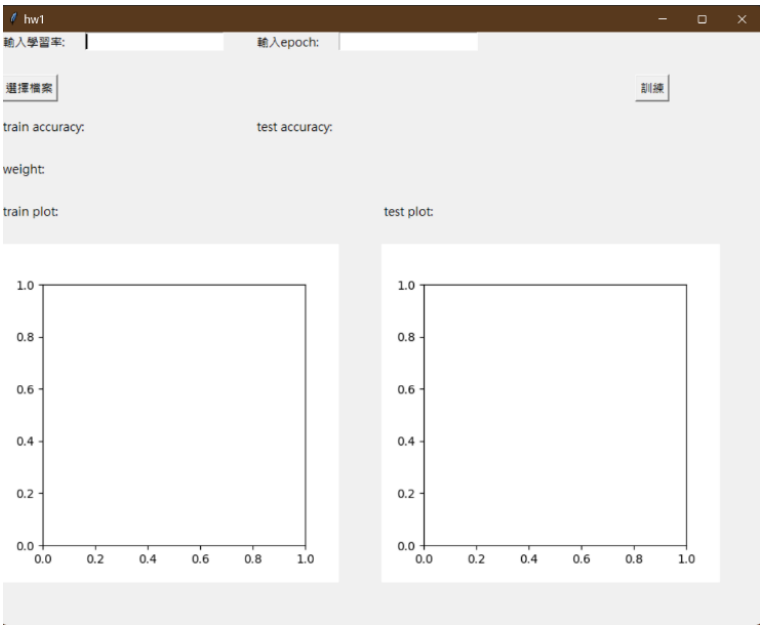
類神經網路作業一

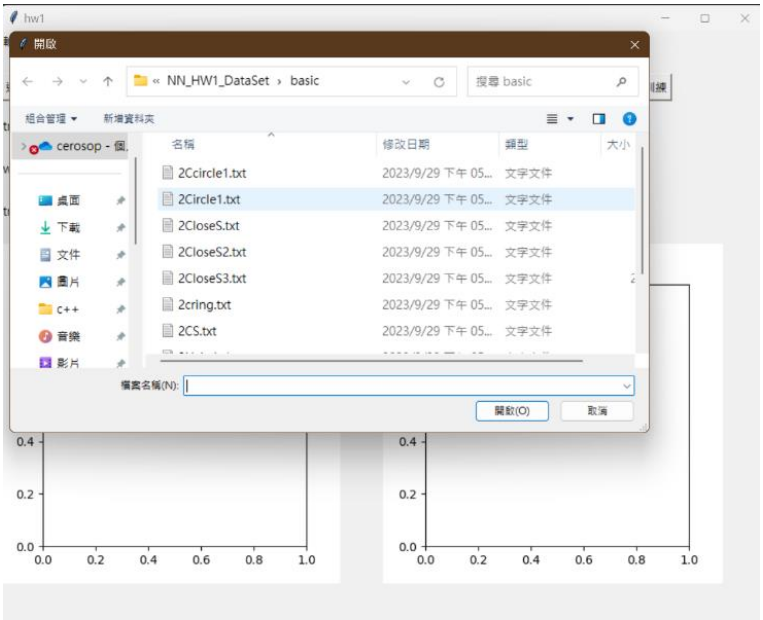
110502516 資工二 A 許尚軒

一、 GUI 說明

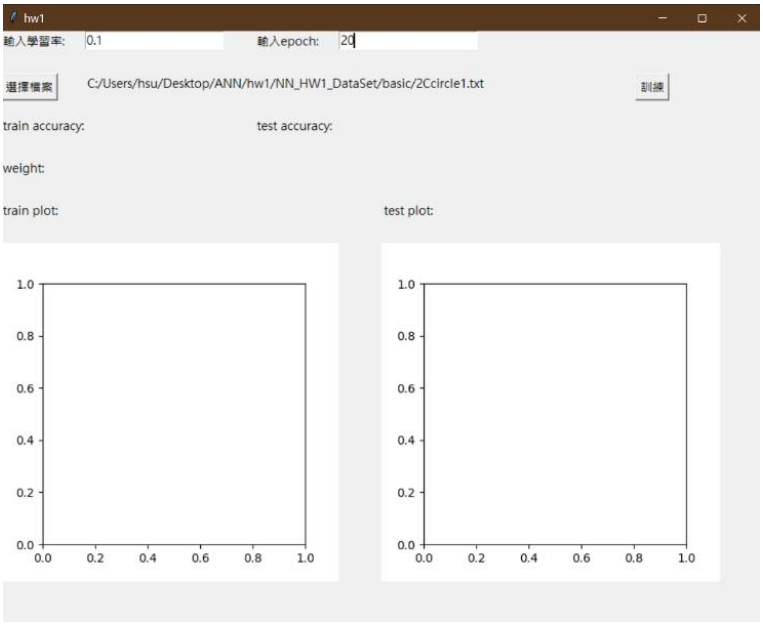
這是**起始畫面**，可以輸入學習率(浮點數)和 epoch(正整數)，也可以按”選擇檔案”選擇要訓練的資料



按下”選擇檔案”後會跳出此視窗選擇要訓練的資料



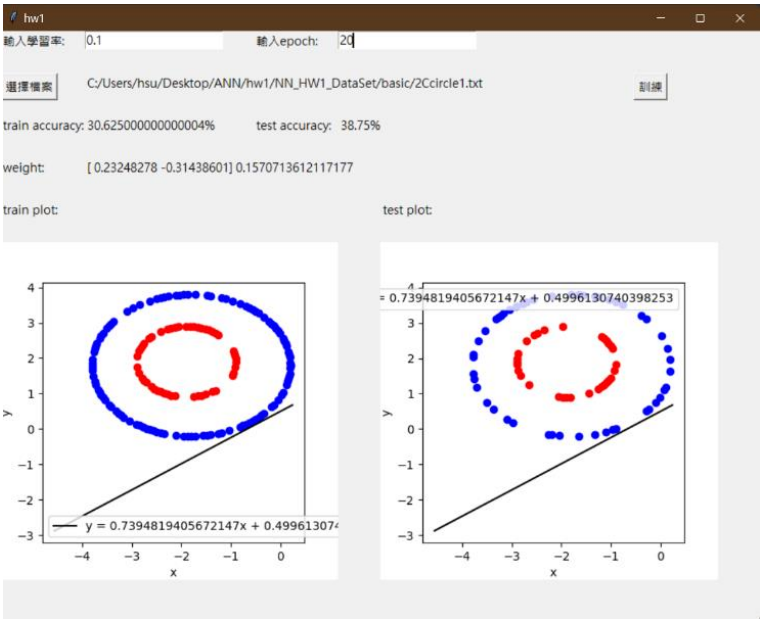
選擇好後會將檔案顯示在畫面上



按下訓練後就會開始訓練，訓練過程中會在，訓練完成後會將結果畫在下

圖，**左邊是 train data，右邊是 test data，紅點代表期望輸出為 1 的點，**

藍點為另一個，黑線為鍵節值和閥值在二維圖上代表的直線



二、 程式碼說明

建立視窗和要顯示在視窗上的物件

```python
#GUI介面
window = tk.Tk()
window.title('hw1')
window.geometry(f"{900}x{700}")

label_m = tk.Label(window, text='輸入學習率:')
label_b = tk.Label(window, text='輸入epoch:')
entry_m = tk.Entry(window)
entry_b = tk.Entry(window)
label_f = tk.Label(window, text='')
label_tra = tk.Label(window, text='train accuracy: ')
label_tea = tk.Label(window, text='test accuracy: ')
label_w = tk.Label(window, text='weight: ')
label_tra1 = tk.Label(window, text='')
label_tea1 = tk.Label(window, text='')
label_w1 = tk.Label(window, text='')
label_trp = tk.Label(window, text='train plot: ')
label_tep = tk.Label(window, text='test plot: ')

fig = Figure(figsize=(4, 4))
ax = fig.add_subplot(111)
canvas = FigureCanvasTkAgg(fig, master=window)

fig2 = Figure(figsize=(4, 4))
ax2 = fig2.add_subplot(111)
canvas2 = FigureCanvasTkAgg(fig2, master=window)
```

建立選取檔案按鈕並存取選取檔名

```python
#選取檔案
file_path = ""

def open_file_dialog():
    global file_path
    file_path = filedialog.askopenfilename()
    label_f.config(text = file_path)

select_file_button = tk.Button(window, text='選擇檔案', command=open_file_dialog)
```

建立訓練按鈕，先獲取資料並隨機分 train data 和 test data

```python
#訓練+畫線
def plot_line():
    data = np.loadtxt(file_path)

    #隨機打亂
    np.random.shuffle(data)

    X = data[:, :2]
    d = data[:, 2]

    num_samples = len(X)

    train_ratio = 2/3
    num_train = int(train_ratio * num_samples)

    X_train = X[:num_train]
    d_train = d[:num_train]

    X_test = X[num_train:]
    d_test = d[num_train:]
```

訓練過程，先隨機設初始鍵節值和閥值，**step_function 即為活化函數**，因為除了 perceptron 外的資料都分類為 1 和 2，因此寫當>=0 時 return 2，其他 return 1，之後跑 epoch 次每筆資料的訓練並以 error 修改鍵節值和閥值，**error 為期望輸出和實際輸出的差**，每筆資料訓練後會輸出該筆資料和改變後的鍵節值和閥值

```
    num_features = 2
    learning_rate = float(entry_m.get())

    #隨機初始鍵節值
    weights = np.random.rand(num_features)
    bias = np.random.rand()

    #測資分類為1和2
    def step_function(x):
        return 2 if x >= 0 else 1

    num_epochs = int(entry_b.get())
    print("w:", weights, "bias:", bias)
    print()

    for epoch in range(num_epochs):
        print(epoch)
        for i in range(num_train):
            #判斷此點對目前鍵節值而言是0還是1
            y = step_function(np.dot(X_train[i], weights) + bias)

            #不合修正
            error = d_train[i] - y
            weights += learning_rate * error * X_train[i]
            bias += learning_rate * error

            print(" ", i, "X:", X_train[i], ", d =", d_train[i], ", y =", y)
            print("  w:", weights, "bias:", bias)
            print()
```

訓練完畢後，會計算正確率，下圖為計算 train 的正確率的程式，計算 test 的程式與之相似

```
    #計算train正確率
    correct_predictions = 0

    print("train:")
    print("w:", weights, "bias:", bias)
    label_w1.config(text = (str(weights) + ' ' + str(bias)))
    print()

    for i in range(len(X_train)):
        y = step_function(np.dot(X_train[i], weights) + bias)
        print(i, "X:", X_train[i], ", d =", d_train[i], ", y =", y)
        print()
        if y == d_train[i]:
            correct_predictions += 1

    accuracy = correct_predictions / len(X_train)
    print(f'Train Accuracy: {accuracy * 100:.2f}%')
    print()
    label_tra1.config(text = (str(accuracy * 100.0) + '%'))
```

畫圖前先去找到該資料中 x_1 的最大最小值

```
    x_max = X.max(0)[0]
    x_min = X.min(0)[0]
```

並以該最大最小值再往外突出 0.2 倍來畫圖，線用黑色畫，點則為紅色和藍色

```
    #畫圖 紅點為1;藍點為2
    m = weights[0] / weights[1]
    m *= -1
    b = bias / weights[1]
    b *= -1

    tmp = abs(x_max) * 0.2
    x = np.linspace(x_min - tmp, x_max + tmp, 100)
    y = m * x + b

    ax.clear()

    ax.plot(x, y, label=f'y = {m}x + {b}', color='black')
    for i, p in enumerate(X_train):
        if d_train[i] == 1:
            ax.scatter(p[0], p[1], color = 'red')
        else:
            ax.scatter(p[0], p[1], color = 'blue')
    ax.set_xlabel('x')
    ax.set_ylabel('y')
    ax.legend()

    canvas.draw()
```

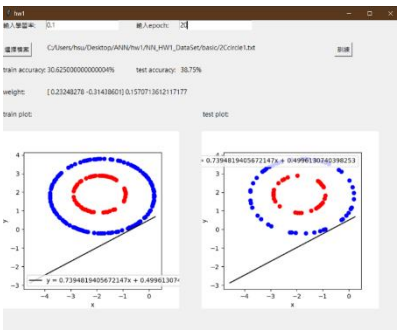
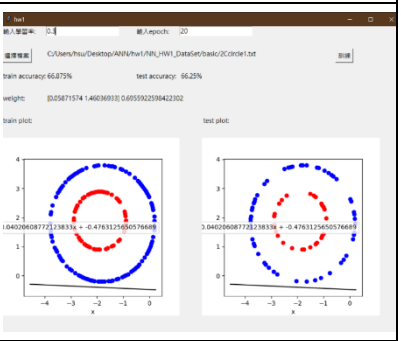
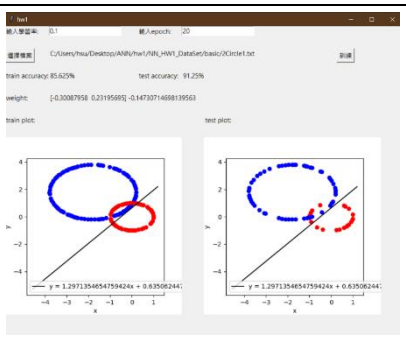
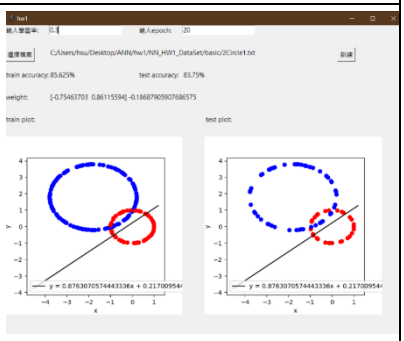
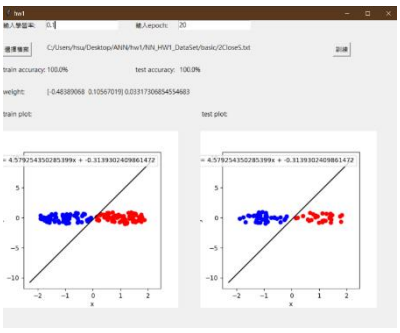
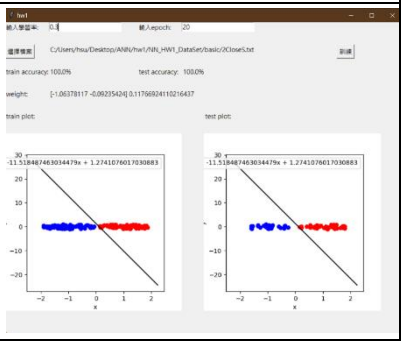
再將上述訓練按鈕的 function 放入訓練按鈕的指令中

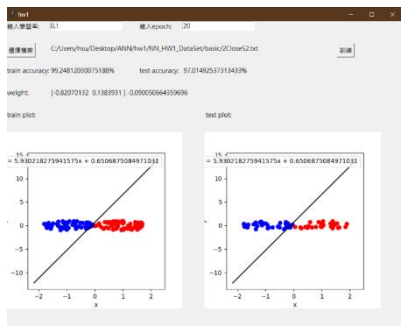
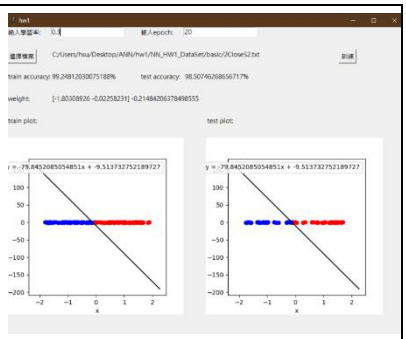
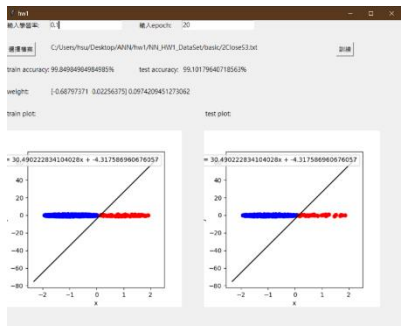
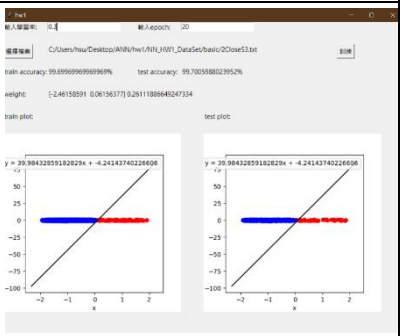
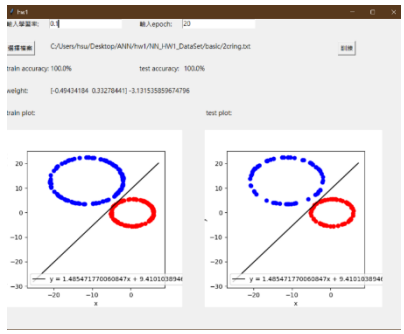
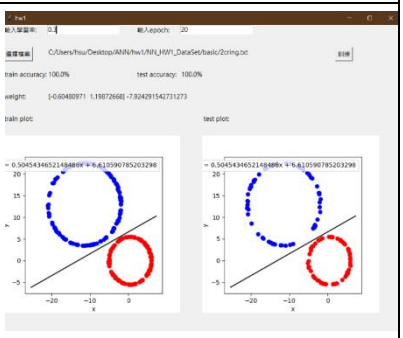
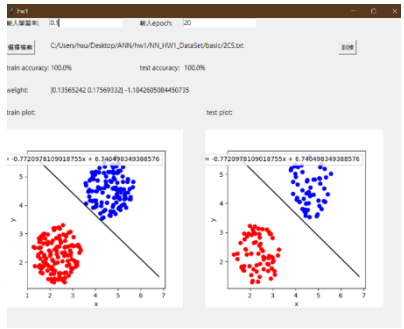
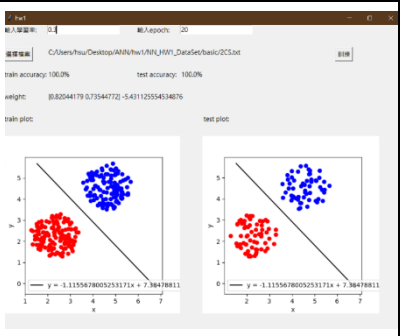
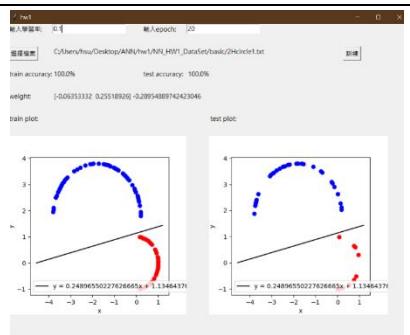
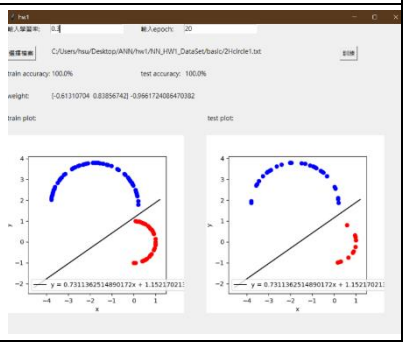
```
plot_button = tk.Button(window, text='訓練', command=plot_line)
```

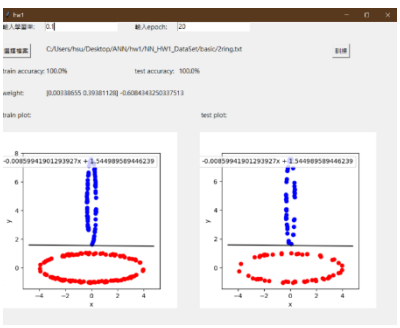
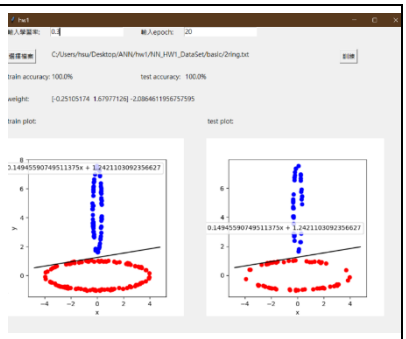
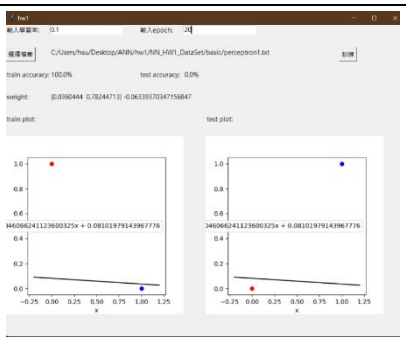
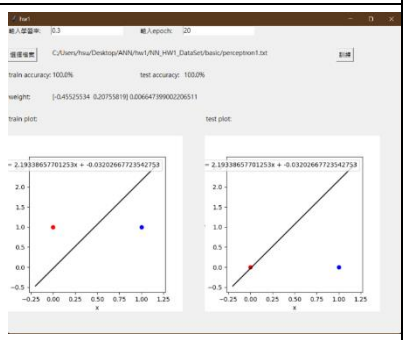
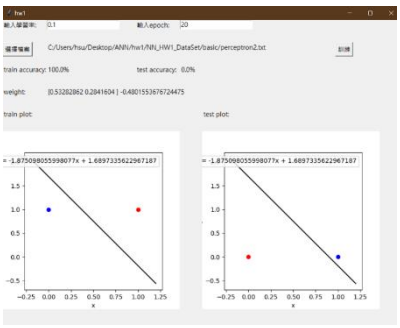
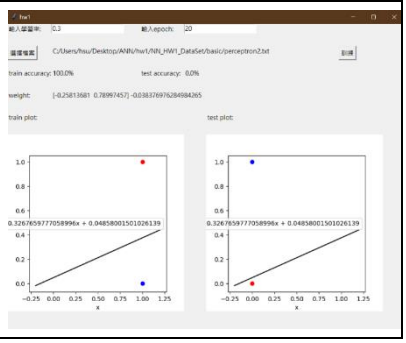
最後就是排版

```
180   #排版
181   label_m.place(x=0, y=0)
182   entry_m.place(x=100, y=0)
183   label_b.place(x=300, y=0)
184   entry_b.place(x=400, y=0)
185   select_file_button.place(x=0, y=50)
186   label_f.place(x=100, y=50)
187   plot_button.place(x=750, y=50)
188   label_tra.place(x=0, y=100)
189   label_tra1.place(x=100, y=100)
190   label_tea.place(x=300, y=100)
191   label_tea1.place(x=400, y=100)
192   label_w.place(x=0, y=150)
193   label_w1.place(x=100, y=150)
194   label_trp.place(x=0, y=200)
195   canvas.get_tk_widget().place(x=0, y=250)
196   label_tep.place(x=450, y=200)
197   canvas2.get_tk_widget().place(x=450, y=250)
198
199   window.mainloop()
```

三、 實驗結果

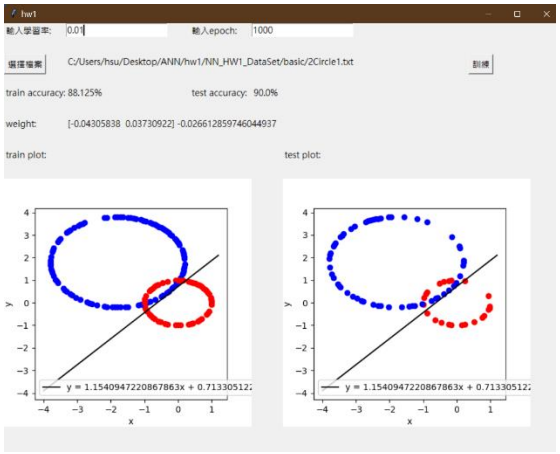
檔案\學習率	0.1	0.3
2Ccircle1		
2Circle1		
2CloseS		

2CloseS2		
2CloseS3		
2cring		
2CS		
2Hcircle1		

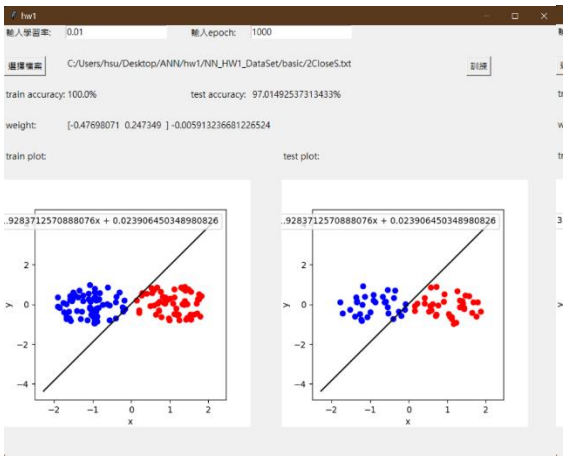
2ring		
perceptron1		
perceptron2		

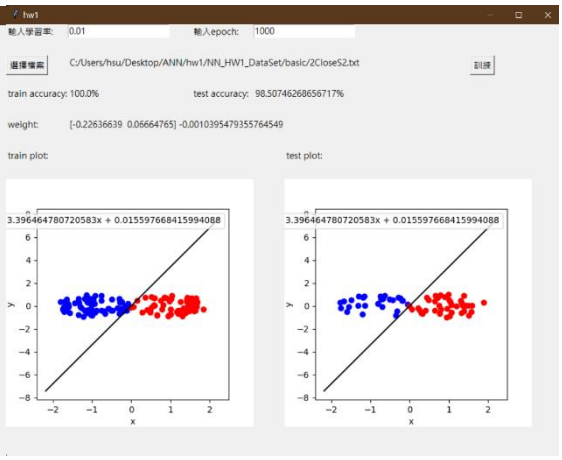
四、實驗結果分析和討論

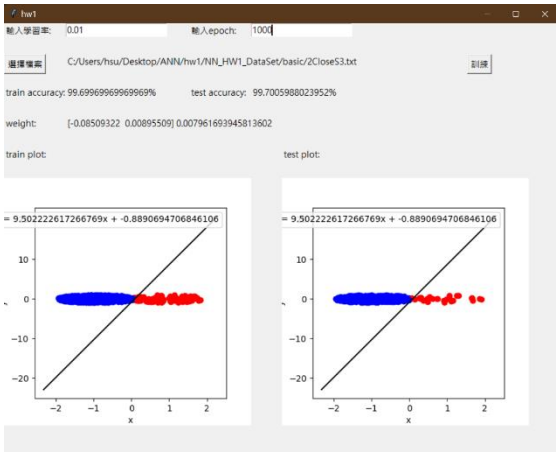
2Ccircle1、2Circle1 都是無法被一刀正確分兩區的，因此可以看到正確率皆無法達到 100%，而 2Circle1 點分布長得像兩圓相交，但為何訓練產生的鍵節值沒準確切在相交點上我認為是因為(1)**epoch 不夠**，為了驗證，我調整 epoch 到 1000 並下調學習率至 0.01，如下圖，可以發現即使訓練這麼多，還是無法準確地剛好切在交點上，我認為是因為(2)**兩圓相交區域與原點的距離不同**的關係導致兩圓錯誤區域對鍵節值的影響不同。



2CloseS、2CloseS2、2CloseS3 皆為兩區分布相近，但是可以被一刀正確區分的圖，所以可以看到訓練後的鍵節值可以達到幾乎 100%的正確率，但為何 train accuracy 沒達到 100%我認為是因為 **epoch 不夠**，因此我將 epoch 調至 1000 且學習率調至 0.01 後，的確正確率都十分接近 100%了，如下圖。







2cring、2CS、2Hcircle1、2ring 皆為兩區分比較開，且可以被一刀正確區分的圖，所以可以看到訓練後的結果可以正確將資料**成功分為兩區**。

Perceptron1、perceptron2 皆為只有 4 個點的圖，因此無論如何調整 epoch 和學習率都**無法精確**地在每次訓練都由 train data 去訓練出正確的感知機來區分 test data。