## A. 程式執行說明

輸入檔案位置、學習率(learning rate)、epoch 次數，程式即可運作

## B. 程式碼簡介

1. import 函式庫

```
import numpy as np
import matplotlib.pyplot as plt
```

2. 畫出圖形

X1 和 Y1 是存 d=1 的資料(mark 是 x)，X0 和 Y0 是存 d=-1 的資料(mark 是 o)，然畫出分布圖

```
def draw(x_test, y_test, W):
    fig = plt.figure()
    ax = fig.add_subplot(111)

    X1 = []
    Y1 = []
    X0 = []
    Y0 = []

    # 畫分散圖
    for i in range(0, len(x_test)):
        Arr = x_test[i]
        if y_test[i] == 1:
            X1.append(Arr[0][1])
            Y1.append(Arr[0][2])
        else:
            X0.append(Arr[0][1])
            Y0.append(Arr[0][2])

    X_Max_val = max(max(X1), max(X0))
    X_Min_val = min(min(X1), min(X0))

    ax.scatter(X1, Y1, marker="x")
    ax.scatter(X0, Y0, marker="o")
```

把訓練完成鍵結值畫函式-w0+w1*x+w2*y=0

移向：y=(-w1/w2)*x+(w0/w2)

因為過程中有可能遇到 w2=0 會導致 error 的狀況，所以寫了一個判斷式

```
# 畫出函式
l = np.linspace(X_Min_val, X_Max_val, 100, dtype=float)

if W[0][2] == 0.:
    a = W[0][0] / W[0][1]
    ax.plot(a*l, l, 'b-')

else:
    b = W[0][0]/W[0][2]
    a = -W[0][1]/W[0][2]
    ax.plot(l, a*l+b, 'b-')
plt.show()
```

3. 讀檔和資料處理

用 open 和 readlines 讀檔

```
def read_data(address):
    Input = open(address)
    data = Input.readlines()

    x_train = []
    x_test = []
    y_train = []
    y_test = []
```

因為資料有空格所以把他去掉

寫一個 for 迴圈把資料分成 training 和 testing

try 和 except 是因為資料有些是 int 有些是 float 才這樣分

```
# 分類train和test資料
index = 0
for line in data:
    buff = line.split(' ')
    if index % 3 == 0:
        try:
            # 才能transpose
            x_test.append(np.array([[-1, int(buff[0]), int(buff[1])]]))
        except:
            # 才能transpose
            x_test.append(np.array([[-1, float(buff[0]), float(buff[1])]]))

        y_test.append(int(buff[2]))

    else:
        try:
            x_train.append(np.array([[-1, int(buff[0]), int(buff[1])]]))
        except:
            x_train.append(
                np.array([[-1, float(buff[0]), float(buff[1])]]))

        y_train.append(int(buff[2]))

    index += 1

y_train = np.array(y_train)
y_test = np.array(y_test)
return x_train, y_train, x_test, y_test
```

4. Sign function

```
def func(x):
    if x >= 0:
        return 1
    else:
        return -1
```

5. 調整鍵結值

```
def modify(w, x, Y, Yp):
    if(Y == 1 and Yp == -1):
        new_w = w-x*learning_rate
    if(Y == -1 and Yp == 1):
        new_w = w+x*learning_rate

    return new_w
```

6. 把 d 進行正規化

有發現一些 txt 檔的 d 是用 1 和 2 表示

原本我用 0 和 1 去分但發現結果不理想

所以想說把 0 改成-1，但好像也沒什麼影響..

```
def norm(List_d1, List_d2):
    Max = List_d1.max()
    Min = List_d1.min()
    Norm_d1 = (List_d1-Min)/(Max-Min)
    Norm_d2 = (List_d2-Min)/(Max-Min)

    for i in range(0, len(Norm_d1)):
        if Norm_d1[i] == 0:
            Norm_d1[i] = -1
    for i in range(0, len(Norm_d2)):
        if Norm_d2[i] == 0:
            Norm_d2[i] = -1

    return Norm_d1, Norm_d2
```

7. 輸入資料

```
# 預設值
W = np.array([0, 0, 0])
learning_rate = 0.01
epoch = 100

# 輸入資料
address = input("輸入資料的位置:")
x_train, y_train, x_test, y_test = read_data(address)
Norm_y_train, Norm_y_test = norm(y_train, y_test)

learning_rate = float(input("輸入learning rate:"))
epoch = int(input("輸入epoch:"))
```

8. Training

鍵結值和 x 做內積再丟到 sign function 判斷+1 還是-1

如果和 d 不一樣就更改鍵結值

最後輸出正確率和鍵結值

```
# training
for k in range(0, epoch):
    wrong = 0
    for i in range(0, len(x_train)):
        Trans_X = x_train[i].T
        V = W.dot(Trans_X)
        Y = func(V)

        if Y != Norm_y_train[i]:
            wrong += 1
            W = modify(W, x_train[i])

print("training correct:", ((len(x_train)-wrong)/len(x_train))*100, "%")
print("W=", W)
```

9. Testing

把最後的鍵結值和 testing data 做內積，再丟進 sign function

如果和 d 一樣就 correct++

最後輸出正確率及用 testing data 和函式做圖

```
# testing
correct = 0
for i in range(0, len(x_test)):
    Trans_X = x_test[i].T
    V = W.dot(Trans_X)
    Y = func(V)

    if Y == Norm_y_test[i]:
        correct += 1

print("testing correct:", (correct/len(x_test))*100, "%")

draw(x_test, Norm_y_test, W)
```

C. 實驗結果

perceptron1

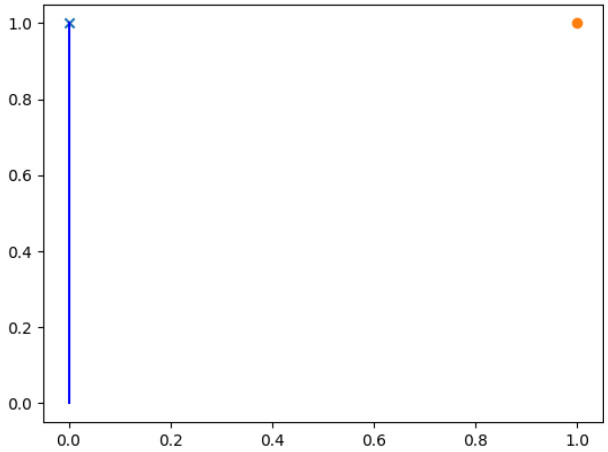
輸入 learning rate:0.01

輸入 epoch:1000

training correct: 100.0 %

W= [[ 0.  -0.01  0.0  ]]

testing correct: 100.0 %



perceptron2

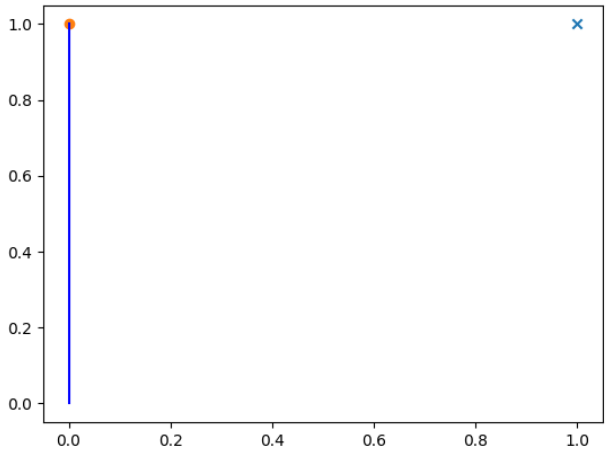
輸入 learning rate:0.01

輸入 epoch:1000

training correct: 100.0 %

W= [[ 0.   -0.01  0.0  ]]

testing correct: 0.0 %




2Ccircle1

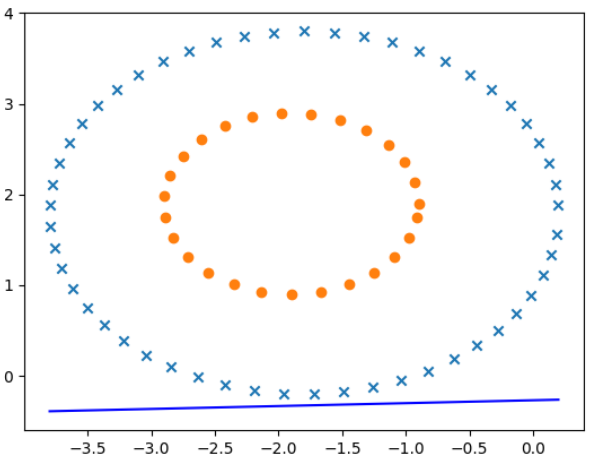
輸入 learning rate:0.01

輸入 epoch:1000

training correct: 96.875 %

W= [[-0.01   -0.00118415  0.03707451]]

testing correct: 66.25 %



2Circle1

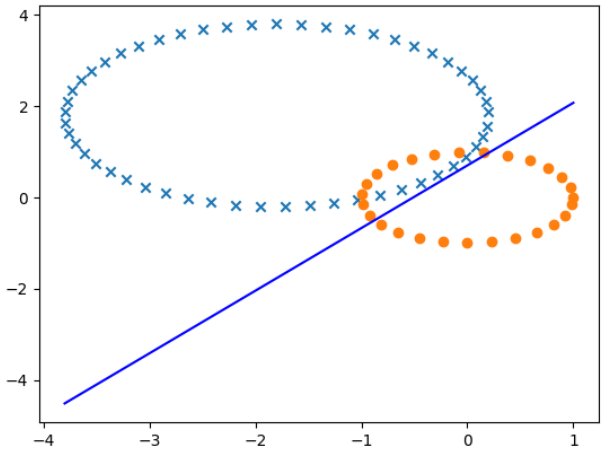
輸入 learning rate:0.01

輸入 epoch:1000

training correct: 93.75 %

W= [[ 0.05  -0.09767712  0.07118387]]

testing correct: 87.5 %




2Circle2

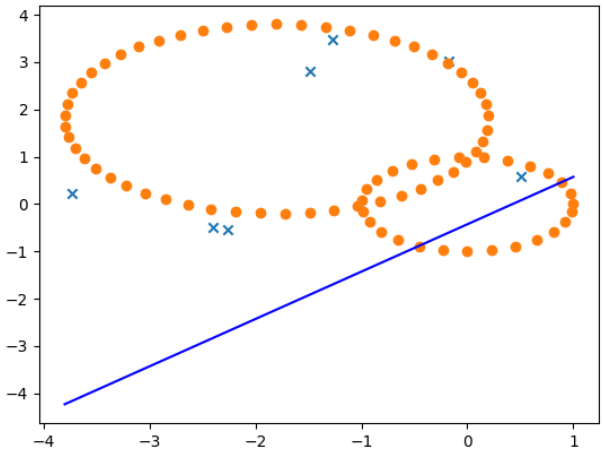
輸入 learning rate:0.01

輸入 epoch:1000

training correct: 18.497109826589593 %

W= [[ -770.3  -1801.50467113  1799.22486166]]

testing correct: 21.839080459770116 %



2CloseS

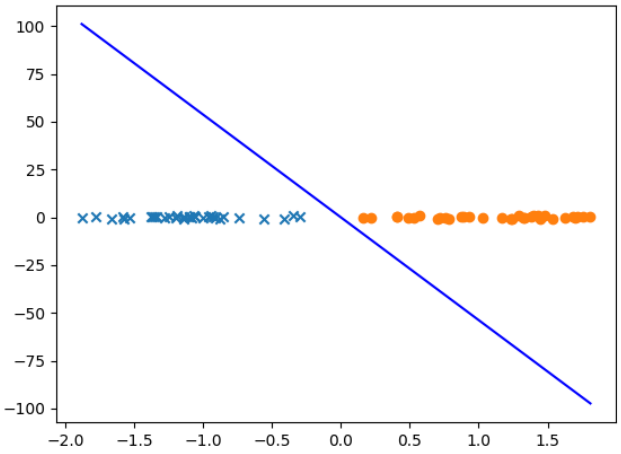
輸入 learning rate:0.01

輸入 epoch:1000

training correct: 100.0 %

W= [[ 0.  -0.0215  -0.0004]]

testing correct: 100.0 %




2CloseS2

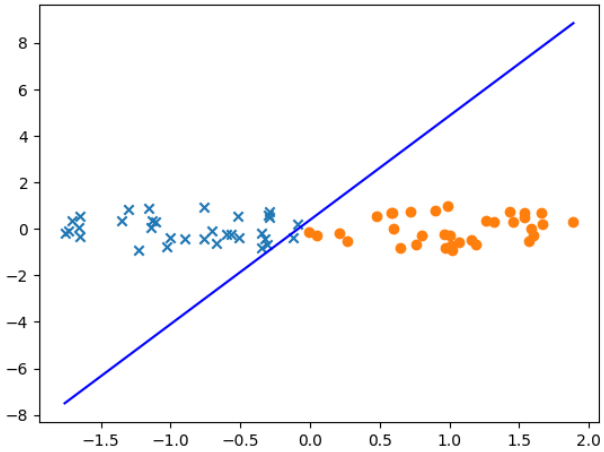
輸入 learning rate:0.01

輸入 epoch:1000

training correct: 100.0 %

W= [[ 0.01   -0.1165  0.026 ]]

testing correct: 98.50746268656717 %



2CloseS3

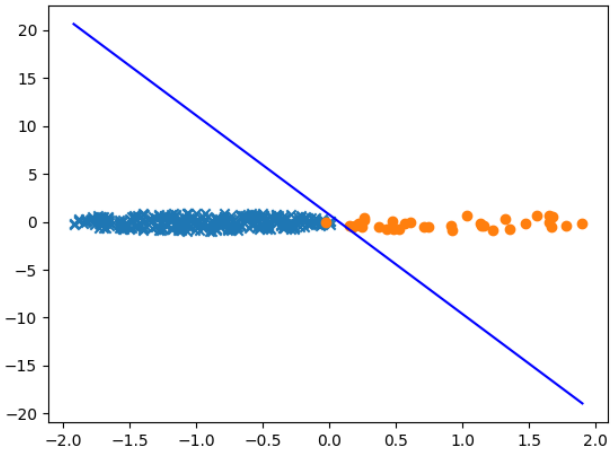
輸入 learning rate:0.01

輸入 epoch:1000

training correct: 100.0 %

W= [[-0.01  -0.1379  -0.0133]]

testing correct: 99.7005988023952 %




2cring

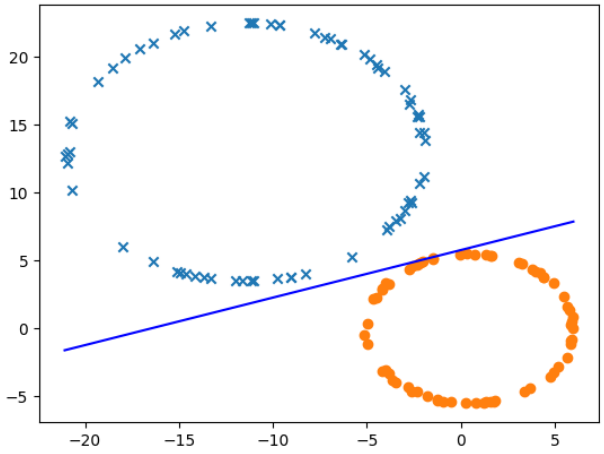
輸入 learning rate:0.01

輸入 epoch:1000

training correct: 100.0 %

W= [[ 0.26  -0.01572  0.0449 ]]

testing correct: 100.0 %



2CS

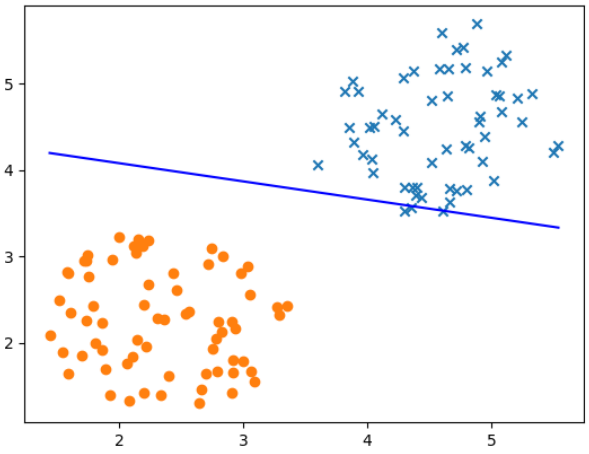
輸入 learning rate:0.01

輸入 epoch:1000

training correct: 100.0 %

W= [[0.24  0.011215  0.053334]]

testing correct: 97.58064516129032 %




2Hcircle1

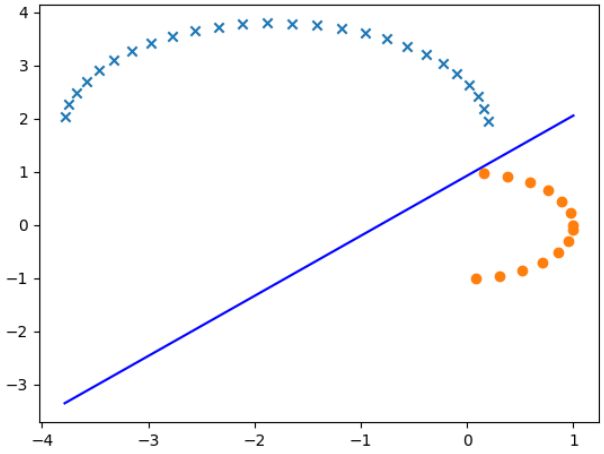
輸入 learning rate:0.01

輸入 epoch:1000

training correct: 100.0 %

W= [[ 0.02  -0.02431455  0.02150713]]

testing correct: 100.0 %



2ring

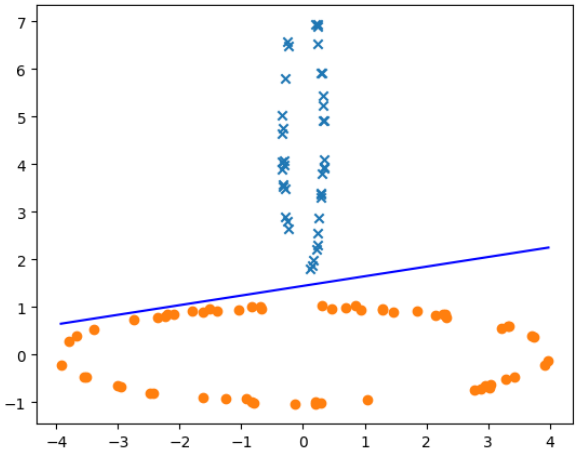
輸入 learning rate:0.01

輸入 epoch:1000

training correct: 100.0 %

W= [[ 0.05    -0.00702  0.0347 ]]

testing correct: 100.0 %




## D. 實驗結果分析及討論

1. 鍵結值

   會發現和法向量同一邊會是期望值是正的，和法向量不同邊會是期望值是負的。而鍵結值最後會影響分割點的函數圖形。

   函式係數的正負號真的很重要！！！

   因為這次實驗發現其中一項正負號錯了，結果就不太理想，到後面才找到這個 bug。

2. 訓練次數

   這次的作業發現大部分 epoch 在 20 以下，把圖形畫出來才會看到有較大的改變，之後圖形就不會有太大的變化了。

3. 學習率

   如果學習率設太大，會導致函式變化的幅度有點大，可能圖形畫出來的結果會不理想。如果設太小，訓練的進度會偏慢。不過因為這次的數據分的漂亮的，用相同 epoch，然後 learning rate 設 0.01、0.1 和 1，最後出來的結果其實差不多。

4. 訓練及測試正確率

   如果是線性可分割的資料，訓練的正確率和測試的正確率通常不會差太多，

而且正確率都蠻高的，不是 100%就是接近 100%(90%以上)，但測試資料的正確率可能就略低一點點。如果是線性不可分割的資料，像 perceptron2、2Ccircle1，測試的結果會比訓練的結果低蠻多的。而 2Circle2 有加入雜訊，結果比 2Circle1 還差。