類神經網路-Hw1-報告-108602532-王鼎元

A、GUI 介紹:

[1] 選擇要用來訓練的資料集

[2] 輸入學習率以及收斂條件，學習率建議低一點，收斂條件是指在訓練組上測試後正確率達到多少%就停止訓練開始測試，輸入 0-100 之間的數字，一開始也是低一點必免一直無法收斂，可以慢慢增加來測試最好的準確率是多少

[3] 開始執行程式

[4] 會顯示訓練完的權重，以畫成線條的公式形式表示

[5] 顯示訓練完的權重在訓練組以及測試組上的準確率有多少

[6] 視覺化的原始資料集，兩種顏色的點代表不同類別

[7] 視覺化的訓練資料集，兩種顏色的點代表不同類別，黑色的線是權重

[8] 視覺化的測試資料集，兩種顏色的點代表不同類別，黑色的線是權重

B、程式碼介紹:

```python
import tkinter as tk
from tkinter.constants import CENTER
import numpy as np
import matplotlib.pyplot as plt
from tkinter import filedialog
from tkinter import messagebox
import random

#initialize
dataset_path = "None"
resFig = np.linspace(-10,10,100)
fig = plt.figure() #定義一個圖像窗口
plt.plot(0, 0, '.') #定義x,y和圖的樣式
plt.title("Full Data Set")
fig.savefig('0.png', dpi = 80)
plt.title("Training Set")
fig.savefig('1.png', dpi = 80)
plt.title("Testing Set")
fig.savefig('2.png', dpi = 80)
plt.close(fig)
trAc = 0.00
teAc = 0.00

#window setting
window = tk.Tk()
window.title('Hw1-108602532-王鼎元')
window.geometry('1140x800')
window.resizable(False, False)
```

Import 函式庫

初始化資料路徑，準確率、視覺化圖表

設定 GUI 視窗

```python
def xyz(data):···

def draw(x, y, z, name, color1, color2, line, w, title):···

def training(trainingData, learningRate, divergeCondition):···

def testing(testingData, w):···

def train(dataset_path, learningRate, divergeCondition):···


def run():···

def importDataset():···
```

30~187 為 function，稍後介紹

```python
#run
start = tk.Button(text="Run", command=run)
start.place(x=250,y=720)

#learning rate
learningRateLabel = tk.Label(text = "Learning Rate :")
learningRateLabel.place(x=135,y=640)

getLearningRate = tk.Entry()
getLearningRate.place(x=270,y=640)

#diverge condition
getDivergeCondition = tk.Entry()
getDivergeCondition.place(x=270,y=680)

divergeConditionLabel = tk.Label(text = "Accuracy rate(%) :")
divergeConditionLabel.place(x=135,y=680)
```

按下按鈕執行 run function

設定各項 Label、按鈕

```python
#dataset
importData = tk.Button(text="Choose Dataset", command=importDataset)
importData.place(x=165,y=580)
currentDataset = tk.Label(text = dataset_path)
currentDataset.place(x=300,y=585)

#results
trainingAccurancy = tk.Label(text="Training accuracy:    "+str(trAc)+"%")
trainingAccurancy.place(x=80,y=480)
testingAccurancy = tk.Label(text="Testing accuracy:    "+str(teAc)+"%")
testingAccurancy.place(x=80,y=510)

#formula
correctFormula = tk.Label(text="Weight:    0*x + 0*y + 0")
correctFormula.place(x=280,y=495)

#pictures
img = tk.PhotoImage(file='0.png')
result = tk.Label(image=img)
result.place(x=30,y=0)

img1 = tk.PhotoImage(file='1.png')
result1 = tk.Label(image=img1)
result1.place(x=570,y=0)

img2 = tk.PhotoImage(file='2.png')
result2 = tk.Label(image=img2)
result2.place(x=570,y=400)

window.mainloop()
```

設定各項 Label、按鈕

主程式結束

```python
def run():
    global dataset_path,learningRate,divergeCondition
    #collect parameters
    if len(getLearningRate.get()) == 0:
        tk.messagebox.showwarning("Warning", "fill Learning Rate")
    else:
        learningRate = float(getLearningRate.get())
        if len(getDivergeCondition.get()) == 0:
            tk.messagebox.showwarning("Warning", "fill Epoch limit")
        else:
            divergeCondition = float(getDivergeCondition.get())

    #training & plot
    if dataset_path == 'None':
        tk.messagebox.showwarning("Warning", "choose training dataset")
    else:
        train(dataset_path, learningRate, divergeCondition)
```

檢查各項參數是否輸入完畢

輸入完畢，執行 train function

```python
def train(dataset_path, learningRate, divergeCondition):
    global trAc
    global teAc
    #讀檔 & list化
    f = open(dataset_path, 'r')
    data = f.read().split("\n")
    f.close()
    if len(data[-1]) == 0:
        del data[-1]
    for i in range(len(data)):
        data[i] = data[i].split(" ")
        for j in range(2):
            data[i][j] = float(data[i][j])
        data[i][2] = int(data[i][2])
```

從選定的路徑中讀取資料並整理，有些資料集最後會有換行，將其移除

```
125     #畫原始資料集圖表
126     x, y, z = xyz(data)
127     draw(x, y, z, "0.png", 'b', 'c', 0, [0,0], "Full Data Set")
128     img = tk.PhotoImage(file='0.png')
129     result.configure(image=img)
130     result.image = img
131     # 隨機排序
132     random.shuffle(data)
133     # 丟進測試 & 訓練資料
134     trainingData = data[:2*len(data)//3]
135     testingData = data[2*len(data)//3:]
136     #訓練
137     trainingData, w, trAc = training(trainingData, learningRate, divergeCondition)
138     #繪製訓練結果
139     x, y, z, = xyz(trainingData)
140     draw(x, y, z, "1.png", 'r', "orange", 1, w, "Training Set")
141     #測試
142     testingData, teAc = testing(testingData, w)
143     #繪製測試結果
144     x, y, z = xyz(testingData)
145     draw(x, y, z, "2.png", 'limegreen', 'green', 1, w, "Testing Set")
146     #、訓練結果圖
147     img1 = tk.PhotoImage(file='1.png')
148     result1.configure(image=img1)
149     result1.image = img1
150     #、測試結果圖
151     img2 = tk.PhotoImage(file='2.png')
152     result2.configure(image=img2)
153     result2.image = img2
154     #、正確率label更新
155     for i in range(len(w)):
156         w[i] = round(w[i], 2)
157     trainingAccurancy.configure(text="Training accuracy: "+str(trAc)+"%")
158     testingAccurancy.configure(text="Testing accuracy:   "+str(teAc)+"%")
159     correctFormula.configure(text="Weight:   ("+str(w[0])+")*x + ("+str(w[1])+")*y + ("+str(w
160
```

如同註解說明
當中 xyz()是用來整理資料以便繪圖的 function
draw()是用來繪圖的 function

```
65  def training(trainingData, learningRate, divergeCondition):
66      w = [random.randint(-1000, 1000)/1000, random.randint(-1000, 1000)/1000, random.randint(
67      while True:
68          #訓練一次
69          for i in range(len(trainingData)):
70              tmp = w[0]*trainingData[i][0] + w[1]*trainingData[i][1] + w[2]
71              #答案錯誤
72              if tmp * (trainingData[i][2]*2-3) < 0:
73                  #更新權重
74                  if tmp < 0:
75                      w[0] = w[0] + trainingData[i][0]*learningRate
76                      w[1] = w[1] + trainingData[i][1]*learningRate
77                      w[2] = w[2] + learningRate
78                  else:
79                      w[0] = w[0] - trainingData[i][0]*learningRate
80                      w[1] = w[1] - trainingData[i][1]*learningRate
81                      w[2] = w[2] - learningRate
82          #測試訓練集
83          acRate = 0
84          z = []
85          for i in range(len(trainingData)):
86              tmp = w[0]*trainingData[i][0] + w[1]*trainingData[i][1] + w[2]
87              if tmp * (trainingData[i][2]*2-3) > 0:
88                  acRate = acRate + 1
89                  z.append(-trainingData[i][2]+3)
90              else:
91                  z.append(trainingData[i][2])
92          finalAc = 100*acRate/len(trainingData)
93          #達到收斂條件
94          if finalAc > divergeCondition:
95              for i in range(len(trainingData)):
96                  trainingData[i][2] = z[i]
97              return trainingData, w, round(finalAc,2)
98
```

判斷單筆資料是否正確

更新權重

計算正確率

達到收斂條件後返回結果

```python
def testing(testingData, w):
    #測試
    acRate = 0
    for i in range(len(testingData)):
        tmp = w[0]*testingData[i][0] + w[1]*testingData[i][1] + w[2]
        if tmp * (testingData[i][2]*2-3) > 0:
            acRate = acRate + 1
            testingData[i][2] = (-testingData[i][2]+3)
    finalAc = round(100*acRate/len(testingData),2)
    return testingData, finalAc
```

計算正確率

```python
def xyz(data):
    x = []
    y = []
    z = []
    for i in range(len(data)):
        x.append(float(data[i][0]))
        y.append(float(data[i][1]))
        z.append(float(data[i][2]))
    return x, y, z

def draw(x, y, z, name, color1, color2, line, w, title):
    fig = plt.figure() #定義一個圖像窗口
    x1 = []
    y1 = []
    x2 = []
    y2 = []
    for i in range(len(x)):
        if z[i] == 2:
            x1.append(x[i])
            y1.append(y[i])
        else:
            x2.append(x[i])
            y2.append(y[i])

    plt.plot(x1, y1, '.',color = color1)
    plt.plot(x2, y2, '.',color = color2)
    x = np.linspace(min(x)-0.1, max(x)+0.1, 100000)
    if line == 1:
        plt.plot(x, (-w[0]*x-w[2])/w[1], '.', color = "black")
    plt.xlim(min(x)-0.1,max(x)+0.1)
    plt.ylim(min(y)-0.1,max(y)+0.1)
    plt.title(title)
    fig.savefig(name, dpi = 80)
    plt.close(fig)
```

分類顏色

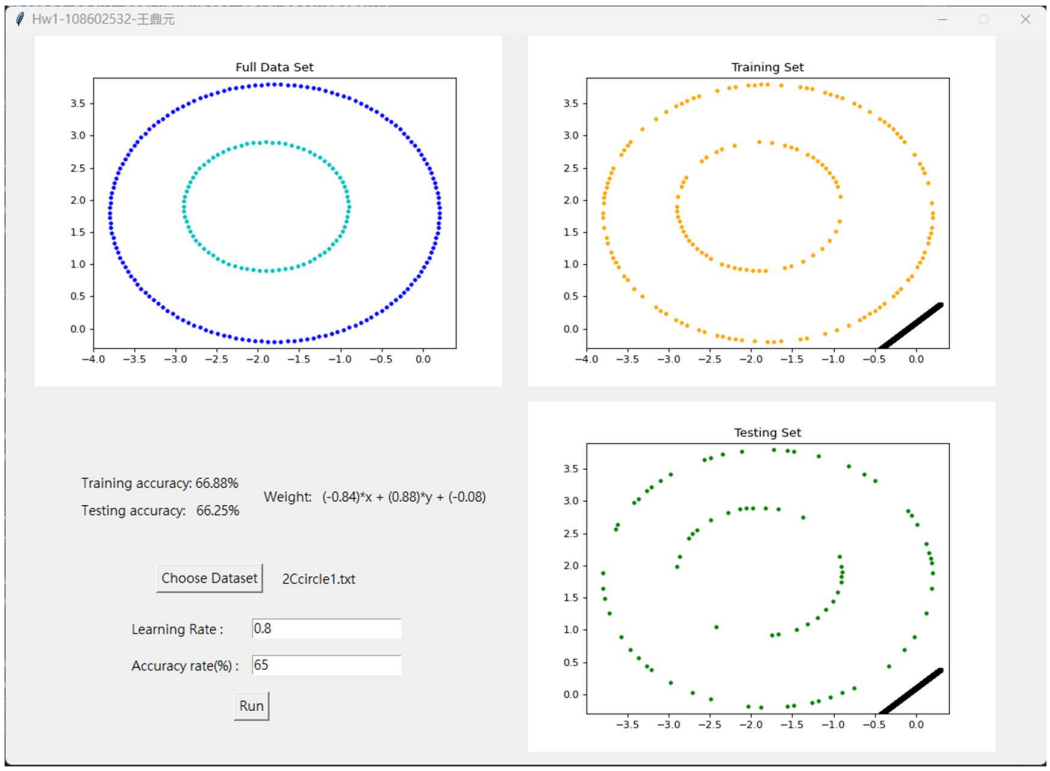
繪圖

```python
def importDataset():
    #open choose file window
    global dataset_path
    dataset_path = filedialog.askopenfilename()
    dataName = dataset_path.split('/')[-1]
    currentDataset.configure(text = dataName)
    currentDataset.place(x=300,y=585)
```

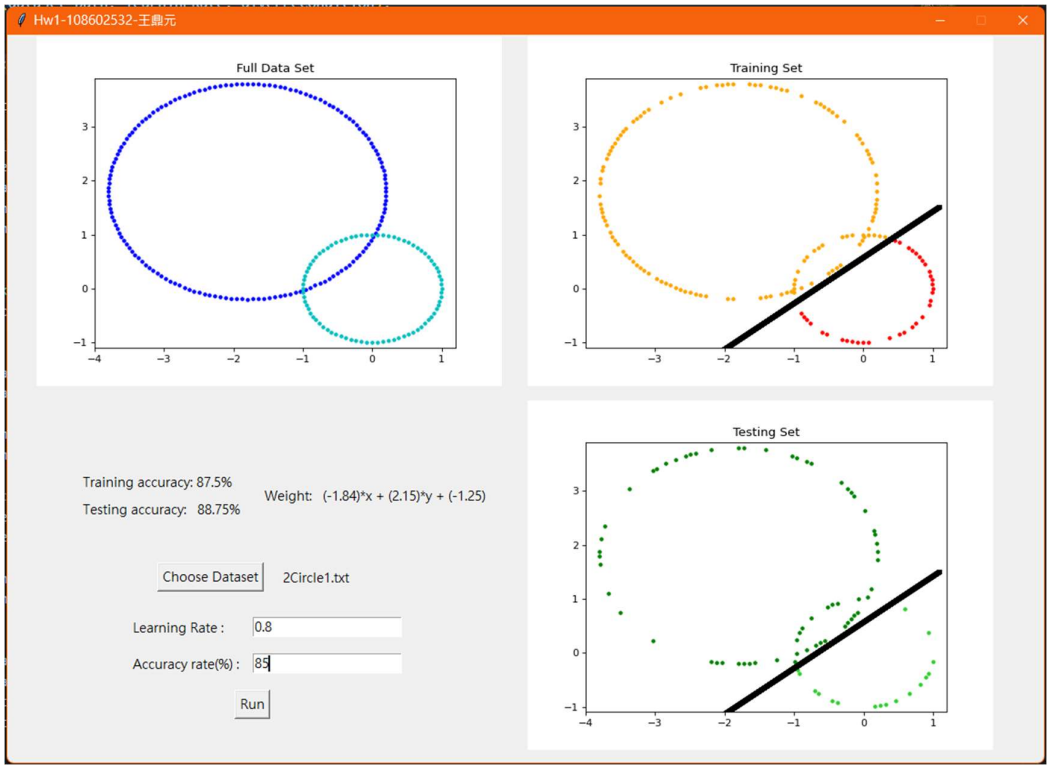
讀檔

C、實驗結果

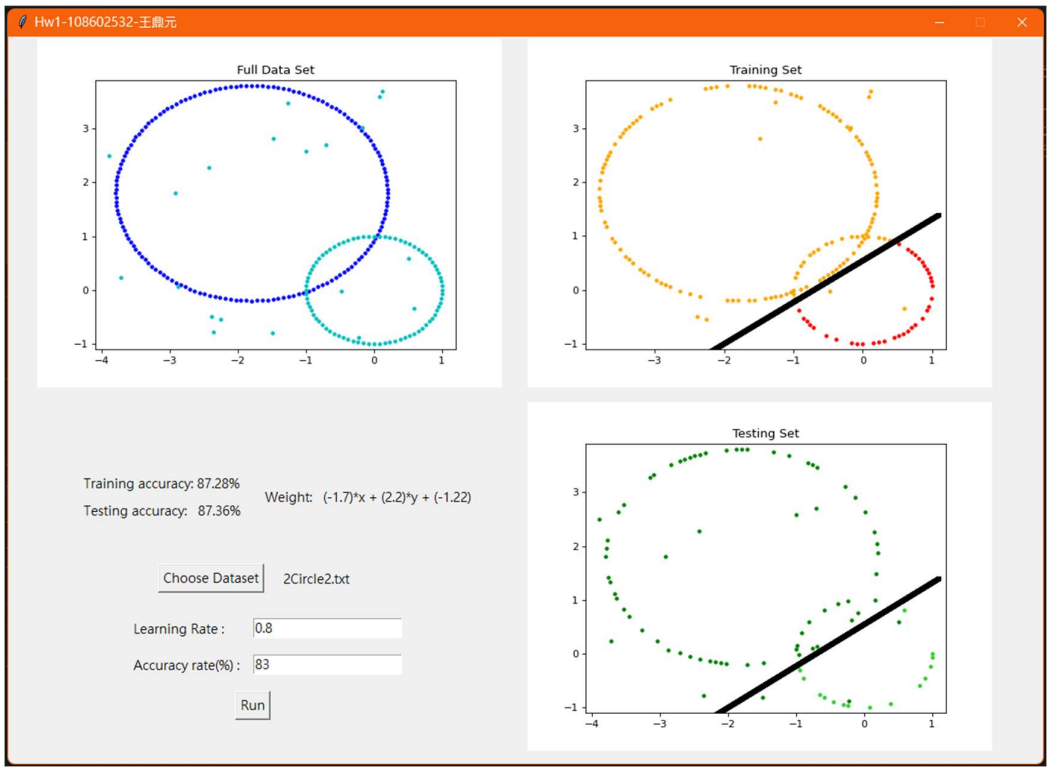
1. 2Ccircle1.txt



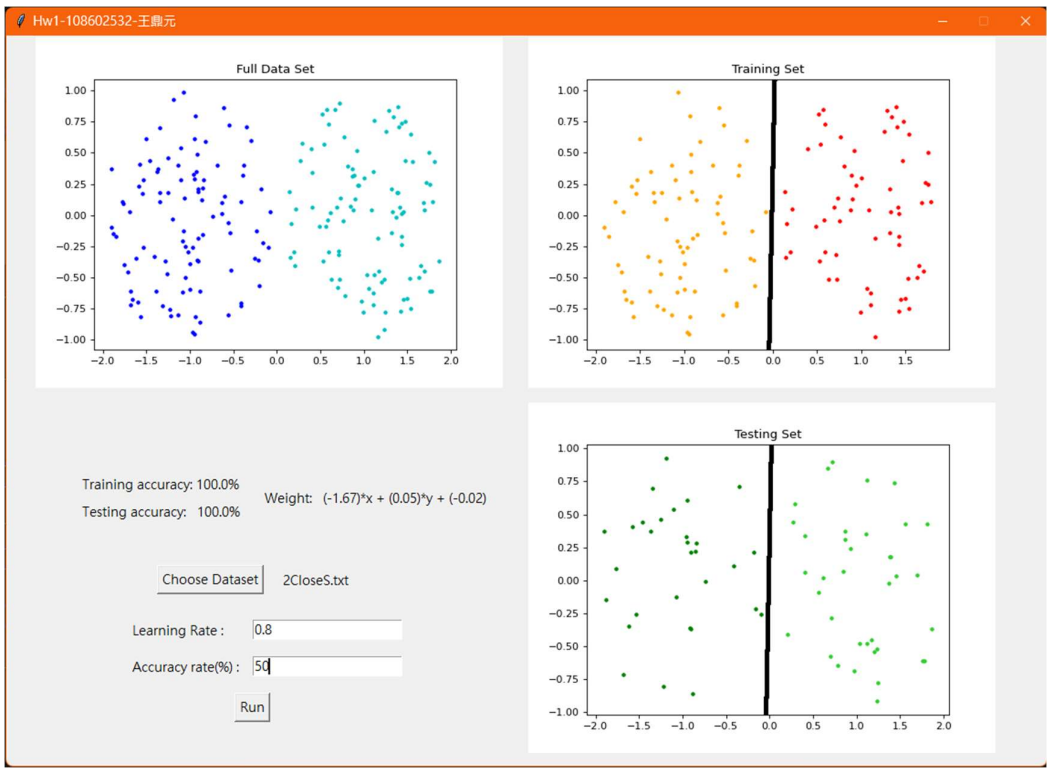
2. 2Circle1.txt



3. 2Circle2.txt



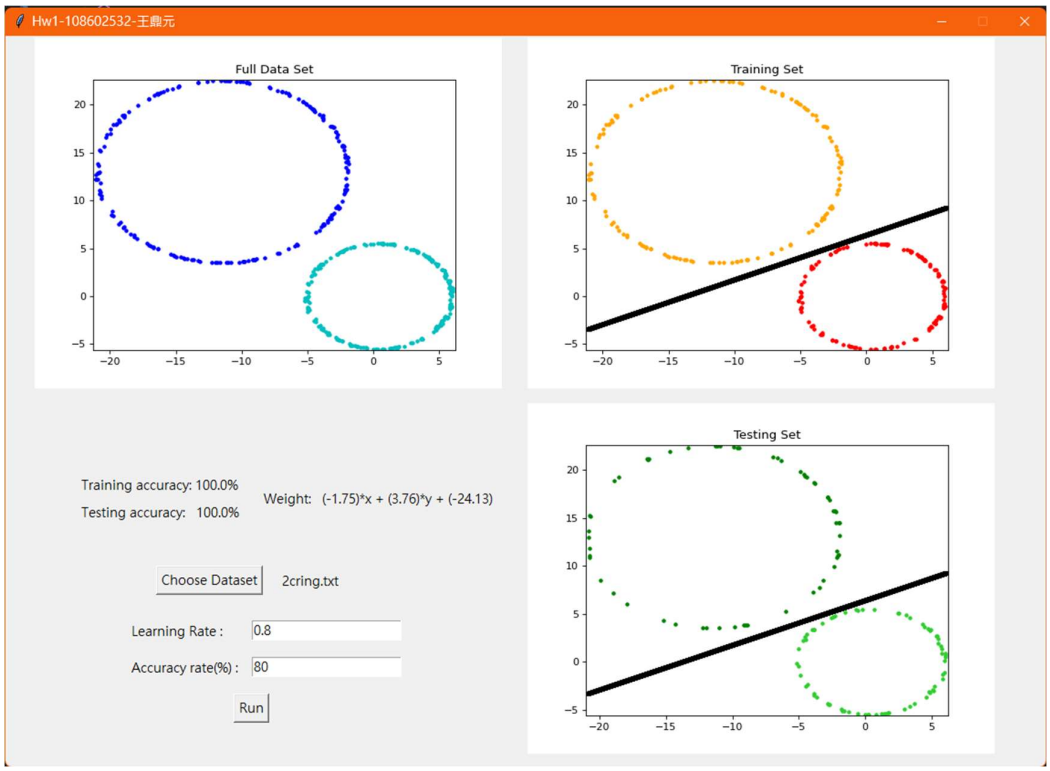
4. 2CloseS.txt



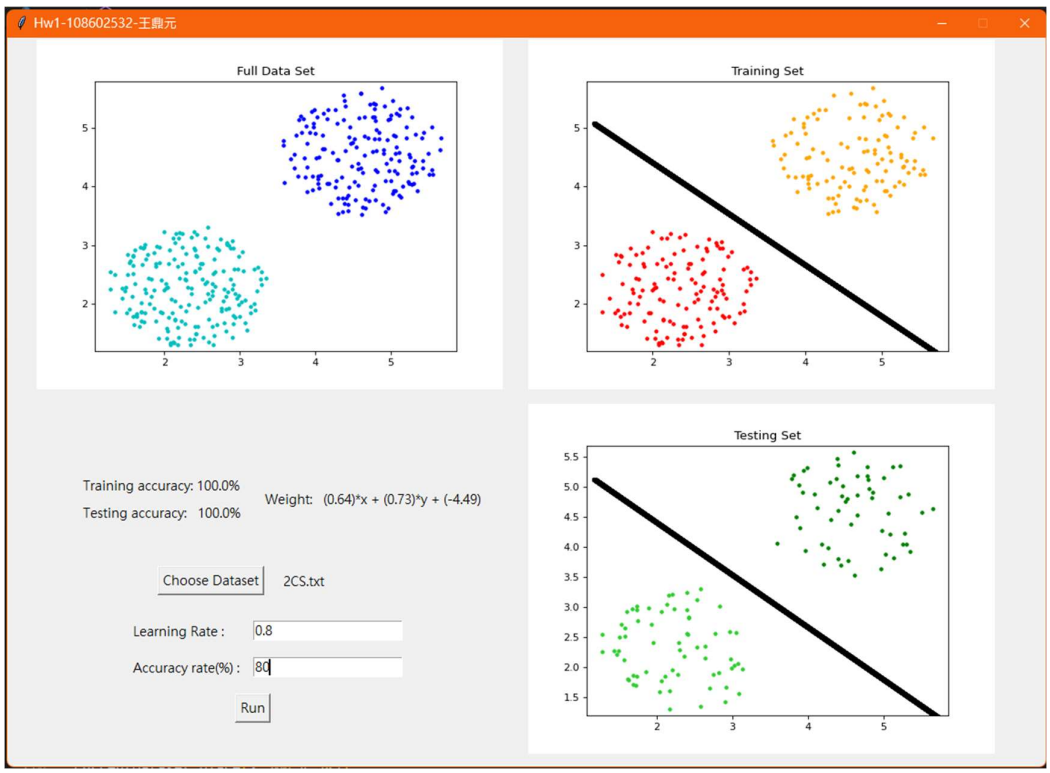
5. 2CloseS2.txt

6. 2CloseS3.txt

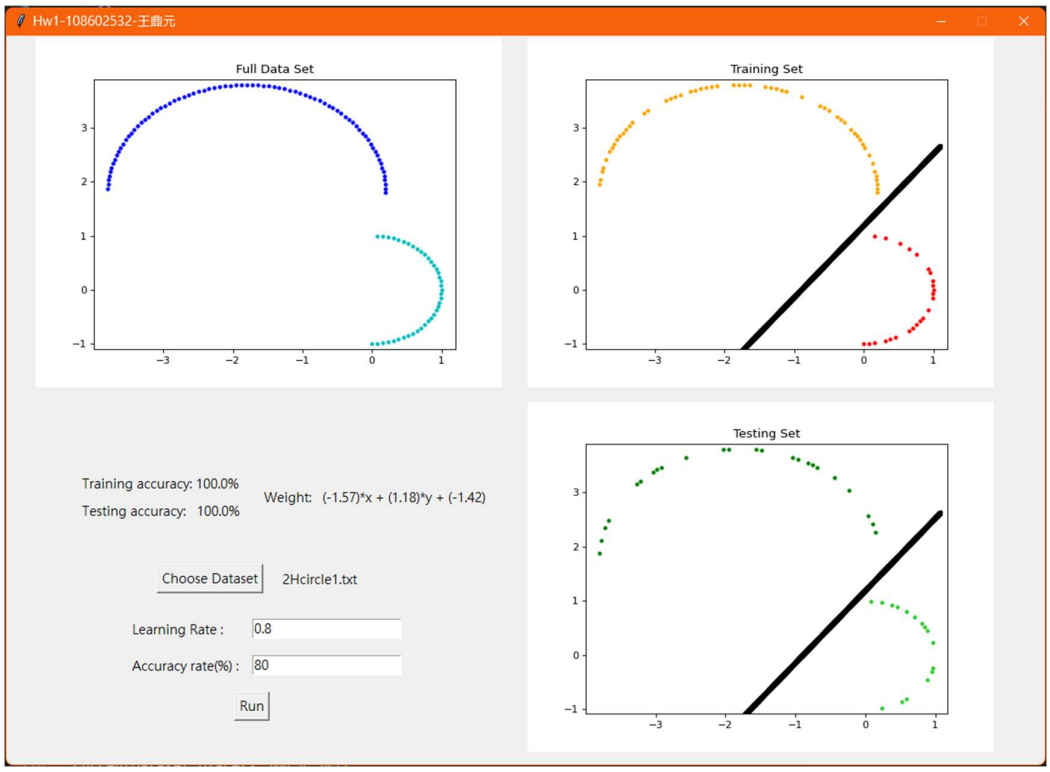
7. 2cring.txt



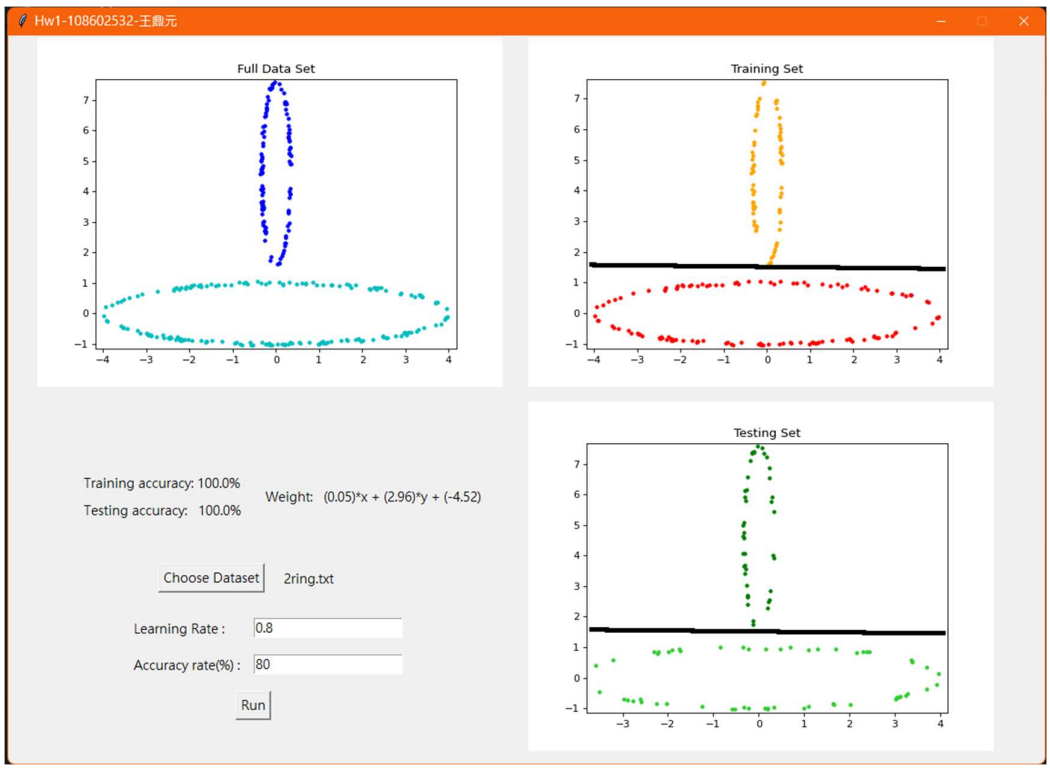
8. 2CS.txt



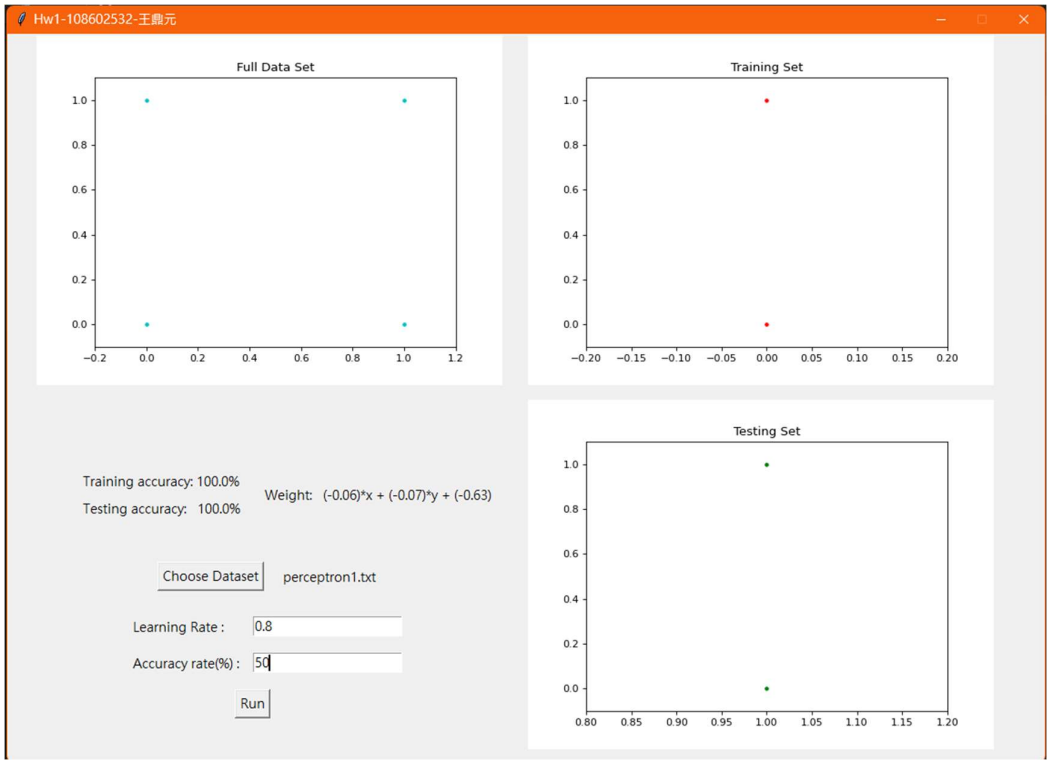
9. 2Hcircle1.txt



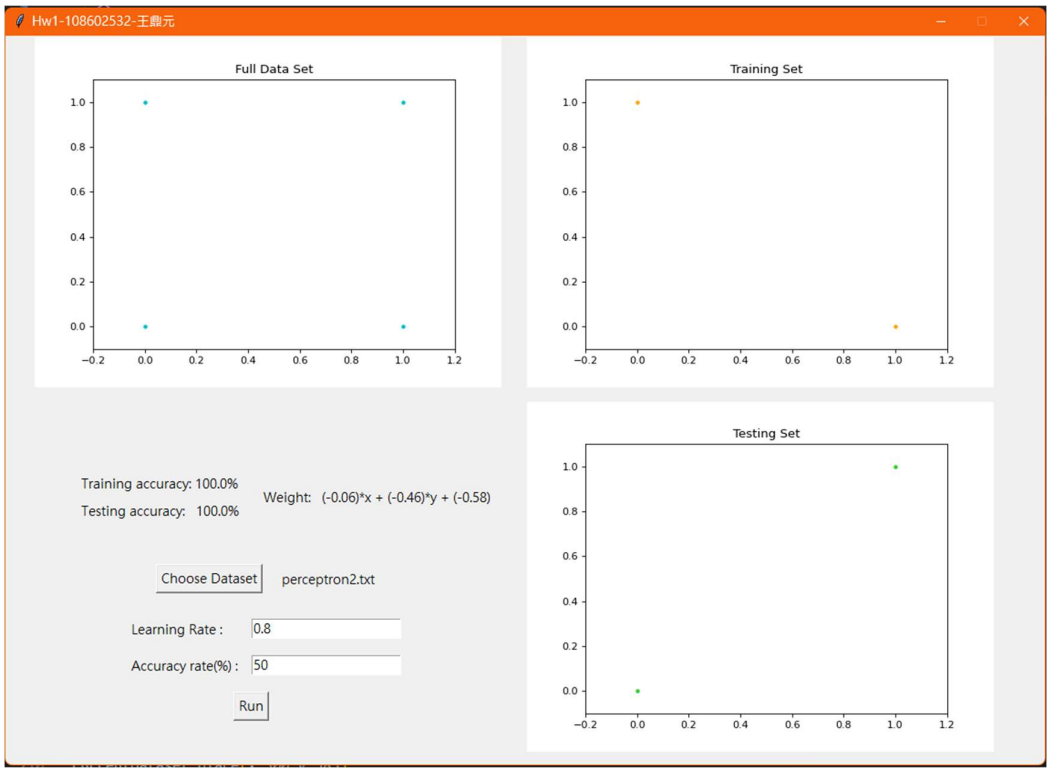
10. 2ring.txt



11. perceptron1.txt *黑線在顯示範圍外，將所有的點分成一類



12. perceptron2.txt *黑線在顯示範圍外，將所有的點分成一類



D、實驗結果分析

[1] 2Ccircle1.txt

此題為同心圓，正確的分割方法應該是要在大小兩個同心圓之間分割，但是我這次設定的神經元只有一次式，因此這題訓練不出來

訓練一次準確率: 50%　　訓練多次準確率: 65%

[2] 2Circle1.txt、2Circle2.txt

兩題圖案長得很像，差別在於第二題的圖案多了一些雜訊點，一樣因為有交錯，因此一次方程式的直線無法切開兩者，但是比起第一題，兩個圓已經有很大一部分是分開的了，因此正確率方面有顯著進步

訓練一次準確率: 80%　　訓練多次準確率: 86%

[3] 2CloseS.txt、2CloseS2、2CloseS3、2CS.txt

四題是同類型的題目，差別只在點的分布，數據量多寡以及稀疏程度而已，這三題兩個類別的點分布都是分散的，一次方程式幾乎可以完全切開，因此準確率極度接近 100%

訓練一次準確率: 99%　　訓練多次準確率: 99%

[4] 2cring.txt、2Hcircle1.txt、2ring.txt

三題也是同類型的題目，與上一個類別的差別是這幾題的分布不是均勻的，是有一個圖形在，但是由於兩個類別也可以完全被一直線分開，因此準確率野是極度接近 100%，與上一個類別差別不大

訓練一次準確率: 100%　　訓練多次準確率: 100%

[5] perceptron1.txt、perceptron2.txt

這兩題是同類型的題目，資料量極少，因此雖然我的截圖上準確率是 100%，但是根據隨機拆分訓練組和測試組的情形，以及一開是設定初始權重的 random number，結果也十分有機會是 50%或是 0%，以四筆資料來看，training set 只有兩組資料，如果剛好得到同一類型的資料那就那就基本上不用訓練了，怎麼測都對，那就算是得到不同類型的也只會有一次的修改機會，因此我認為這兩題對感知機來說基本上等於用猜的

訓練一次準確率: ?%　　訓練多次準確率: ?%