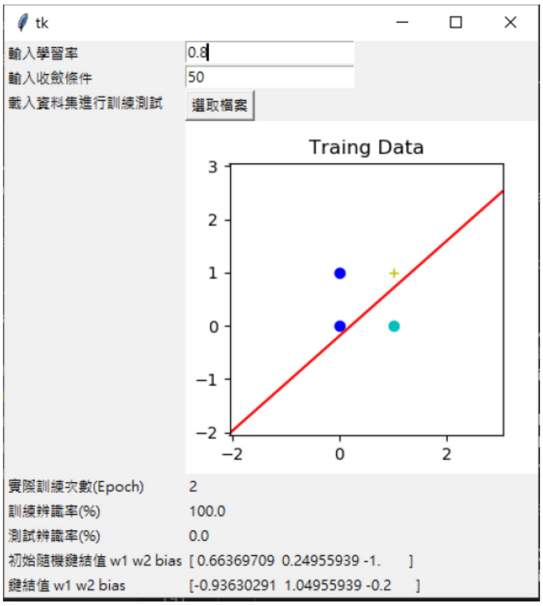
108522050 賴映如 類神經網路作業 1 - 設計感知機類神經網路

A. 程式執行說明:

輸入學習率及收斂條件，並選取 Dataset 檔案，即會開始訓練資料，並顯示出訓練圖和資訊於下。

註: 深藍點:訓練資料第一類 青藍點:訓練資料第二類 黃十字點:測試資料



輸入學習率 0.8
輸入收斂條件 50
載入資料集進行訓練測試
實際訓練次數(Epoch) 2
訓練辨識率(%) 100.0
測試辨識率(%) 0.0
初始隨機鍵結值 w1 w2 bias [0.66369709 0.24955939 -1.]
鍵結值 w1 w2 bias [-0.93630291 1.04955939 -0.2]

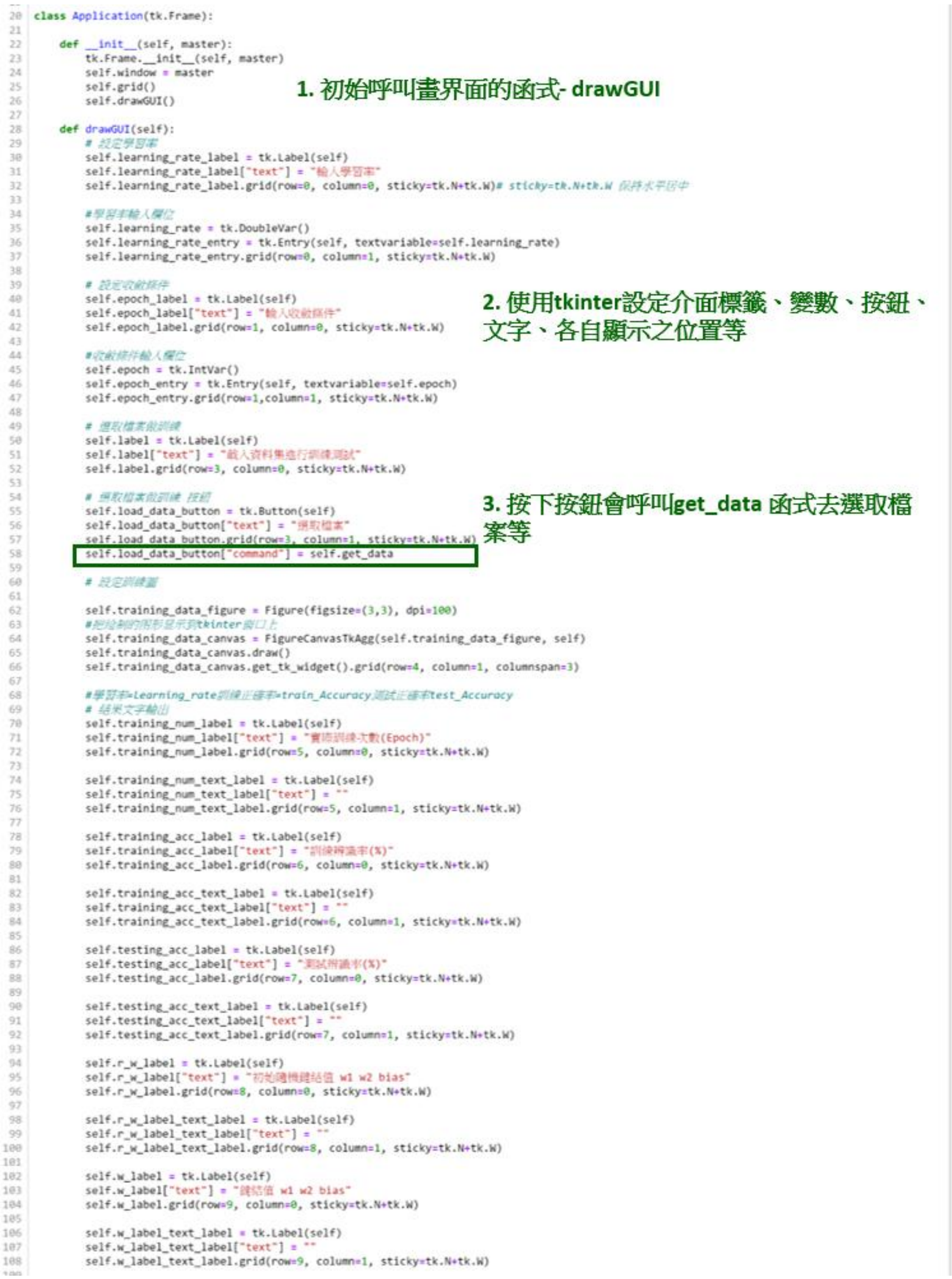
B. 程式簡介:

1. 分 2 個 class 於 2 個 py 檔:

   i. Application 類別: 負責介面顯示、載入檔案、根據 Perceptron 結果顯示圖形及資訊。

   ii. Perceptron 類別: 負責將 Application 傳入的檔案轉為感知機資料模式，如分割陣列等，並做感知機訓練以及最後算出辨識率。

2. 以下分步驟 1~21 依序對程式詳細做說明:

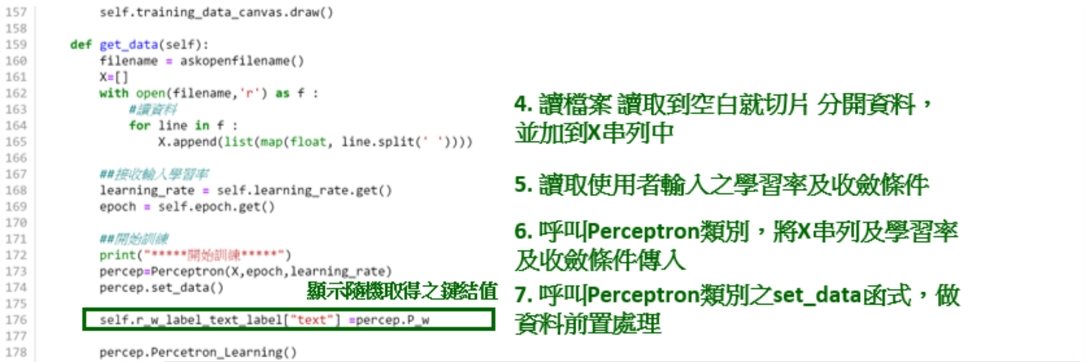


```python
class Application(tk.Frame):

    def __init__(self, master):
        tk.Frame.__init__(self, master)
        self.window = master
        self.grid()
        self.drawGUI()

    def drawGUI(self):
        # 設定學習率
        self.learning_rate_label = tk.Label(self)
        self.learning_rate_label["text"] = "輸入學習率"
        self.learning_rate_label.grid(row=0, column=0, sticky=tk.N+tk.W)# sticky=tk.N+tk.W 保持水平居中

        #學習率輸入欄位
        self.learning_rate = tk.DoubleVar()
        self.learning_rate_entry = tk.Entry(self, textvariable=self.learning_rate)
        self.learning_rate_entry.grid(row=0, column=1, sticky=tk.N+tk.W)

        # 設定收斂條件
        self.epoch_label = tk.Label(self)
        self.epoch_label["text"] = "輸入收斂條件"
        self.epoch_label.grid(row=1, column=0, sticky=tk.N+tk.W)

        #收斂條件輸入欄位
        self.epoch = tk.IntVar()
        self.epoch_entry = tk.Entry(self, textvariable=self.epoch)
        self.epoch_entry.grid(row=1,column=1, sticky=tk.N+tk.W)

        # 選取檔案做訓練
        self.label = tk.Label(self)
        self.label["text"] = "載入資料集進行訓練測試"
        self.label.grid(row=3, column=0, sticky=tk.N+tk.W)

        # 選取檔案做訓練 按鈕
        self.load_data_button = tk.Button(self)
        self.load_data_button["text"] = "選取檔案"
        self.load_data_button.grid(row=3, column=1, sticky=tk.N+tk.W)
        self.load_data_button["command"] = self.get_data

        # 設定訓練圖

        self.training_data_figure = Figure(figsize=(3,3), dpi=100)
        #把绘制的图形显示到tkinter窗口上
        self.training_data_canvas = FigureCanvasTkAgg(self.training_data_figure, self)
        self.training_data_canvas.draw()
        self.training_data_canvas.get_tk_widget().grid(row=4, column=1, columnspan=3)

        #學習率=Learning_rate訓練正確率=train_Accuracy測試正確率test_Accuracy
        # 結果文字輸出
        self.training_num_label = tk.Label(self)
        self.training_num_label["text"] = "實際訓練次數(Epoch)"
        self.training_num_label.grid(row=5, column=0, sticky=tk.N+tk.W)

        self.training_num_text_label = tk.Label(self)
        self.training_num_text_label["text"] = ""
        self.training_num_text_label.grid(row=5, column=1, sticky=tk.N+tk.W)

        self.training_acc_label = tk.Label(self)
        self.training_acc_label["text"] = "訓練辨識率(%)"
        self.training_acc_label.grid(row=6, column=0, sticky=tk.N+tk.W)

        self.training_acc_text_label = tk.Label(self)
        self.training_acc_text_label["text"] = ""
        self.training_acc_text_label.grid(row=6, column=1, sticky=tk.N+tk.W)

        self.testing_acc_label = tk.Label(self)
        self.testing_acc_label["text"] = "測試辨識率(%)"
        self.testing_acc_label.grid(row=7, column=0, sticky=tk.N+tk.W)

        self.testing_acc_text_label = tk.Label(self)
        self.testing_acc_text_label["text"] = ""
        self.testing_acc_text_label.grid(row=7, column=1, sticky=tk.N+tk.W)

        self.r_w_label = tk.Label(self)
        self.r_w_label["text"] = "初始隨機鍵結值 w1 w2 bias"
        self.r_w_label.grid(row=8, column=0, sticky=tk.N+tk.W)

        self.r_w_label_text_label = tk.Label(self)
        self.r_w_label_text_label["text"] = ""
        self.r_w_label_text_label.grid(row=8, column=1, sticky=tk.N+tk.W)

        self.w_label = tk.Label(self)
        self.w_label["text"] = "鍵結值 w1 w2 bias"
        self.w_label.grid(row=9, column=0, sticky=tk.N+tk.W)

        self.w_label_text_label = tk.Label(self)
        self.w_label_text_label["text"] = ""
        self.w_label_text_label.grid(row=9, column=1, sticky=tk.N+tk.W)
```

1. 初始呼叫畫界面的函式- drawGUI

2. 使用tkinter設定介面標籤、變數、按鈕、文字、各自顯示之位置等

3. 按下按鈕會呼叫get_data 函式去選取檔案等



```
        self.training_data_canvas.draw()

    def get_data(self):
        filename = askopenfilename()
        X=[]
        with open(filename,'r') as f :
            #讀資料
            for line in f :
                X.append(list(map(float, line.split(' '))))

        ##接收輸入學習率
        learning_rate = self.learning_rate.get()
        epoch = self.epoch.get()

        ##開始訓練
        print("*****開始訓練*****")
        percep=Perceptron(X,epoch,learning_rate)
        percep.set_data()

        self.r_w_label_text_label["text"] =percep.P_w

        percep.Percetron_Learning()
        training_
        print("**
        print("tr

        self.trai
        self.trai
        self.w_la

        #self.wei

        testing_a
        print("te


        self.test

        self.Draw

window = tk.Tk()
app = Applicatior
window.mainloop()


# In[ ]:





# In[ ]:
```

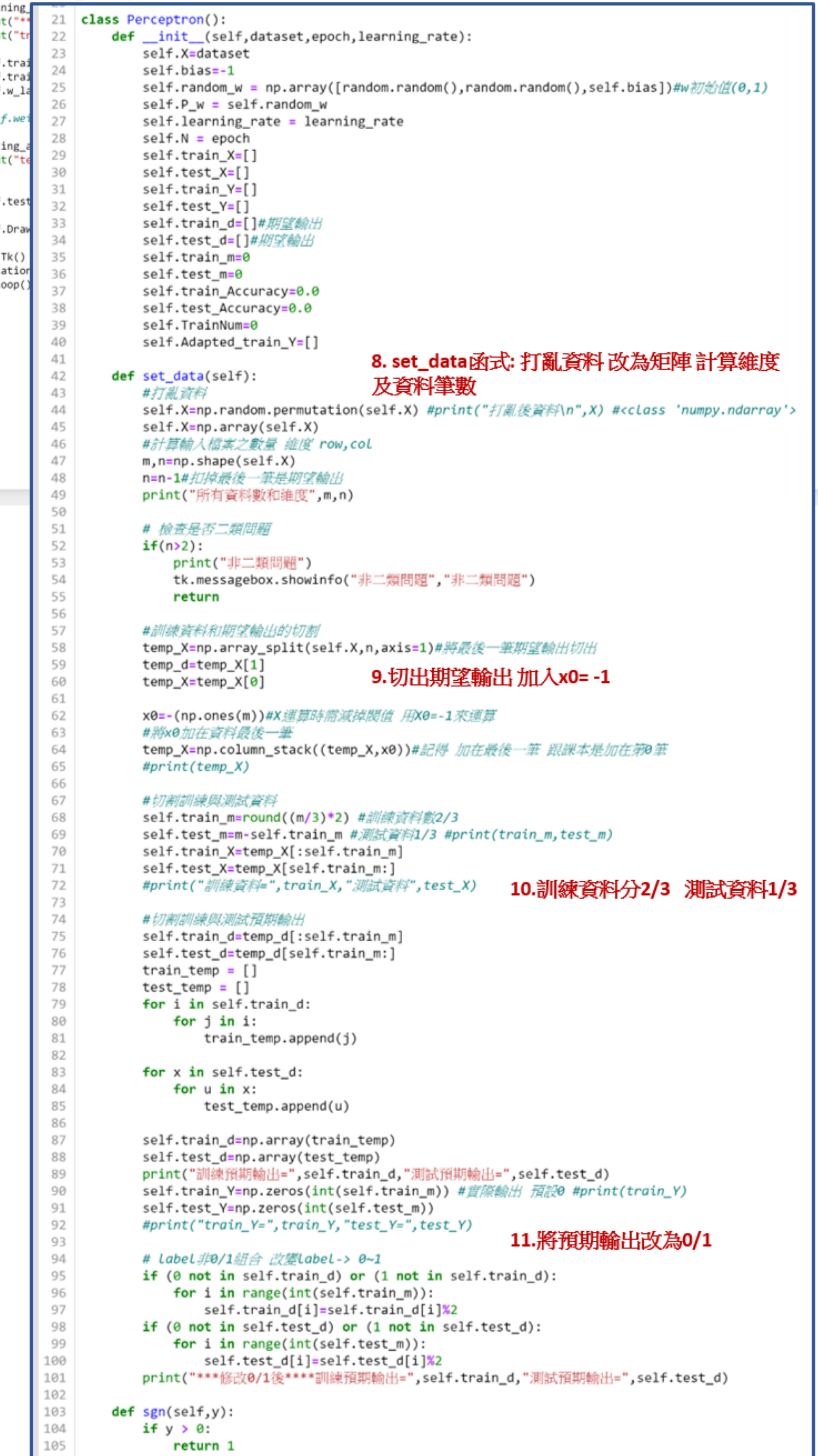
顯示隨機取得之鍵結值

4. 讀檔案 讀取到空白就切片 分開資料，並加到X串列中

5. 讀取使用者輸入之學習率及收斂條件

6. 呼叫Perceptron類別，將X串列及學習率及收斂條件傳入

7. 呼叫Perceptron類別之set_data函式，做資料前置處理



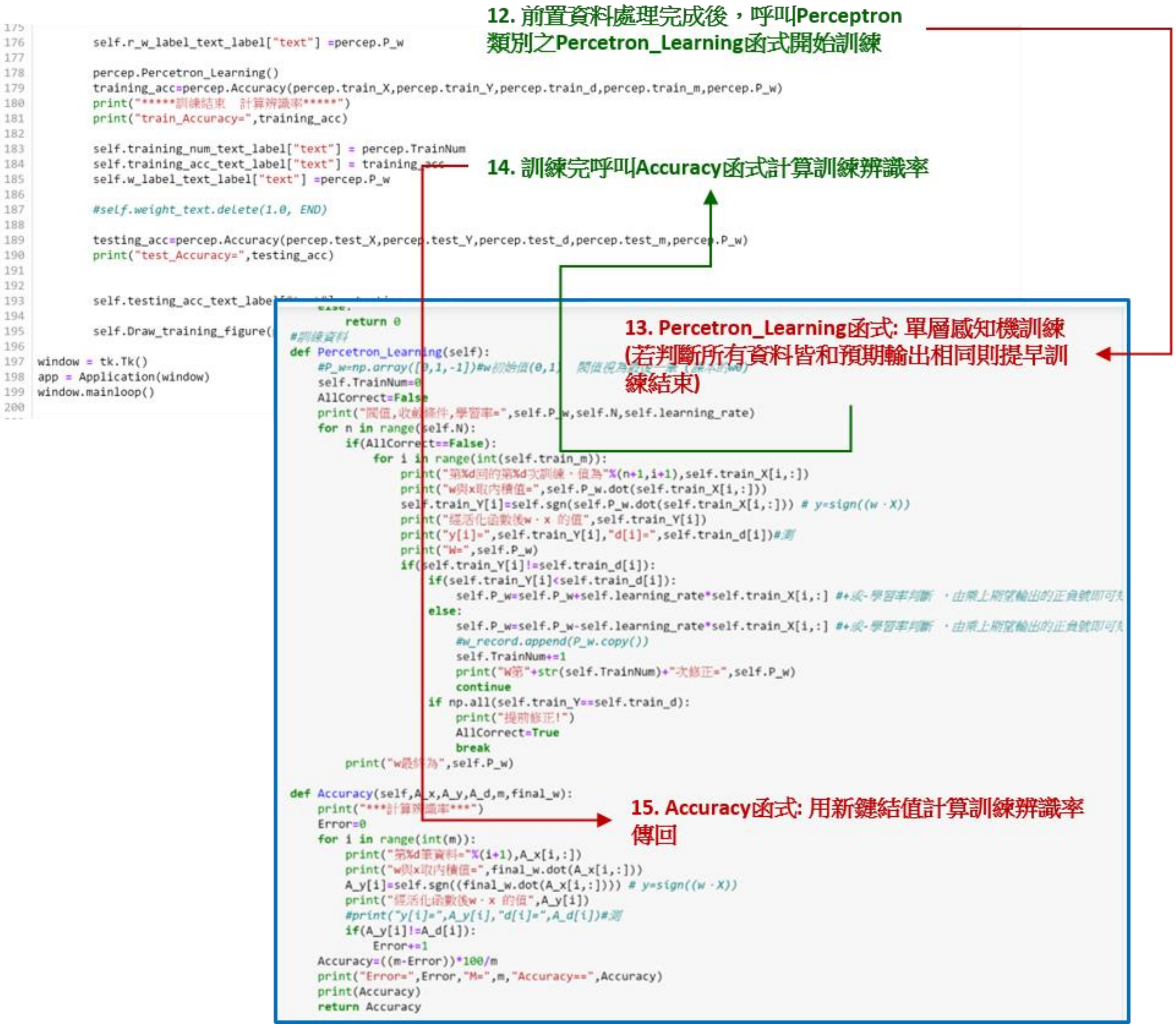
```
class Perceptron():
    def __init__(self,dataset,epoch,learning_rate):
        self.X=dataset
        self.bias=-1
        self.random_w = np.array([random.random(),random.random(),self.bias])#w初始值(0,1)
        self.P_w = self.random_w
        self.learning_rate = learning_rate
        self.N = epoch
        self.train_X=[]
        self.test_X=[]
        self.train_Y=[]
        self.test_Y=[]
        self.train_d=[]#期望輸出
        self.test_d=[]#期望輸出
        self.train_m=0
        self.test_m=0
        self.train_Accuracy=0.0
        self.test_Accuracy=0.0
        self.TrainNum=0
        self.Adapted_train_Y=[]

    def set_data(self):
        #打亂資料
        self.X=np.random.permutation(self.X) #print("打亂後資料\n",X) #<class 'numpy.ndarray'>
        self.X=np.array(self.X)
        #計算輸入檔案之數量 維度 row,col
        m,n=np.shape(self.X)
        n=n-1#扣掉最後一筆是期望輸出
        print("所有資料數和維度",m,n)

        # 檢查是否二類問題
        if(n>2):
            print("非二類問題")
            tk.messagebox.showinfo("非二類問題","非二類問題")
            return

        #訓練資料和期望輸出的切割
        temp_X=np.array_split(self.X,n,axis=1)#將最後一筆期望輸出切出
        temp_d=temp_X[1]
        temp_X=temp_X[0]

        x0=-(np.ones(m))#X運算時需減掉閥值 用X0=-1來運算
        #將x0加在資料最後一筆
        temp_X=np.column_stack((temp_X,x0))#記得 加在最後一筆 跟課本是加在第0筆
        #print(temp_X)

        #切割訓練與測試資料
        self.train_m=round((m/3)*2) #訓練資料數2/3
        self.test_m=m-self.train_m #測試資料1/3 #print(train_m,test_m)
        self.train_X=temp_X[:self.train_m]
        self.test_X=temp_X[self.train_m:]
        #print("訓練資料=",train_X,"測試資料",test_X)

        #切割訓練與測試預期輸出
        self.train_d=temp_d[:self.train_m]
        self.test_d=temp_d[self.train_m:]
        train_temp = []
        test_temp = []
        for i in self.train_d:
            for j in i:
                train_temp.append(j)

        for x in self.test_d:
            for u in x:
                test_temp.append(u)

        self.train_d=np.array(train_temp)
        self.test_d=np.array(test_temp)
        print("訓練預期輸出=",self.train_d,"測試預期輸出=",self.test_d)
        self.train_Y=np.zeros(int(self.train_m)) #實際輸出 預設0 #print(train_Y)
        self.test_Y=np.zeros(int(self.test_m))
        #print("train_Y=",train_Y,"test_Y=",test_Y)

        # label非0/1組合 改變label-> 0~1
        if (0 not in self.train_d) or (1 not in self.train_d):
            for i in range(int(self.train_m)):
                self.train_d[i]=self.train_d[i]%2
        if (0 not in self.test_d) or (1 not in self.test_d):
            for i in range(int(self.test_m)):
                self.test_d[i]=self.test_d[i]%2
        print("***修改0/1後****訓練預期輸出=",self.train_d,"測試預期輸出=",self.test_d)

    def sgn(self,y):
        if y > 0:
            return 1
```

8. set_data函式: 打亂資料 改為矩陣 計算維度及資料筆數

9.切出期望輸出 加入x0= -1

10.訓練資料分2/3　測試資料1/3

11.將預期輸出改為0/1



12. 前置資料處理完成後，呼叫Perceptron類別之Percetron_Learning函式開始訓練

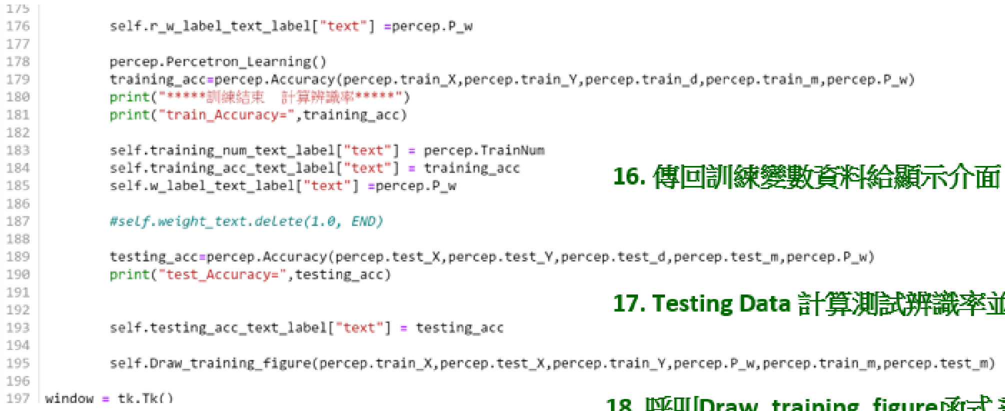
14. 訓練完呼叫Accuracy函式計算訓練辨識率

13. Percetron_Learning函式: 單層感知機訓練(若判斷所有資料皆和預期輸出相同則提早訓練結束)

15. Accuracy函式: 用新鍵結值計算訓練辨識率傳回

```
        self.r_w_label_text_label["text"] =percep.P_w

        percep.Percetron_Learning()
        training_acc=percep.Accuracy(percep.train_X,percep.train_Y,percep.train_d,percep.train_m,percep.P_w)
        print("*****訓練結束  計算辨識率*****")
        print("train_Accuracy=",training_acc)

        self.training_num_text_label["text"] = percep.TrainNum
        self.training_acc_text_label["text"] = training_acc
        self.w_label_text_label["text"] =percep.P_w

        #self.weight_text.delete(1.0, END)

        testing_acc=percep.Accuracy(percep.test_X,percep.test_Y,percep.test_d,percep.test_m,percep.P_w)
        print("test_Accuracy=",testing_acc)


        self.testing_acc_text_label[

        self.Draw_training_figure(

window = tk.Tk()
app = Application(window)
window.mainloop()
```

```
        else:
            return 0
    #訓練資料
    def Percetron_Learning(self):
        #P_w=np.array([0,1,-1])#w初始值(0,1)   閥值視為最後一筆 (課本的w0)
        self.TrainNum=0
        AllCorrect=False
        print("閥值,收斂條件,學習率=",self.P_w,self.N,self.learning_rate)
        for n in range(self.N):
            if(AllCorrect==False):
                for i in range(int(self.train_m)):
                    print("第%d回的第%d次訓練，值為"%(n+1,i+1),self.train_X[i,:])
                    print("w與x取內積值=",self.P_w.dot(self.train_X[i,:]))
                    self.train_Y[i]=self.sgn(self.P_w.dot(self.train_X[i,:])) # y=sign((w‧X))
                    print("經活化函數後w‧x 的值",self.train_Y[i])
                    print("y[i]=",self.train_Y[i],"d[i]=",self.train_d[i])#測
                    print("W=",self.P_w)
                    if(self.train_Y[i]!=self.train_d[i]):
                        if(self.train_Y[i]<self.train_d[i]):
                            self.P_w=self.P_w+self.learning_rate*self.train_X[i,:] #+或-學習率判斷  ，由乘上期望輸出的正負號即可
                        else:
                            self.P_w=self.P_w-self.learning_rate*self.train_X[i,:] #+或-學習率判斷  ，由乘上期望輸出的正負號即可
                            #w_record.append(P_w.copy())
                            self.TrainNum+=1
                            print("W第"+str(self.TrainNum)+"次修正=",self.P_w)
                            continue
                    if np.all(self.train_Y==self.train_d):
                        print("提前修正!")
                        AllCorrect=True
                        break
            print("w最終為",self.P_w)

    def Accuracy(self,A_x,A_y,A_d,m,final_w):
        print("***計算辨識率***")
        Error=0
        for i in range(int(m)):
            print("第%d筆資料="%(i+1),A_x[i,:])
            print("w與x取內積值=",final_w.dot(A_x[i,:]))
            A_y[i]=self.sgn((final_w.dot(A_x[i,:]))) # y=sign((w‧X))
            print("經活化函數後w‧x 的值",A_y[i])
            #print("y[i]=",A_y[i],"d[i]=",A_d[i])#測
            if(A_y[i]!=A_d[i]):
                Error+=1
        Accuracy=((m-Error))*100/m
        print("Error=",Error,"M=",m,"Accuracy==",Accuracy)
        print(Accuracy)
        return Accuracy
```

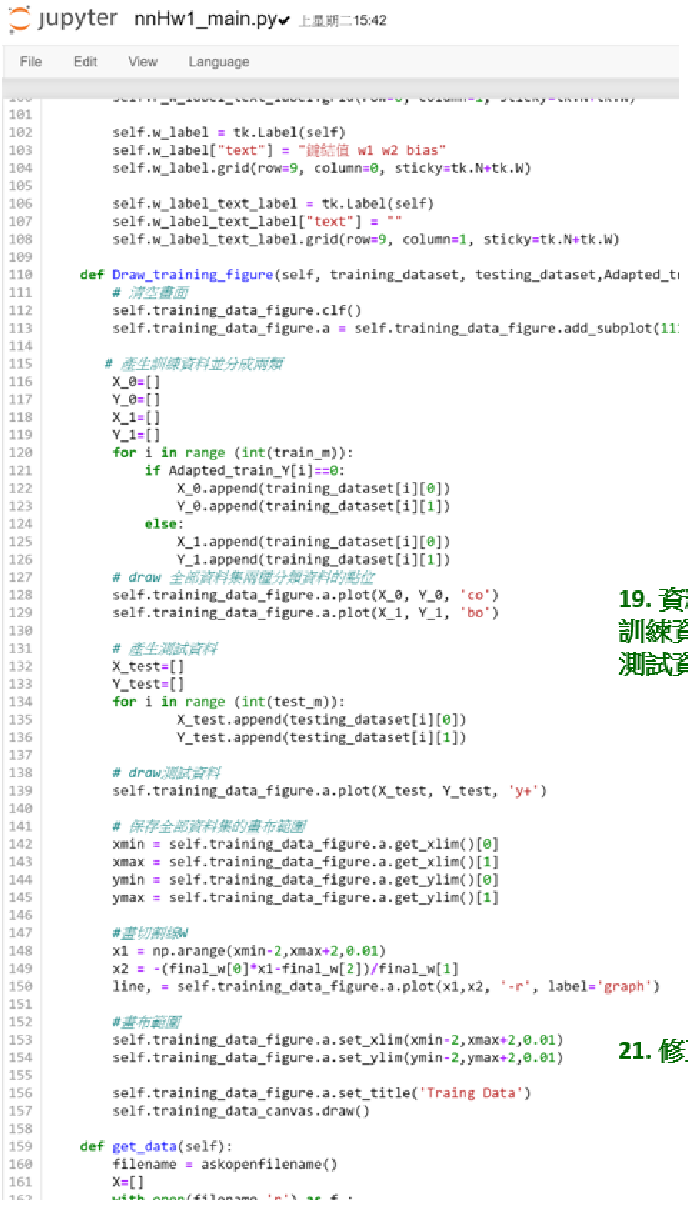


```
        self.r_w_label_text_label["text"] =percep.P_w

        percep.Percetron_Learning()
        training_acc=percep.Accuracy(percep.train_X,percep.train_Y,percep.train_d,percep.train_m,percep.P_w)
        print("*****訓練結束   計算辨識率*****")
        print("train_Accuracy=",training_acc)

        self.training_num_text_label["text"] = percep.TrainNum
        self.training_acc_text_label["text"] = training_acc
        self.w_label_text_label["text"] =percep.P_w

        #self.weight_text.delete(1.0, END)

        testing_acc=percep.Accuracy(percep.test_X,percep.test_Y,percep.test_d,percep.test_m,percep.P_w)
        print("test_Accuracy=",testing_acc)


        self.testing_acc_text_label["text"] = testing_acc

        self.Draw_training_figure(percep.train_X,percep.test_X,percep.train_Y,percep.P_w,percep.train_m,percep.test_m)

window = tk.Tk()
```

16. 傳回訓練變數資料給顯示介面

17. Testing Data 計算測試辨識率並傳回顯示介面

18. 呼叫Draw_training_figure函式 畫出訓練圖



```

        self.w_label = tk.Label(self)
        self.w_label["text"] = "鏈結值 w1 w2 bias"
        self.w_label.grid(row=9, column=0, sticky=tk.N+tk.W)

        self.w_label_text_label = tk.Label(self)
        self.w_label_text_label["text"] = ""
        self.w_label_text_label.grid(row=9, column=1, sticky=tk.N+tk.W)

    def Draw_training_figure(self, training_dataset, testing_dataset,Adapted_tı
        # 清空畫面
        self.training_data_figure.clf()
        self.training_data_figure.a = self.training_data_figure.add_subplot(11

       # 產生訓練資料並分成兩類
        X_0=[]
        Y_0=[]
        X_1=[]
        Y_1=[]
        for i in range (int(train_m)):
            if Adapted_train_Y[i]==0:
                X_0.append(training_dataset[i][0])
                Y_0.append(training_dataset[i][1])
            else:
                X_1.append(training_dataset[i][0])
                Y_1.append(training_dataset[i][1])
        # draw 全部資料集兩種分類資料的點位
        self.training_data_figure.a.plot(X_0, Y_0, 'co')
        self.training_data_figure.a.plot(X_1, Y_1, 'bo')

        # 產生測試資料
        X_test=[]
        Y_test=[]
        for i in range (int(test_m)):
                X_test.append(testing_dataset[i][0])
                Y_test.append(testing_dataset[i][1])

        # draw測試資料
        self.training_data_figure.a.plot(X_test, Y_test, 'y+')

        # 保存全部資料集的畫布範圍
        xmin = self.training_data_figure.a.get_xlim()[0]
        xmax = self.training_data_figure.a.get_xlim()[1]
        ymin = self.training_data_figure.a.get_ylim()[0]
        ymax = self.training_data_figure.a.get_ylim()[1]

        #畫切割線W
        x1 = np.arange(xmin-2,xmax+2,0.01)
        x2 = -(final_w[0]*x1-final_w[2])/final_w[1]
        line, = self.training_data_figure.a.plot(x1,x2, '-r', label='graph')

        #畫布範圍
        self.training_data_figure.a.set_xlim(xmin-2,xmax+2,0.01)
        self.training_data_figure.a.set_ylim(ymin-2,ymax+2,0.01)

        self.training_data_figure.a.set_title('Traing Data')
        self.training_data_canvas.draw()

    def get_data(self):
        filename = askopenfilename()
        X=[]
        with open(filename,'r') as f :
```

19. 資料分類顯示
訓練資料(不同顏色)
測試資料 (不同符號)

20. 依據訓練最後鏈結值計算方程式畫分線

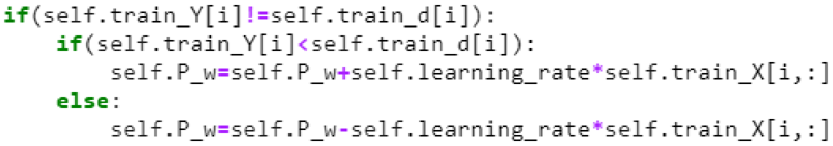
21. 修正顯示範圍

C. 實驗結果分析及討論<鍵結值、訓練次數、學習率、訓練正確率......(詳如圖)>

1. perceptron1

Log 視窗首先顯示資料維度、預期輸出、修改為 0/1 後的預期輸出、鍵結值(log 寫錯不是閥值)、收斂條件、學習率，便開始根據收斂條件 N 做 N 回訓練，每回的第 i 次訓練代表當次的第 i 筆資料。

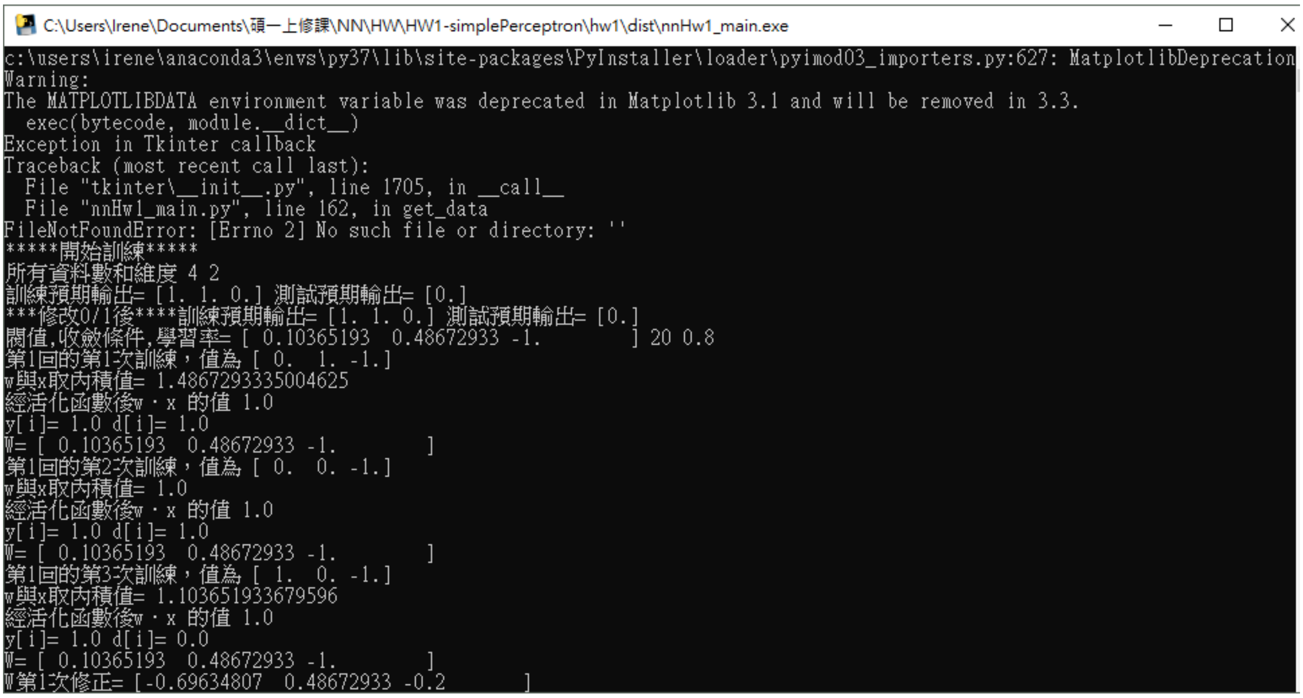
其中，訓練將 x 資料與鍵結值 w 取內積，再呼叫 sgn( )函式使其依據正負成為 1 或 0，再與已經過修正為 0/1 知預期輸出做比對，相同則維持鍵結值，不同則更正鍵結值，小於則加上學習率乘以該筆 x，反之則為加。



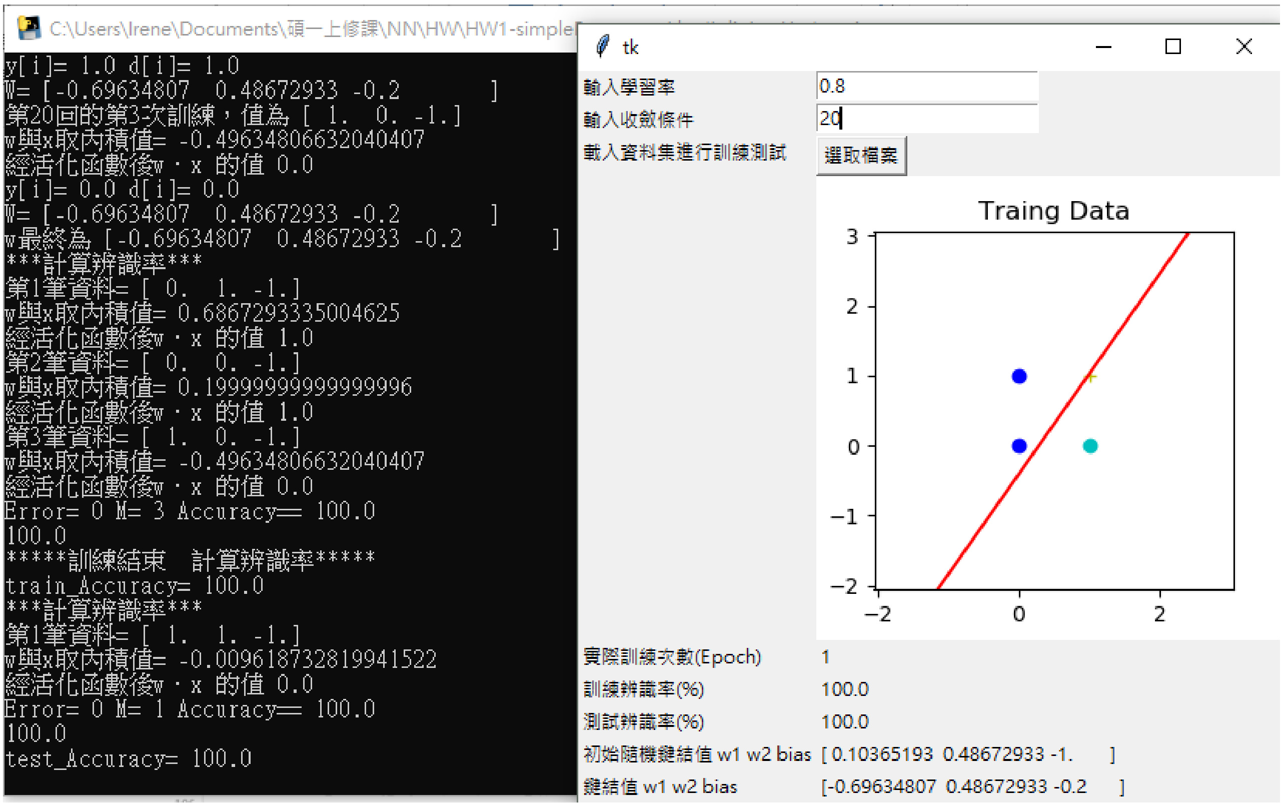
```
if(self.train_Y[i]!=self.train_d[i]):
    if(self.train_Y[i]<self.train_d[i]):
        self.P_w=self.P_w+self.learning_rate*self.train_X[i,:]
    else:
        self.P_w=self.P_w-self.learning_rate*self.train_X[i,:]
```

當**訓練資料全符合預期輸出或者已到達收斂條件，則訓練停止**。以最後的鍵結值去計算訓練辨識率與測試辨識率。

最後將資料及圖示顯示於視窗中。


C:\Users\Irene\Documents\碩一上修課\NN\HW\HW1-simplePerceptron\hw1\dist\nnHw1_main.exe

```
c:\users\irene\anaconda3\envs\py37\lib\site-packages\PyInstaller\loader\pyimod03_importers.py:627: MatplotlibDeprecation
Warning:
The MATPLOTLIBDATA environment variable was deprecated in Matplotlib 3.1 and will be removed in 3.3.
  exec(bytecode, module.__dict__)
Exception in Tkinter callback
Traceback (most recent call last):
  File "tkinter\__init__.py", line 1705, in __call__
  File "nnHw1_main.py", line 162, in get_data
FileNotFoundError: [Errno 2] No such file or directory: ''
*****開始訓練*****
所有資料數和維度 4 2
訓練預期輸出= [1. 1. 0.] 測試預期輸出= [0.]
***修改0/1後****訓練預期輸出= [1. 1. 0.] 測試預期輸出= [0.]
閥值,收斂條件,學習率= [ 0.10365193  0.48672933 -1.        ] 20 0.8
第1回的第1次訓練，值為 [ 0.  1. -1.]
w與x取內積值= 1.4867293335004625
經活化函數後w．x 的值 1.0
y[i]= 1.0 d[i]= 1.0
W= [ 0.10365193  0.48672933 -1.        ]
第1回的第2次訓練，值為 [ 0.  0. -1.]
w與x取內積值= 1.0
經活化函數後w．x 的值 1.0
y[i]= 1.0 d[i]= 1.0
W= [ 0.10365193  0.48672933 -1.        ]
第1回的第3次訓練，值為 [ 1.  0. -1.]
w與x取內積值= 1.103651933679596
經活化函數後w．x 的值 1.0
y[i]= 1.0 d[i]= 0.0
W= [ 0.10365193  0.48672933 -1.        ]
W第1次修正= [-0.69634807  0.48672933 -0.2       ]
```



```
y[i]= 1.0 d[i]= 1.0
W= [-0.69634807  0.48672933 -0.2       ]
第20回的第3次訓練，值為 [ 1.  0. -1.]
w與x取內積值= -0.49634806632040407
經活化函數後w．x 的值 0.0
y[i]= 0.0 d[i]= 0.0
W= [-0.69634807  0.48672933 -0.2       ]
w最終為 [-0.69634807  0.48672933 -0.2       ]
***計算辨識率***
第1筆資料= [ 0.  1. -1.]
w與x取內積值= 0.6867293335004625
經活化函數後w．x 的值 1.0
第2筆資料= [ 0.  0. -1.]
w與x取內積值= 0.19999999999999996
經活化函數後w．x 的值 1.0
第3筆資料= [ 1.  0. -1.]
w與x取內積值= -0.49634806632040407
經活化函數後w．x 的值 0.0
Error= 0 M= 3 Accuracy== 100.0
100.0
*****訓練結束  計算辨識率*****
train_Accuracy= 100.0
***計算辨識率***
第1筆資料= [ 1.  1. -1.]
w與x取內積值= -0.009618732819941522
經活化函數後w．x 的值 0.0
Error= 0 M= 1 Accuracy== 100.0
100.0
test_Accuracy= 100.0
```

輸入學習率 0.8

輸入收斂條件 20

載入資料集進行訓練測試

實際訓練次數(Epoch) 1

訓練辨識率(%) 100.0

測試辨識率(%) 100.0

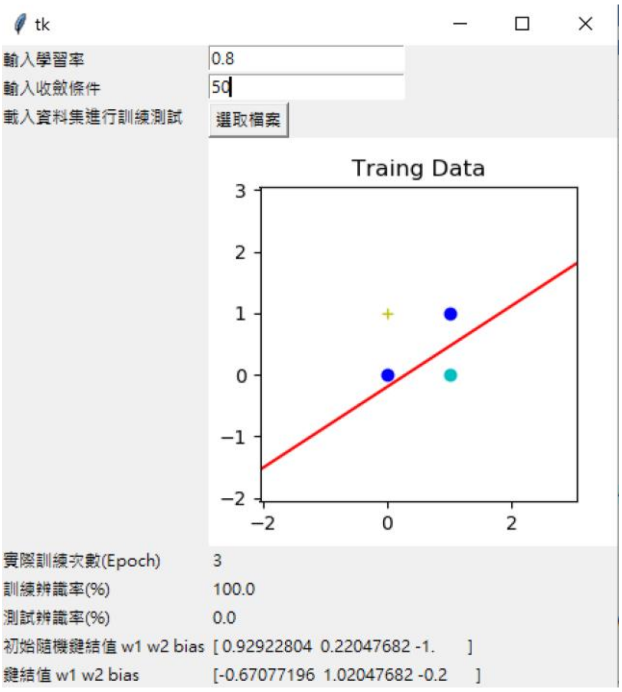
初始隨機鍵結值 w1 w2 bias [0.10365193 0.48672933 -1.   ]

鍵結值 w1 w2 bias [-0.69634807 0.48672933 -0.2   ]

※其後資料將省略 Log 視窗，僅呈現結果圖式式窗。

2. perceptron2

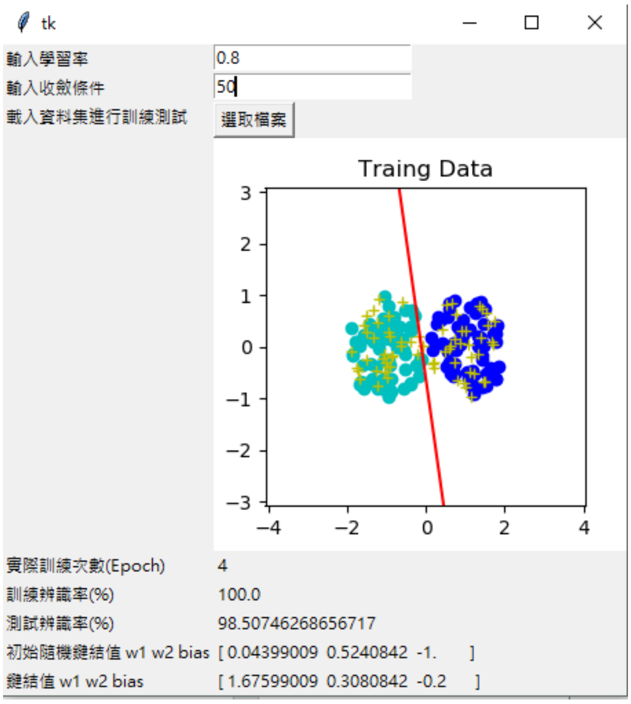
測試辨識率因資料太少(4 筆的 1/3，僅 1 筆)，而剛好預估錯誤為 0



輸入學習率	0.8
輸入收斂條件	50
載入資料集進行訓練測試	
實際訓練次數(Epoch)	3
訓練辨識率(%)	100.0
測試辨識率(%)	0.0
初始隨機鍵結值 w1 w2 bias	[0.92922804 0.22047682 -1.]
鍵結值 w1 w2 bias	[-0.67077196 1.02047682 -0.2]

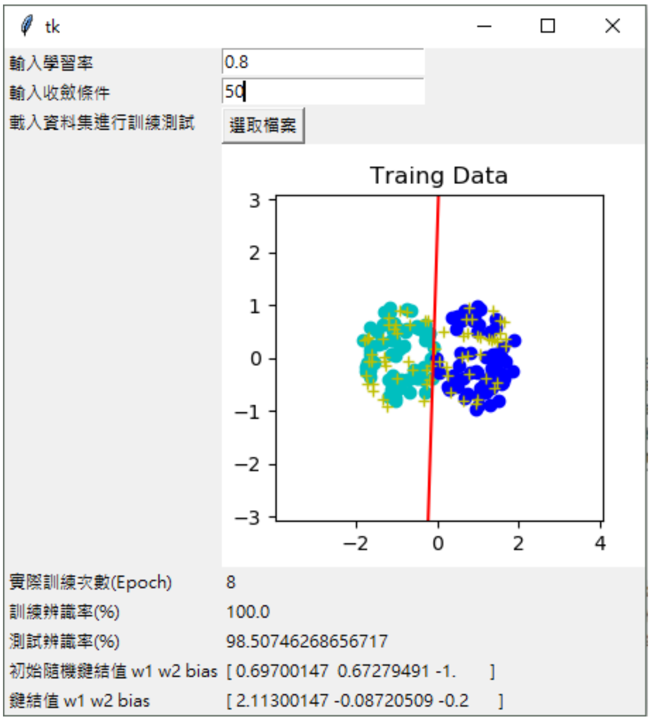
3. 2CloseS

資料單純分群，訓練效果及測試辨識率結不錯，近乎 100%



輸入學習率	0.8
輸入收斂條件	50
載入資料集進行訓練測試	
實際訓練次數(Epoch)	4
訓練辨識率(%)	100.0
測試辨識率(%)	98.50746268656717
初始隨機鍵結值 w1 w2 bias	[0.04399009 0.5240842 -1.]
鍵結值 w1 w2 bias	[1.67599009 0.3080842 -0.2]

4. 2CloseS2



輸入學習率 0.8

輸入收斂條件 50

載入資料集進行訓練測試

實際訓練次數(Epoch) 8

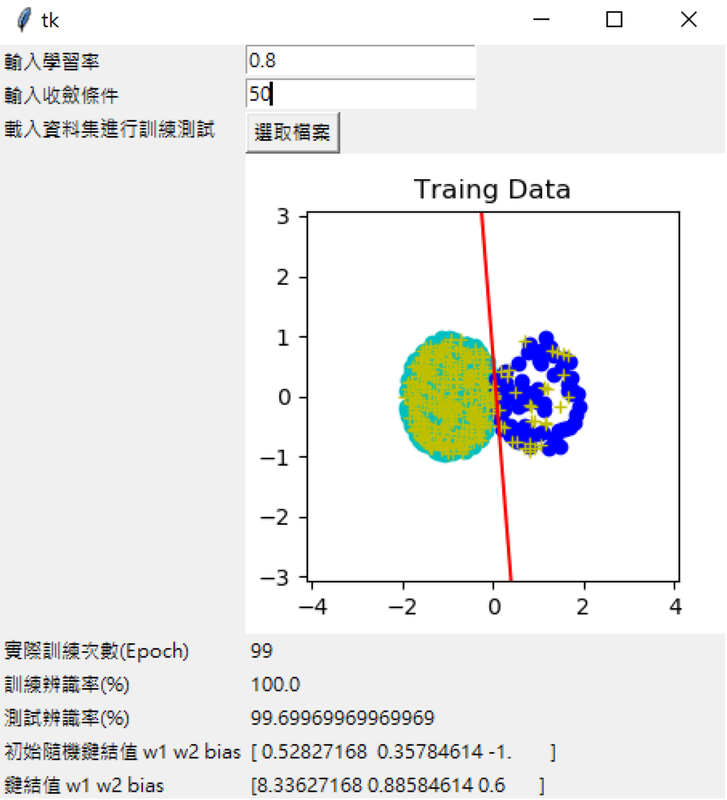
訓練辨識率(%) 100.0

測試辨識率(%) 98.50746268656717

初始隨機鍵結值 w1 w2 bias [0.69700147 0.67279491 -1.   ]

鍵結值 w1 w2 bias [2.11300147 -0.08720509 -0.2   ]

5. 2CloseS3



輸入學習率 0.8

輸入收斂條件 50

載入資料集進行訓練測試

實際訓練次數(Epoch) 99

訓練辨識率(%) 100.0

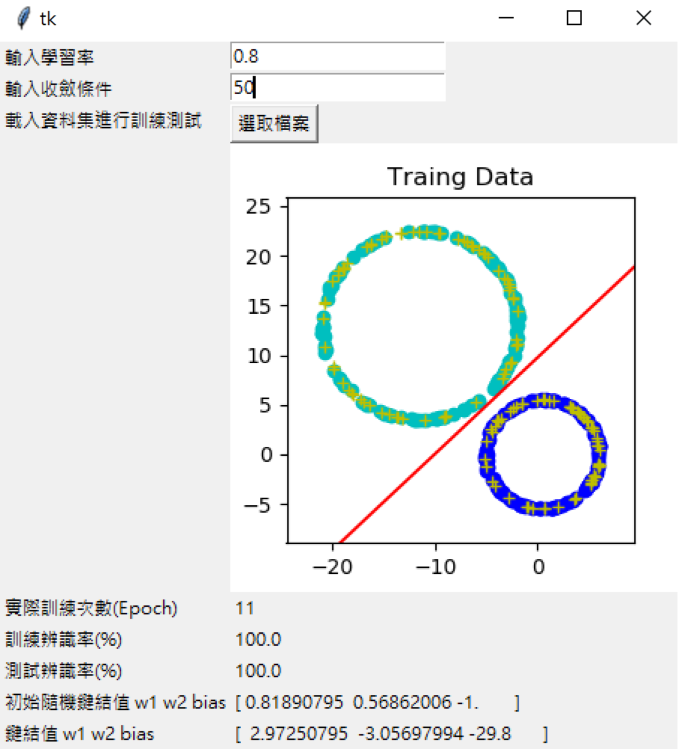
測試辨識率(%) 99.69969969969969

初始隨機鍵結值 w1 w2 bias [0.52827168 0.35784614 -1.   ]

鍵結值 w1 w2 bias [8.33627168 0.88584614 0.6   ]

6. 2cring

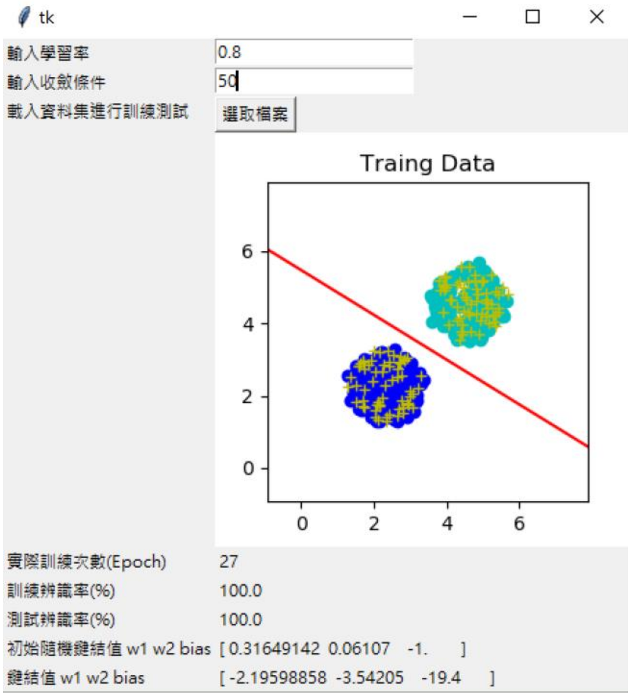
由圖可見同樣為單純分 2 群資料，因此訓練效果及測試辨識率結皆能達到 100%



輸入學習率	0.8
輸入收斂條件	50
載入資料集進行訓練測試	
實際訓練次數(Epoch)	11
訓練辨識率(%)	100.0
測試辨識率(%)	100.0
初始隨機鍵結值 w1 w2 bias	[0.81890795 0.56862006 -1.]
鍵結值 w1 w2 bias	[2.97250795 -3.05697994 -29.8]

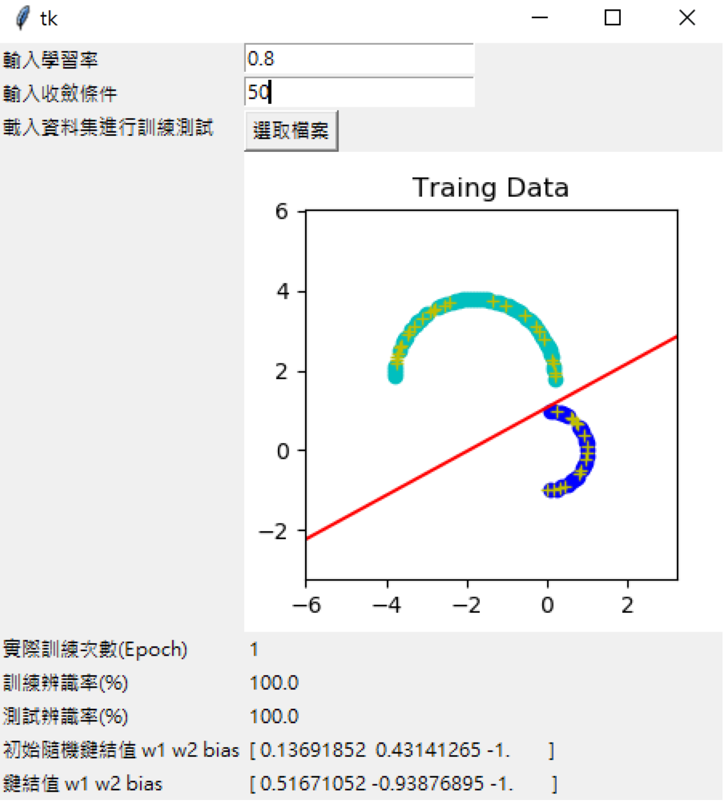
7. 2CS

由圖可顯著見其分 2 群，因此訓練效果及測試辨識率結皆能達到 100%



輸入學習率	0.8
輸入收斂條件	50
載入資料集進行訓練測試	
實際訓練次數(Epoch)	27
訓練辨識率(%)	100.0
測試辨識率(%)	100.0
初始隨機鍵結值 w1 w2 bias	[0.31649142 0.06107 -1.]
鍵結值 w1 w2 bias	[-2.19598858 -3.54205 -19.4]

8. 2Hcircle1



輸入學習率 0.8

輸入收斂條件 50

載入資料集進行訓練測試

實際訓練次數(Epoch) 1

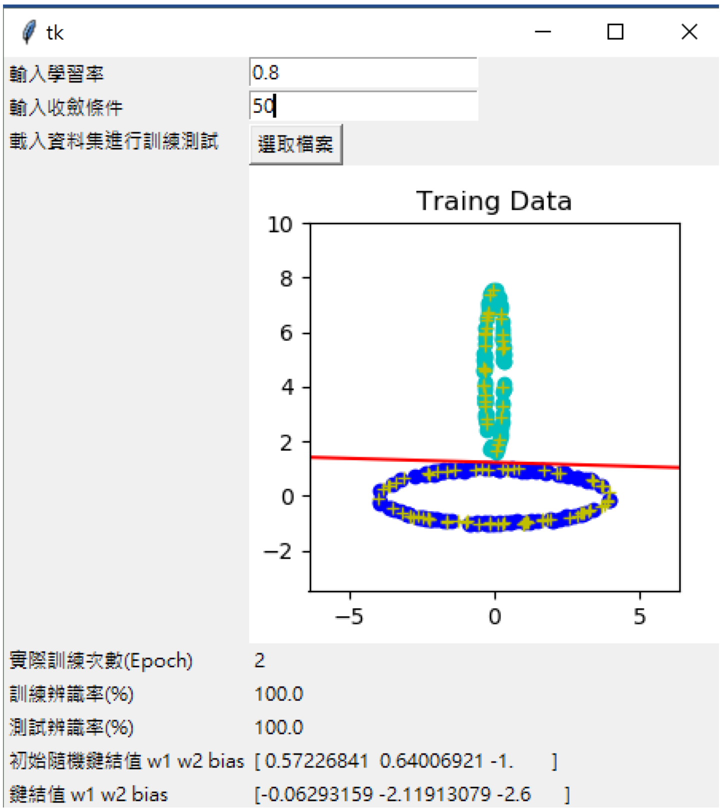
訓練辨識率(%) 100.0

測試辨識率(%) 100.0

初始隨機鍵結值 w1 w2 bias [0.13691852 0.43141265 -1.   ]

鍵結值 w1 w2 bias [0.51671052 -0.93876895 -1.   ]

9. 2ring



輸入學習率 0.8

輸入收斂條件 50

載入資料集進行訓練測試

實際訓練次數(Epoch) 2

訓練辨識率(%) 100.0

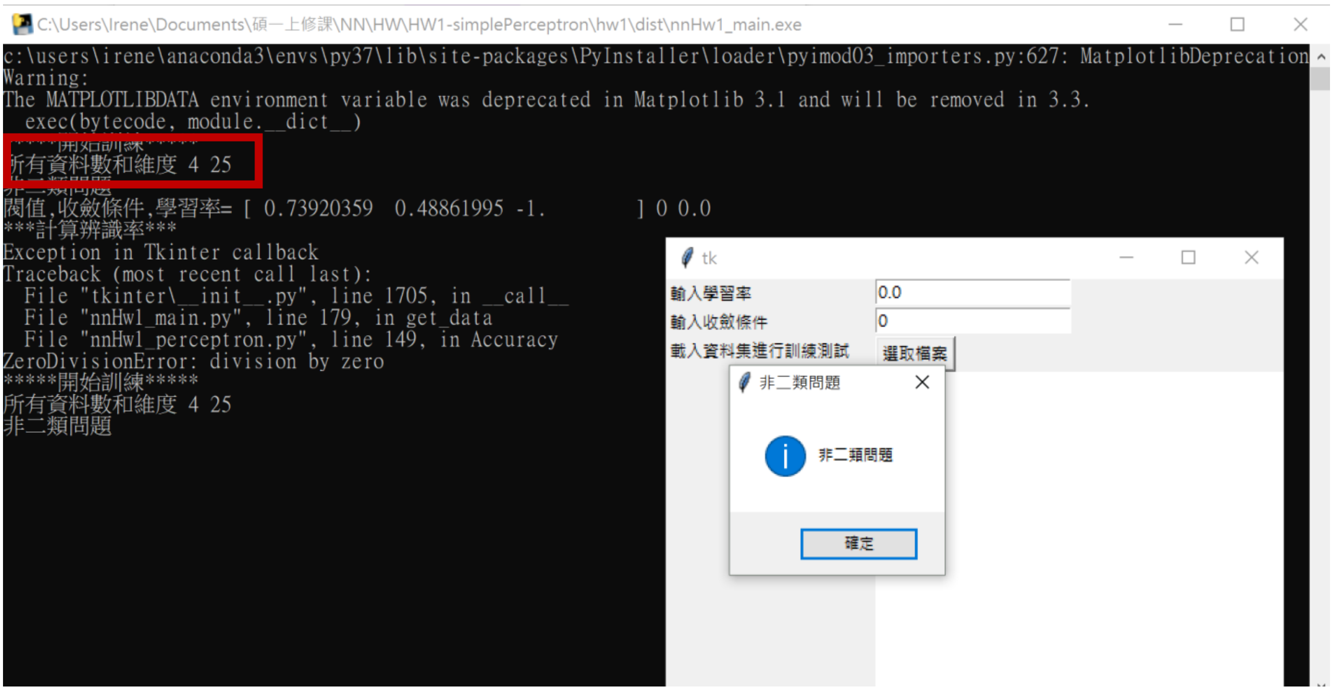
測試辨識率(%) 100.0

初始隨機鍵結值 w1 w2 bias [0.57226841 0.64006921 -1.   ]

鍵結值 w1 w2 bias [-0.06293159 -2.11913079 -2.6   ]

10. Number

資料維度過高(25)，非二類資料，難於單層感知機做訓練。



C:\Users\Irene\Documents\碩一上修課\NN\HW\HW1-simplePerceptron\hw1\dist\nnHw1_main.exe

```
c:\users\irene\anaconda3\envs\py37\lib\site-packages\PyInstaller\loader\pyimod03_importers.py:627: MatplotlibDeprecation
Warning:
The MATPLOTLIBDATA environment variable was deprecated in Matplotlib 3.1 and will be removed in 3.3.
  exec(bytecode, module.__dict__)
*****開始訓練*****
所有資料數和維度 4 25
非二類問題
閥值,收斂條件,學習率= [ 0.73920359  0.48861995 -1.        ] 0 0.0
***計算辨識率***
Exception in Tkinter callback
Traceback (most recent call last):
  File "tkinter\__init__.py", line 1705, in __call__
  File "nnHw1_main.py", line 179, in get_data
  File "nnHw1_perceptron.py", line 149, in Accuracy
ZeroDivisionError: division by zero
*****開始訓練*****
所有資料數和維度 4 25
非二類問題
```

tk

輸入學習率 0.0

輸入收斂條件 0

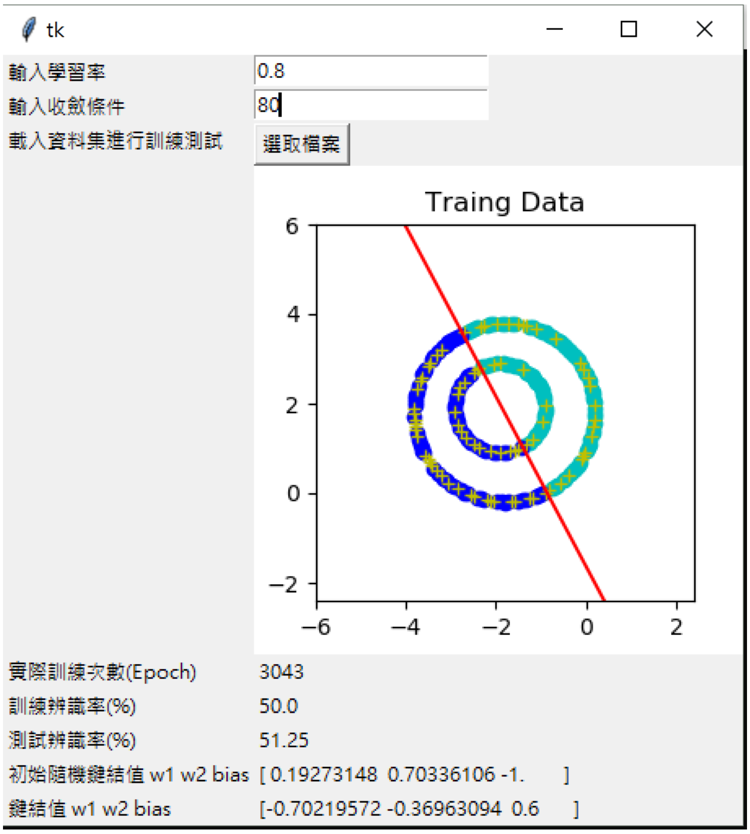
載入資料集進行訓練測試

非二類問題

非二類問題

11. 2Ccircle1

結果不甚理想，可能須多層感知機才能將其更好地分類。此外訓練次數顯示有錯誤，可能程式有 bug，會再做修正。



輸入學習率 0.8

輸入收斂條件 80

載入資料集進行訓練測試

實際訓練次數(Epoch) 3043

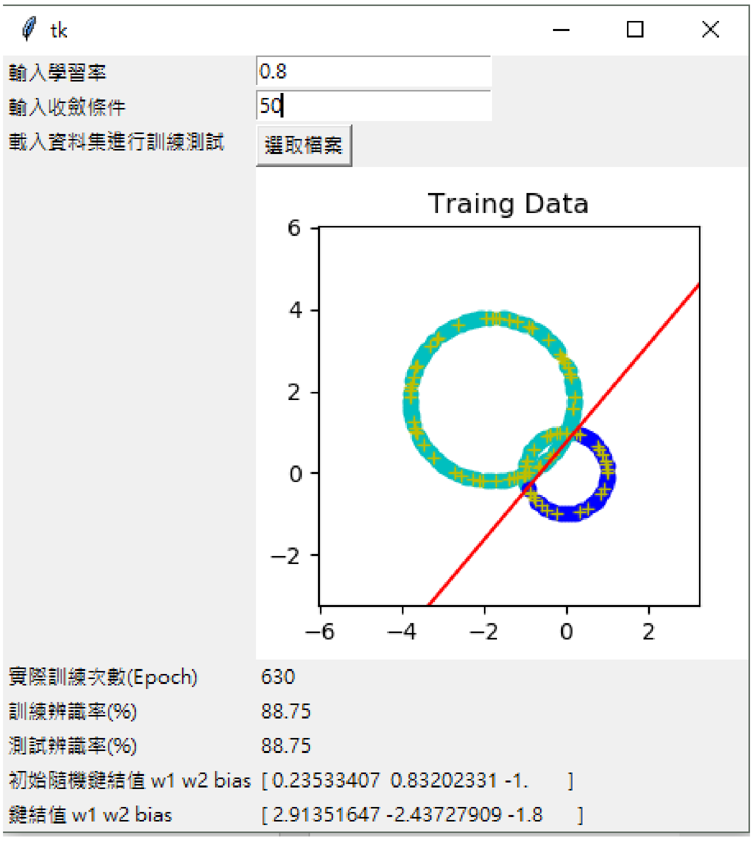
訓練辨識率(%) 50.0

測試辨識率(%) 51.25

初始隨機鍵結值 w1 w2 bias [0.19273148 0.70336106 -1.]

鍵結值 w1 w2 bias [-0.70219572 -0.36963094 0.6]

12. 2Circle1



輸入學習率 0.8

輸入收斂條件 50

載入資料集進行訓練測試

實際訓練次數(Epoch) 630

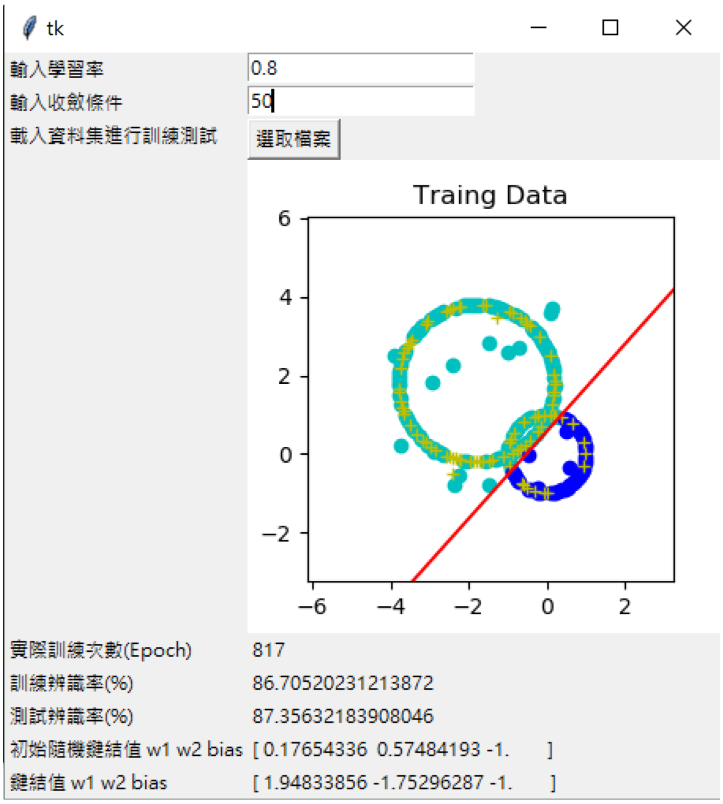
訓練辨識率(%) 88.75

測試辨識率(%) 88.75

初始隨機鍵結值 w1 w2 bias [0.23533407 0.83202331 -1.]

鍵結值 w1 w2 bias [2.91351647 -2.43727909 -1.8]

13. 2Circle2



輸入學習率 0.8

輸入收斂條件 50

載入資料集進行訓練測試

實際訓練次數(Epoch) 817

訓練辨識率(%) 86.70520231213872

測試辨識率(%) 87.35632183908046

初始隨機鍵結值 w1 w2 bias [0.17654336 0.57484193 -1.]

鍵結值 w1 w2 bias [1.94833856 -1.75296287 -1.]