NYCU Pattern Recognition, Homework 2

Deadline: April 6, 23:59

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_AT0828/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 m_i (i=1, 2) of each 2 classes on training data

```
[7] ## Your code HERE
    m1, m2 = c1.mean(axis=0), c2.mean(axis=0)
```

```
[8] print(f"mean vector of class 1: {m1}", f"mean vector of class 2: {m2}")

    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

```
[9] ## Your code HERE
    s1 = np.matmul((c1-m1).T, c1-m1 )
    s2 = np.matmul((c2-m2).T, c2-m2 )
    sw = s1+s2
```

```
[10] assert sw.shape == (2,2)
     print(f"Within-class scatter matrix SW: {sw}")

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

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

```
[11] ## Your code HERE
     sb = np.matmul( (m1-m2).reshape((2,1)), (m1-m2).reshape((1,2)) )
```

```
[12] assert sb.shape == (2,2)
     print(f"Between-class scatter matrix SB: {sb}")

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

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

```
## Your code HERE
eig_vals, eig_vecs = np.linalg.eig( np.matmul(np.linalg.inv(sw), sb) )
idxs = np.argsort(abs(eig_vals),axis=0) # idxs [] is the order of the element
eig_vals = eig_vals[idxs[-1]] # idxs[-1] is index of max eigenvalue
eig_vecs = eig_vecs[: ,idxs[-1]]
w = eig_vecs.reshape((2,1))
```

```
assert w.shape == (2,1)
print(f" Fisher's linear discriminant: {w}")
```

```
Fisher's linear discriminant: [[ 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)

```
[26] # It is the nearest-mean-neighbor rule
     # Accuracy = 0.908
     prjx_test = np.matmul(x_test, w) # y = xw
     #print(prjx_test)
     prjm1 = np.matmul(m1, w)
     #print(prjm1)
     prjm2 = np.matmul(m2, w)
     test_result = np.array([]) # Put 0 or 1 (class number)
     for xt in prjx_test:
         if abs(prjm1 - xt) < abs(prjm2 - xt) :
             cls = 0
         else:
             cls = 1
         #print(nearest_index)
         test_result = np.append(test_result, np.array([ cls ]))
         # Append class of x_train[nearest_index] in test_result
```

```
[27] from sklearn.metrics import accuracy_score
     y_pred = test_result
     acc = accuracy_score(y_test, y_pred)
```

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

     Accuracy of test-set 0.908
```

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)

3
秒

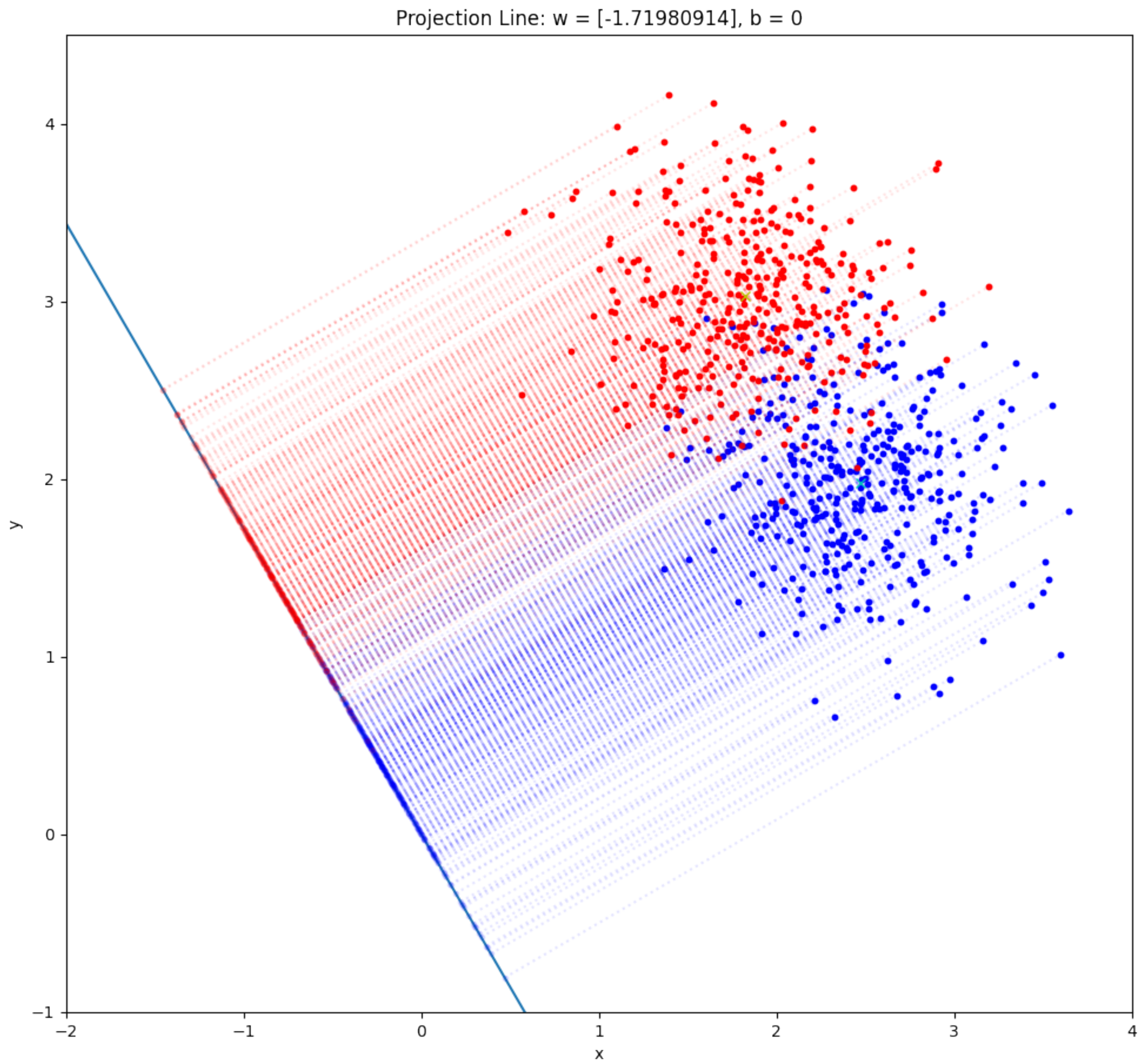
```python
plt.figure(figsize = (6*2, 5.5*2), dpi = 100)
plt.title('Projection Line: w = {}, b = {}'.format(w[1]/w[0],0))

plt.xlabel('x')
plt.ylabel('y')
plt.scatter(c1[:,0:1], c1[:,1:], s=10, c='blue')
plt.scatter(c2[:,0:1], c2[:,1:], s=10, c='red')


# Plot project line
x = np.linspace(-2,4,200)
y = w[1]/w[0] * x
plt.plot(x, y)
```

```python
# Project data point on the project line
res = w
r = res.reshape(2,)
n2 = np.linalg.norm(r)**2
for pt in c1:
        prj = r * r.dot(pt) / n2
        plt.plot([prj[0], pt[0]], [prj[1], pt[1]], 'b.:', alpha=0.1)
for pt in c2:
        prj = r * r.dot(pt) / n2
        plt.plot([prj[0], pt[0]], [prj[1], pt[1]], 'r.:', alpha=0.1)

# Plot m1, m2
plt.plot(m1[0], m1[1], 'cx')
plt.plot(m2[0], m2[1], 'yx')
plt.gca().axis('square')
plt.xlim(-2, 4)
plt.ylim(-1, 4.5)
plt.savefig('FLD result.jpg')
plt.show()
```



Projection Line: w = [-1.71980914], b = 0
y
x
4
3
2
1
0
−1
−2
−1
0
1
2
3
4

Part. 2, Questions (40%):

1. (10%) 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 $w \propto (m2 - m1)$.

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

$\rightarrow \frac{\partial L}{\partial w} = m_2 - m_1 + 2\lambda w$

Setting the gradient to zero gives

$m_2 - m_1 + 2\lambda w = 0 \Rightarrow w = -\frac{1}{2\lambda}(m_2 - m_1)$

form which it follows that $w \propto m_2 - m_1$

※

2. (15%) By making use of (eq 1), (eq 2), (eq 3), (eq 4), and (eq 5), show that the Fisher criterion (eq 6) can be written in the form (eq 7).

Note that x, m_1, m_2 can be transformed to y, m_1, m_2 by w

$$(m_2 - m_1)^2 = [w^T(m_2 - m_1)][w^T(m_2 - m_1)]^T \quad (\text{by eq } 2)$$

$$= w^T(m_2 - m_1)(m_2 - m_1)^T w = w^T S_B w \quad (\text{by eq } 4) \quad \dots ①$$

$$s_1^2 + s_2^2 = \sum_{n \in C_1} (y_n - m_1)^2 + \sum_{n \in C_2} (y_n - m_2)^2 \quad (\text{by eq } 3)$$

$$= \sum_{n \in C_1} (w^T x_n - w^T m_1)(w^T x_n - w^T m_1)^T + \sum_{n \in C_2} (w^T x_n - w^T m_2)(w^T x_n - w^T m_2)^T \quad (\text{by eq } 1 \& 2)$$

$$= \sum_{n \in C_1} w^T(x_n - m_1)[w^T(x_n - m_1)]^T + \sum_{n \in C_2} w^T(x_n - m_2)[w^T(x_n - m_2)]^T$$

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

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

$$= w^T S_W w \quad (\text{by eq } 5) \quad \dots ②$$

From ①、② equation: $J(w) = \dfrac{(m_2 - m_1)^2}{s_1^2 + s_2^2} = \dfrac{w^T S_B w}{w^T S_W w}$

(eq 1) $$y = \mathbf{w}^{\mathrm{T}}\mathbf{x}.$$

(eq 2) $$m_2 - m_1 = \mathbf{w}^{\mathrm{T}}(\mathbf{m}_2 - \mathbf{m}_1)$$

(eq 3) $$s_k^2 = \sum_{n \in \mathcal{C}_k} (y_n - m_k)^2$$

(eq 4) $$\mathbf{S}_{\mathrm{B}} = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^{\mathrm{T}}$$

(eq 5) $$\mathbf{S}_{\mathrm{W}} = \sum_{n \in \mathcal{C}_1} (\mathbf{x}_n - \mathbf{m}_1)(\mathbf{x}_n - \mathbf{m}_1)^{\mathrm{T}} + \sum_{n \in \mathcal{C}_2} (\mathbf{x}_n - \mathbf{m}_2)(\mathbf{x}_n - \mathbf{m}_2)^{\mathrm{T}}$$

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

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

3. (15%) By making use of the result (eq 8) for the derivative of the logistic sigmoid, show that the derivative of the error function (eq 9) for the logistic regression model is given by (eq 10), where $y_n = \sigma(a_n)$, $a_n = w^T\varphi_n$.

We start by computing the derivative of eq 9 w.r.t. y_n

$$\frac{\partial E}{\partial y_n} = \frac{1-t_n}{1-y_n} - \frac{t_n}{y_n} = \frac{y_n(1-t_n) - t_n(1-y_n)}{y_n(1-y_n)}$$

$$= \frac{y_n - y_n t_n - t_n + y_n t_n}{y_n(1-y_n)} = \frac{y_n - t_n}{y_n(1-y_n)} \quad \ldots ①$$

From eq 8, and $y_n = \sigma(a_n)$, we see that

$$\frac{\partial y_n}{\partial a_n} = \frac{\partial \sigma(a_n)}{\partial a_n} = \sigma(a_n)(1-\sigma(a_n)) = y_n(1-y_n) \quad \ldots ②$$

Finally, from $a_n = w^T\phi_n$, we see that

$$\nabla a_n = \frac{\partial a_n}{\partial w} = \phi_n \quad \ldots ③$$

Combining ①, ② and ③ using the chain rule, we obtain

$$\nabla E = \frac{\partial E}{\partial w} = \sum_{n=1}^{N} \frac{\partial E}{\partial y_n} \frac{\partial y_n}{\partial a_n} \frac{\partial a_n}{\partial w}$$

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

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

(eq 8) $$\frac{d\sigma}{da} = \sigma(1-\sigma).$$

(eq 9) $$E(\mathbf{w}) = -\ln p(\mathbf{t}|\mathbf{w}) = -\sum_{n=1}^{N} \{t_n \ln y_n + (1-t_n)\ln(1-y_n)\}$$

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

※