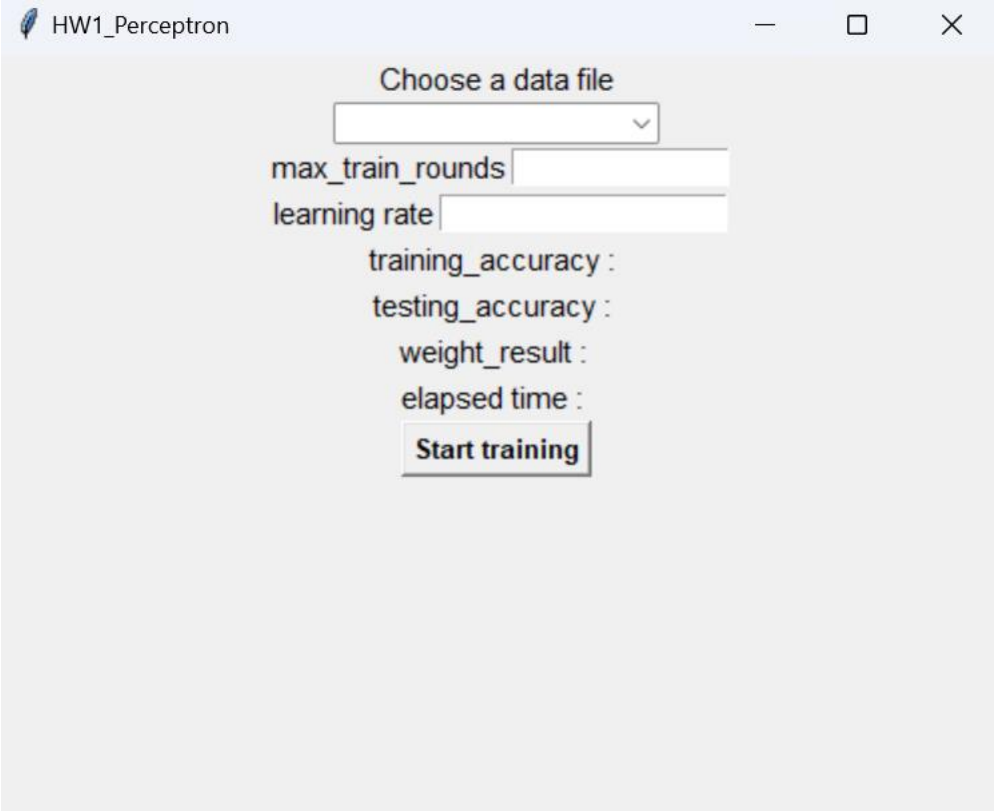
## 一、程式執行說明

這是尚未選取檔案前的 GUI 畫面




# GUI 介面執行在 if __name__ == "__main__":

1. 自訂一個下拉選單(combobox)選取指定的 txt 檔案

2. 輸入訓練次數上限(只能為整數)

3. 輸入學習率，接著按下 start training 開始執行程式

4. 訓練完成後，訓練正確率、測試正確率、鍵結值、訓練時間會顯示出結果

## 二、程式碼簡介

用到的函式為 numpy, matplotlib, os, time, tkinter

**※主要 Function 說明:**

1. reform_data

```
def reform_data(dataset):
    np.random.shuffle(dataset)
    train_num = int(2 * dataset.shape[0] / 3)

    data_train = dataset[:train_num, ...]
    data_test = dataset[train_num:, ...]
    data_train_x, data_train_y, data_train_d = np.hsplit(data_train, dataset.shape[1])
    data_train = np.hstack((data_train_x, data_train_y))

    data_test_x, data_test_y, data_test_d = np.hsplit(data_test, dataset.shape[1])
    data_test = np.hstack((data_test_x, data_test_y))

    return data_train, data_train_d, data_test, data_test_d
```

先將 dataset 內容打散，再透過 numpy 陣列的切割(split)和合併(stack)操作

將當前選取 txt 檔的資料，劃分成訓練資料及測試資料

2. plot_data

```
def plot_data(w, data_array, data_array_d, chosen_file_name, is_train, is_test):
    train_or_test = ""
    if is_train == 1 and is_test == 0: train_or_test = "Training Data"
    elif is_train == 0 and is_test == 1: train_or_test = "Testing Data"

    d_max, d_min = np.amax(data_array_d), np.amin(data_array_d)

    if d_max == 1.0 and d_min == 0.0:
        for i in range(data_array.shape[0]):
            if data_array_d[i][0] == 0.0:
                plt.scatter(data_array[i][0], data_array[i][1], c = 'blue', s = 10)
            else:
                plt.scatter(data_array[i][0], data_array[i][1], c = 'red', s = 10)
    elif d_max == 2.0 and d_min == 1.0:
        for i in range(data_array.shape[0]):
            if data_array_d[i][0] == 1.0:
                plt.scatter(data_array[i][0], data_array[i][1], c = 'blue', s = 10)
            else:
                plt.scatter(data_array[i][0], data_array[i][1], c = 'red', s = 10)

    # find maximum, minimum in index 0(value x) of data_array
    max_x, min_x = np.max(data_array[..., 0]), np.min(data_array[..., 0])

    # linear eq: w1x+w2y-w0=0 (w0 is bias) ---> y = -(w1x-w0)/w2
    max_y, min_y = -1*(w[1]*max_x - w[0])/w[2], -1*(w[1]*min_x - w[0])/w[2]

    chosen_file_name = chosen_file_name.split('.')[0] + f' ({train_or_test})'
    save_path = "D:\\大三上\\類神經網路\\hw1\\hw1_截圖"
    plt.plot([min_x, max_x], [min_y, max_y])
    plt.title(chosen_file_name)
    plt.xlabel('weight  ---  w0(bias):{:.3f}   w1:{:.3f}   w2:{:.3f}'.format(w[0], w[1], w[2]))
    plt.savefig(save_path + '\\' + chosen_file_name)
    plt.show()
    plt.close()
```

* is_train、is_test 是用來分辨當前傳入的資料是訓練用還是測試用

根據每個點的座標(x, y)及期望輸出(d)，區分顏色標示在圖上

找出 data 的 x 最大和最小值，再根據 w(鍵結值)去算出 y 的最大和最小值

最後，將兩個座標點連接畫出直線方程式即可

3. Train

```python
def Train():
    start_time = time.time()
    # get the chosen txt file
    file_name = selected_file.get()
    print(f'目前 {file_name} 執行結果:')
    file_name_path = os.path.join(data_path, file_name)
    dataset = np.loadtxt(file_name_path, delimiter = ' ') # delimiter default = space
    data_train, data_train_d, data_test, data_test_d = reform_data(dataset)

    # initialize weight, max_iter and learning rate
    w = np.random.rand(3)
    X = np.array([-1, np.random.rand(), np.random.rand()]) # init X[0] = -1

    # Entry.get() is a string, change it to integer or floating point
    max_iter = int(Max_training_iter_entry.get())
    learning_rate = float(learning_rate_entry.get())

    # find max, min in data_train_d
    data_train_d_max, data_train_d_min = np.amax(data_train_d), np.amin(data_train_d)
    # find max, min in data_test_d
    data_test_d_max, data_test_d_min = np.amax(data_test_d), np.amin(data_test_d)

    N = max_iter

    while N != 0:
        for i in range(data_train.shape[0]):
            X = np.array([-1, data_train[i][0], data_train[i][1]])
            cur_D = data_train_d[i][0]
            if (data_train_d_max == 2.0 and data_train_d_min == 1.0):
                cur_D -= 1
            dot_product = np.dot(w.T, X)
            # check whether need to adjust "current_weight" or not
            if is_match_d(dot_product) != int(cur_D):
                if is_match_d(dot_product) == 1 and int(cur_D) == 0:
                    w = w - learning_rate * X
                elif is_match_d(dot_product) == 0 and int(cur_D) == 1:
```

```
                w = w + learning_rate * X
            # from data_train[0] to retrain the weight
            i = 0
            N -= 1 # iteration_number minus 1
            continue
        # iteration_number minus 1
        N -= 1
        if N == 0:
            break
weight_result_label.configure(text = "weight_result : " + '\nw0(bias):{:.3f}, w1:{:.3f}, w2:{:.3f}'.format(w[0], w[1], w[2]))
print(f'鍵結值{w}')
end_time = time.time()
time_label.configure(text = f'elapsed time : {round(end_time - start_time, 3)} seconds')

# cal train_accuracy and round to third decimal place
train_success_num = 0
for j in range(data_train.shape[0]):
    X = np.array([-1, data_train[j][0], data_train[j][1]])
    cur_D = data_train_d[j][0]
    if (data_train_d_max == 2.0 and data_train_d_min == 1.0):
        cur_D -= 1
    cur_dot = X.dot(w.T)
    if is_match_d(cur_dot) == int(cur_D):
        train_success_num += 1
train_accuracy = round((train_success_num / data_train.shape[0]) * 100, 3)
training_accuracy_label.configure(text = "training_accuracy : " + f'{train_accuracy}%')
print(f'訓練資料正確率:{train_accuracy}%')

# cal test_accuracy and round to third decimal place
test_success_num = 0
for j in range(data_test.shape[0]):
    X = np.array([-1, data_test[j][0], data_test[j][1]])
    cur_D = data_test_d[j][0]
    if (data_test_d_max == 2.0 and data_test_d_min == 1.0):
        cur_D -= 1
    cur_dot = X.dot(w.T)
```

```
    if is_match_d(cur_dot) == int(cur_D):
        test_success_num += 1
test_accuracy = round((test_success_num / data_test.shape[0]) * 100, 3)
testing_accuracy_label.configure(text = "testing_accuracy : " + f'{test_accuracy}%')
print(f'測試資料正確率:{test_accuracy}%')

# plot training result
plot_data(w, data_train, data_train_d, file_name, 1, 0)
# plot testing result
plot_data(w, data_test, data_test_d, file_name, 0, 1)
```

將選取的檔案透過 reform_data 分割

自訂初始鍵結值(w = np.random.rand(3)), w0 是 bias

X 的話初始 X0 = -1, X1 和 X2 則隨意設置數字(0~1)

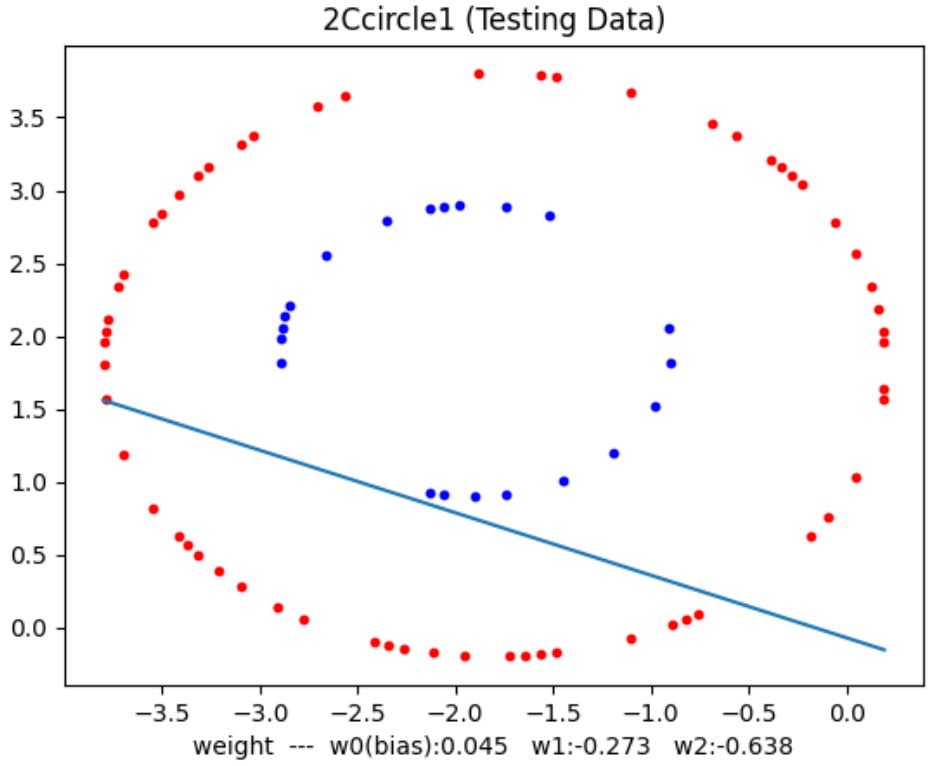
透過 while loop，根據 numpy 內積運算去調整鍵結值

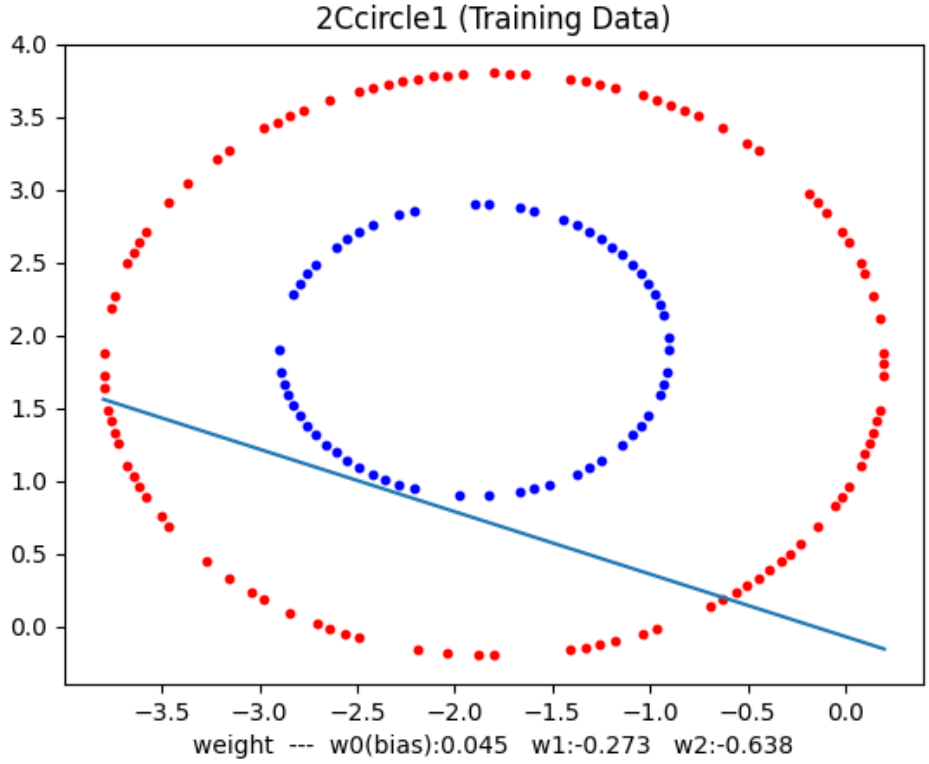
超過輸入的訓練次數就結束訓練

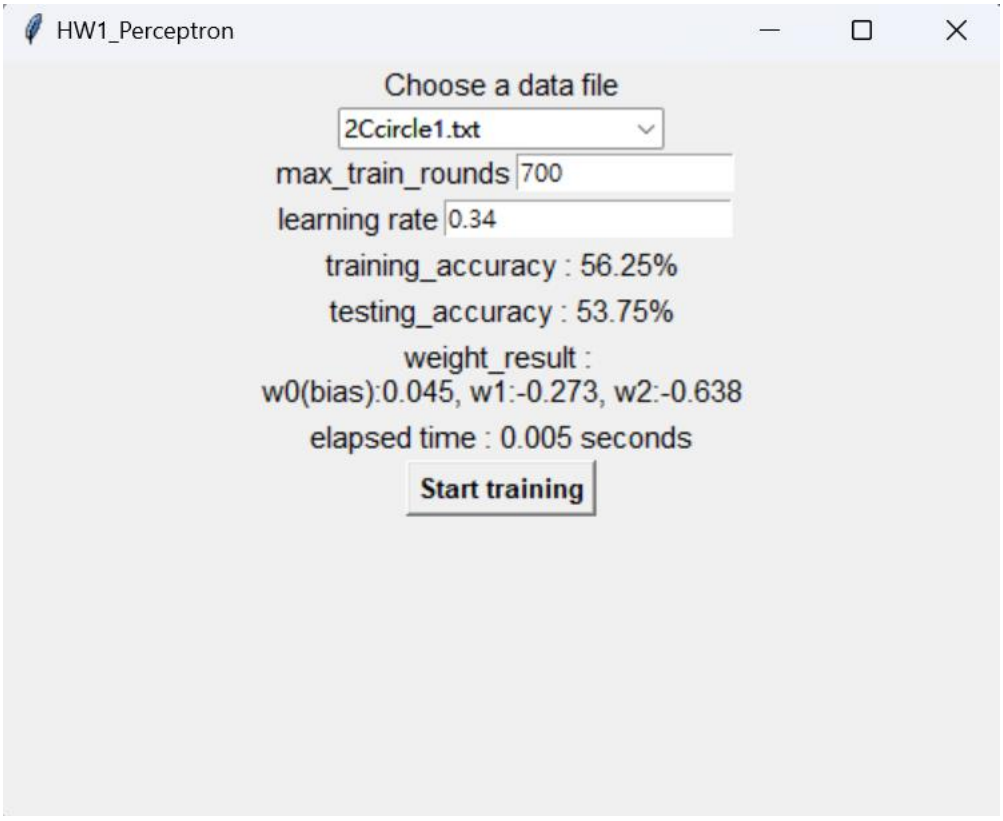
最後再分別對訓練資料及測試資料做運算的鍵結值做正確率判斷，

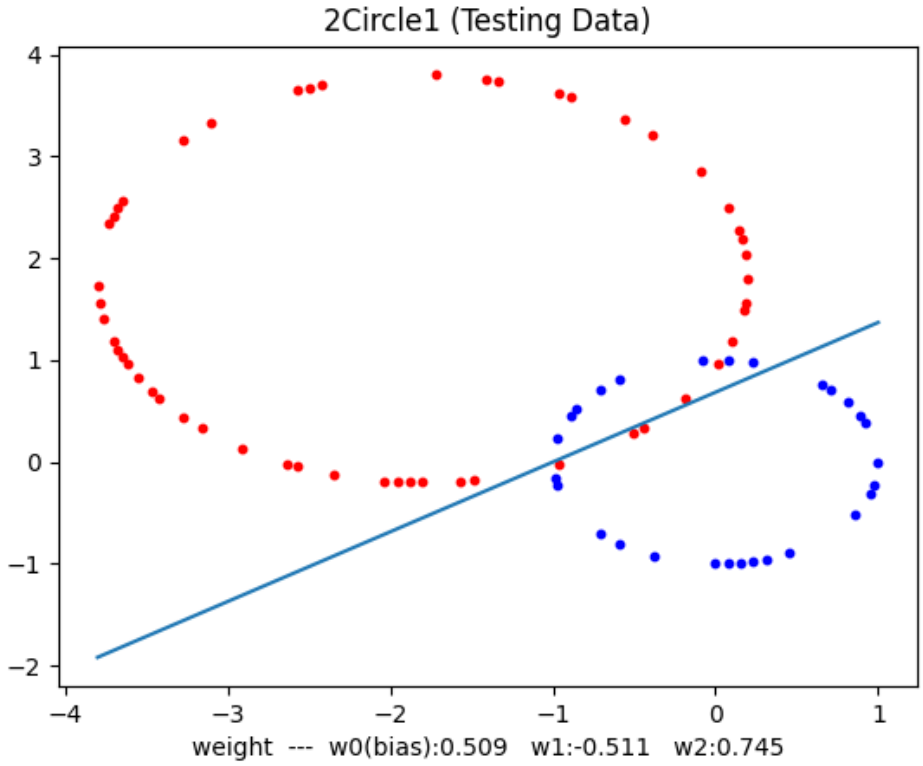
將 Train 結果透過 plot_data 畫出來並顯示在 GUI 介面上

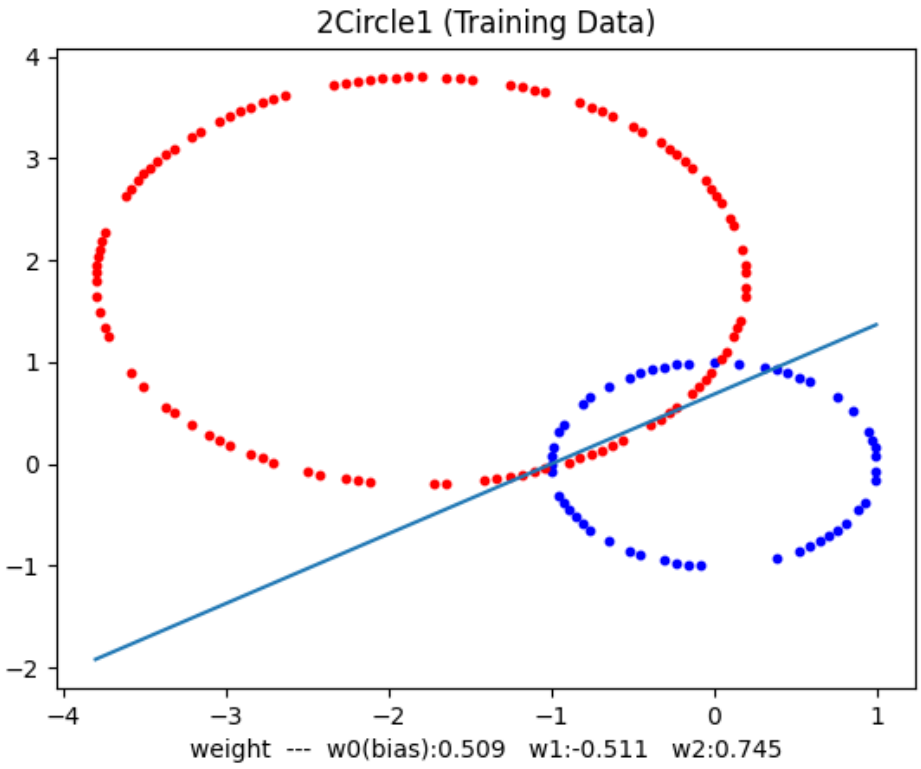
## 三、實驗結果

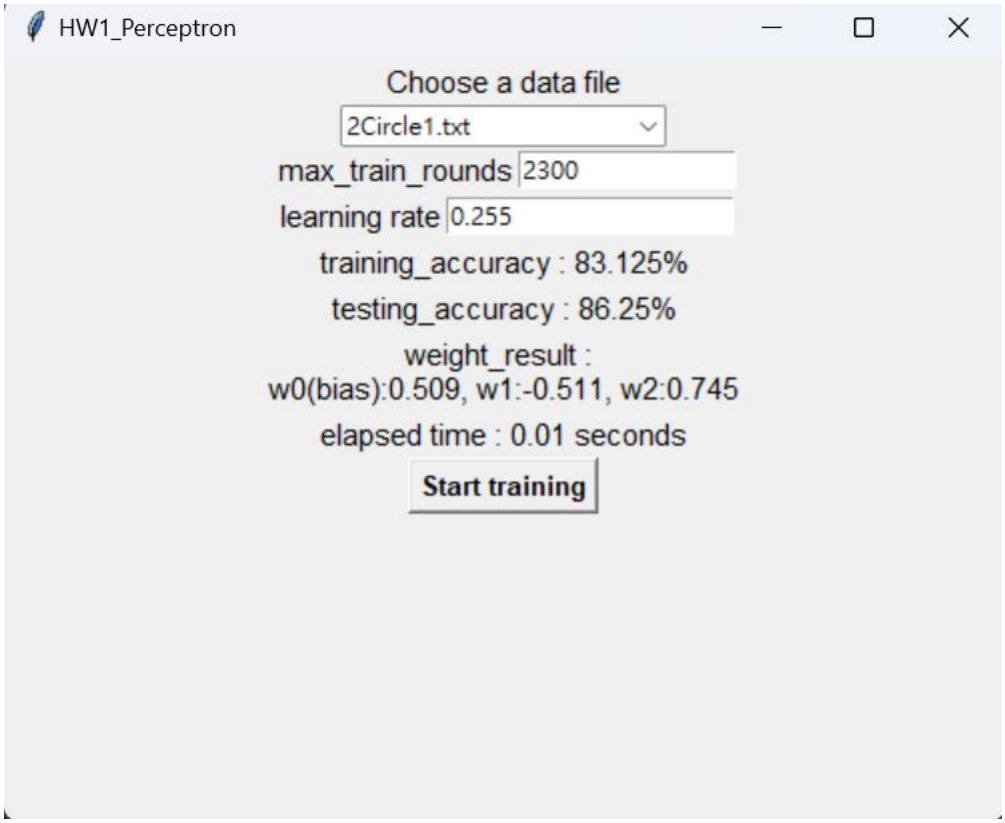

2Ccircle1 (Testing Data)
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−3.5
−3.0
−2.5
−2.0
−1.5
−1.0
−0.5
0.0
weight --- w0(bias):0.045 w1:-0.273 w2:-0.638

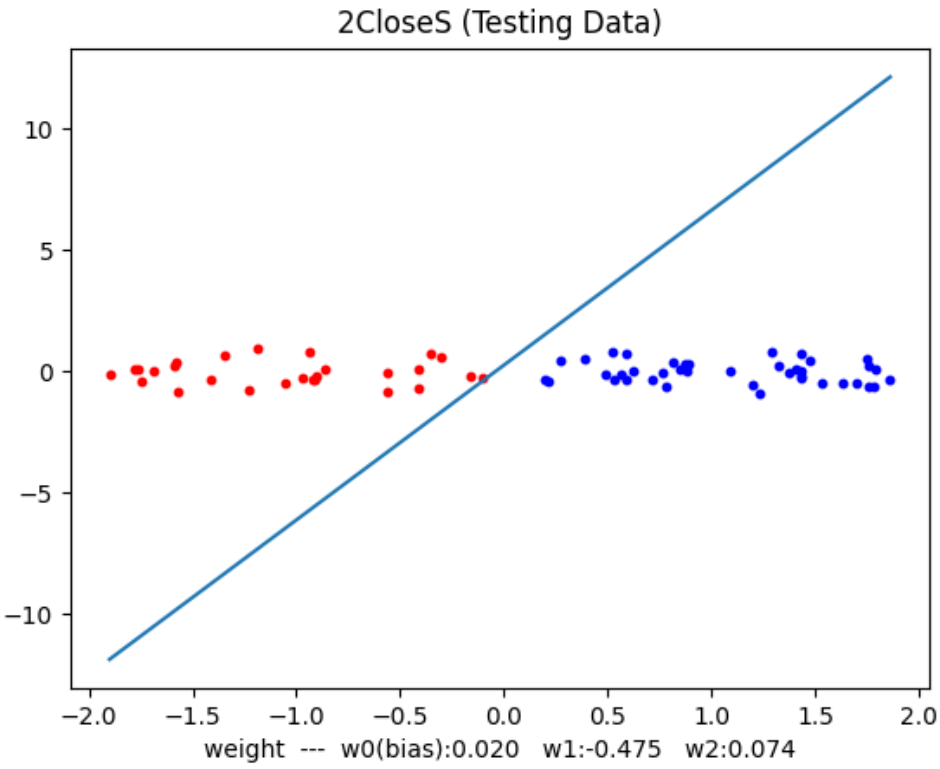

2Ccircle1 (Training Data)
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−3.5
−3.0
−2.5
−2.0
−1.5
−1.0
−0.5
0.0
weight --- w0(bias):0.045 w1:-0.273 w2:-0.638

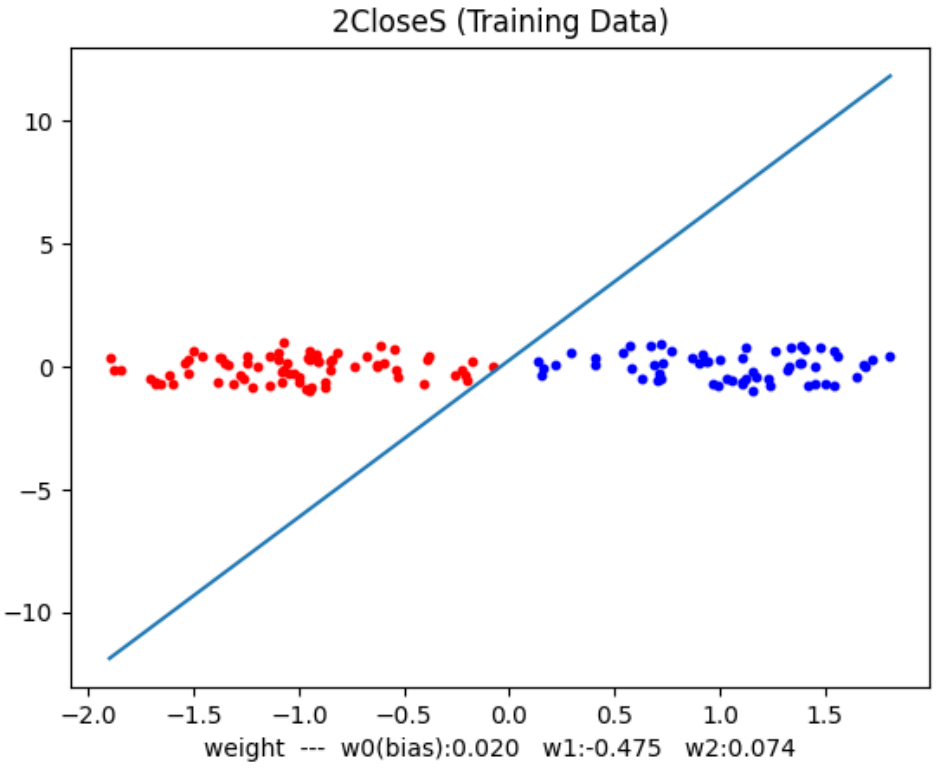

HW1_Perceptron
Choose a data file
2Ccircle1.txt
max_train_rounds 700
learning rate 0.34
training_accuracy : 56.25%
testing_accuracy : 53.75%
weight_result :
w0(bias):0.045, w1:-0.273, w2:-0.638
elapsed time : 0.005 seconds
Start training

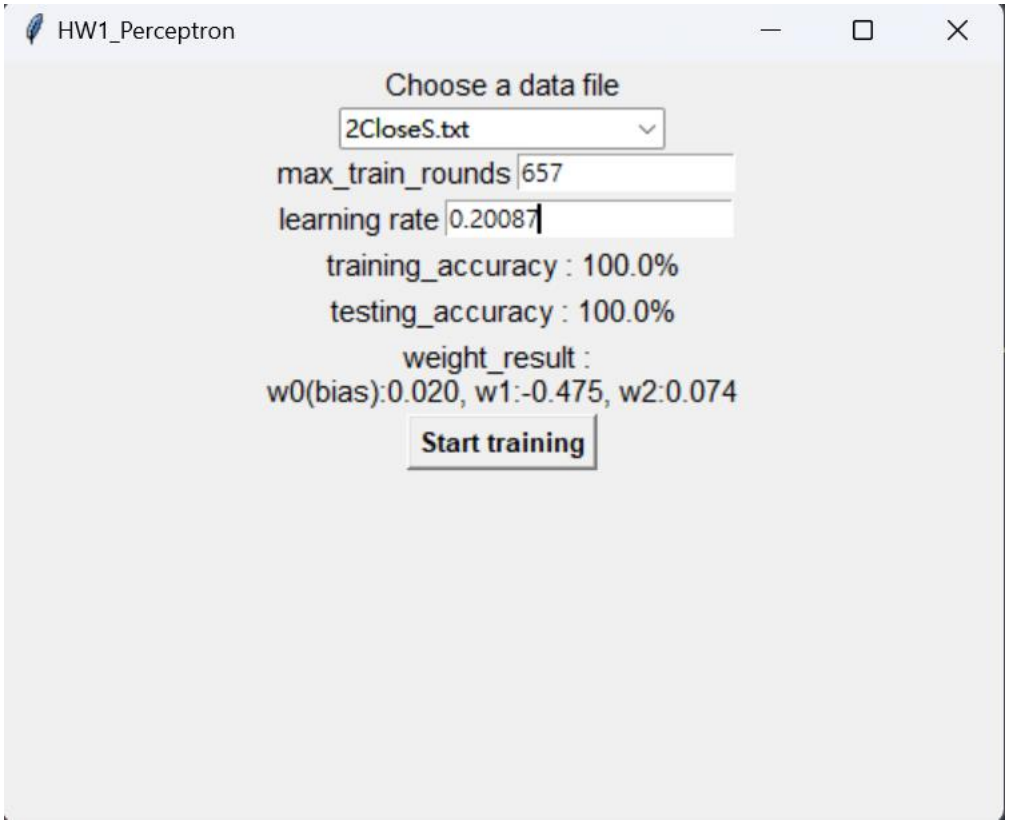

2Circle1 (Testing Data)
4
3
2
1
0
−1
−2
−4
−3
−2
−1
0
1
weight --- w0(bias):0.509 w1:-0.511 w2:0.745

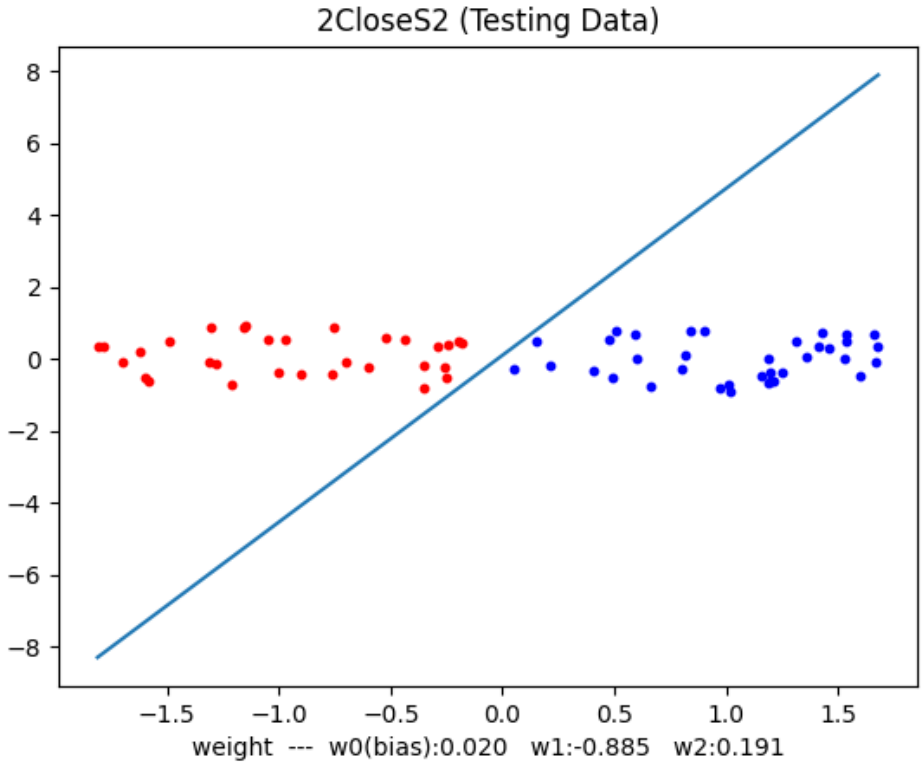

2Circle1 (Training Data)
4
3
2
1
0
−1
−2
−4
−3
−2
−1
0
1
weight --- w0(bias):0.509 w1:-0.511 w2:0.745

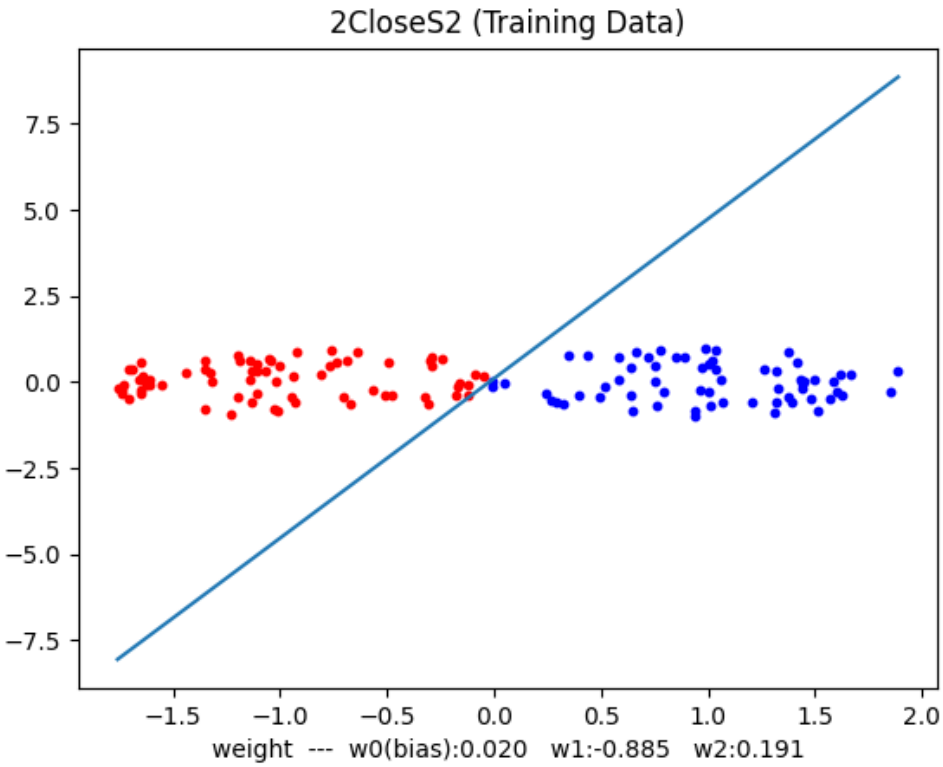

HW1_Perceptron
Choose a data file
2Circle1.txt
max_train_rounds 2300
learning rate 0.255
training_accuracy : 83.125%
testing_accuracy : 86.25%
weight_result :
w0(bias):0.509, w1:-0.511, w2:0.745
elapsed time : 0.01 seconds
Start training

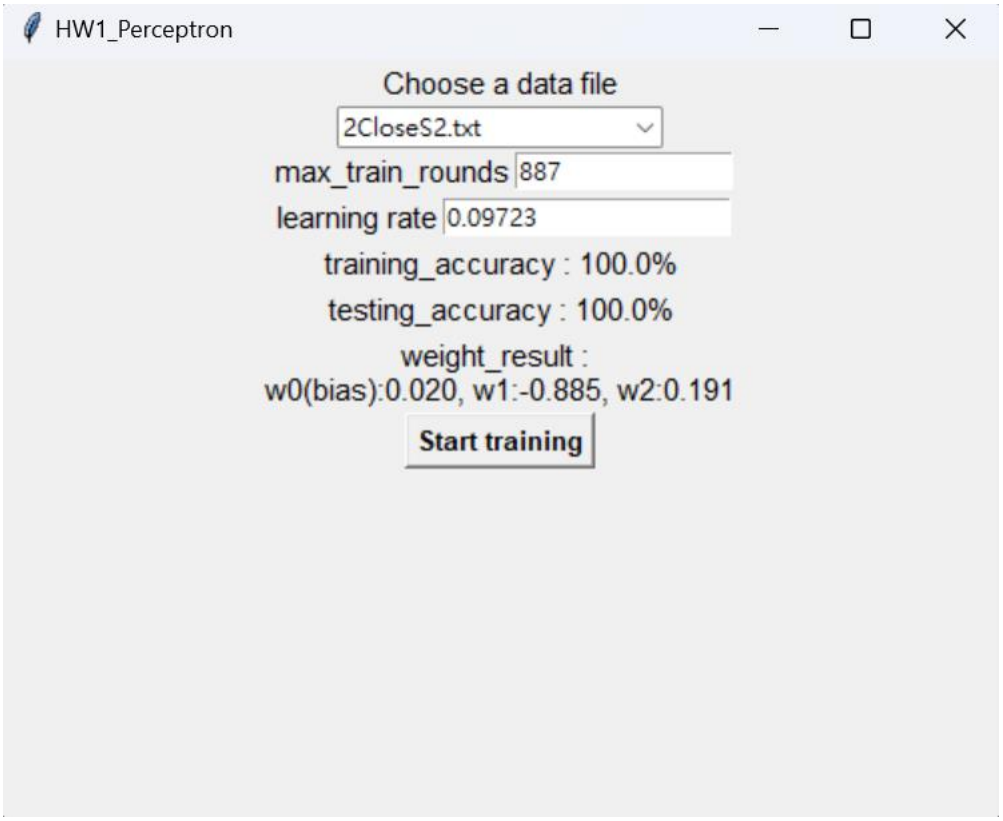

2CloseS (Testing Data)
10
5
0
−5
−10
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
weight --- w0(bias):0.020 w1:-0.475 w2:0.074

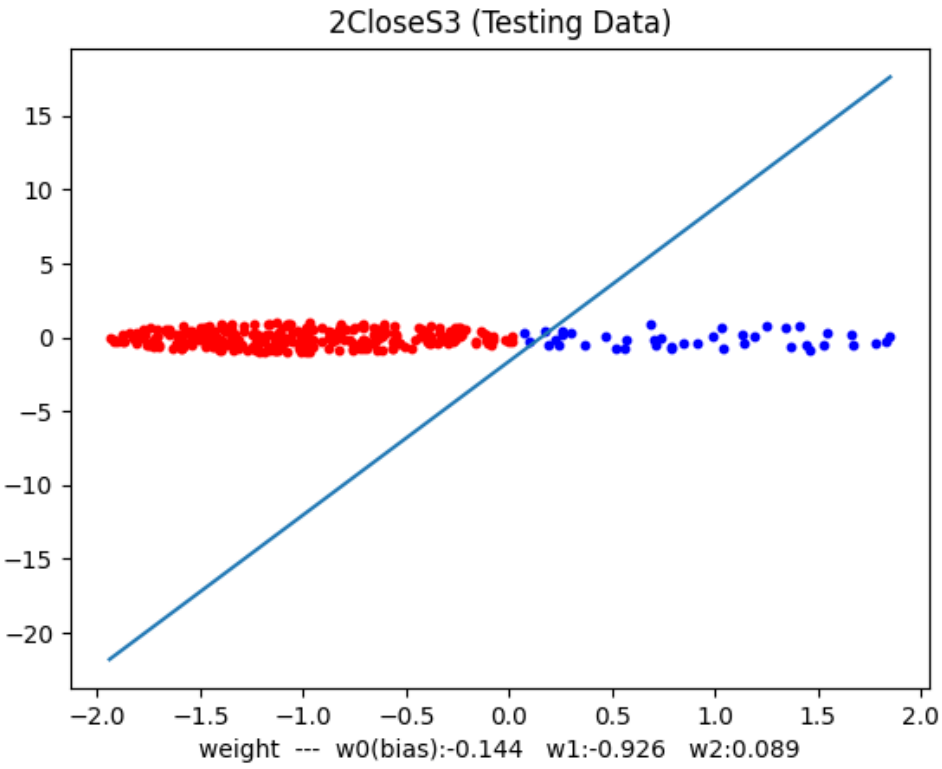

2CloseS (Training Data)
10
5
0
−5
−10
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
weight --- w0(bias):0.020 w1:-0.475 w2:0.074

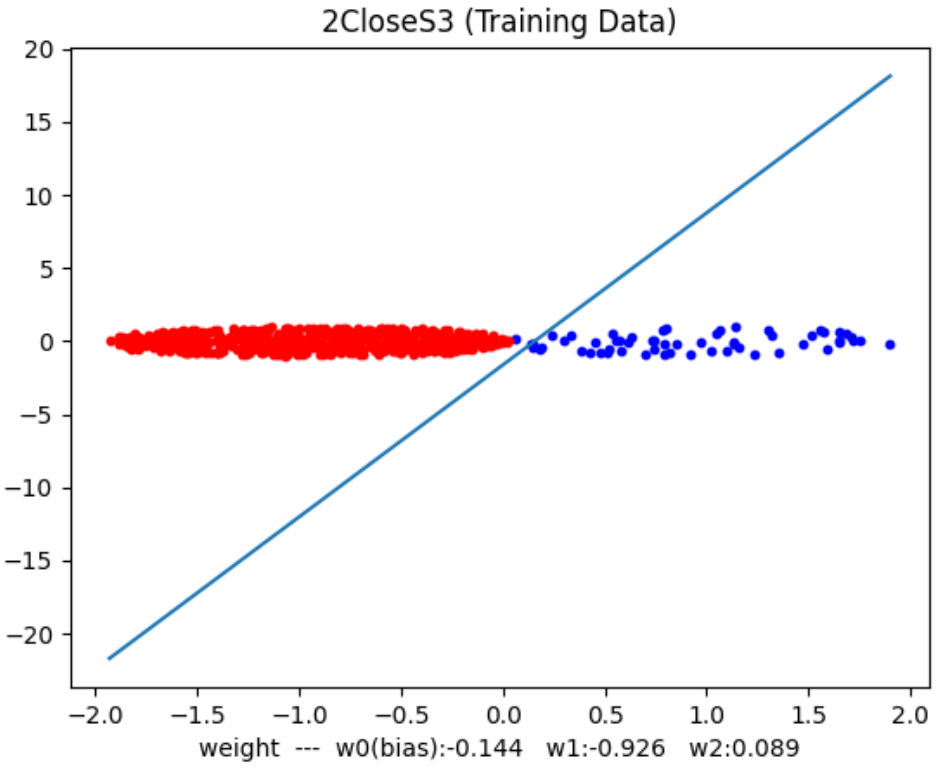

HW1_Perceptron
Choose a data file
2CloseS.txt
max_train_rounds 657
learning rate 0.20087
training_accuracy : 100.0%
testing_accuracy : 100.0%
weight_result :
w0(bias):0.020, w1:-0.475, w2:0.074
Start training

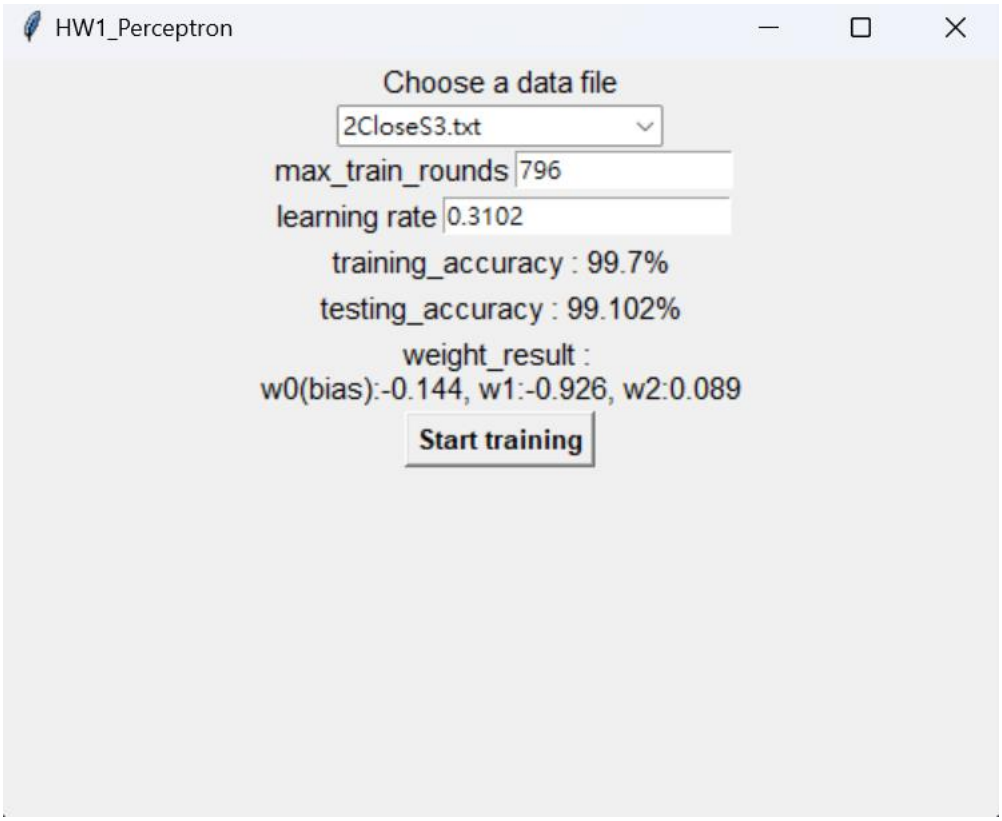

2CloseS2 (Testing Data)
8
6
4
2
0
−2
−4
−6
−8
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
weight --- w0(bias):0.020 w1:-0.885 w2:0.191

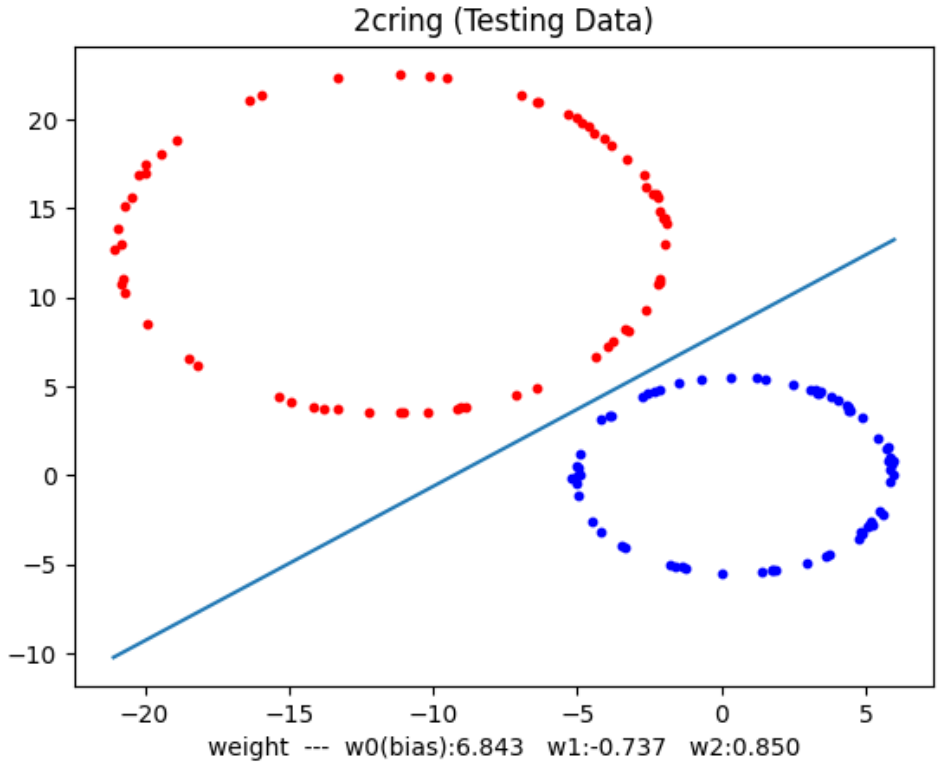

2CloseS2 (Training Data)
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
weight --- w0(bias):0.020 w1:-0.885 w2:0.191

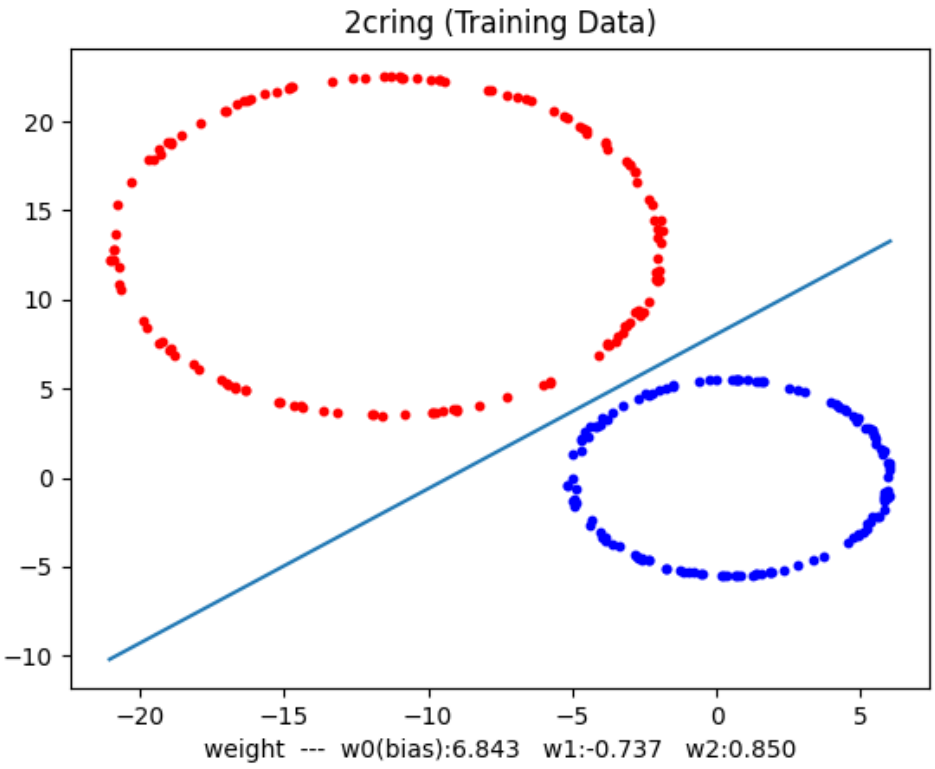

HW1_Perceptron
Choose a data file
2CloseS2.txt
max_train_rounds 887
learning rate 0.09723
training_accuracy : 100.0%
testing_accuracy : 100.0%
weight_result :
w0(bias):0.020, w1:-0.885, w2:0.191
Start training

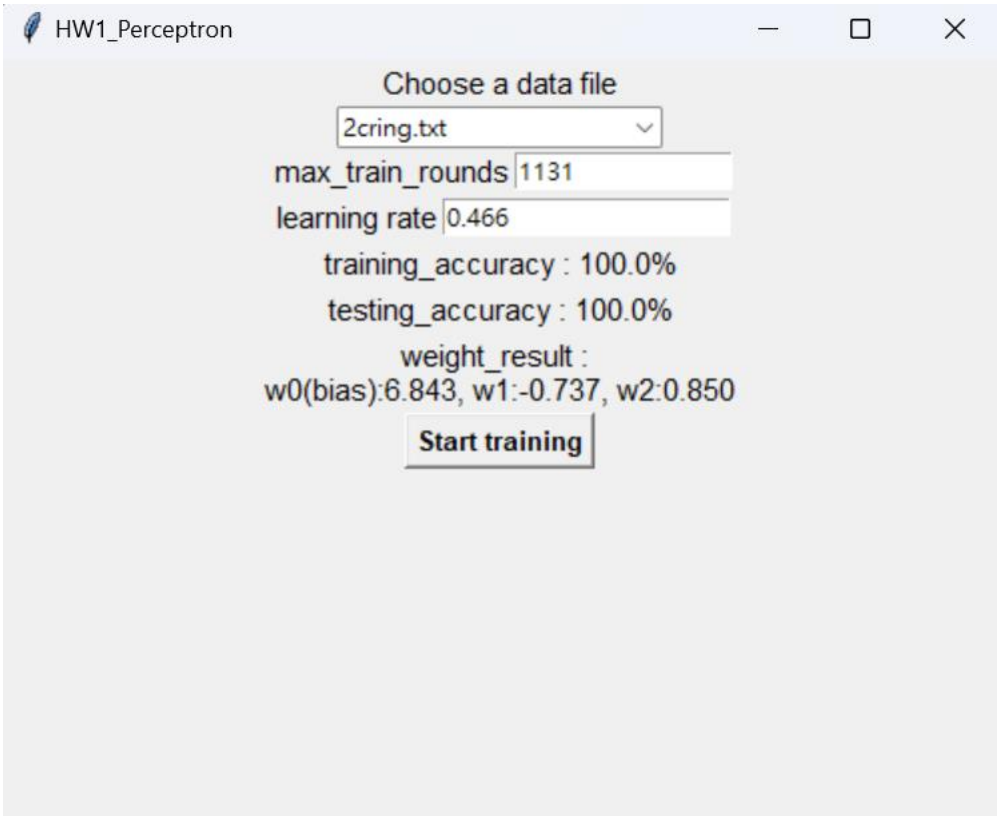

2CloseS3 (Testing Data)
15
10
5
0
−5
−10
−15
−20
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
weight --- w0(bias):-0.144 w1:-0.926 w2:0.089

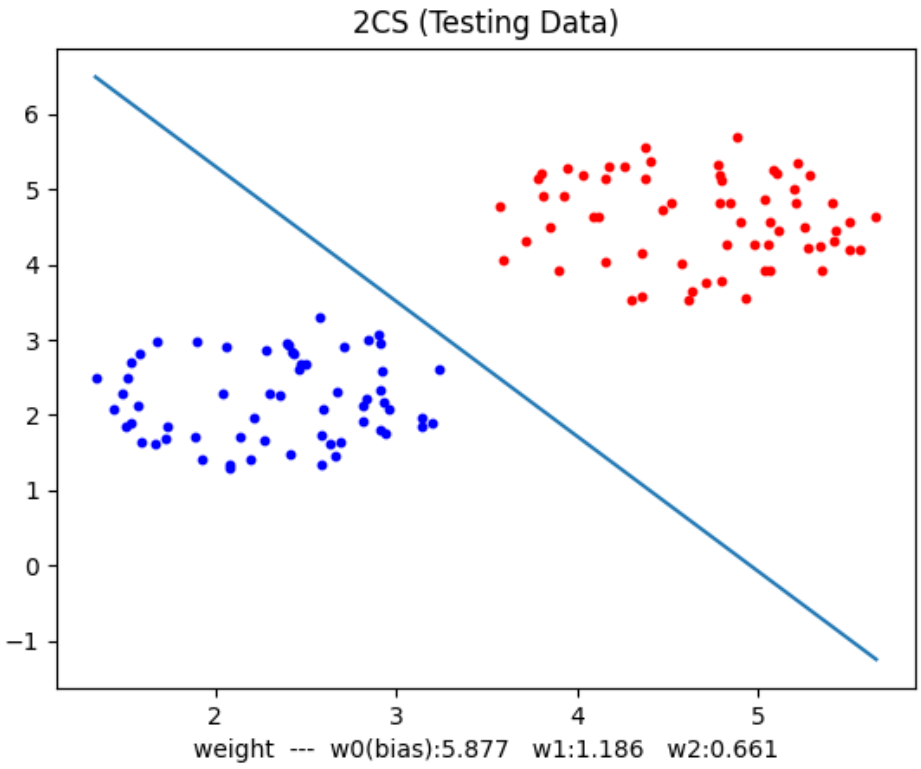

2CloseS3 (Training Data)
20
15
10
5
0
−5
−10
−15
−20
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
weight --- w0(bias):-0.144 w1:-0.926 w2:0.089


HW1_Perceptron
Choose a data file
2CloseS3.txt
max_train_rounds 796
learning rate 0.3102
training_accuracy : 99.7%
testing_accuracy : 99.102%
weight_result :
w0(bias):-0.144, w1:-0.926, w2:0.089
Start training

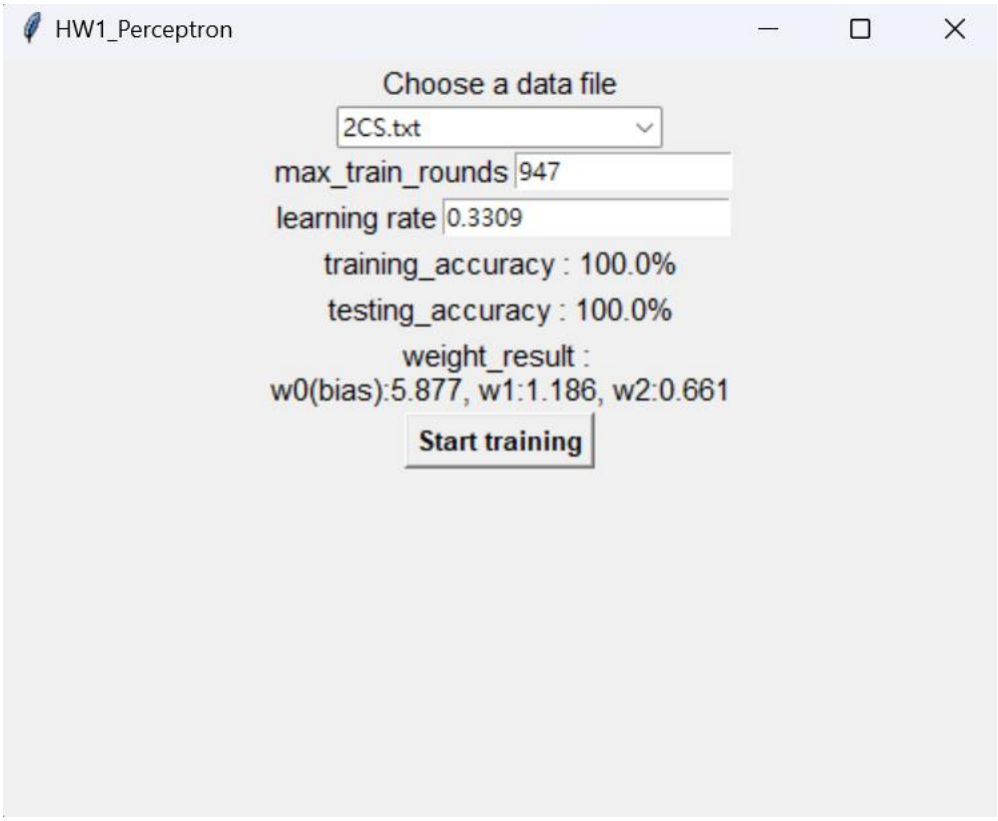

2cring (Testing Data)
20
15
10
5
0
−5
−10
−20
−15
−10
−5
0
5
weight --- w0(bias):6.843 w1:-0.737 w2:0.850

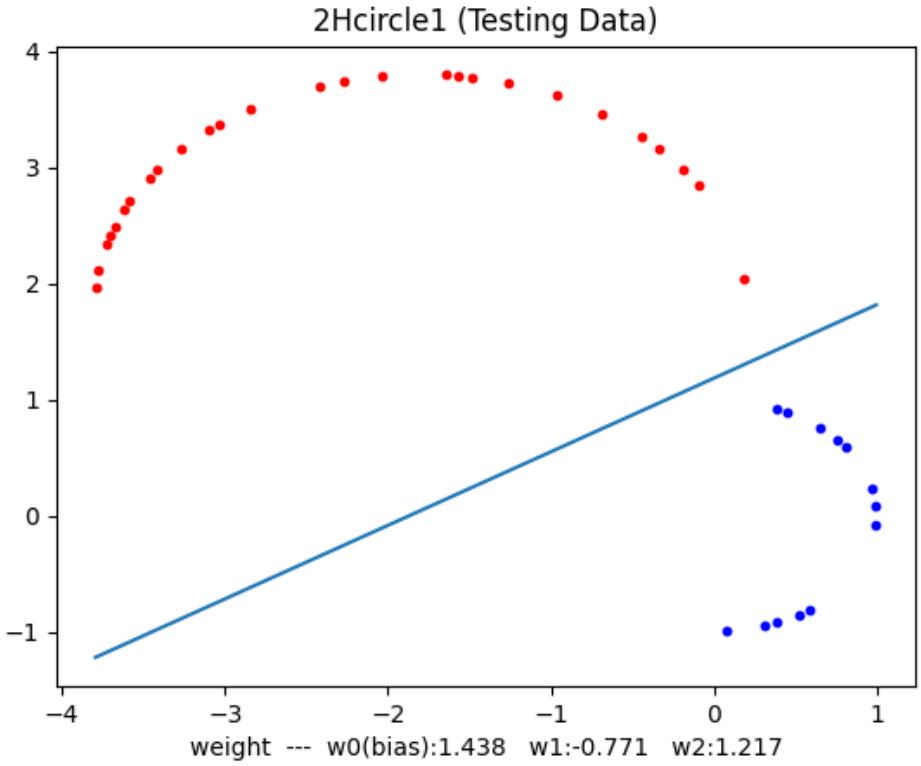

2cring (Training Data)
20
15
10
5
0
−5
−10
−20
−15
−10
−5
0
5
weight --- w0(bias):6.843 w1:-0.737 w2:0.850

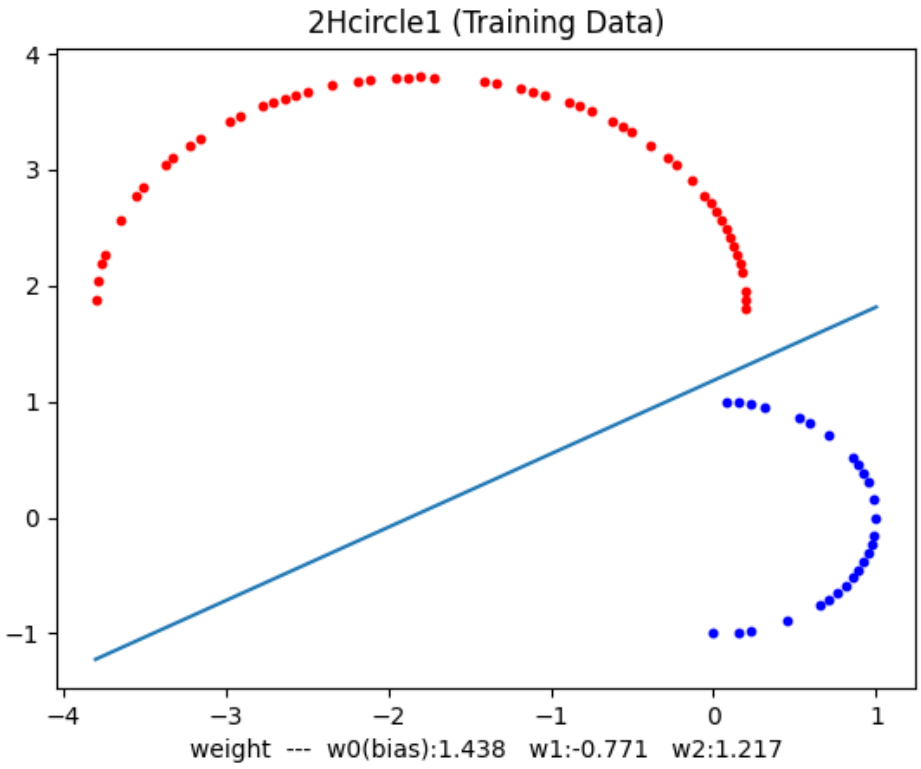

HW1_Perceptron
Choose a data file
2cring.txt
max_train_rounds 1131
learning rate 0.466
training_accuracy : 100.0%
testing_accuracy : 100.0%
weight_result :
w0(bias):6.843, w1:-0.737, w2:0.850
Start training

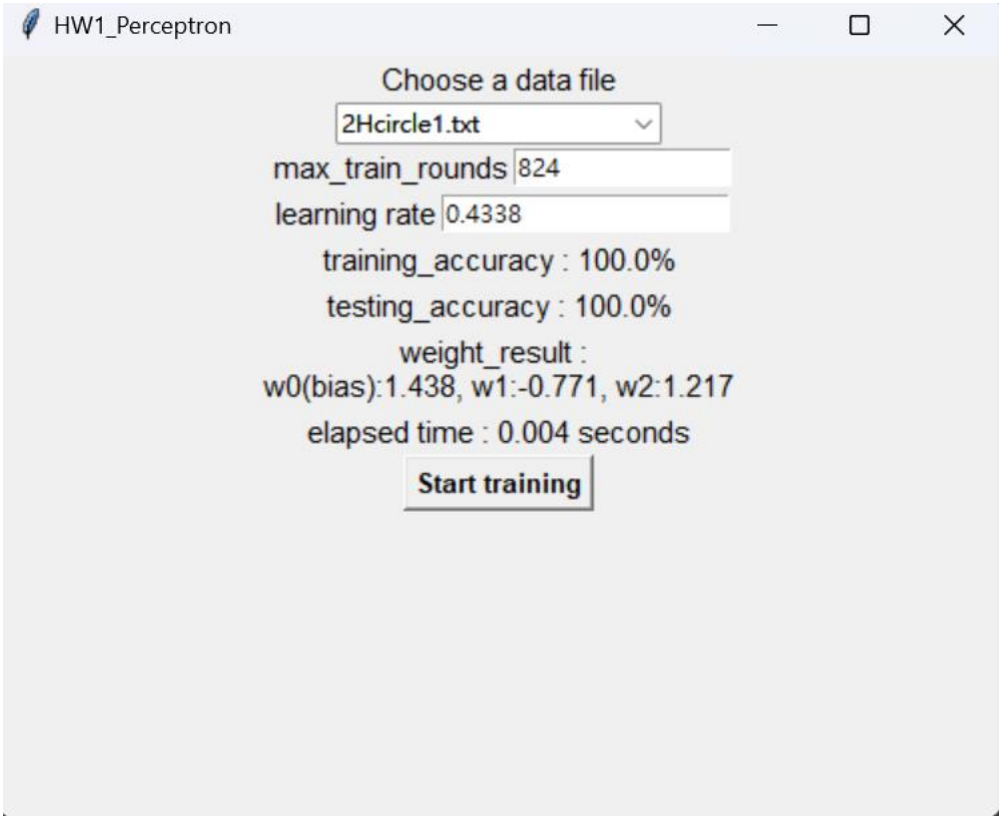

2CS (Testing Data)
6
5
4
3
2
1
0
−1
2
3
4
5
weight --- w0(bias):5.877 w1:1.186 w2:0.661

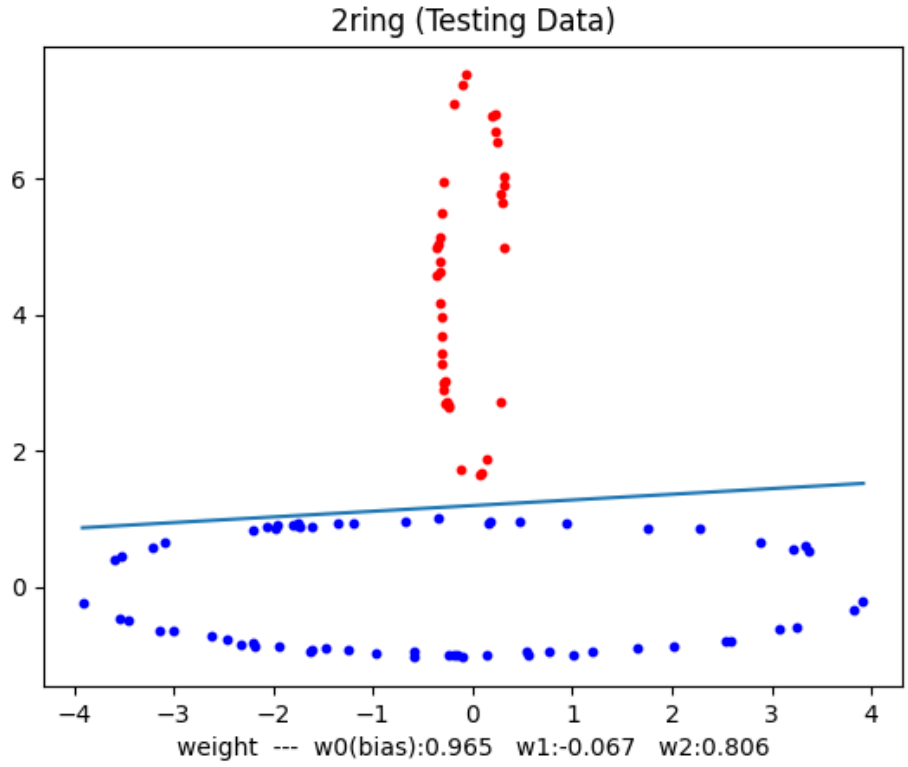

2CS (Training Data)
6
5
4
3
2
1
0
−1
2
3
4
5
weight --- w0(bias):5.877 w1:1.186 w2:0.661

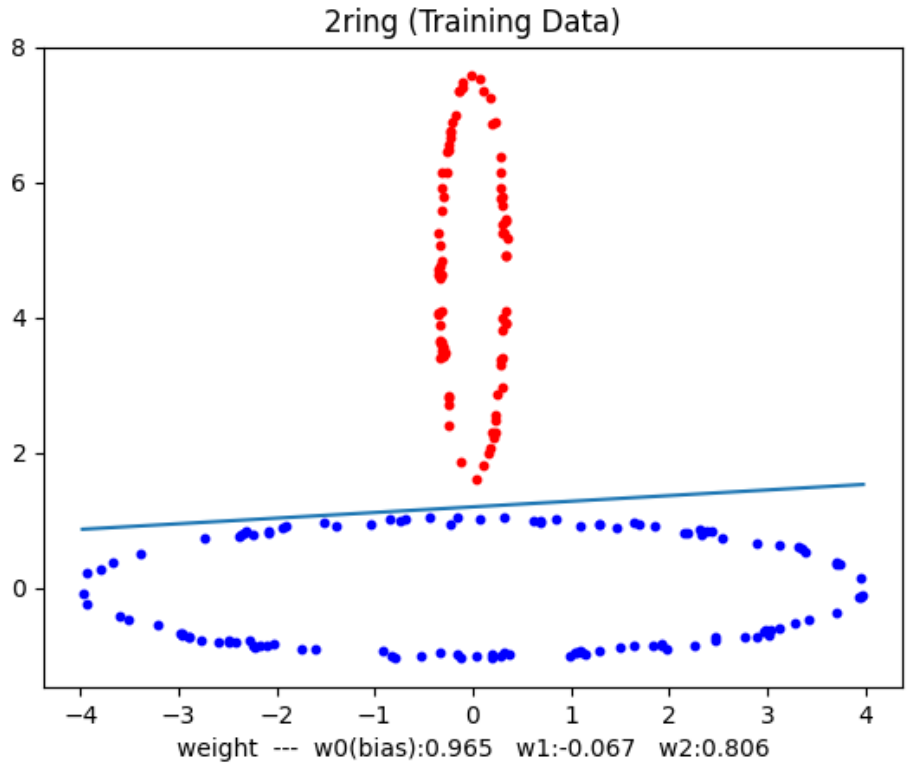

HW1_Perceptron
Choose a data file
2CS.txt
max_train_rounds 947
learning rate 0.3309
training_accuracy : 100.0%
testing_accuracy : 100.0%
weight_result :
w0(bias):5.877, w1:1.186, w2:0.661
Start training

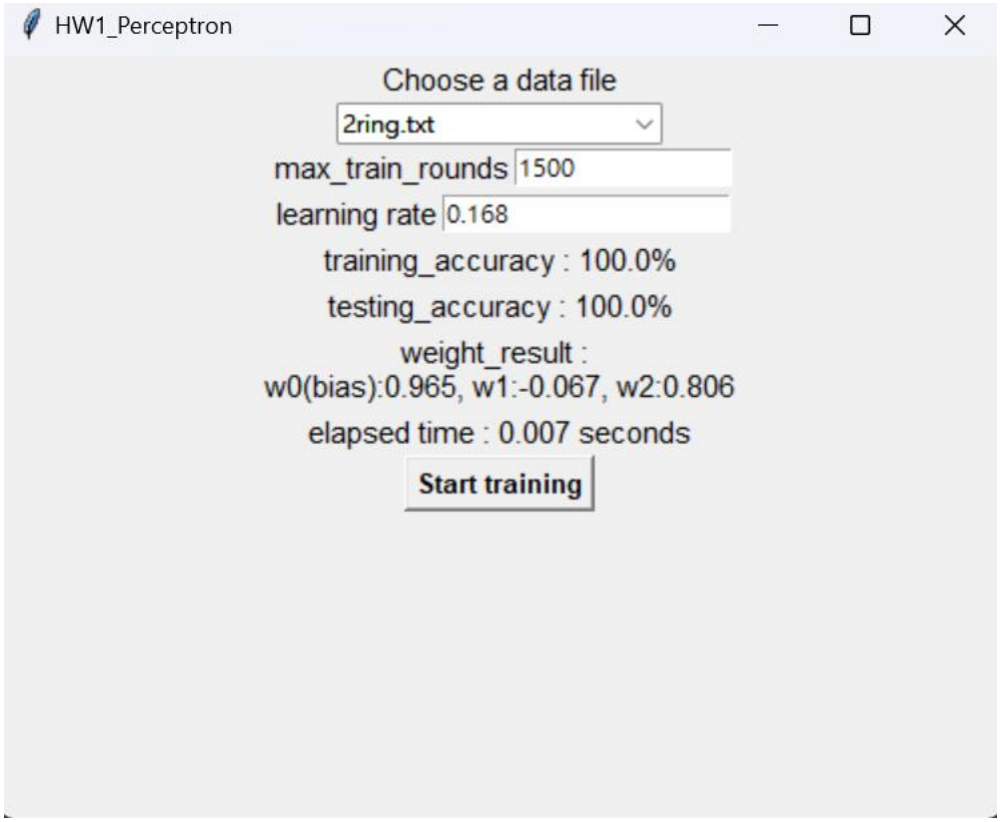

2Hcircle1 (Testing Data)
4
3
2
1
0
−1
−4
−3
−2
−1
0
1
weight --- w0(bias):1.438 w1:-0.771 w2:1.217

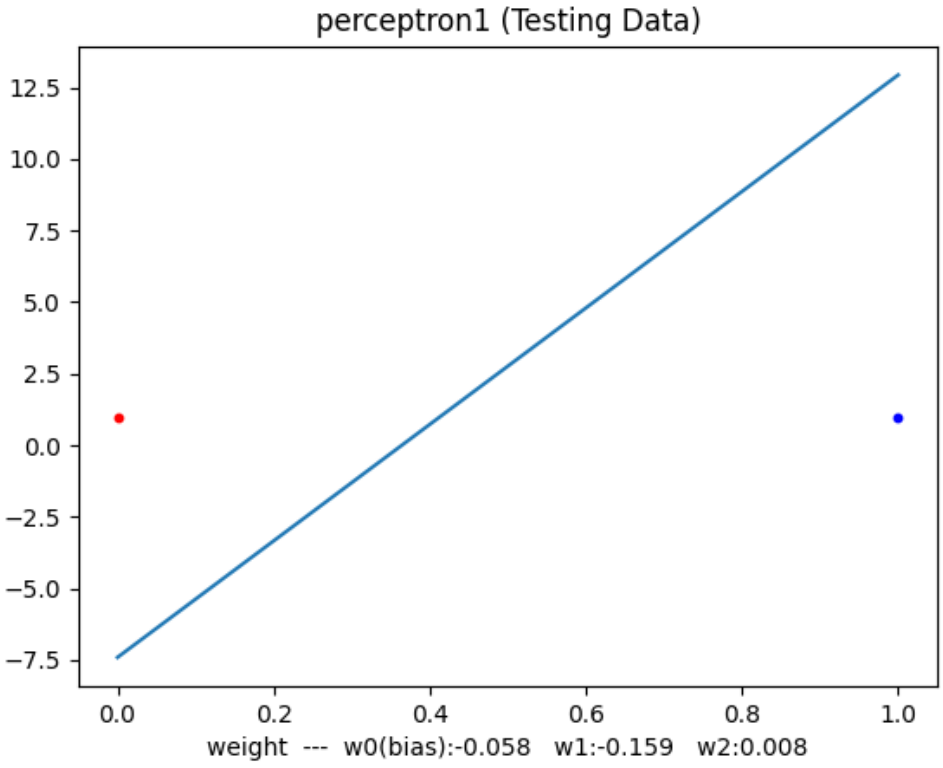

2Hcircle1 (Training Data)
4
3
2
1
0
−1
−4
−3
−2
−1
0
1
weight --- w0(bias):1.438 w1:-0.771 w2:1.217

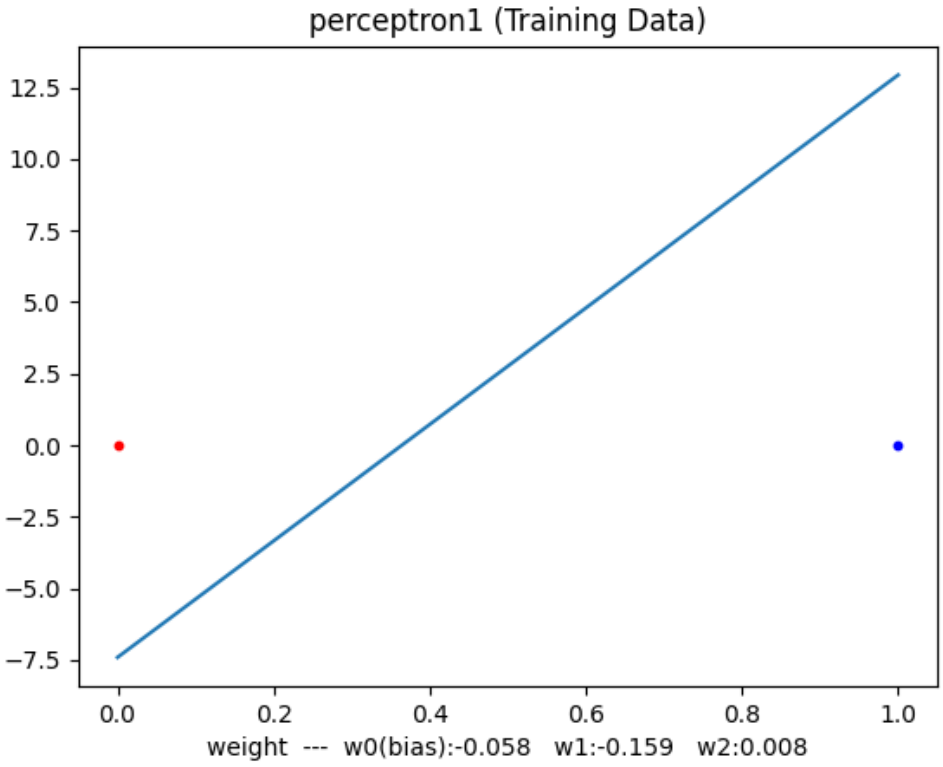

HW1_Perceptron
Choose a data file
2Hcircle1.txt
max_train_rounds 824
learning rate 0.4338
training_accuracy : 100.0%
testing_accuracy : 100.0%
weight_result :
w0(bias):1.438, w1:-0.771, w2:1.217
elapsed time : 0.004 seconds
Start training

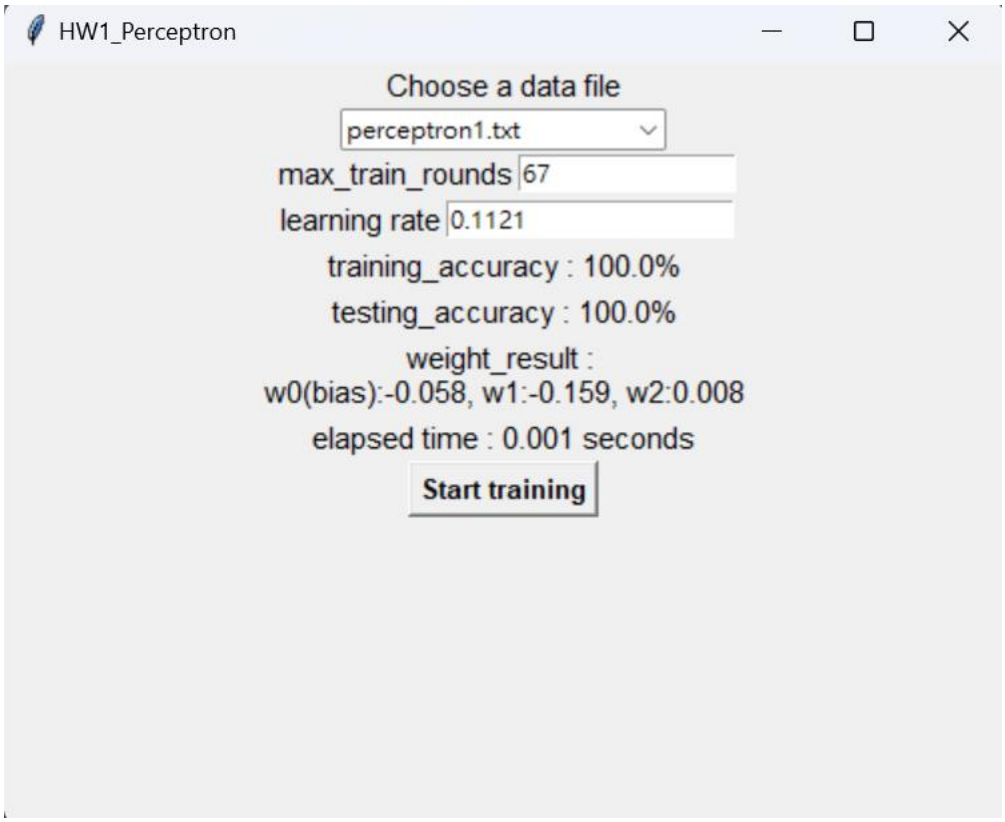

2ring (Testing Data)
6
4
2
0
−4
−3
−2
−1
0
1
2
3
4
weight --- w0(bias):0.965 w1:-0.067 w2:0.806

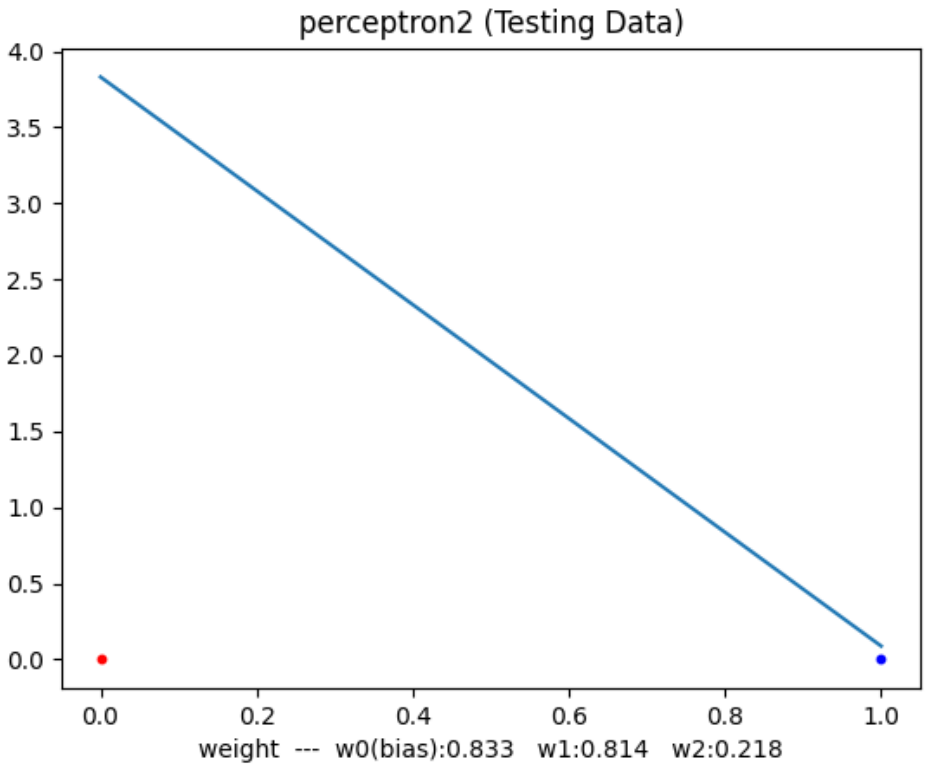

2ring (Training Data)
8
6
4
2
0
−4
−3
−2
−1
0
1
2
3
4
weight --- w0(bias):0.965 w1:-0.067 w2:0.806

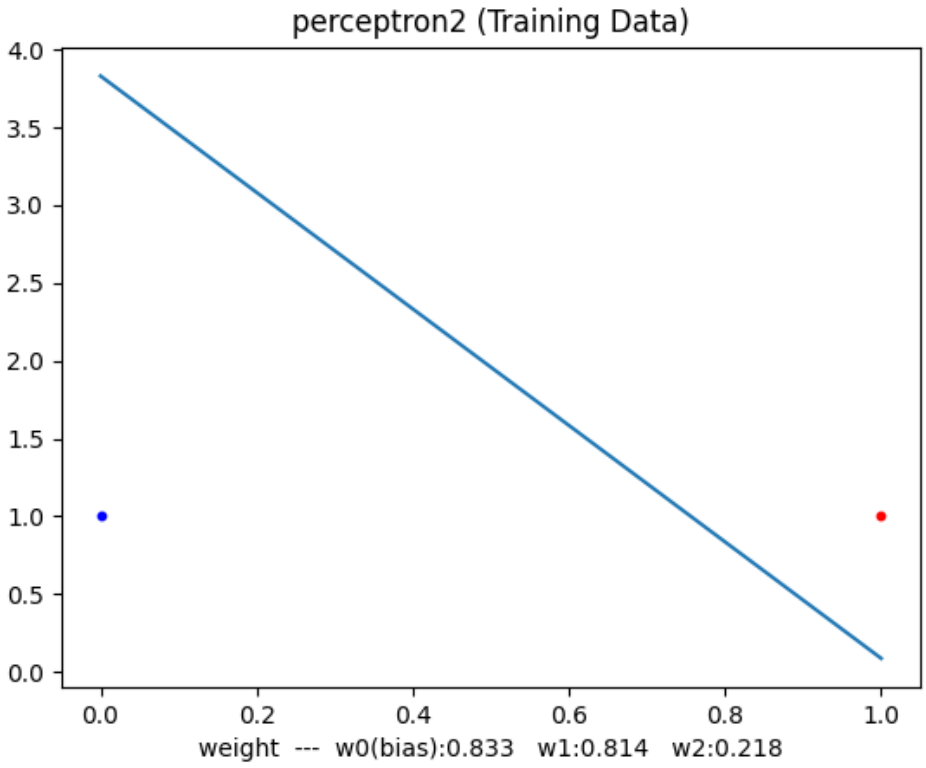

HW1_Perceptron
Choose a data file
2ring.txt
max_train_rounds 1500
learning rate 0.168
training_accuracy : 100.0%
testing_accuracy : 100.0%
weight_result :
w0(bias):0.965, w1:-0.067, w2:0.806
elapsed time : 0.007 seconds
Start training

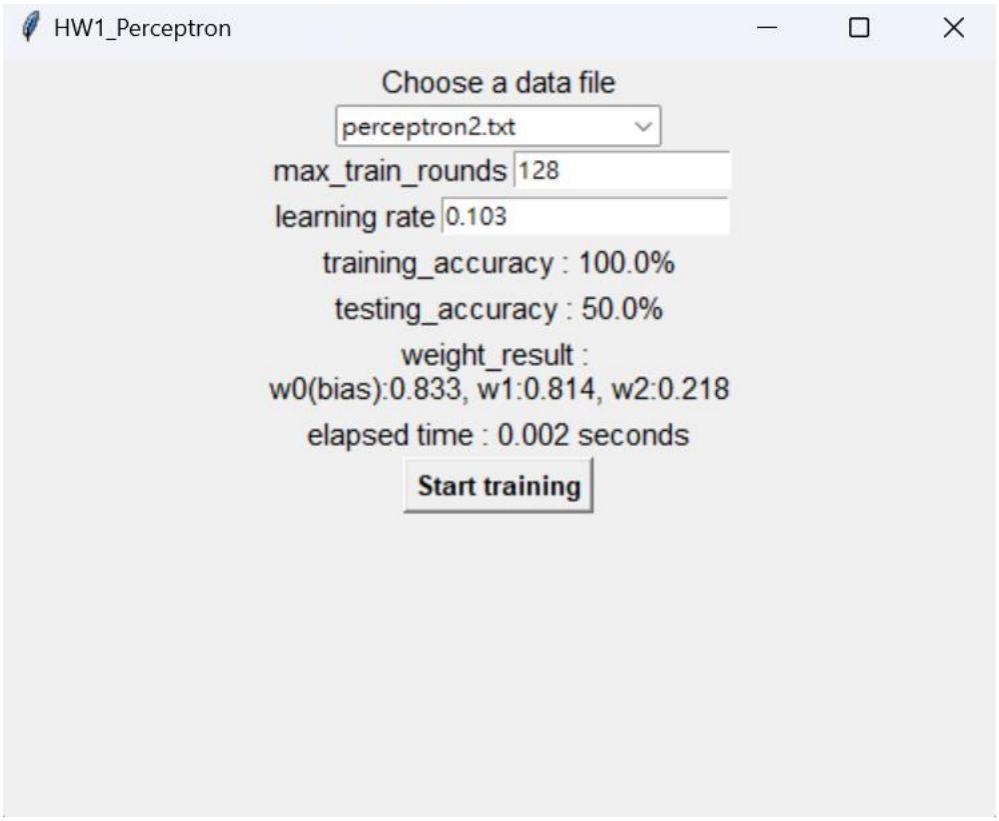

perceptron1 (Testing Data)
12.5
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
0.0
0.2
0.4
0.6
0.8
1.0
weight --- w0(bias):-0.058 w1:-0.159 w2:0.008


perceptron1 (Training Data)
12.5
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
−7.5
0.0
0.2
0.4
0.6
0.8
1.0
weight --- w0(bias):-0.058 w1:-0.159 w2:0.008


HW1_Perceptron
Choose a data file
perceptron1.txt
max_train_rounds 67
learning rate 0.1121
training_accuracy : 100.0%
testing_accuracy : 100.0%
weight_result :
w0(bias):-0.058, w1:-0.159, w2:0.008
elapsed time : 0.001 seconds
Start training


perceptron2 (Testing Data)
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
weight --- w0(bias):0.833 w1:0.814 w2:0.218


perceptron2 (Training Data)
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
weight --- w0(bias):0.833 w1:0.814 w2:0.218




## 四、實驗結果分析及討論

基本上大部分資料正確率都蠻高的，除了 2Ccircle1 和 perceptron2

2Ccircle1 因為資料點明顯不分群，所以準確率多數測試都低於 5 成

(2Ccircle1 常常執行按下去 python 出現無回應，不太清楚是不是資料點明顯

不分群，造成運算上出現狀況)

perceptron2 推測是資料點太少，所以縱使拉高訓練次數，測試資料結果最高

只有 50%，也可能是學習率設定不佳所導致，在實作上仍有改進空間

(perceptron1 和 2 訓練次數拉太高或學習率拉高常常做圖出現缺漏，有時候卻

又正常，猜測樣本空間太少也許也對作圖有影響)

此外，原先我預測訓練次數拉到 1000 以上必然會大幅提高正確率

但根據手動測試結果，發現在幾百到 1000 初頭範圍，影響較大的還是學習率通常超過 0.4 正確率就會有點往下掉(當然，跟資料點離散程度也有關)

大三以前完全沒有學過 python 資料分析、機器學習、GUI 模組，這次花了非常多時間去學習、參考資料，雖然實驗結果無法每項都達 100%，但整體而言，數據呈現還算令人滿意!

* 有些圖示沒有標示 elapsed_time，是因為這是中途測試時所加的功能但發現訓練時間基本上跟訓練次數最相關，while loop 跑調整鍵結值的部分對時間影響並不大，因此，就沒重新將已經截圖好的幾筆數據打掉重練