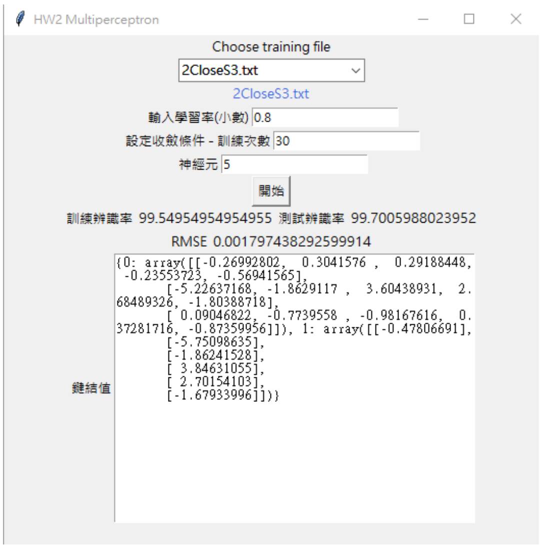
108522050 賴映如 類神經網路作業 2 – Multiperceptron

A. 程式執行說明:



Choose training file

2CloseS3.txt

2CloseS3.txt

輸入學習率(小數) 0.8

設定收斂條件 - 訓練次數 30

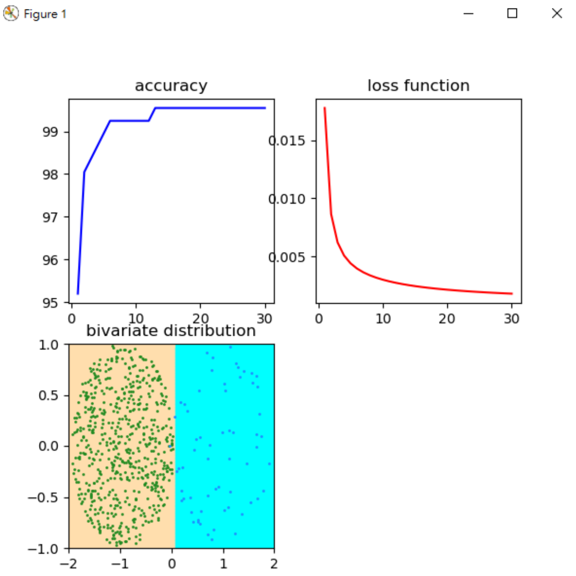
神經元 5

訓練辨識率 99.54954954954955 測試辨識率 99.7005988023952

RMSE 0.001797438292599914

鍵結值

```
{0: array([[-0.26992802,  0.3041576 ,  0.29188448,
 -0.23553723, -0.56941565],
       [-5.22637168, -1.8629117 ,  3.60438931,  2.
68489326, -1.80388718],
       [ 0.09046822, -0.7739558 , -0.98167616,  0.
37281716, -0.87359956]]), 1: array([[-0.47806691],
       [-5.75098635],
       [-1.86241528],
       [ 3.84631055],
       [ 2.70154103],
       [-1.67933996]])}
```

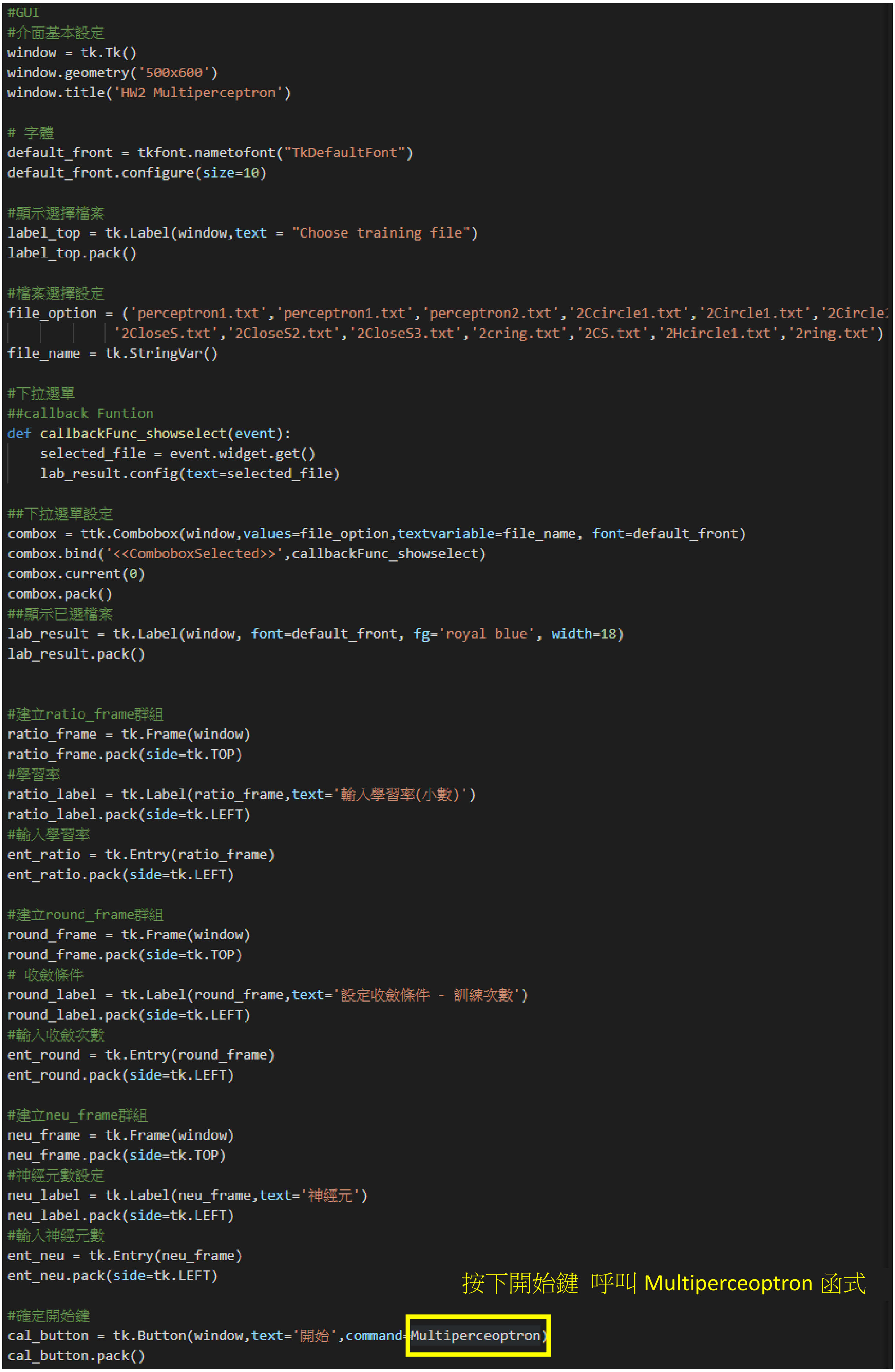


1. 選擇檔案 (選像僅含二維檔) – 藍字顯示已選檔案
2. 輸入學習率(小數)
3. 輸入收斂條件 (訓練次數限制)
4. 輸入神經元數量
5. 開始訓練
6. 介面顯示
    i. training data 辨識率
    ii. RMSE (均方根誤差)
    iii. 空白處顯示鍵結值
        1. array 0 是每筆資料分別對應幾顆神經元而有對應維度的 weight
        2. array1 是輸出層的 weight
7. 另一視窗跳出顯示兩維兩群的圖形介面(依照原始資料的期望值以不同顏色表示)& 辨識率、均方根誤差之折線圖

B. 程式簡介: [詳情直接打在程式註解上! : >]

#GUI 介面基本設定

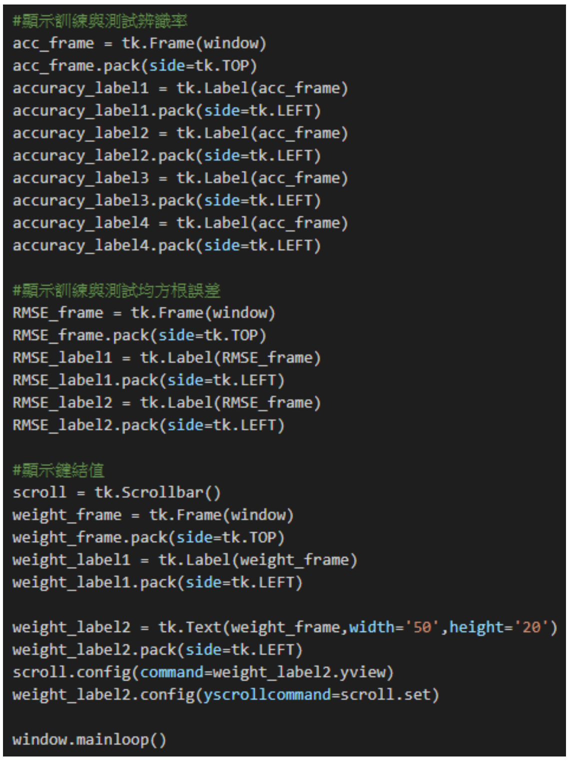
(下拉選單匯入檔案此僅提供二維資料選擇)



```
#GUI
#介面基本設定
window = tk.Tk()
window.geometry('500x600')
window.title('HW2 Multiperceptron')

# 字體
default_front = tkfont.nametofont("TkDefaultFont")
default_front.configure(size=10)

#顯示選擇檔案
label_top = tk.Label(window,text = "Choose training file")
label_top.pack()

#檔案選擇設定
file_option = ('perceptron1.txt','perceptron1.txt','perceptron2.txt','2Ccircle1.txt','2Circle1.txt','2Circle
                '2CloseS.txt','2CloseS2.txt','2CloseS3.txt','2cring.txt','2CS.txt','2Hcircle1.txt','2ring.txt')
file_name = tk.StringVar()

#下拉選單
##callback Funtion
def callbackFunc_showselect(event):
    selected_file = event.widget.get()
    lab_result.config(text=selected_file)

##下拉選單設定
combox = ttk.Combobox(window,values=file_option,textvariable=file_name, font=default_front)
combox.bind('<<ComboboxSelected>>',callbackFunc_showselect)
combox.current(0)
combox.pack()
##顯示已選檔案
lab_result = tk.Label(window, font=default_front, fg='royal blue', width=18)
lab_result.pack()


#建立ratio_frame群組
ratio_frame = tk.Frame(window)
ratio_frame.pack(side=tk.TOP)
#學習率
ratio_label = tk.Label(ratio_frame,text='輸入學習率(小數)')
ratio_label.pack(side=tk.LEFT)
#輸入學習率
ent_ratio = tk.Entry(ratio_frame)
ent_ratio.pack(side=tk.LEFT)

#建立round_frame群組
round_frame = tk.Frame(window)
round_frame.pack(side=tk.TOP)
# 收斂條件
round_label = tk.Label(round_frame,text='設定收斂條件 - 訓練次數')
round_label.pack(side=tk.LEFT)
#輸入收斂次數
ent_round = tk.Entry(round_frame)
ent_round.pack(side=tk.LEFT)

#建立neu_frame群組
neu_frame = tk.Frame(window)
neu_frame.pack(side=tk.TOP)
#神經元數設定
neu_label = tk.Label(neu_frame,text='神經元')
neu_label.pack(side=tk.LEFT)
#輸入神經元數
ent_neu = tk.Entry(neu_frame)
ent_neu.pack(side=tk.LEFT)

#確定開始鍵
cal_button = tk.Button(window,text='開始',command=Multiperceoptron)
cal_button.pack()
```

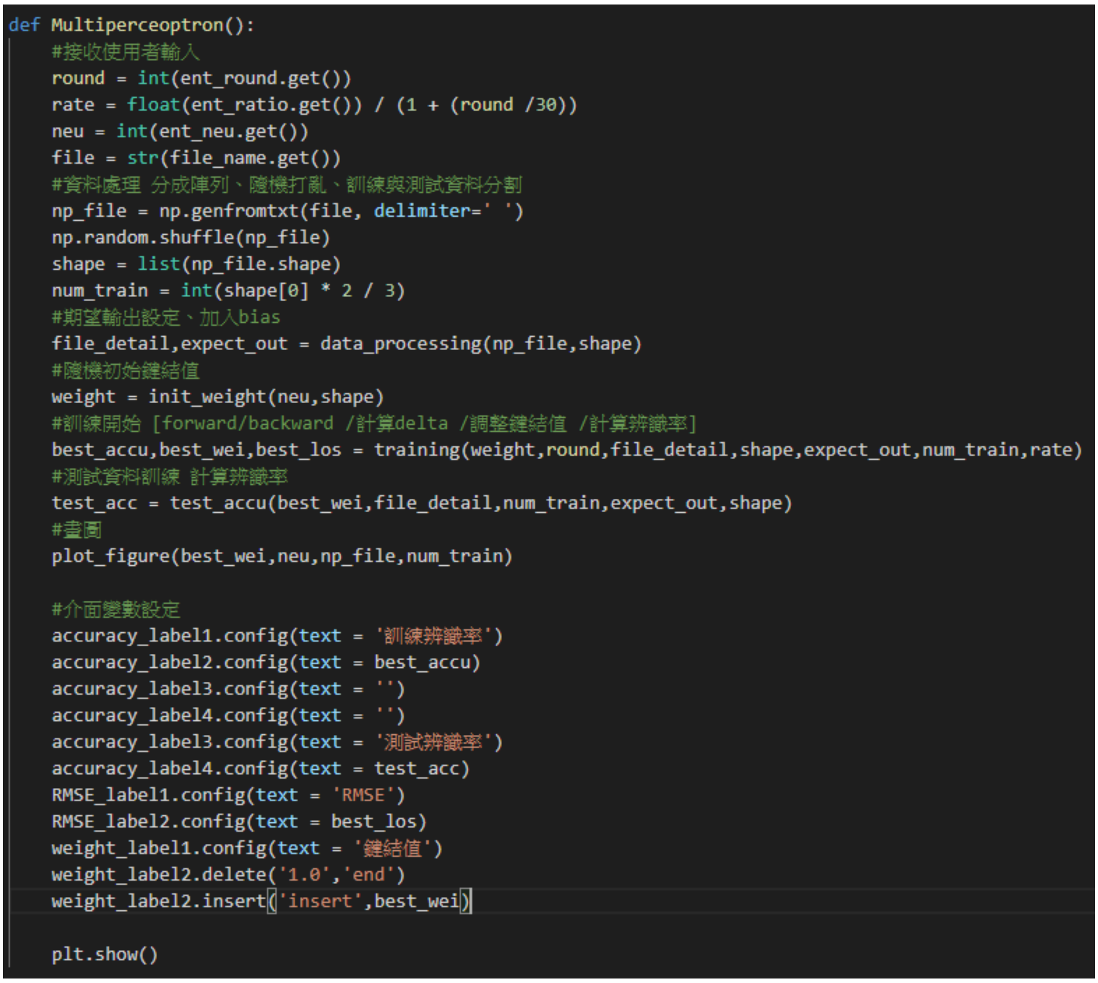
按下開始鍵 呼叫 Multiperceoptron 函式

#GUI 介面 顯示設定



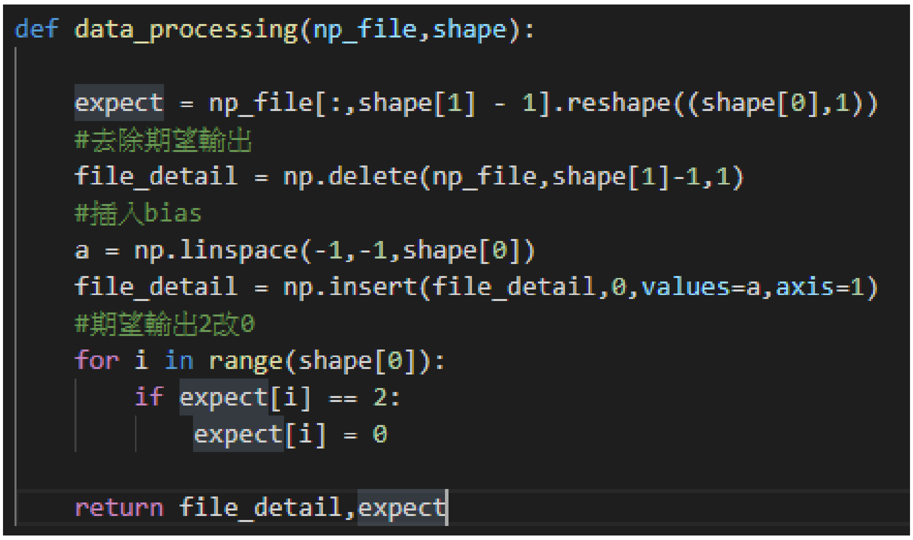
```
#顯示訓練與測試辨識率
acc_frame = tk.Frame(window)
acc_frame.pack(side=tk.TOP)
accuracy_label1 = tk.Label(acc_frame)
accuracy_label1.pack(side=tk.LEFT)
accuracy_label2 = tk.Label(acc_frame)
accuracy_label2.pack(side=tk.LEFT)
accuracy_label3 = tk.Label(acc_frame)
accuracy_label3.pack(side=tk.LEFT)
accuracy_label4 = tk.Label(acc_frame)
accuracy_label4.pack(side=tk.LEFT)

#顯示訓練與測試均方根誤差
RMSE_frame = tk.Frame(window)
RMSE_frame.pack(side=tk.TOP)
RMSE_label1 = tk.Label(RMSE_frame)
RMSE_label1.pack(side=tk.LEFT)
RMSE_label2 = tk.Label(RMSE_frame)
RMSE_label2.pack(side=tk.LEFT)

#顯示鍵結值
scroll = tk.Scrollbar()
weight_frame = tk.Frame(window)
weight_frame.pack(side=tk.TOP)
weight_label1 = tk.Label(weight_frame)
weight_label1.pack(side=tk.LEFT)

weight_label2 = tk.Text(weight_frame,width='50',height='20')
weight_label2.pack(side=tk.LEFT)
scroll.config(command=weight_label2.yview)
weight_label2.config(yscrollcommand=scroll.set)

window.mainloop()
```

# GUI 介面 顯示設定



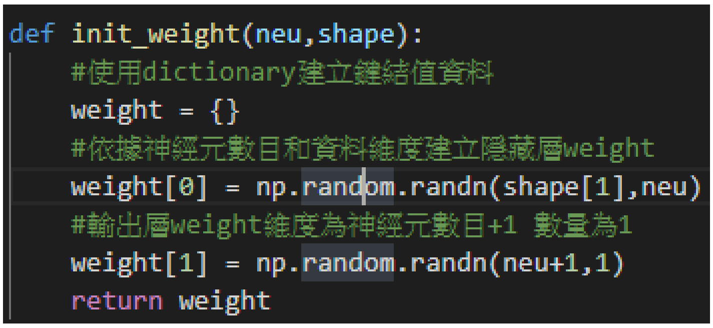
```
def Multiperceoptron():
    #接收使用者輸入
    round = int(ent_round.get())
    rate = float(ent_ratio.get()) / (1 + (round /30))
    neu = int(ent_neu.get())
    file = str(file_name.get())
    #資料處理 分成陣列、隨機打亂、訓練與測試資料分割
    np_file = np.genfromtxt(file, delimiter=' ')
    np.random.shuffle(np_file)
    shape = list(np_file.shape)
    num_train = int(shape[0] * 2 / 3)
    #期望輸出設定、加入bias
    file_detail,expect_out = data_processing(np_file,shape)
    #隨機初始鍵結值
    weight = init_weight(neu,shape)
    #訓練開始 [forward/backward /計算delta /調整鍵結值 /計算辨識率]
    best_accu,best_wei,best_los = training(weight,round,file_detail,shape,expect_out,num_train,rate)
    #測試資料訓練 計算辨識率
    test_acc = test_accu(best_wei,file_detail,num_train,expect_out,shape)
    #畫圖
    plot_figure(best_wei,neu,np_file,num_train)

    #介面變數設定
    accuracy_label1.config(text = '訓練辨識率')
    accuracy_label2.config(text = best_accu)
    accuracy_label3.config(text = '')
    accuracy_label4.config(text = '')
    accuracy_label3.config(text = '測試辨識率')
    accuracy_label4.config(text = test_acc)
    RMSE_label1.config(text = 'RMSE')
    RMSE_label2.config(text = best_los)
    weight_label1.config(text = '鍵結值')
    weight_label2.delete('1.0','end')
    weight_label2.insert('insert',best_wei)

    plt.show()
```

#將資料處理 讀空白分割成陣列、隨機打亂、插入 bias、訓練與測試資料分割



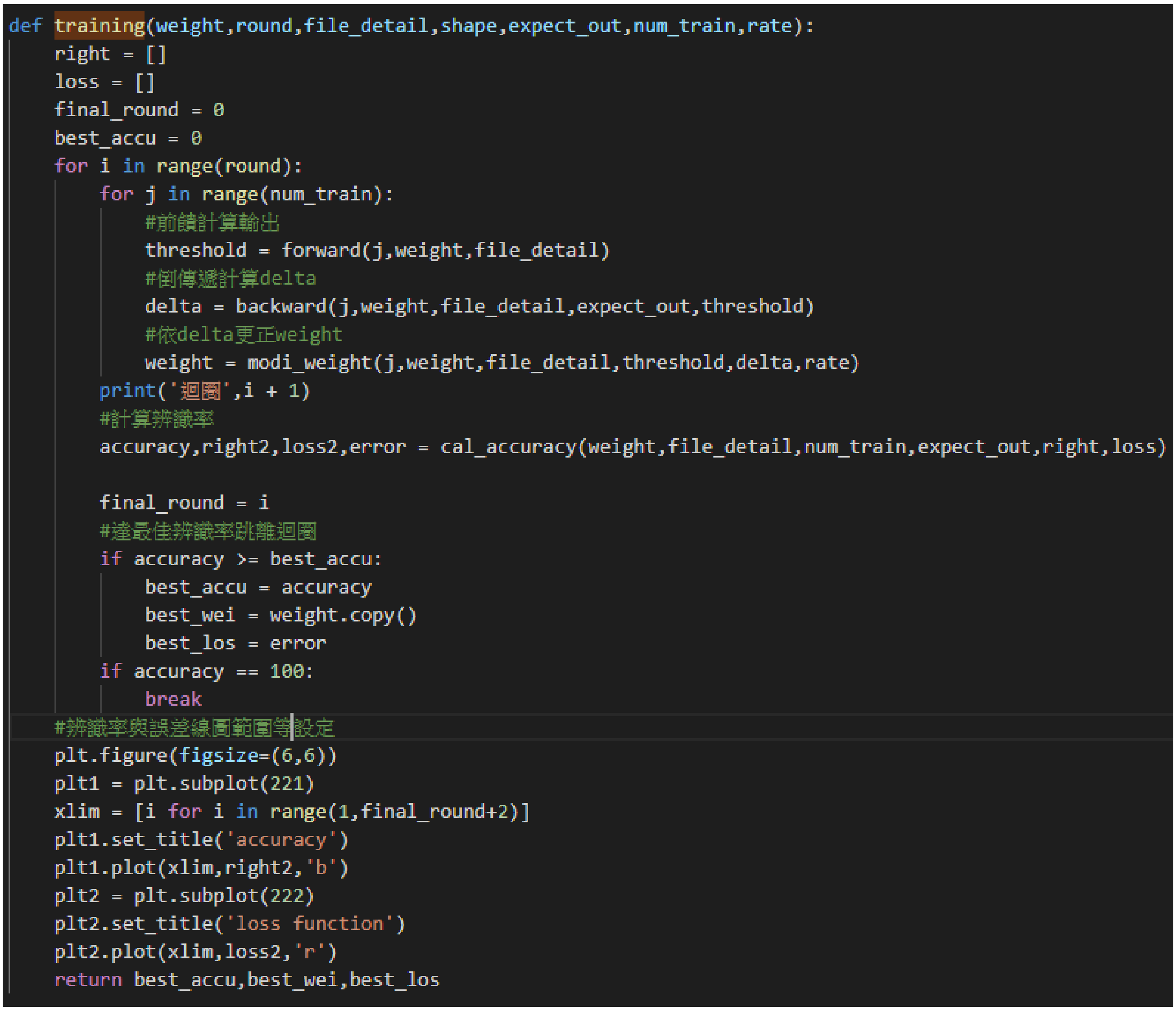
```
def data_processing(np_file,shape):

    expect = np_file[:,shape[1] - 1].reshape((shape[0],1))
    #去除期望輸出
    file_detail = np.delete(np_file,shape[1]-1,1)
    #插入bias
    a = np.linspace(-1,-1,shape[0])
    file_detail = np.insert(file_detail,0,values=a,axis=1)
    #期望輸出2改0
    for i in range(shape[0]):
        if expect[i] == 2:
            expect[i] = 0

    return file_detail,expect
```

#隨機初始鍵結值設定



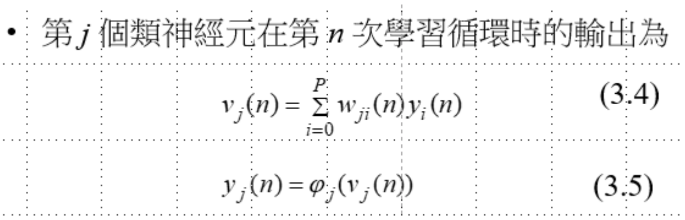
```
def init_weight(neu,shape):
    #使用dictionary建立鍵結值資料
    weight = {}
    #依據神經元數目和資料維度建立隱藏層weight
    weight[0] = np.random.randn(shape[1],neu)
    #輸出層weight維度為神經元數目+1 數量為1
    weight[1] = np.random.randn(neu+1,1)
    return weight
```

#訓練開始 [forward/backward /計算 delta /調整鍵結值 /計算辨識率]



```
def training(weight,round,file_detail,shape,expect_out,num_train,rate):
    right = []
    loss = []
    final_round = 0
    best_accu = 0
    for i in range(round):
        for j in range(num_train):
            #前饋計算輸出
            threshold = forward(j,weight,file_detail)
            #倒傳遞計算delta
            delta = backward(j,weight,file_detail,expect_out,threshold)
            #依delta更正weight
            weight = modi_weight(j,weight,file_detail,threshold,delta,rate)
        print('迴圈',i + 1)
        #計算辨識率
        accuracy,right2,loss2,error = cal_accuracy(weight,file_detail,num_train,expect_out,right,loss)

        final_round = i
        #達最佳辨識率跳離迴圈
        if accuracy >= best_accu:
            best_accu = accuracy
            best_wei = weight.copy()
            best_los = error
        if accuracy == 100:
            break
    #辨識率與誤差線圖範圍等設定
    plt.figure(figsize=(6,6))
    plt1 = plt.subplot(221)
    xlim = [i for i in range(1,final_round+2)]
    plt1.set_title('accuracy')
    plt1.plot(xlim,right2,'b')
    plt2 = plt.subplot(222)
    plt2.set_title('loss function')
    plt2.plot(xlim,loss2,'r')
    return best_accu,best_wei,best_los
```

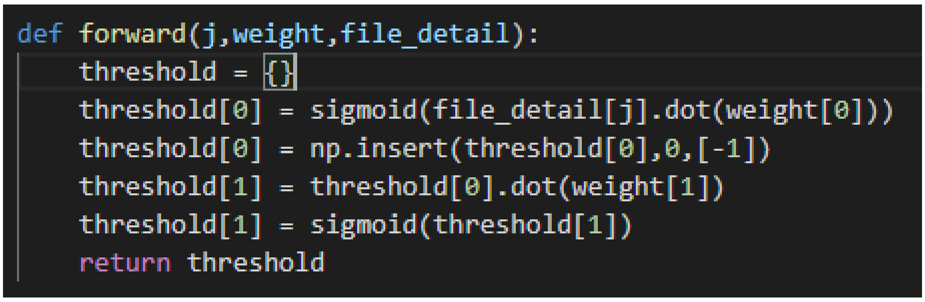
# forward 計算輸出



- 第 j 個類神經元在第 n 次學習循環時的輸出為

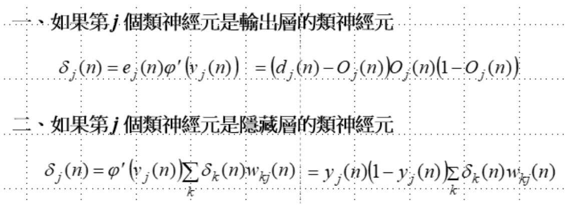
$$v_j(n) = \sum_{i=0}^{P} w_{ji}(n) y_i(n) \qquad (3.4)$$

$$y_j(n) = \varphi_j(v_j(n)) \qquad (3.5)$$



```
def forward(j,weight,file_detail):
    threshold = {}
    threshold[0] = sigmoid(file_detail[j].dot(weight[0]))
    threshold[0] = np.insert(threshold[0],0,[-1])
    threshold[1] = threshold[0].dot(weight[1])
    threshold[1] = sigmoid(threshold[1])
    return threshold
```

# backward 計算 delta

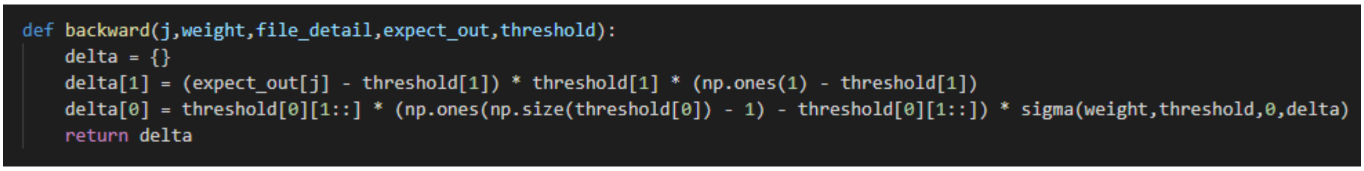


一、如果第 j 個類神經元是輸出層的類神經元

$$\delta_j(n) = e_j(n)\varphi'(v_j(n)) = (d_j(n) - O_j(n))O_j(n)(1 - O_j(n))$$

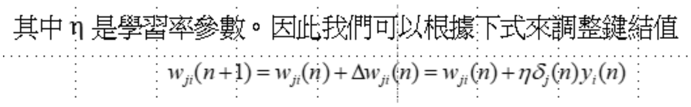
二、如果第 j 個類神經元是隱藏層的類神經元

$$\delta_j(n) = \varphi'(v_j(n))\sum_k \delta_k(n)w_{kj}(n) = y_j(n)(1 - y_j(n))\sum_k \delta_k(n)w_{kj}(n)$$



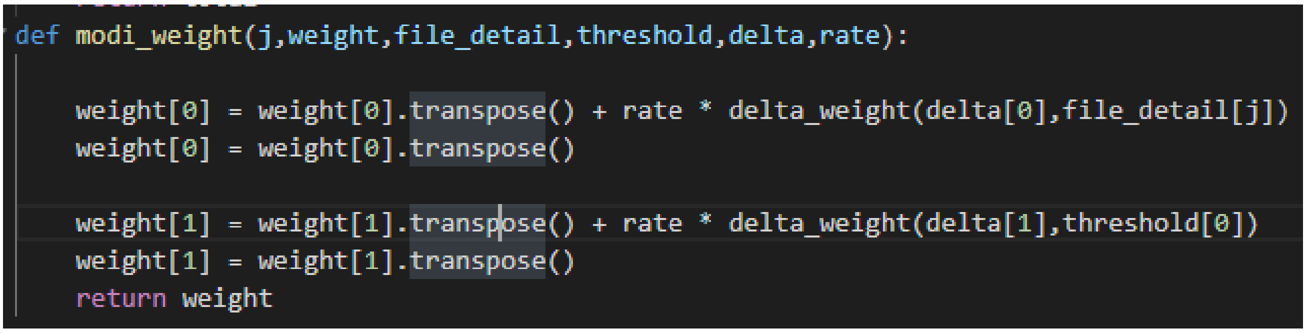
```
def backward(j,weight,file_detail,expect_out,threshold):
    delta = {}
    delta[1] = (expect_out[j] - threshold[1]) * threshold[1] * (np.ones(1) - threshold[1])
    delta[0] = threshold[0][1::] * (np.ones(np.size(threshold[0]) - 1) - threshold[0][1::]) * sigma(weight,threshold,0,delta)
    return delta
```

# 依據倒傳遞算出之 delta 調整 weight



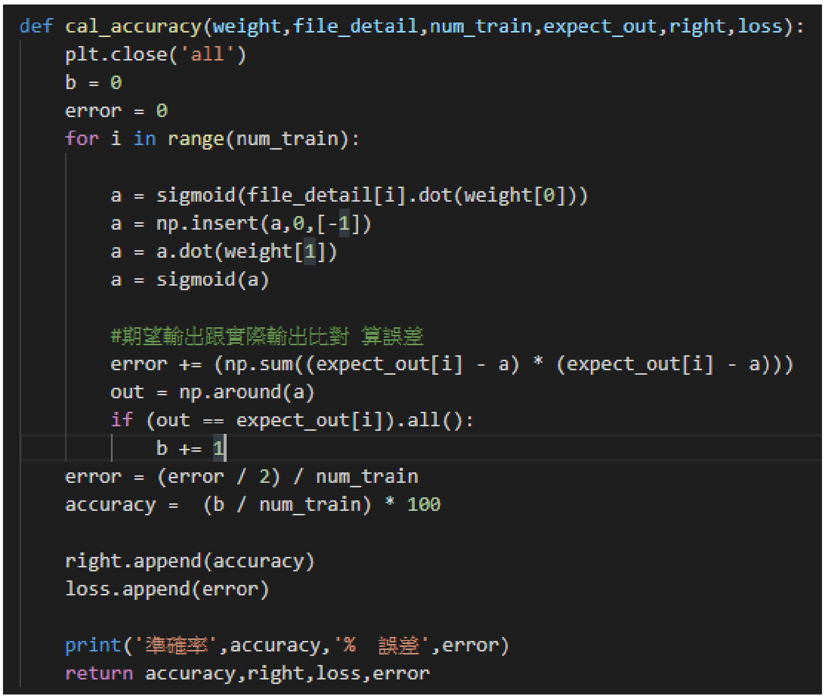
其中 η 是學習率參數。因此我們可以根據下式來調整鍵結值

$$w_{ji}(n+1) = w_{ji}(n) + \Delta w_{ji}(n) = w_{ji}(n) + \eta\delta_j(n)y_i(n)$$



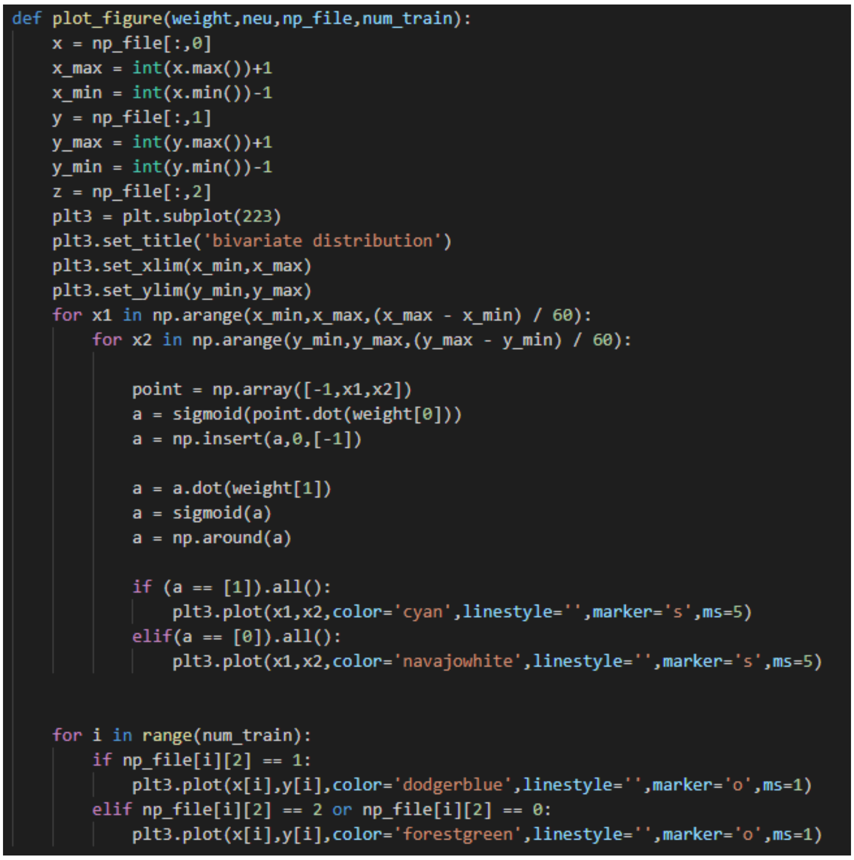
```
def modi_weight(j,weight,file_detail,threshold,delta,rate):

    weight[0] = weight[0].transpose() + rate * delta_weight(delta[0],file_detail[j])
    weight[0] = weight[0].transpose()

    weight[1] = weight[1].transpose() + rate * delta_weight(delta[1],threshold[0])
    weight[1] = weight[1].transpose()
    return weight
```

# 計算辨識率及誤差



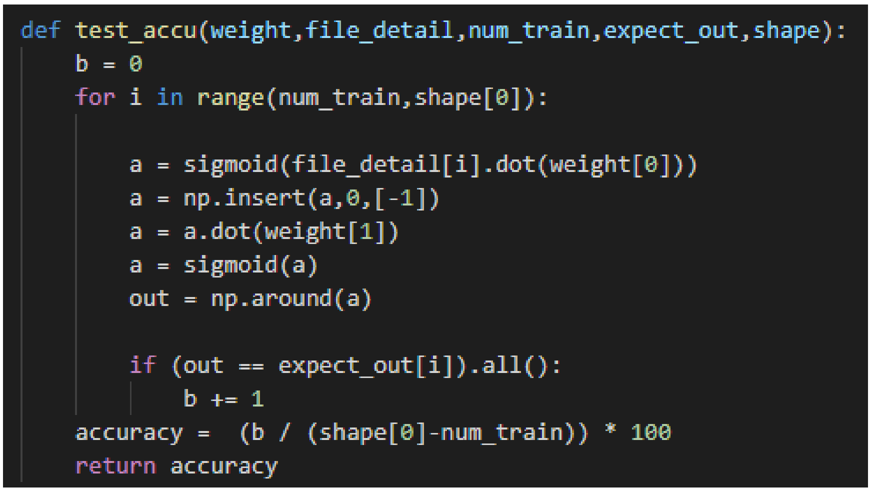
```
def cal_accuracy(weight,file_detail,num_train,expect_out,right,loss):
    plt.close('all')
    b = 0
    error = 0
    for i in range(num_train):

        a = sigmoid(file_detail[i].dot(weight[0]))
        a = np.insert(a,0,[-1])
        a = a.dot(weight[1])
        a = sigmoid(a)

        #期望輸出跟實際輸出比對 算誤差
        error += (np.sum((expect_out[i] - a) * (expect_out[i] - a)))
        out = np.around(a)
        if (out == expect_out[i]).all():
            b += 1
    error = (error / 2) / num_train
    accuracy =  (b / num_train) * 100

    right.append(accuracy)
    loss.append(error)

    print('準確率',accuracy,'%  誤差',error)
    return accuracy,right,loss,error
```

# 依座標軸每點(60*60= 3600 點)帶入訓練畫出背景顏色分類分布圖　再將訓練資料畫上 可清楚看出分類正確與否



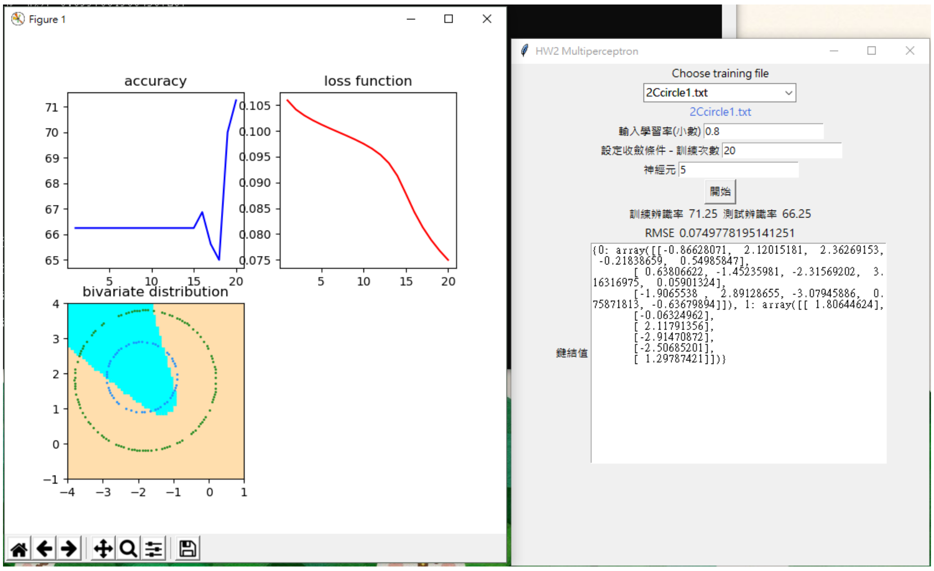
```
def plot_figure(weight,neu,np_file,num_train):
    x = np_file[:,0]
    x_max = int(x.max())+1
    x_min = int(x.min())-1
    y = np_file[:,1]
    y_max = int(y.max())+1
    y_min = int(y.min())-1
    z = np_file[:,2]
    plt3 = plt.subplot(223)
    plt3.set_title('bivariate distribution')
    plt3.set_xlim(x_min,x_max)
    plt3.set_ylim(y_min,y_max)
    for x1 in np.arange(x_min,x_max,(x_max - x_min) / 60):
        for x2 in np.arange(y_min,y_max,(y_max - y_min) / 60):

            point = np.array([-1,x1,x2])
            a = sigmoid(point.dot(weight[0]))
            a = np.insert(a,0,[-1])

            a = a.dot(weight[1])
            a = sigmoid(a)
            a = np.around(a)

            if (a == [1]).all():
                plt3.plot(x1,x2,color='cyan',linestyle='',marker='s',ms=5)
            elif(a == [0]).all():
                plt3.plot(x1,x2,color='navajowhite',linestyle='',marker='s',ms=5)


    for i in range(num_train):
        if np_file[i][2] == 1:
            plt3.plot(x[i],y[i],color='dodgerblue',linestyle='',marker='o',ms=1)
        elif np_file[i][2] == 2 or np_file[i][2] == 0:
            plt3.plot(x[i],y[i],color='forestgreen',linestyle='',marker='o',ms=1)
```

# 計算測試資料辨識率



```
def test_accu(weight,file_detail,num_train,expect_out,shape):
    b = 0
    for i in range(num_train,shape[0]):

        a = sigmoid(file_detail[i].dot(weight[0]))
        a = np.insert(a,0,[-1])
        a = a.dot(weight[1])
        a = sigmoid(a)
        out = np.around(a)

        if (out == expect_out[i]).all():
            b += 1
    accuracy =  (b / (shape[0]-num_train)) * 100
    return accuracy
```

c. 實驗結果與分析討論:



Choose training file

2Ccircle1.txt

2Ccircle1.txt

輸入學習率(小數) 0.8

設定收斂條件 - 訓練次數 20

神經元 5

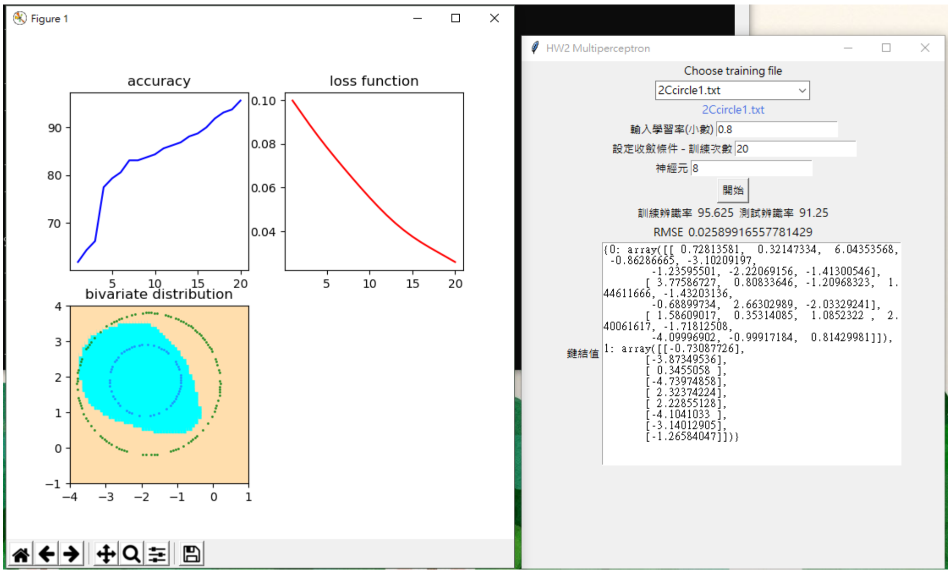
訓練辨識率 71.25 測試辨識率 66.25

RMSE 0.0749778195141251

鍵結值

```
{0: array([[-0.86628071,  2.12015181,  2.36269153,
 -0.21838659,  0.54985847],
       [ 0.63806622, -1.45235981, -2.31569202,  3.
16316975,  0.05901324],
       [-1.9065538 ,  2.89128655, -3.07945886,  0.
75871813, -0.63679894]]), 1: array([[ 1.80644624],
       [-0.06324962],
       [ 2.11791356],
       [-2.91470872],
       [-2.50685201],
       [ 1.29787421]])}
```

增加神經元數 － 可有效增加辨識率:



Choose training file

2Ccircle1.txt

2Ccircle1.txt

輸入學習率(小數) 0.8

設定收斂條件 - 訓練次數 20

神經元 8

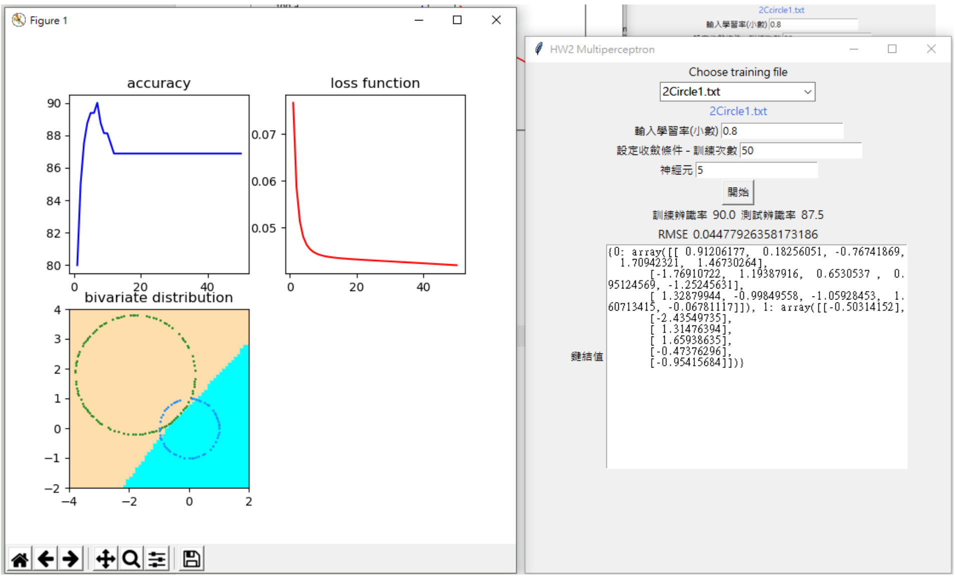
訓練辨識率 95.625 測試辨識率 91.25

RMSE 0.02589916557781429

鍵結值

```
{0: array([[ 0.72813581,  0.32147334,  6.04353568,
 -0.86286665, -3.10209197,
        -1.23595501, -2.22069156, -1.41300546],
       [ 3.77586727,  0.80833646, -1.20968323,  1.
44611666, -1.43203136,
        -0.68899734,  2.66302989, -2.03329241],
       [ 1.58609017,  0.35314085,  1.0852322 ,  2.
40061617, -1.71812508,
        -4.09996902, -0.99917184,  0.81429981]]),
1: array([[-0.73087726],
       [-3.87349536],
       [ 0.3455058 ],
       [-4.73974858],
       [ 2.32374224],
       [ 2.22855128],
       [-4.1041033 ],
       [-3.14012905],
       [-1.26584047]])}
```

增加訓練次數　－ 可有效增加辨識率:



Choose training file

2Circle1.txt

2Circle1.txt

輸入學習率(小數) 0.8

設定收斂條件 - 訓練次數 50

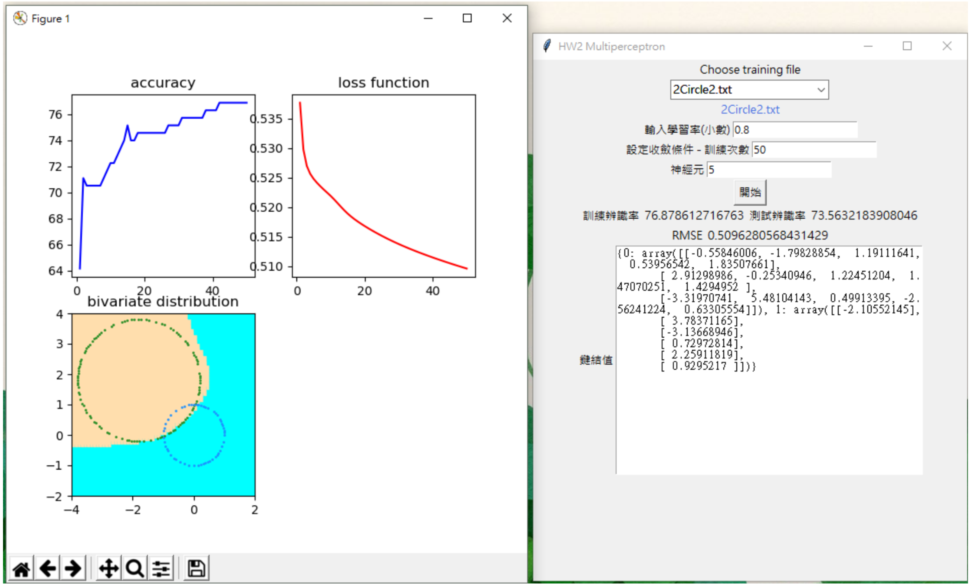
神經元 5

訓練辨識率 90.0 測試辨識率 87.5

RMSE 0.04477926358173186

鍵結值

```
{0: array([[ 0.91206177,  0.18256051, -0.76741869,
  1.70942321,  1.46730264],
       [-1.76910722,  1.19387916,  0.6530537 ,  0.
95124569, -1.25245631],
       [ 1.32879944, -0.99849558, -1.05928453,  1.
60713415, -0.06781117]]), 1: array([[-0.50314152],
       [-2.43549735],
       [ 1.31476394],
       [ 1.65938635],
       [-0.47376296],
       [-0.95415684]])}
```



Choose training file

2Circle2.txt

2Circle2.txt

輸入學習率(小數) 0.8

設定收斂條件 - 訓練次數 50

神經元 5

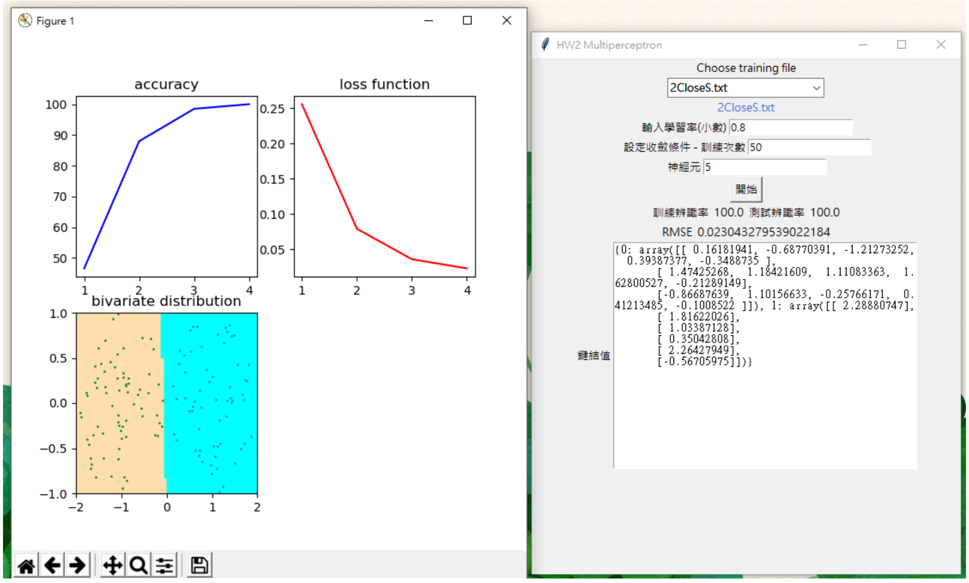
開始

訓練辨識率 76.878612716763 測試辨識率 73.5632183908046

RMSE 0.5096280568431429

鍵結值

```
{0: array([[-0.55846006, -1.79828854,  1.19111641,
  0.53956542,  1.83507661],
       [ 2.91298986, -0.25340946,  1.22451204,  1.
47070251,  1.4294952 ],
       [-3.31970741,  5.48104143,  0.49913395, -2.
56241224,  0.63305554]]), 1: array([[-2.10552145],
       [ 3.78371165],
       [-3.13668946],
       [ 0.72972814],
       [ 2.25911819],
       [ 0.9295217 ]])}
```



Choose training file

2CloseS.txt

2CloseS.txt

輸入學習率(小數) 0.8

設定收斂條件 - 訓練次數 50

神經元 5

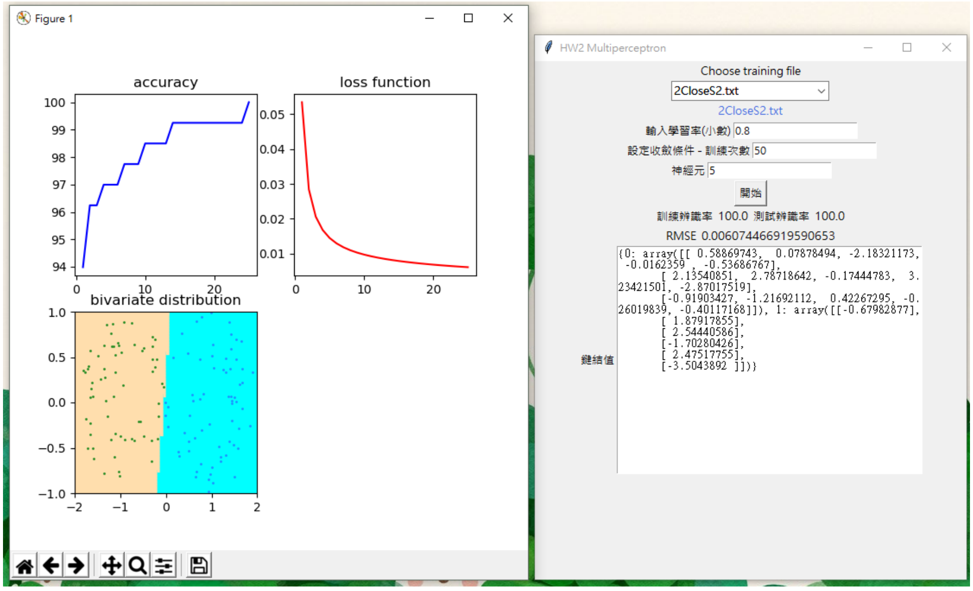
開始

訓練辨識率 100.0 測試辨識率 100.0

RMSE 0.023043279539022184

鍵結值

```
{0: array([[ 0.16181941, -0.68770391, -1.21273252,
  0.39387377, -0.3488735 ],
       [ 1.47425268,  1.18421609,  1.11083363,  1.
62800527, -0.21289149],
       [-0.86687639,  1.10156633, -0.25766171,  0.
41213485, -0.1008522 ]]), 1: array([[ 2.28880747],
       [ 1.81622026],
       [ 1.03387128],
       [ 0.35042808],
       [ 2.26427949],
       [-0.56705975]])}
```



Choose training file

2CloseS2.txt

2CloseS2.txt

輸入學習率(小數) 0.8

設定收斂條件 - 訓練次數 50

神經元 5

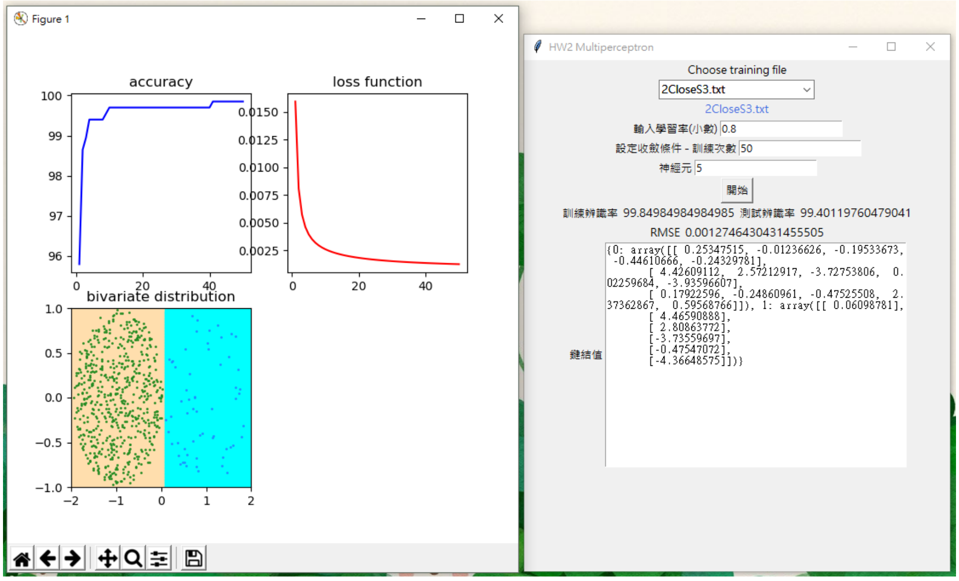
開始

訓練辨識率 100.0 測試辨識率 100.0

RMSE 0.006074466919590653

鍵結值

```
{0: array([[ 0.58869743,  0.07878494, -2.18321173,
 -0.0162359 , -0.53686767],
       [ 2.13540851,  2.78718642, -0.17444783,  3.
23421501, -2.87017519],
       [-0.91903427, -1.21692112,  0.42267295, -0.
26019839, -0.40117168]]), 1: array([[-0.67982877],
       [ 1.87917855],
       [ 2.54440586],
       [-1.70280426],
       [ 2.47517755],
       [-3.5043892 ]])}
```



2CloseS3.txt

2CloseS3.txt

輸入學習率(小數) 0.8

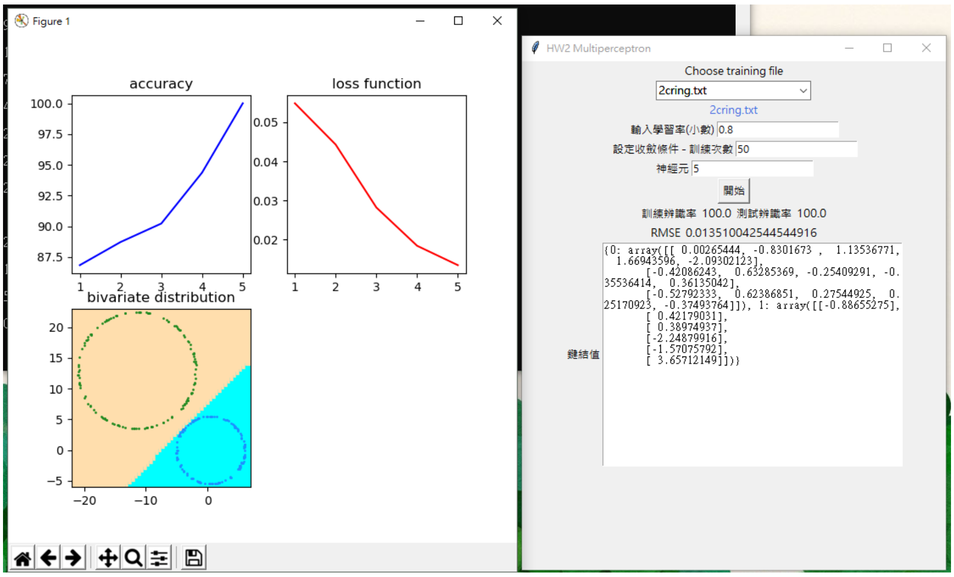
設定收斂條件 - 訓練次數 50

神經元 5

訓練辨識率 99.84984984984985 測試辨識率 99.40119760479041

RMSE 0.0012746430431455505

```
{0: array([[ 0.25347515, -0.01236626, -0.19533673,
 -0.44610666, -0.24329781],
       [ 4.42609112,  2.57212917, -3.72753806,  0.
02259684, -3.93596607],
       [ 0.17922596, -0.24860961, -0.47525508,  2.
37362867,  0.59568766]]), 1: array([[ 0.06098781],
       [ 4.46590888],
       [ 2.80863772],
       [-3.73559697],
       [-0.47547072],
       [-4.36648575]])}
```



2cring.txt

2cring.txt

輸入學習率(小數) 0.8

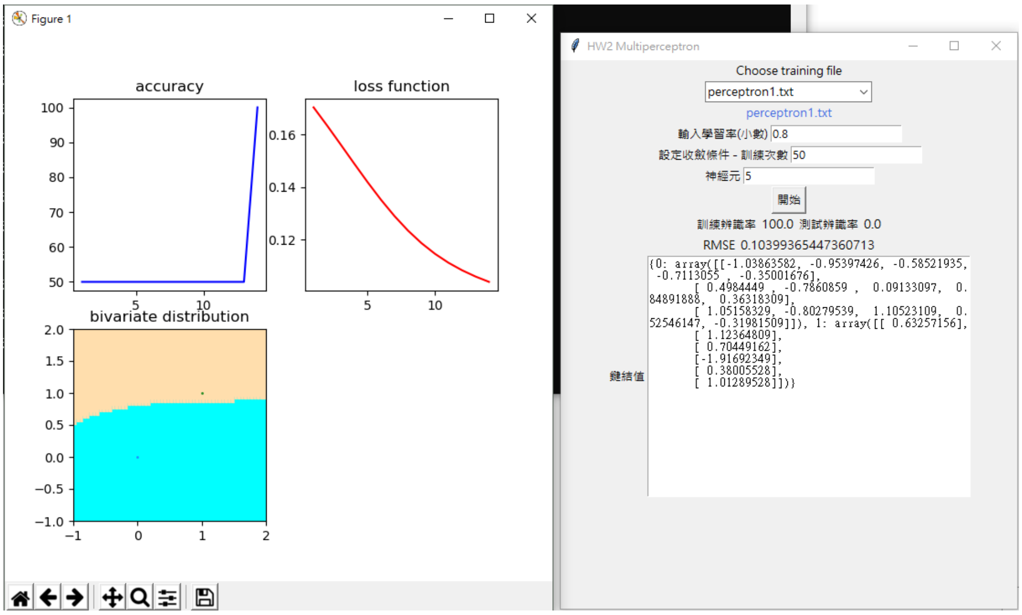
設定收斂條件 - 訓練次數 50

神經元 5

訓練辨識率 100.0 測試辨識率 100.0

RMSE 0.013510042544544916

```
{0: array([[ 0.00265444, -0.8301673 ,  1.13536771,
  1.66943596, -2.09302123],
       [-0.42086243,  0.63285369, -0.25409291, -0.
35536414,  0.36135042],
       [-0.52792333,  0.62386851,  0.27544925,  0.
25170923, -0.37493764]]), 1: array([[-0.88655275],
       [ 0.42179031],
       [ 0.38974937],
       [-2.24879916],
       [-1.57075792],
       [ 3.65712149]])}
```



perceptron1.txt

perceptron1.txt

輸入學習率(小數) 0.8

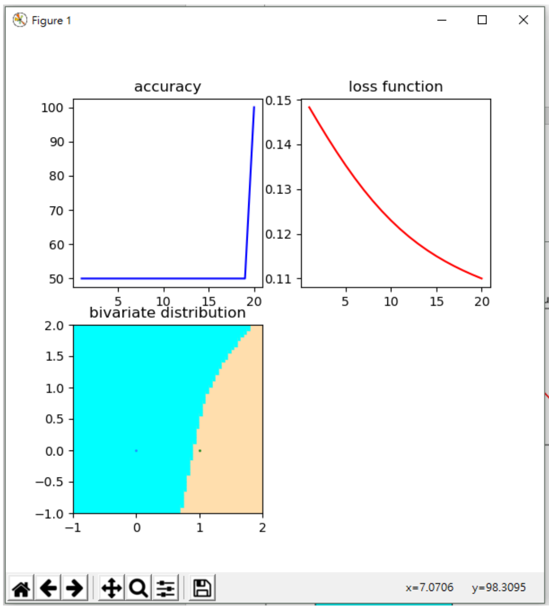
設定收斂條件 - 訓練次數 50

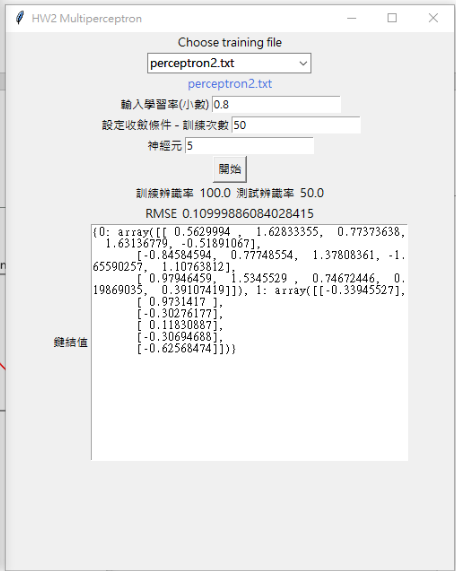
神經元 5

訓練辨識率 100.0 測試辨識率 0.0

RMSE 0.10399365447360713

```
{0: array([[-1.03863582, -0.95397426, -0.58521935,
 -0.7113055 , -0.35001676],
       [ 0.4984449 , -0.7860859 ,  0.09133097,  0.
84891888,  0.36318309],
       [ 1.05158329, -0.80279539,  1.10523109,  0.
52546147, -0.31981509]]), 1: array([[ 0.63257156],
       [ 1.12364809],
       [ 0.70449162],
       [-1.91692349],
       [ 0.38005528],
       [ 1.01289528]])}
```





Choose training file

perceptron2.txt

perceptron2.txt

輸入學習率(小數) 0.8

設定收斂條件 - 訓練次數 50

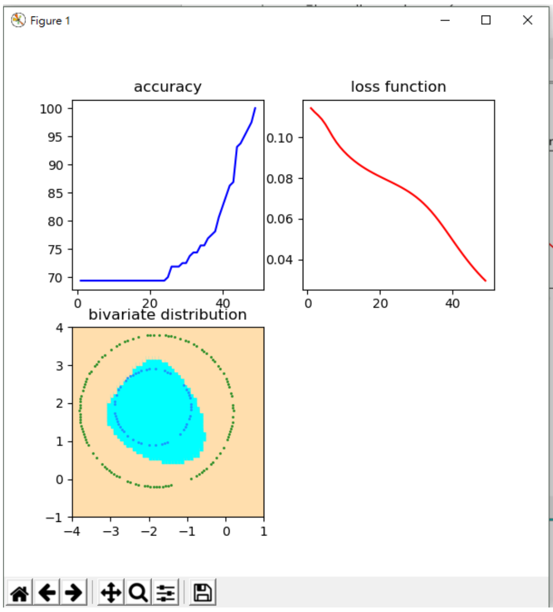
神經元 5

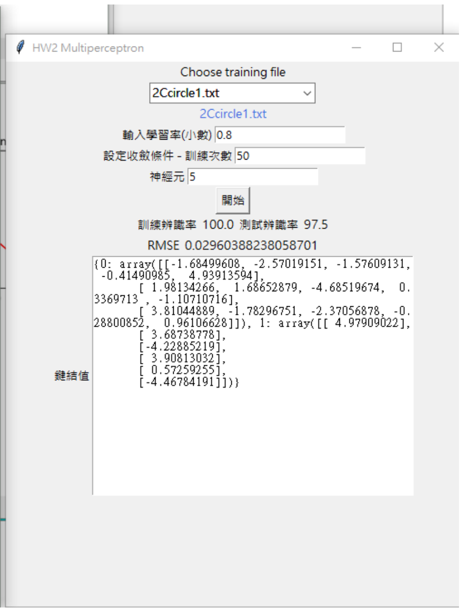
訓練辨識率 100.0 測試辨識率 50.0

RMSE 0.10999886084028415

鍵結值

```
{0: array([[ 0.5629994 ,  1.62833355,  0.77373638,
  1.63136779, -0.51891067],
       [-0.84584594,  0.77748554,  1.37808361, -1.
65590257,  1.10763812],
       [ 0.97946459,  1.5345529 ,  0.74672446,  0.
19869035,  0.39107419]]), 1: array([[-0.33945527],
       [ 0.9731417 ],
       [-0.30276177],
       [ 0.11830887],
       [-0.30694688],
       [-0.62568474]])}
```





Choose training file

2Ccircle1.txt

2Ccircle1.txt

輸入學習率(小數) 0.8

設定收斂條件 - 訓練次數 50

神經元 5

訓練辨識率 100.0 測試辨識率 97.5

RMSE 0.02960388238058701

鍵結值

```
{0: array([[-1.68499608, -2.57019151, -1.57609131,
 -0.41490985,  4.93913594],
       [ 1.98134266,  1.68652879, -4.68519674,  0.
3369713 , -1.10710716],
       [ 3.81044889, -1.78296751, -2.37056878, -0.
28800852,  0.96106628]]), 1: array([[ 4.97909022],
       [ 3.68738778],
       [-4.22885219],
       [ 3.90813032],
       [ 0.57259255],
       [-4.46784191]])}
```

D.加分項目

可設定輸入神經元數目