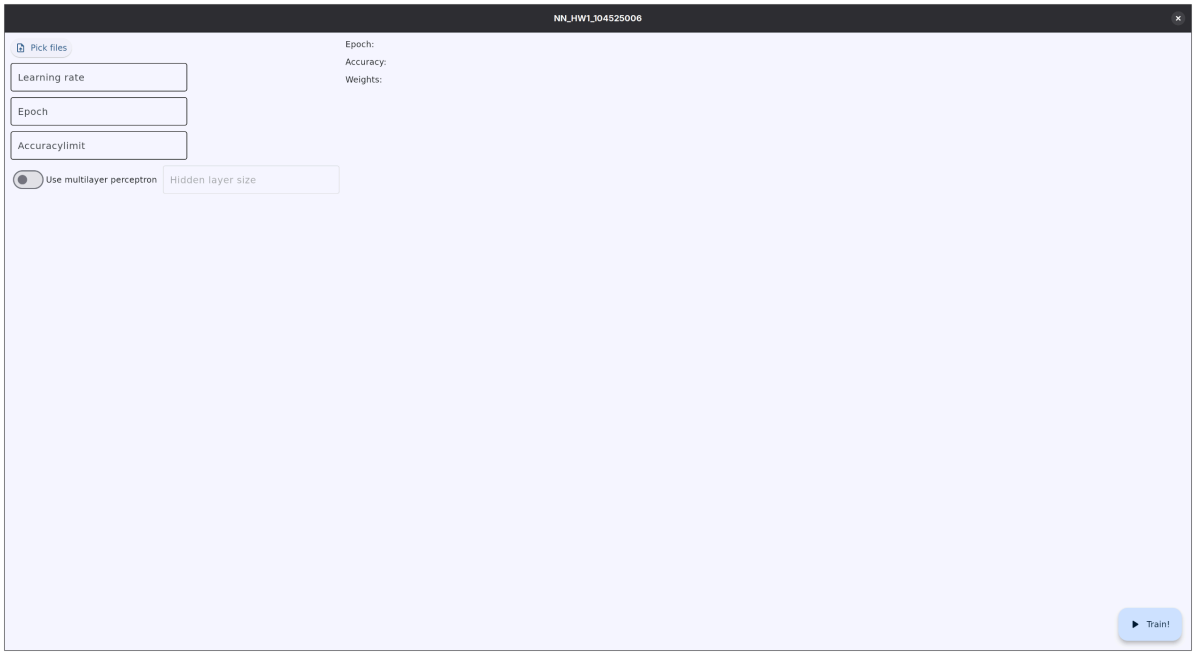
一 加分題完成功能

- [x] 三維資料圖形顯示介面
- [x] 能夠處理多維資料(四維以上)
- [ ] 數字辨識
- [ ] 可辨識兩群以上的資料

二 程式執行說明 (GUI 功能說明)



- 左側顯示可調整參數
  - 選擇 Data(選中後在右側顯示檔名)
  - 學習率
  - 執行 Epoch 數
  - 準確率限制(達到設定值後停止訓練)
  - 選擇是否使用多層感知機
    - 設定隱藏層大小
- 右側顯示訓練結果
  - 訓練的 Epoch 數
  - Test 資料的準確率
  - 鍵結值
- 右下方為訓練按鈕, 若為 2 或 3 維資料, 會將結果圖彈出顯示

三 程式碼簡介

三.1 單層感知機

```
for row in train_data:
  input = np.array([-1, *row[:-1]])
  if weight @ input < 0 and row[-1] == 1:
    weight = weight + learning_rate * input
  elif weight @ input > 0 and row[-1] == 0:
    weight = weight - learning_rate * input
```

計算網路輸出值: $y(n) = \varphi[w^T(n)x(n)]$

$\varphi(v) = \begin{cases} +1 \text{ if } v \geq 0 \\ -1 \text{ if } v < 0 \end{cases}$

調整鍵結值向量: $w(n+1) = \begin{cases} w(n) \text{ if 分類正確} \\ w(n) + \eta x(n) \text{ if } y(n) < 0 \\ w(n) - \eta x(n) \text{ if } y(n) \geq 0 \end{cases}$

三.2 多層感知機

三.2.a 前饋

```
y = [row[:-1]]
  for weight in weights:
    y.append(lib.sigmoid(weight @ [-1, *y[-1]]))
```

計算網路輸出值: $y(n) = \varphi[w^T(n)x(n)]$

$\varphi(v) = \begin{cases} +1 \text{ if } v \geq 0 \\ -1 \text{ if } v < 0 \end{cases}$

三.2.b 倒傳遞

```
def delta_final(prediction, output):
  return (output - prediction) * prediction * (1 - prediction)
```

$\delta_j = e_j(n)\varphi'\big(v_j(n)\big) = \big(d_{j(n)} - O_j(n)\big)O_{j(n)}\big(1 - O_j(n)\big)$

```
for layer in range(len(layer_size)-2, 0, -1):
  delta = [y[layer] * (1 - y[layer]) * (weights[layer].T[1:] @ delta[0]), *delta]
```

$\delta_{j(n)} = \varphi'\big(v_{j(n)}\big)\sum_k \delta_k(n)w_{kj}(n) = y_j(n)\big(1 - y_j(n)\big)\sum_k \delta_{k(n)}w_{kj}(n)$

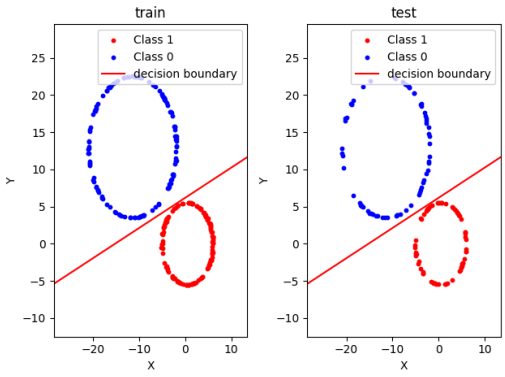
三.2.c 調整鍵結值

```
for layer in range(len(weights)):
  weights[layer] += learning_rate * np.outer(delta[layer], [-1, *y[layer]])
```

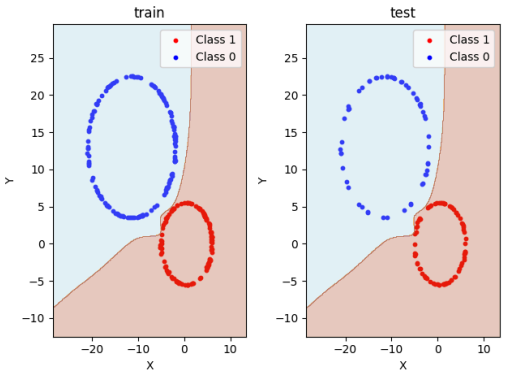
$w_{ji} = w_{ji} + \Delta w_{ji} = w_{ji} + \eta \times \delta_j(n) \times y_i(n)$

四 實驗結果

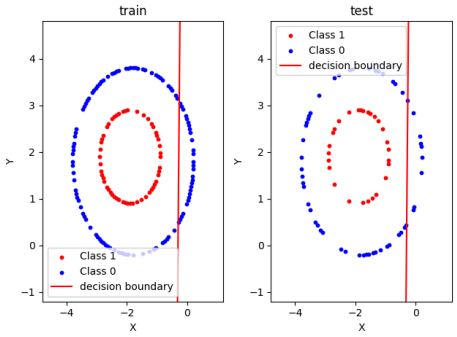
四.1 基本題



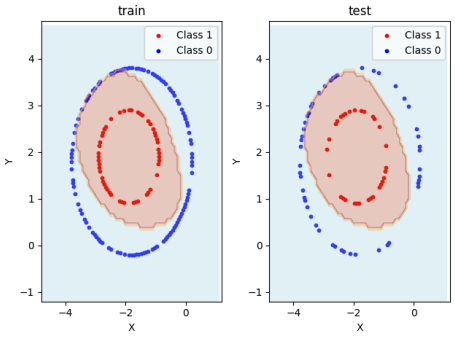
2cring



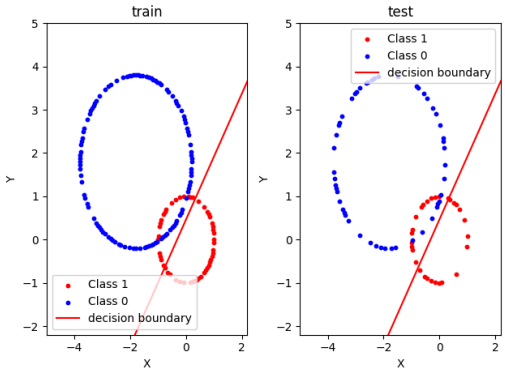
2cring (mlp)



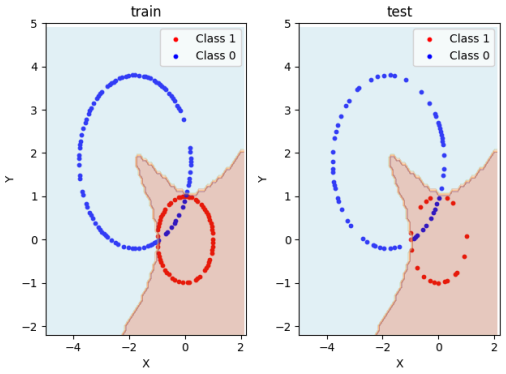
2Ccircle1



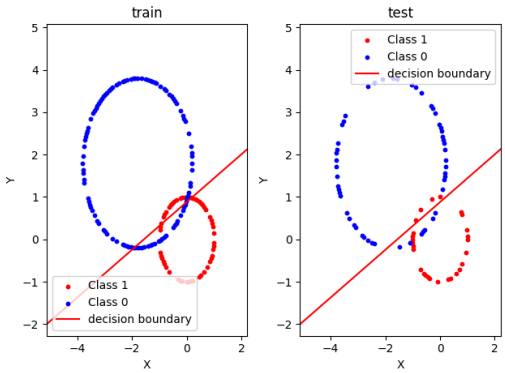
2Ccircle1 (mlp)



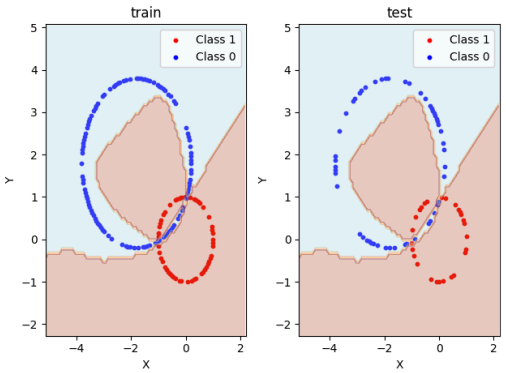
2Circle1



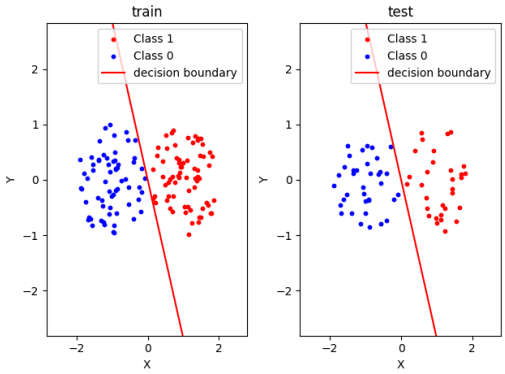
2Circle1 (mlp)



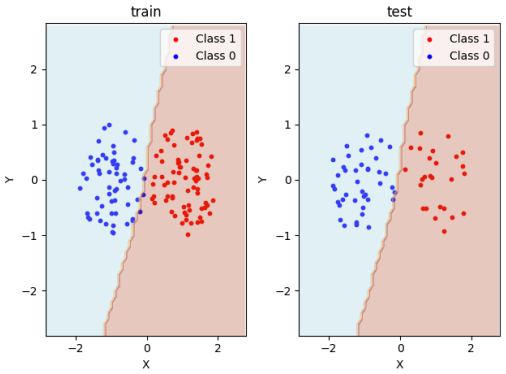
2Circle2



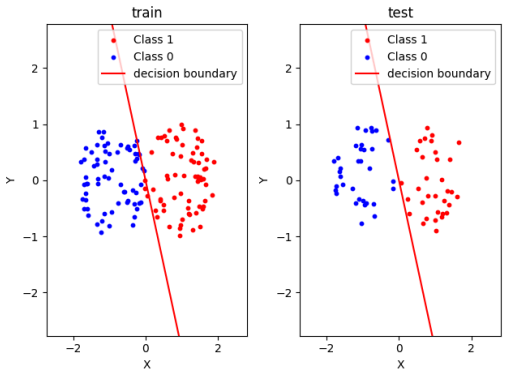
2Circle2 (mlp)



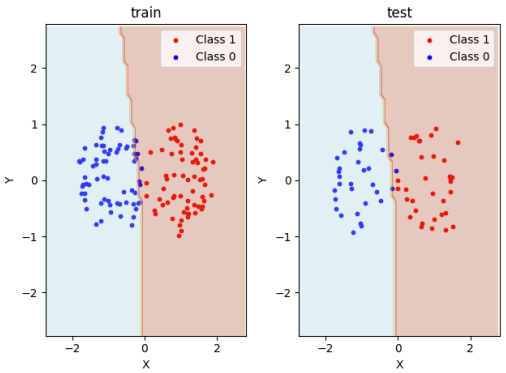
2CloseS



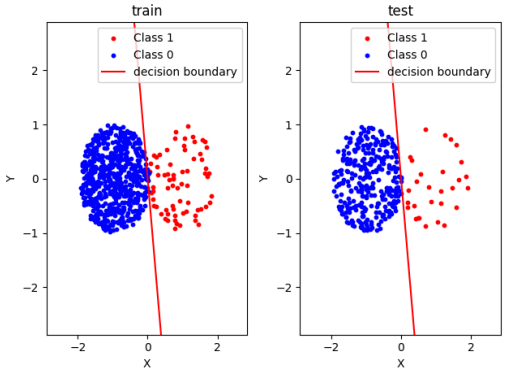
2CloseS (mlp)



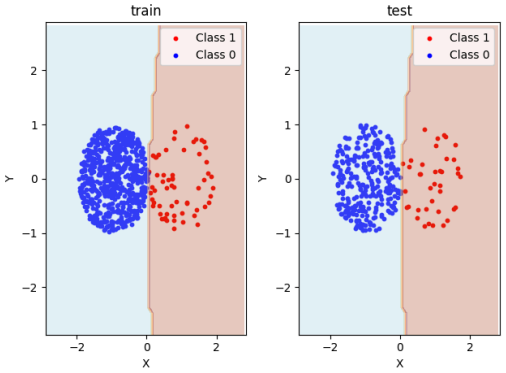
2CloseS2



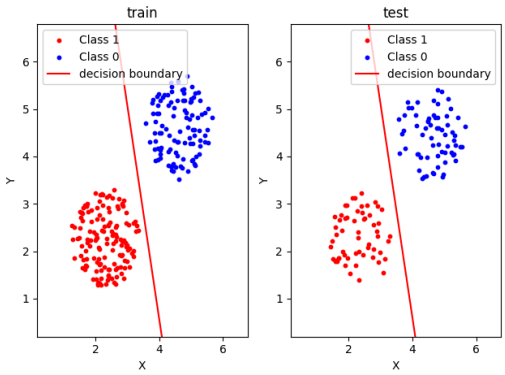
2CloseS2 (mlp)



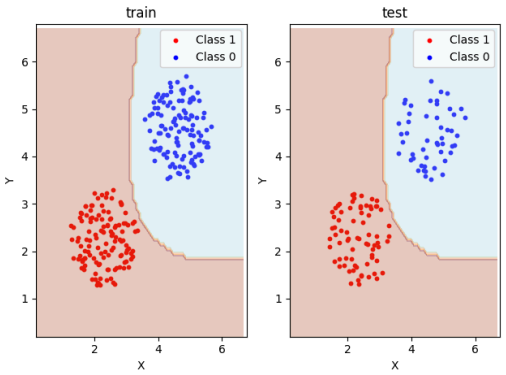
2CloseS3



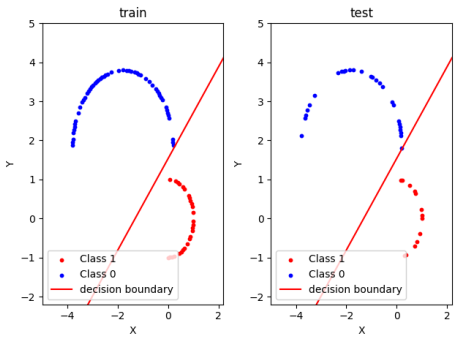
2CloseS3 (mlp)

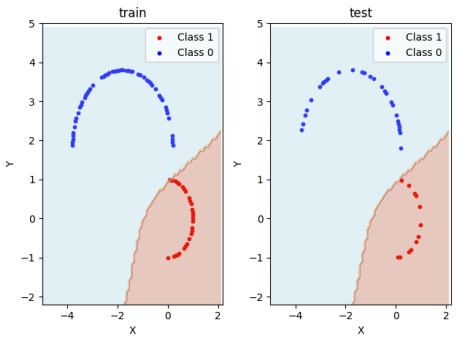


2CS

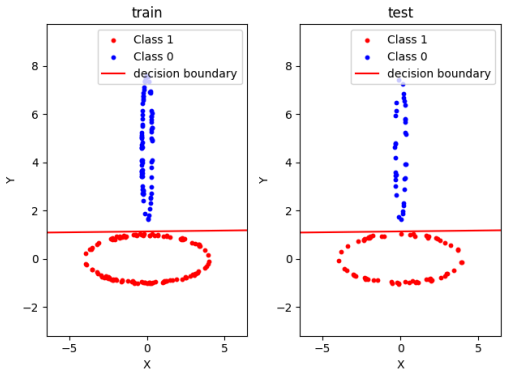


2CS (mlp)

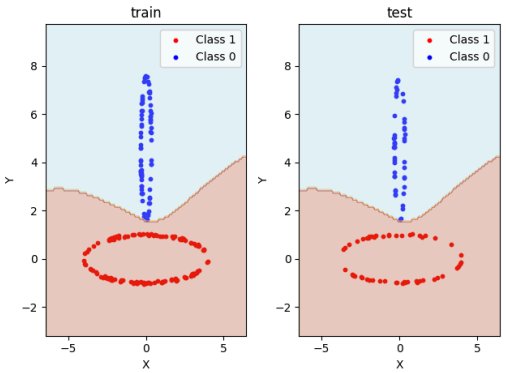




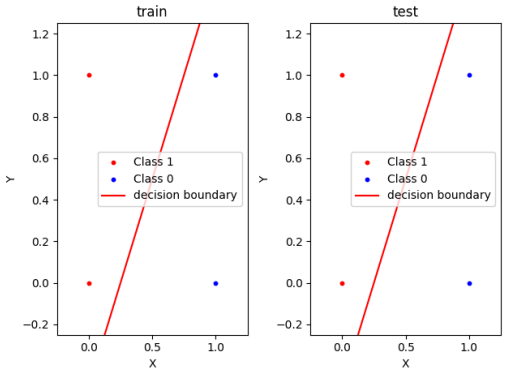
2Hcircle1



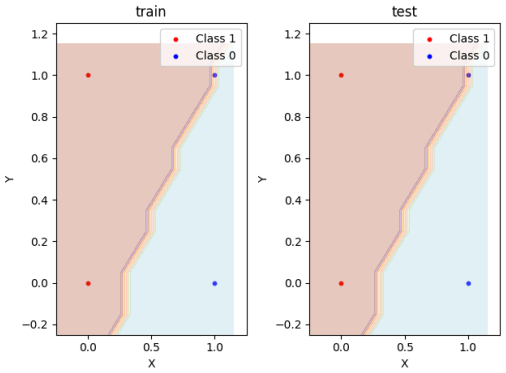
2Hcircle1 (mlp)



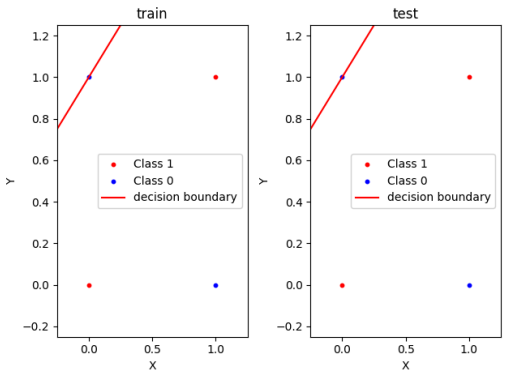
2ring



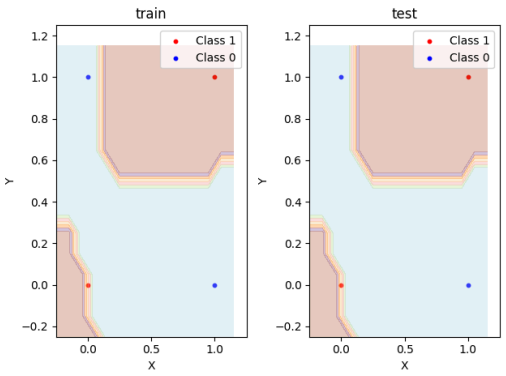
2ring (mlp)



perceptron1



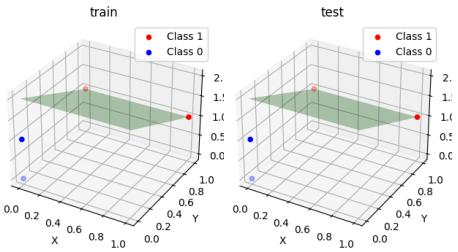
perceptron1 (mlp)



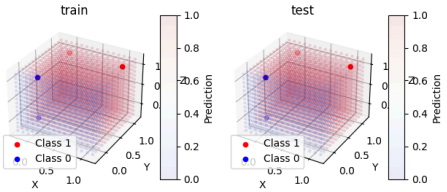
perceptron2

perceptron2 (mlp)

四.2 加分題



perceptron3



perceptron3 (mlp)

五 實驗結果分析及討論

1. 訓練次數過小容易造成未擬和, 訓練次數過大容易造成過擬和
   - 就結果而論應該選 Test 準確率最高的情況
   - 就實際考量(初始值為隨機情況下), 在 Train Data 的準確率上升變得非常緩慢時就可以考慮停止訓練
2. 多層感知機
   - 隱藏層設成 2, 3 時效果比只有一層好很多, 超過 3 層提升不明顯且時間花費巨大
   - 隱藏層越往輸出靠近每層神經元數目漸漸減少似乎有較好效果