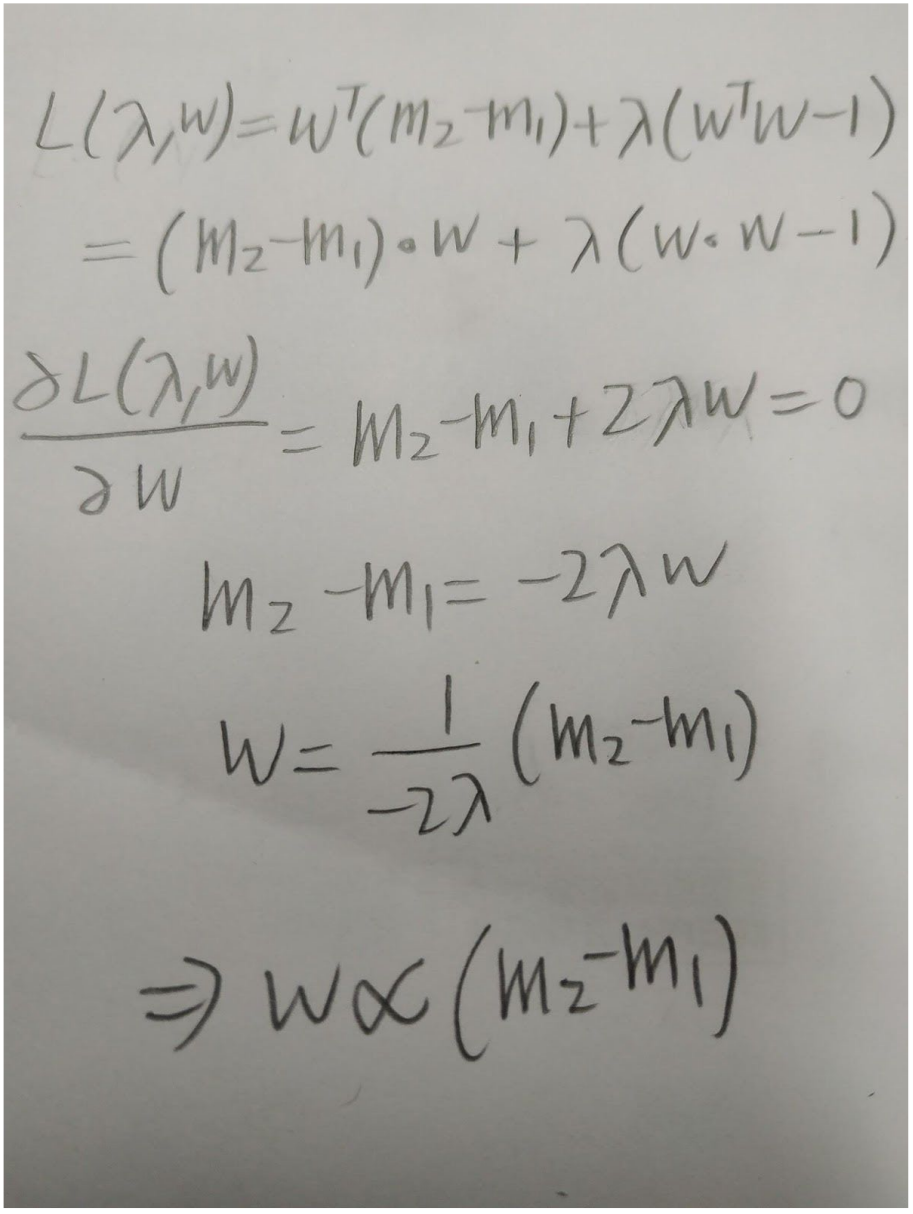

```
plt.figure(figsize=(4.8, 10))
plt.plot(c1[:, 0], c1[:, 1], '.', color='blue')
plt.plot(c2[:, 0], c2[:, 1], '.', color='green')
slope = w[1, 0] / w[0, 0]
intercept = 3
x = np.linspace(-0.3, 2, 300)
plt.plot(x, slope * x + intercept, color='red')
a = np.array([[1/slope, 1], [-slope, 1]])
for i in range(c1[:, 0].size):
    b = np.array([1 / slope * c1[i, 0] + c1[i, 1], intercept])
    proj = np.linalg.solve(a, b)
    plt.plot([c1[i, 0], proj[0]], [c1[i, 1], proj[1]], ':b')
    plt.plot(proj[0], proj[1], '.', color='blue')
for i in range(c2[:, 0].size):
    b = np.array([1 / slope * c2[i, 0] + c2[i, 1], intercept])
    proj = np.linalg.solve(a, b)
    plt.plot([c2[i, 0], proj[0]], [c2[i, 1], proj[1]], ':g')
    plt.plot(proj[0], proj[1], '.', color='green')
plt.axis("scaled")
plt.title("Projection line: w=" + str(slope) + " b=" +
str(intercept))
plt.show()
```



Part. 2, Questions (40%):

1. (20%) Show that maximization of the class separation criterion given by $L(\lambda, w) = w^T (m2 - m1) + \lambda(w^T w - 1)$ with respect to w, using a Lagrange multiplier to enforce the constraint $w^T w = 1$, leads to the result that

$w \propto (m2 - m1)$.



2. (20%) Using (eq 1) and (eq 2), derive the result (eq 3) for the posterior class probability in the two-class generative model with Gaussian densities, and verify the results (eq 4) and (eq 5) for the parameters w and w0.

(eq 1)
$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} = \frac{1}{1+\exp(-a)} = \sigma(a)$$

(eq 2)
$$a = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$$

(eq 3)
$$p(\mathcal{C}_1|\mathbf{x}) = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0)$$

(eq 4)
$$\mathbf{w} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$$

(eq 5)
$$w_0 = -\frac{1}{2}\boldsymbol{\mu}_1^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2 + \ln \frac{p(\mathcal{C}_1)}{p(\mathcal{C}_2)}$$

$$a = \ln \frac{P(x|C_1)P(C_1)}{P(x|C_2)P(C_2)} = \ln \frac{P(x|C_1)}{P(x|C_2)} + \ln \frac{P(C_1)}{P(C_2)}$$

$$P(x|C_1) = \frac{1}{(2\pi)^{D/2}} \frac{1}{\sigma_1} e^{-\frac{1}{2}\left(\frac{x-\mu_1}{\sigma_1}\right)^2}$$

$$P(x|C_2) = \frac{1}{(2\pi)^{D/2}} \frac{1}{\sigma_2} e^{-\frac{1}{2}\left(\frac{x-\mu_2}{\sigma_2}\right)^2}$$

$$\frac{P(x|C_1)}{P(x|C_2)} = \frac{\frac{1}{(2\pi)^{D/2}} \frac{1}{\sigma_1} e^{-\frac{1}{2}\left(\frac{x-\mu_1}{\sigma_1}\right)^2}}{\frac{1}{(2\pi)^{D/2}} \frac{1}{\sigma_2} e^{-\frac{1}{2}\left(\frac{x-\mu_2}{\sigma_2}\right)^2}}$$

$\sigma_1 = \sigma_2 = \sigma$

$$e^{-\frac{1}{2}\left(\frac{-2(\mu_1-\mu_2)x + \mu_1^2 - \mu_2^2}{\sigma^2}\right)}$$

$$a = -\frac{1}{2}\left(\frac{-2(\mu_1-\mu_2)x + \mu_1^2 - \mu_2^2}{\sigma^2}\right)$$

$$= \frac{\mu_1-\mu_2}{\sigma^2} x - \frac{1}{2}\frac{\mu_1^2-\mu_2^2}{\sigma^2} + \ln \frac{P(C_1)}{P(C_2)}$$

$$= wx + w_0$$

$$w = \frac{\mu_1-\mu_2}{\sigma^2} = \Sigma^{-1}(\mu_1-\mu_2)$$

$$w_0 = -\frac{1}{2}(\mu_1^2-\mu_2^2)\frac{1}{\sigma^2} + \ln \frac{P(C_1)}{P(C_2)}$$

$$= -\frac{1}{2}(\mu_1^T \Sigma^{-1} \mu_1) + \frac{1}{2}(\mu_2^T \Sigma^{-1} \mu_2) + \ln \frac{P(C_1)}{P(C_2)}$$