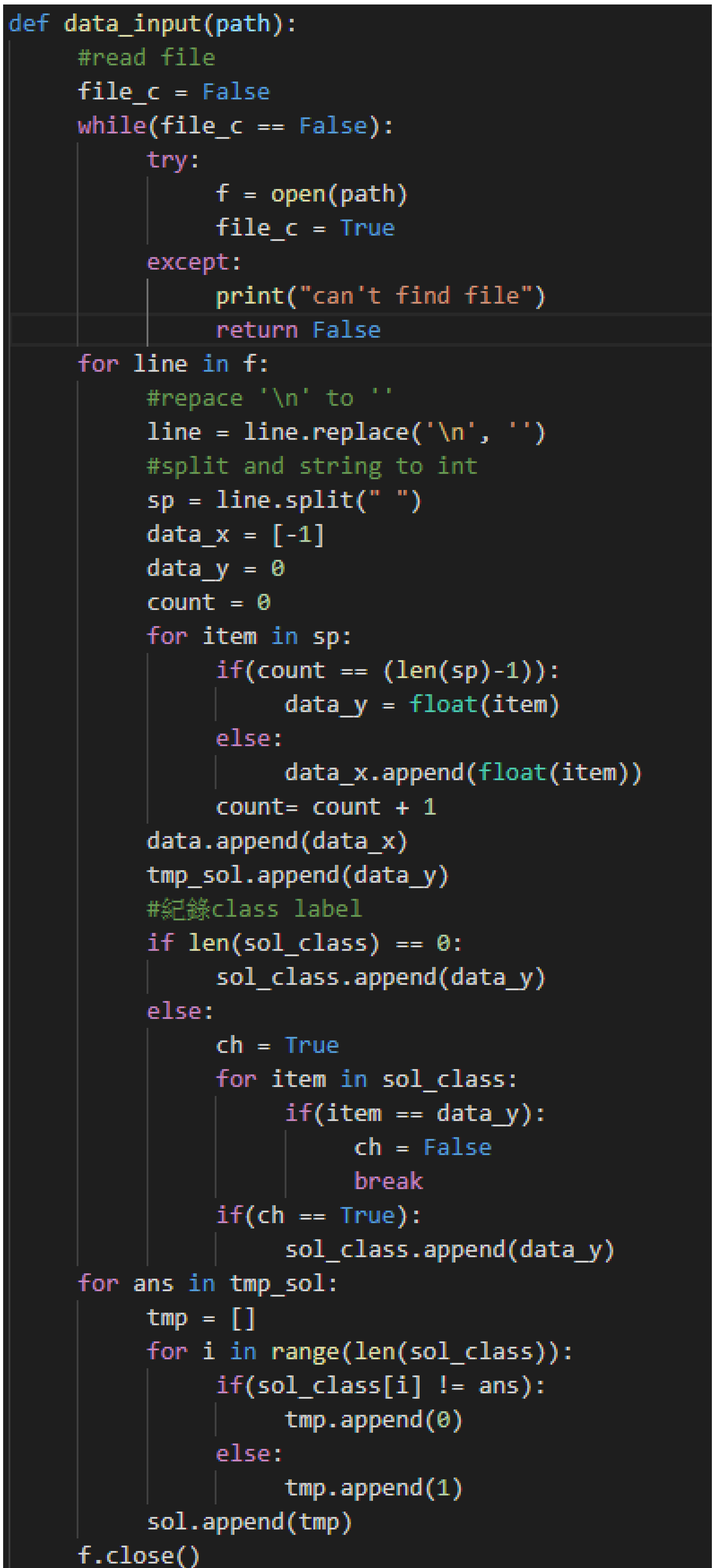
類神經網路 HW1 書面報告

105502302 林觀明

- 程式簡介
  - Data input

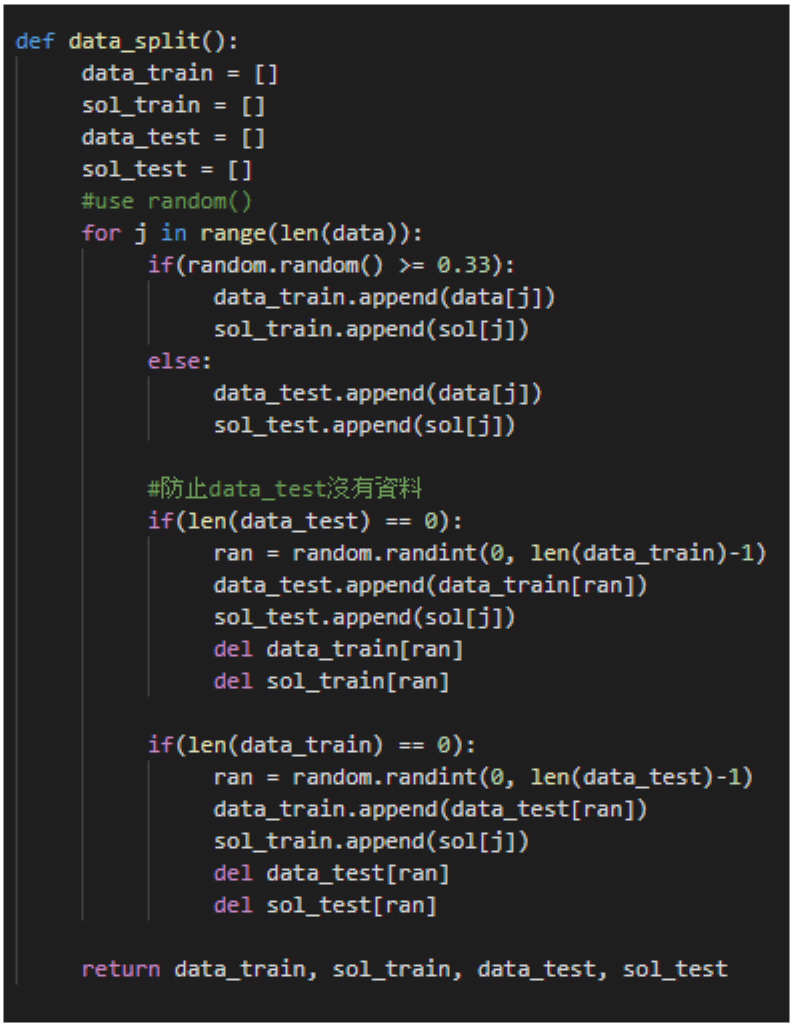


```python
def data_input(path):
    #read file
    file_c = False
    while(file_c == False):
        try:
            f = open(path)
            file_c = True
        except:
            print("can't find file")
            return False
    for line in f:
        #repace '\n' to ''
        line = line.replace('\n', '')
        #split and string to int
        sp = line.split(" ")
        data_x = [-1]
        data_y = 0
        count = 0
        for item in sp:
            if(count == (len(sp)-1)):
                data_y = float(item)
            else:
                data_x.append(float(item))
            count= count + 1
        data.append(data_x)
        tmp_sol.append(data_y)
        #紀錄class label
        if len(sol_class) == 0:
            sol_class.append(data_y)
        else:
            ch = True
            for item in sol_class:
                if(item == data_y):
                    ch = False
                    break
            if(ch == True):
                sol_class.append(data_y)
    for ans in tmp_sol:
        tmp = []
        for i in range(len(sol_class)):
            if(sol_class[i] != ans):
                tmp.append(0)
            else:
                tmp.append(1)
        sol.append(tmp)
    f.close()
```

類神經網路 HW1 書面報告

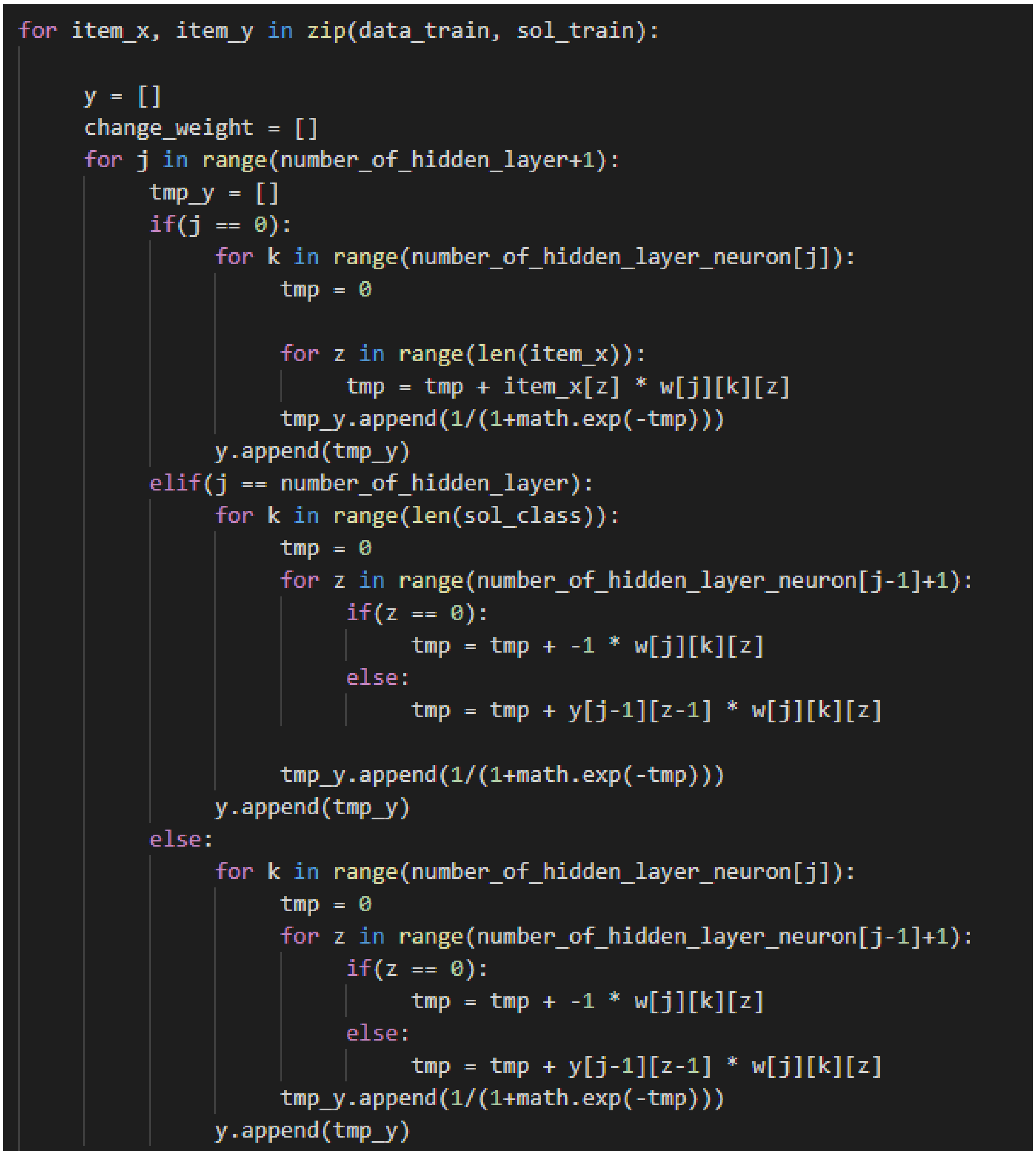
105502302 林觀明

- Data 分割



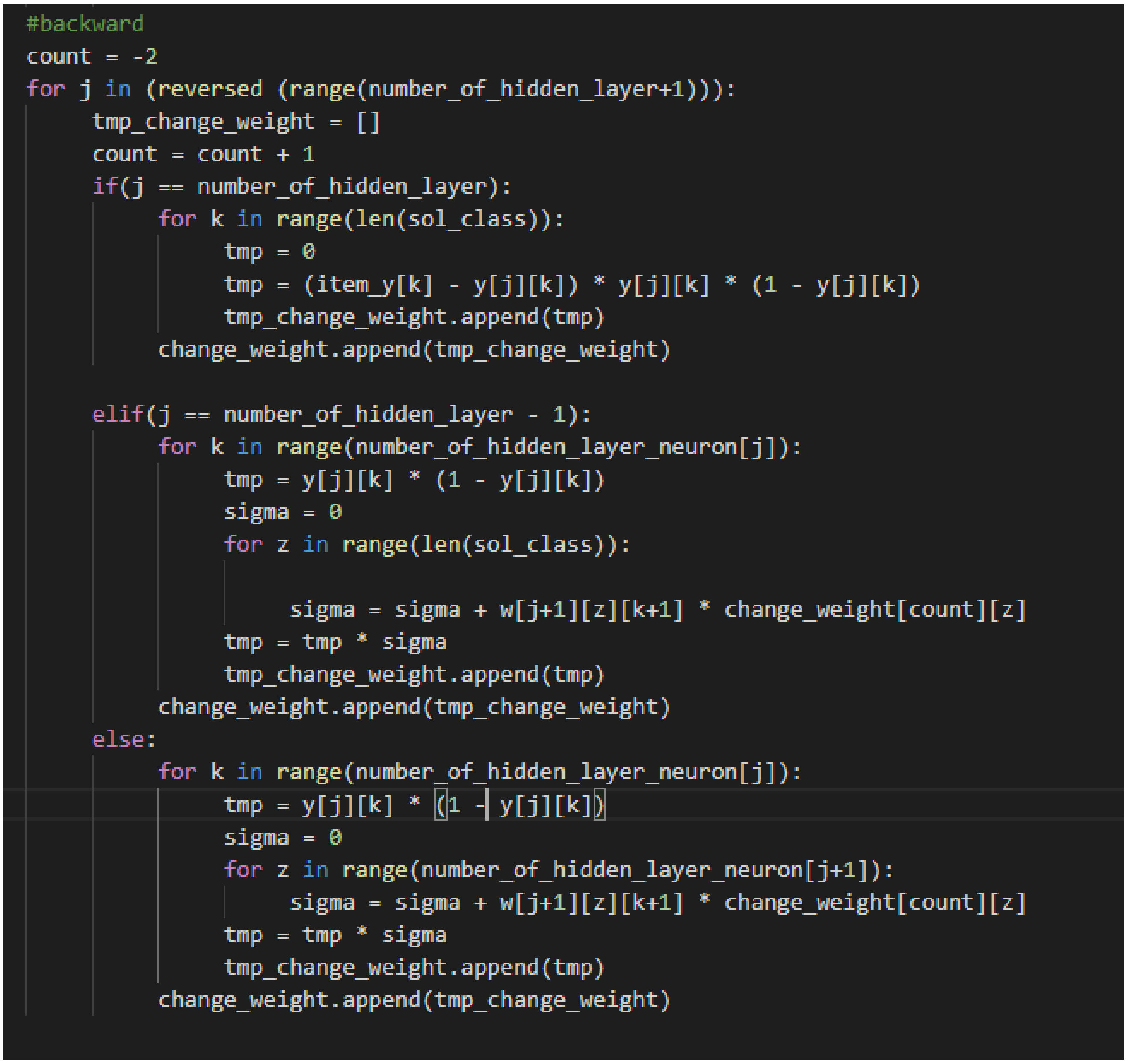
```python
def data_split():
    data_train = []
    sol_train = []
    data_test = []
    sol_test = []
    #use random()
    for j in range(len(data)):
        if(random.random() >= 0.33):
            data_train.append(data[j])
            sol_train.append(sol[j])
        else:
            data_test.append(data[j])
            sol_test.append(sol[j])

        #防止data_test沒有資料
        if(len(data_test) == 0):
            ran = random.randint(0, len(data_train)-1)
            data_test.append(data_train[ran])
            sol_test.append(sol[j])
            del data_train[ran]
            del sol_train[ran]

        if(len(data_train) == 0):
            ran = random.randint(0, len(data_test)-1)
            data_train.append(data_test[ran])
            sol_train.append(sol[j])
            del data_test[ran]
            del sol_test[ran]

    return data_train, sol_train, data_test, sol_test
```

- Train
  - 前饋



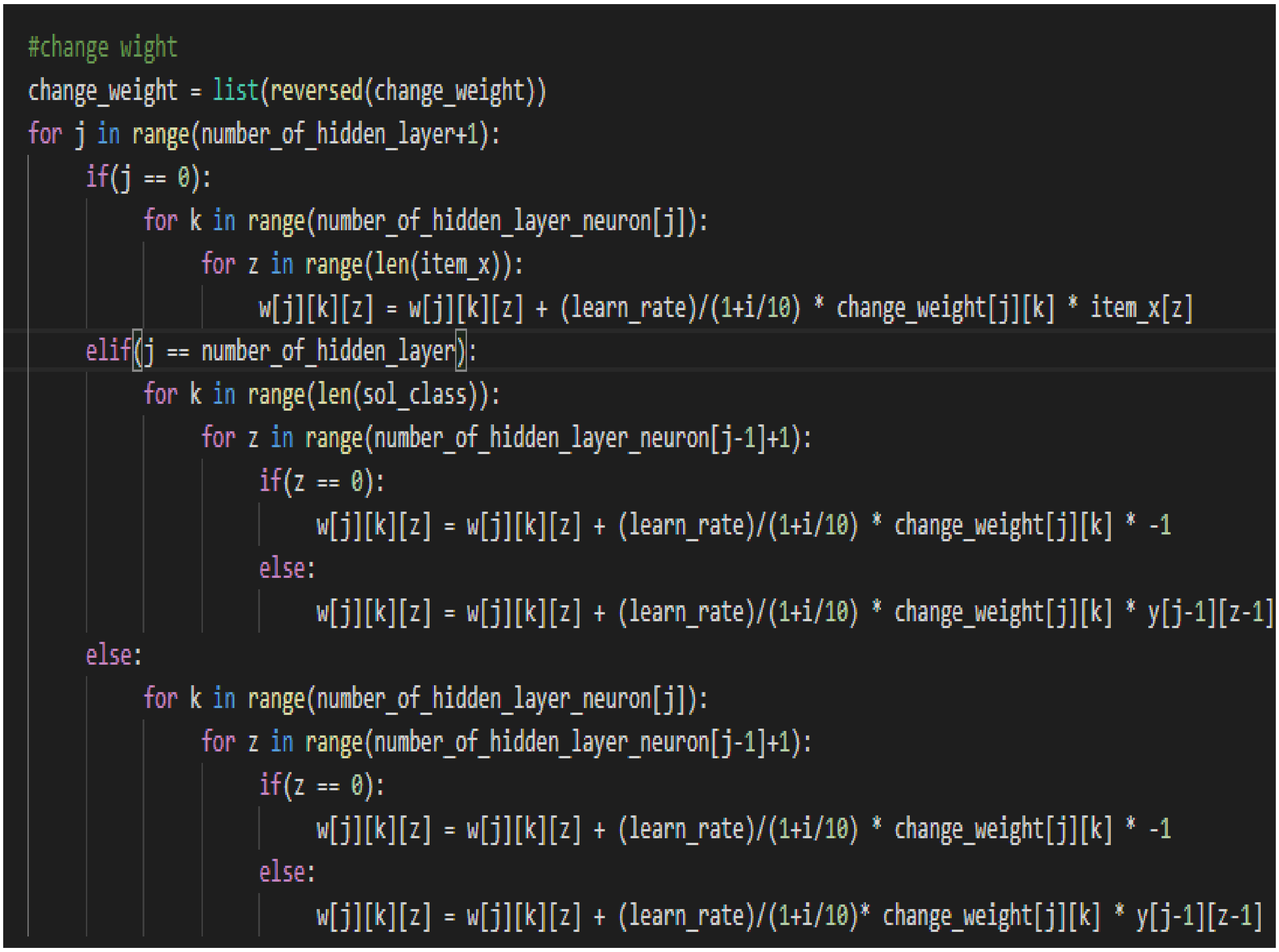
```
for item_x, item_y in zip(data_train, sol_train):

    y = []
    change_weight = []
    for j in range(number_of_hidden_layer+1):
        tmp_y = []
        if(j == 0):
            for k in range(number_of_hidden_layer_neuron[j]):
                tmp = 0

                for z in range(len(item_x)):
                    tmp = tmp + item_x[z] * w[j][k][z]
                tmp_y.append(1/(1+math.exp(-tmp)))
            y.append(tmp_y)
        elif(j == number_of_hidden_layer):
            for k in range(len(sol_class)):
                tmp = 0
                for z in range(number_of_hidden_layer_neuron[j-1]+1):
                    if(z == 0):
                        tmp = tmp + -1 * w[j][k][z]
                    else:
                        tmp = tmp + y[j-1][z-1] * w[j][k][z]

                tmp_y.append(1/(1+math.exp(-tmp)))
            y.append(tmp_y)
        else:
            for k in range(number_of_hidden_layer_neuron[j]):
                tmp = 0
                for z in range(number_of_hidden_layer_neuron[j-1]+1):
                    if(z == 0):
                        tmp = tmp + -1 * w[j][k][z]
                    else:
                        tmp = tmp + y[j-1][z-1] * w[j][k][z]
                tmp_y.append(1/(1+math.exp(-tmp)))
            y.append(tmp_y)
```

- Train
    - 倒饋



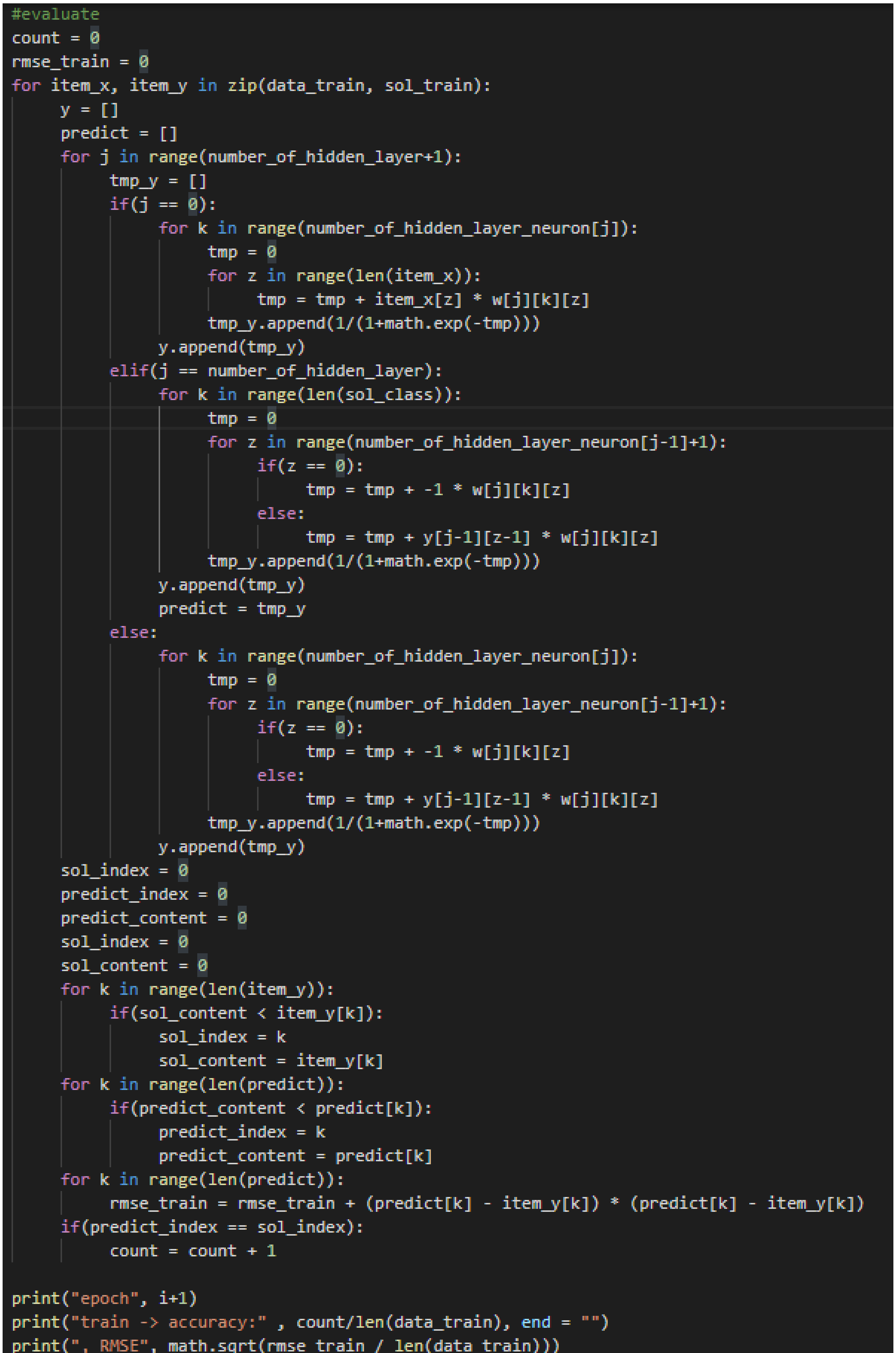
```
#backward
count = -2
for j in (reversed (range(number_of_hidden_layer+1))):
    tmp_change_weight = []
    count = count + 1
    if(j == number_of_hidden_layer):
        for k in range(len(sol_class)):
            tmp = 0
            tmp = (item_y[k] - y[j][k]) * y[j][k] * (1 - y[j][k])
            tmp_change_weight.append(tmp)
        change_weight.append(tmp_change_weight)

    elif(j == number_of_hidden_layer - 1):
        for k in range(number_of_hidden_layer_neuron[j]):
            tmp = y[j][k] * (1 - y[j][k])
            sigma = 0
            for z in range(len(sol_class)):

                sigma = sigma + w[j+1][z][k+1] * change_weight[count][z]
            tmp = tmp * sigma
            tmp_change_weight.append(tmp)
        change_weight.append(tmp_change_weight)
    else:
        for k in range(number_of_hidden_layer_neuron[j]):
            tmp = y[j][k] * (1 - y[j][k])
            sigma = 0
            for z in range(number_of_hidden_layer_neuron[j+1]):
                sigma = sigma + w[j+1][z][k+1] * change_weight[count][z]
            tmp = tmp * sigma
            tmp_change_weight.append(tmp)
        change_weight.append(tmp_change_weight)
```

- Train
    - 改變權重



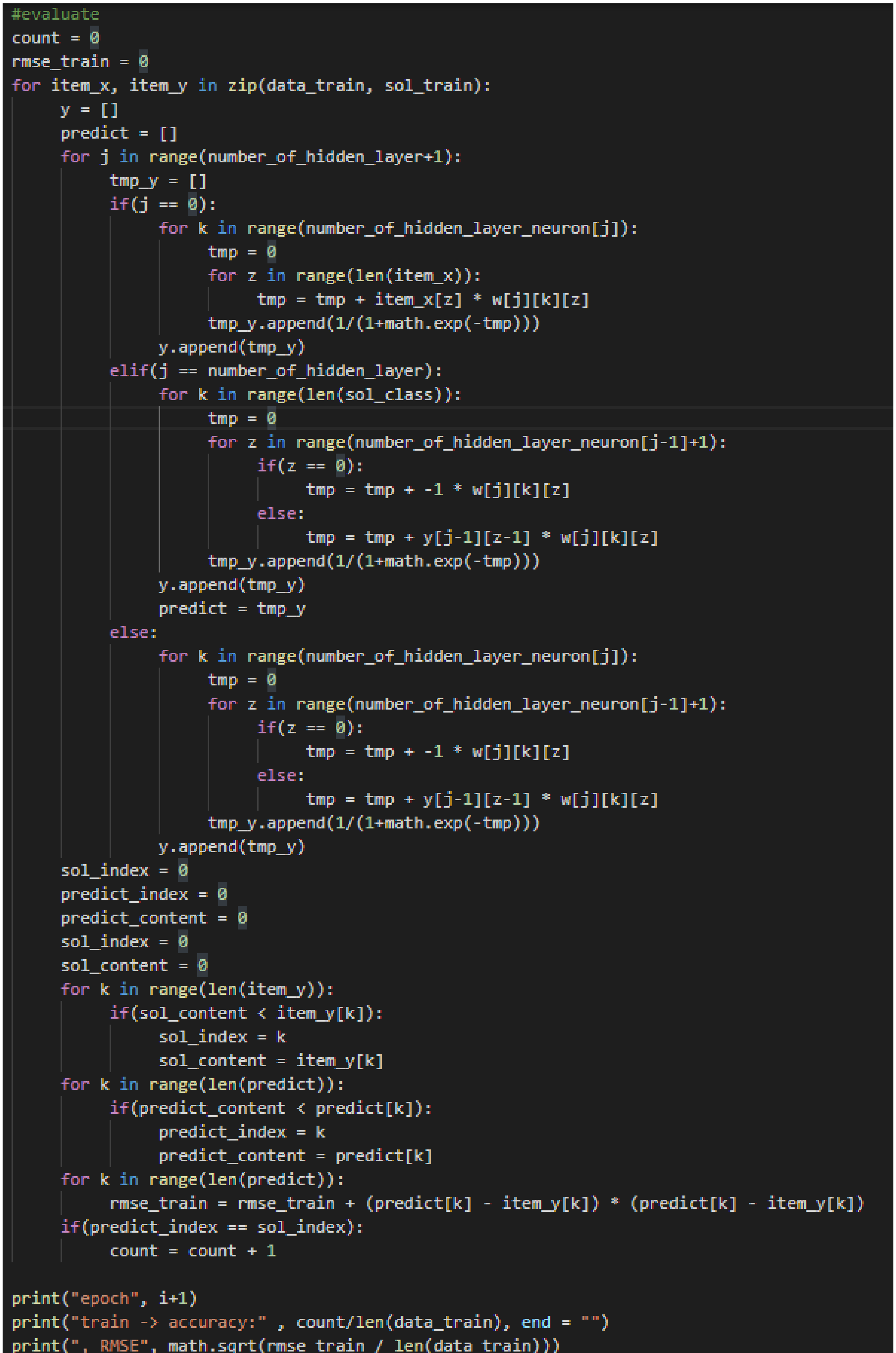
```
#change wight
change_weight = list(reversed(change_weight))
for j in range(number_of_hidden_layer+1):
    if(j == 0):
        for k in range(number_of_hidden_layer_neuron[j]):
            for z in range(len(item_x)):
                w[j][k][z] = w[j][k][z] + (learn_rate)/(1+i/10) * change_weight[j][k] * item_x[z]
    elif(j == number_of_hidden_layer):
        for k in range(len(sol_class)):
            for z in range(number_of_hidden_layer_neuron[j-1]+1):
                if(z == 0):
                    w[j][k][z] = w[j][k][z] + (learn_rate)/(1+i/10) * change_weight[j][k] * -1
                else:
                    w[j][k][z] = w[j][k][z] + (learn_rate)/(1+i/10) * change_weight[j][k] * y[j-1][z-1]
    else:
        for k in range(number_of_hidden_layer_neuron[j]):
            for z in range(number_of_hidden_layer_neuron[j-1]+1):
                if(z == 0):
                    w[j][k][z] = w[j][k][z] + (learn_rate)/(1+i/10) * change_weight[j][k] * -1
                else:
                    w[j][k][z] = w[j][k][z] + (learn_rate)/(1+i/10)* change_weight[j][k] * y[j-1][z-1]
```

- Train
    - 精準度

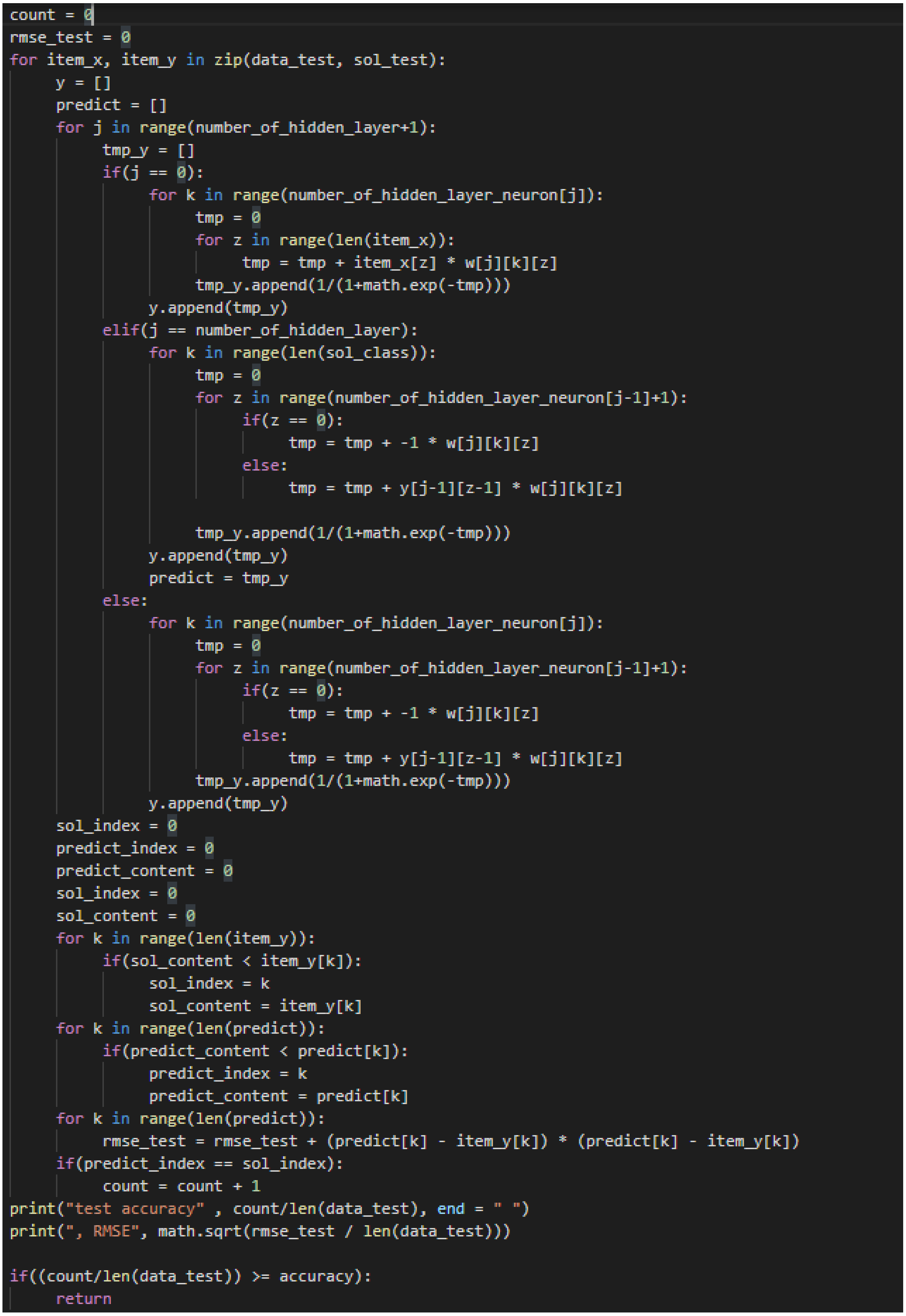


```
#evaluate
count = 0
rmse_train = 0
for item_x, item_y in zip(data_train, sol_train):
    y = []
    predict = []
    for j in range(number_of_hidden_layer+1):
        tmp_y = []
        if(j == 0):
            for k in range(number_of_hidden_layer_neuron[j]):
                tmp = 0
                for z in range(len(item_x)):
                    tmp = tmp + item_x[z] * w[j][k][z]
                tmp_y.append(1/(1+math.exp(-tmp)))
            y.append(tmp_y)
        elif(j == number_of_hidden_layer):
            for k in range(len(sol_class)):
                tmp = 0
                for z in range(number_of_hidden_layer_neuron[j-1]+1):
                    if(z == 0):
                        tmp = tmp + -1 * w[j][k][z]
                    else:
                        tmp = tmp + y[j-1][z-1] * w[j][k][z]
                tmp_y.append(1/(1+math.exp(-tmp)))
            y.append(tmp_y)
            predict = tmp_y
        else:
            for k in range(number_of_hidden_layer_neuron[j]):
                tmp = 0
                for z in range(number_of_hidden_layer_neuron[j-1]+1):
                    if(z == 0):
                        tmp = tmp + -1 * w[j][k][z]
                    else:
                        tmp = tmp + y[j-1][z-1] * w[j][k][z]
                tmp_y.append(1/(1+math.exp(-tmp)))
            y.append(tmp_y)
    sol_index = 0
    predict_index = 0
    predict_content = 0
    sol_index = 0
    sol_content = 0
    for k in range(len(item_y)):
        if(sol_content < item_y[k]):
            sol_index = k
            sol_content = item_y[k]
    for k in range(len(predict)):
        if(predict_content < predict[k]):
            predict_index = k
            predict_content = predict[k]
    for k in range(len(predict)):
        rmse_train = rmse_train + (predict[k] - item_y[k]) * (predict[k] - item_y[k])
    if(predict_index == sol_index):
        count = count + 1

print("epoch", i+1)
print("train -> accuracy:" , count/len(data_train), end = "")
print(", RMSE", math.sqrt(rmse_train / len(data_train)))
```

- Train
    - 精準度

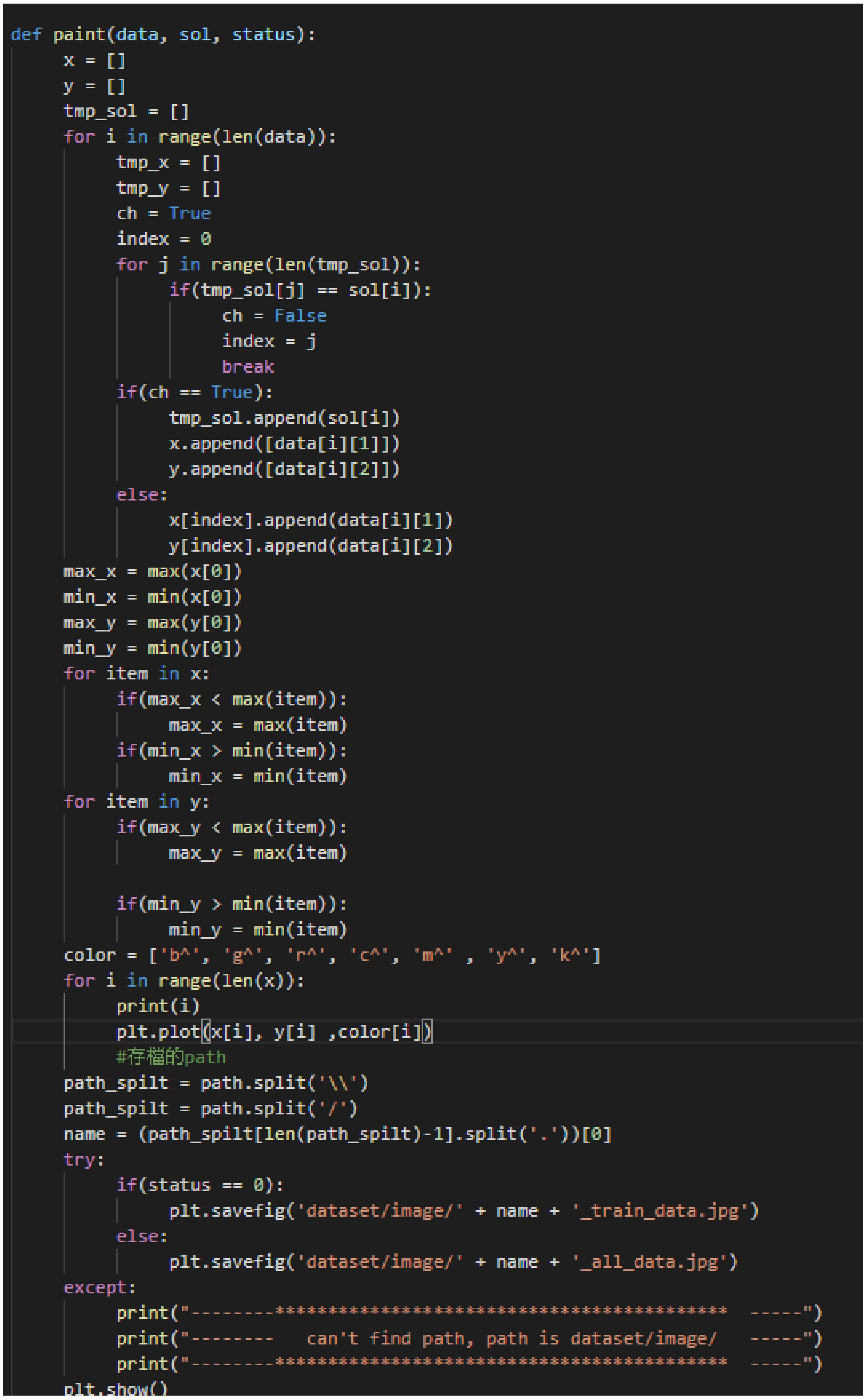


```
#evaluate
count = 0
rmse_train = 0
for item_x, item_y in zip(data_train, sol_train):
    y = []
    predict = []
    for j in range(number_of_hidden_layer+1):
        tmp_y = []
        if(j == 0):
            for k in range(number_of_hidden_layer_neuron[j]):
                tmp = 0
                for z in range(len(item_x)):
                    tmp = tmp + item_x[z] * w[j][k][z]
                tmp_y.append(1/(1+math.exp(-tmp)))
            y.append(tmp_y)
        elif(j == number_of_hidden_layer):
            for k in range(len(sol_class)):
                tmp = 0
                for z in range(number_of_hidden_layer_neuron[j-1]+1):
                    if(z == 0):
                        tmp = tmp + -1 * w[j][k][z]
                    else:
                        tmp = tmp + y[j-1][z-1] * w[j][k][z]
                tmp_y.append(1/(1+math.exp(-tmp)))
            y.append(tmp_y)
            predict = tmp_y
        else:
            for k in range(number_of_hidden_layer_neuron[j]):
                tmp = 0
                for z in range(number_of_hidden_layer_neuron[j-1]+1):
                    if(z == 0):
                        tmp = tmp + -1 * w[j][k][z]
                    else:
                        tmp = tmp + y[j-1][z-1] * w[j][k][z]
                tmp_y.append(1/(1+math.exp(-tmp)))
            y.append(tmp_y)
    sol_index = 0
    predict_index = 0
    predict_content = 0
    sol_index = 0
    sol_content = 0
    for k in range(len(item_y)):
        if(sol_content < item_y[k]):
            sol_index = k
            sol_content = item_y[k]
    for k in range(len(predict)):
        if(predict_content < predict[k]):
            predict_index = k
            predict_content = predict[k]
    for k in range(len(predict)):
        rmse_train = rmse_train + (predict[k] - item_y[k]) * (predict[k] - item_y[k])
    if(predict_index == sol_index):
        count = count + 1

print("epoch", i+1)
print("train -> accuracy:" , count/len(data_train), end = "")
print(", RMSE", math.sqrt(rmse_train / len(data_train)))
```



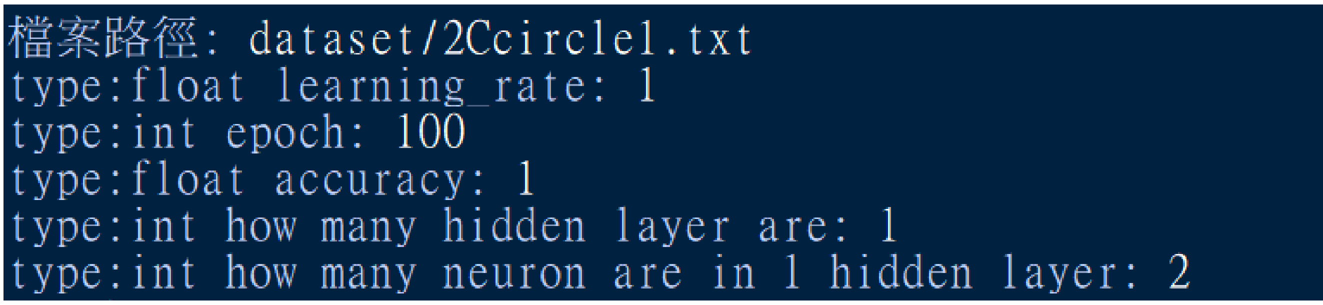
```python
count = 0
rmse_test = 0
for item_x, item_y in zip(data_test, sol_test):
    y = []
    predict = []
    for j in range(number_of_hidden_layer+1):
        tmp_y = []
        if(j == 0):
            for k in range(number_of_hidden_layer_neuron[j]):
                tmp = 0
                for z in range(len(item_x)):
                    tmp = tmp + item_x[z] * w[j][k][z]
                tmp_y.append(1/(1+math.exp(-tmp)))
            y.append(tmp_y)
        elif(j == number_of_hidden_layer):
            for k in range(len(sol_class)):
                tmp = 0
                for z in range(number_of_hidden_layer_neuron[j-1]+1):
                    if(z == 0):
                        tmp = tmp + -1 * w[j][k][z]
                    else:
                        tmp = tmp + y[j-1][z-1] * w[j][k][z]

                tmp_y.append(1/(1+math.exp(-tmp)))
            y.append(tmp_y)
            predict = tmp_y
        else:
            for k in range(number_of_hidden_layer_neuron[j]):
                tmp = 0
                for z in range(number_of_hidden_layer_neuron[j-1]+1):
                    if(z == 0):
                        tmp = tmp + -1 * w[j][k][z]
                    else:
                        tmp = tmp + y[j-1][z-1] * w[j][k][z]
                tmp_y.append(1/(1+math.exp(-tmp)))
            y.append(tmp_y)
    sol_index = 0
    predict_index = 0
    predict_content = 0
    sol_index = 0
    sol_content = 0
    for k in range(len(item_y)):
        if(sol_content < item_y[k]):
            sol_index = k
            sol_content = item_y[k]
    for k in range(len(predict)):
        if(predict_content < predict[k]):
            predict_index = k
            predict_content = predict[k]
    for k in range(len(predict)):
        rmse_test = rmse_test + (predict[k] - item_y[k]) * (predict[k] - item_y[k])
    if(predict_index == sol_index):
        count = count + 1
print("test accuracy" , count/len(data_test), end = " ")
print(", RMSE", math.sqrt(rmse_test / len(data_test)))

if((count/len(data_test)) >= accuracy):
    return
```

- Train
  - 畫圖



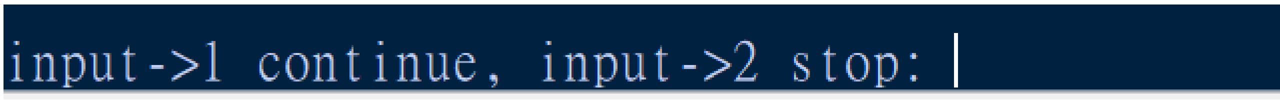
```python
def paint(data, sol, status):
    x = []
    y = []
    tmp_sol = []
    for i in range(len(data)):
        tmp_x = []
        tmp_y = []
        ch = True
        index = 0
        for j in range(len(tmp_sol)):
            if(tmp_sol[j] == sol[i]):
                ch = False
                index = j
                break
        if(ch == True):
            tmp_sol.append(sol[i])
            x.append([data[i][1]])
            y.append([data[i][2]])
        else:
            x[index].append(data[i][1])
            y[index].append(data[i][2])
    max_x = max(x[0])
    min_x = min(x[0])
    max_y = max(y[0])
    min_y = min(y[0])
    for item in x:
        if(max_x < max(item)):
            max_x = max(item)
        if(min_x > min(item)):
            min_x = min(item)
    for item in y:
        if(max_y < max(item)):
            max_y = max(item)

        if(min_y > min(item)):
            min_y = min(item)
    color = ['b^', 'g^', 'r^', 'c^', 'm^' , 'y^', 'k^']
    for i in range(len(x)):
        print(i)
        plt.plot(x[i], y[i] ,color[i])
        #存檔的path
    path_spilt = path.split('\\')
    path_spilt = path.split('/')
    name = (path_spilt[len(path_spilt)-1].split('.'))[0]
    try:
        if(status == 0):
            plt.savefig('dataset/image/' + name + '_train_data.jpg')
        else:
            plt.savefig('dataset/image/' + name + '_all_data.jpg')
    except:
        print("--------********************************************  -----")
        print("--------   can't find path, path is dataset/image/   -----")
        print("--------********************************************  -----")
    plt.show()
```

- 程式執行說明
  1. 首先打開執行檔(會等很久)
  2. 輸入檔案路徑(有path.txt 上有路徑可以複製貼上)、learning_rate、epoch、accuracy、需要幾層神經元與第幾層需要幾個神經元



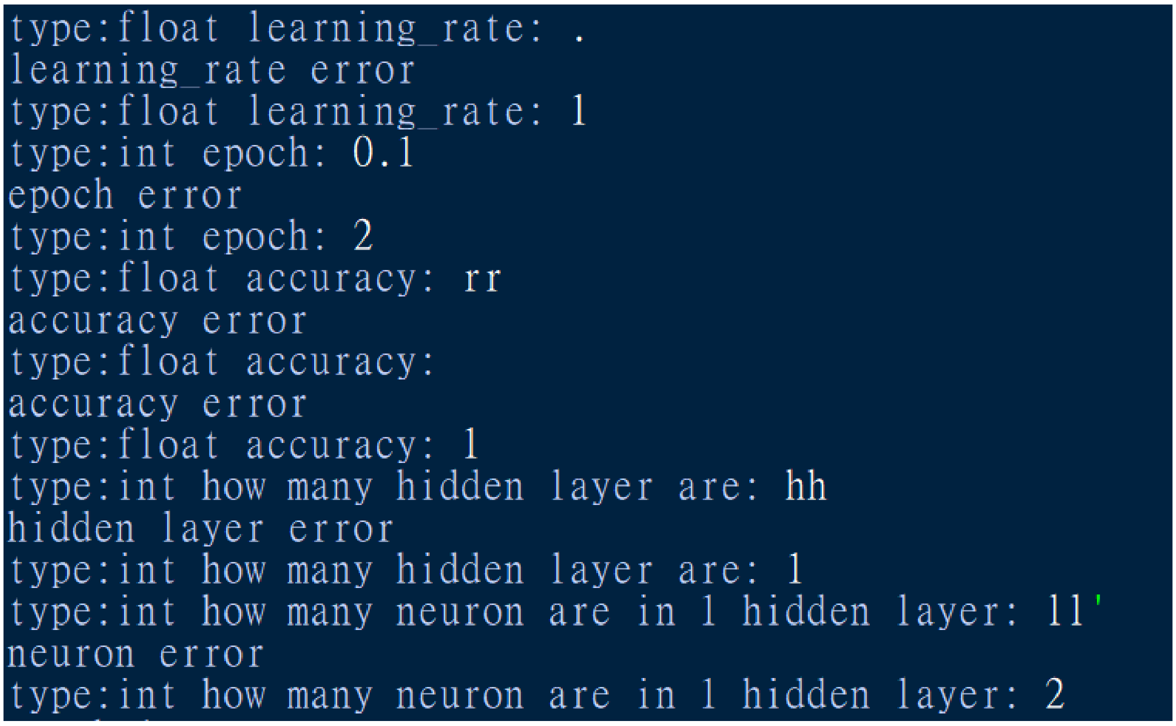
```
檔案路徑: dataset/2Ccircle1.txt
type:float learning_rate: 1
type:int epoch: 100
type:float accuracy: 1
type:int how many hidden layer are: 1
type:int how many neuron are in 1 hidden layer: 2
```

  3. 如果找不到檔案 就會出現 input->1 continue, input->2 stop，輸入1可以繼續 輸入2就會停止



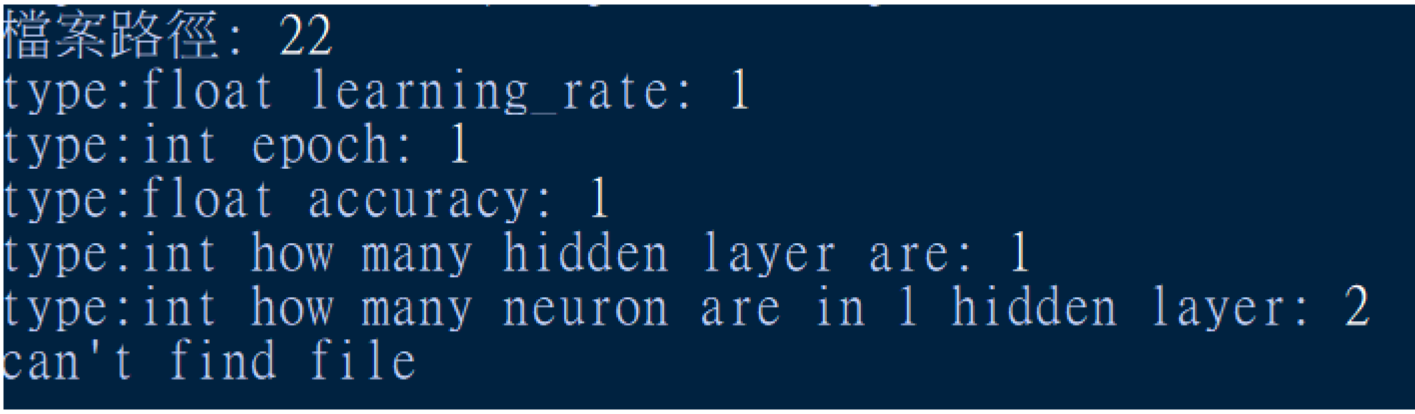
```
input->1 continue, input->2 stop: |
```

  4. 另外如果learning_rate, epoch, accuracy輸入錯誤需要重新輸入



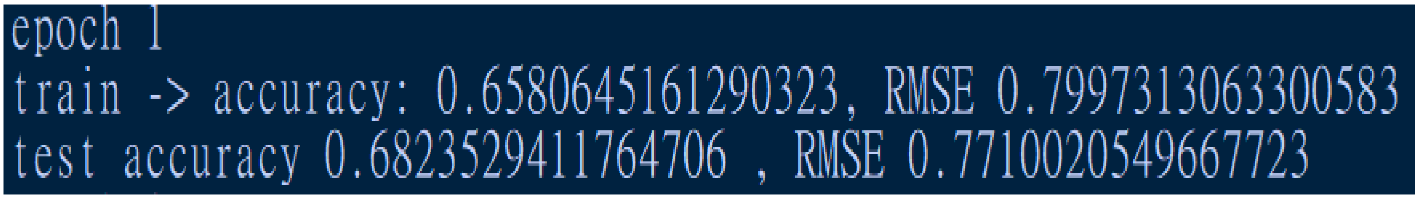
```
type:float learning_rate: .
learning_rate error
type:float learning_rate: 1
type:int epoch: 0.1
epoch error
type:int epoch: 2
type:float accuracy: rr
accuracy error
type:float accuracy:
accuracy error
type:float accuracy: 1
type:int how many hidden layer are: hh
hidden layer error
type:int how many hidden layer are: 1
type:int how many neuron are in 1 hidden layer: 11'
neuron error
type:int how many neuron are in 1 hidden layer: 2
```

5. 可以確定file是否存在



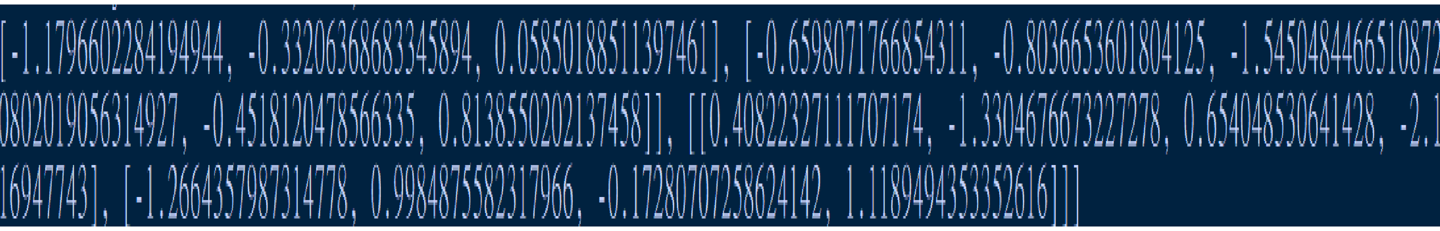
```
檔案路徑: 22
type:float learning_rate: 1
type:int epoch: 1
type:float accuracy: 1
type:int how many hidden layer are: 1
type:int how many neuron are in 1 hidden layer: 2
can't find file
```

5. 輸入都正確就會開始執行，每個epoch都會輸出train的accuracy, test的accurac與RMSE

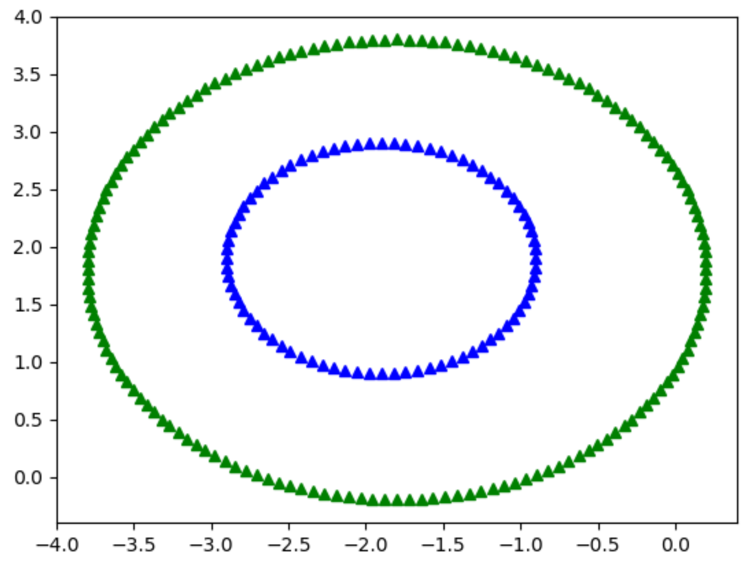


```
epoch 1
train -> accuracy: 0.6580645161290323, RMSE 0.7997313063300583
test accuracy 0.6823529411764706 , RMSE 0.7710020549667723
```

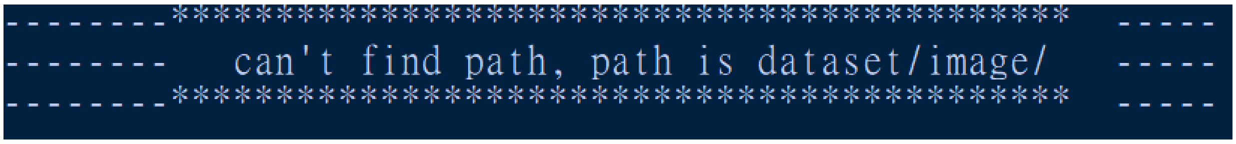
5. 訓練完之後會跑出第一張圖且會顯示權重



```
[-1.1796602284194944, -0.33206368683345894, 0.05850188511397461], [-0.6598071766854311, -0.8036653601804125, -1.5450484466510872
0802019056314927, -0.4518120478566335, 0.8138550202137458]], [[0.40822327111707174, -1.3304676673227278, 0.654048530641428, -2.1
1694743], [-1.2664357987314778, 0.9984875582317966, -0.17280707258624142, 1.1189494353352616]]]
```

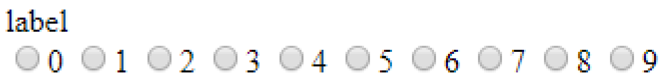


8. 出現的圖檔會存在dataset/image/，如果找不到目錄會出現以下圖片，不影響程式執行。

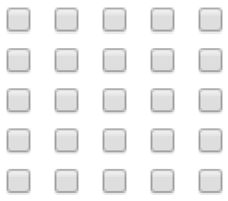


```
--------*******************************************  -----
--------   can't find path, path is dataset/image/   -----
--------*******************************************  -----
```

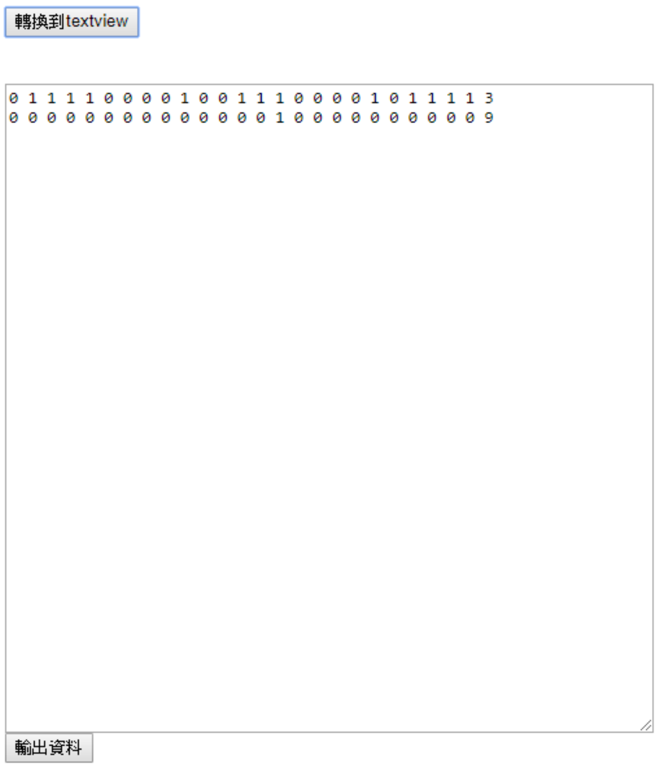
- 程式執行說明(index.html)
  1. 用瀏覽器打開
  2. 選擇label



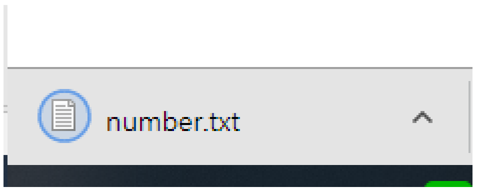
  3. **選擇bit**



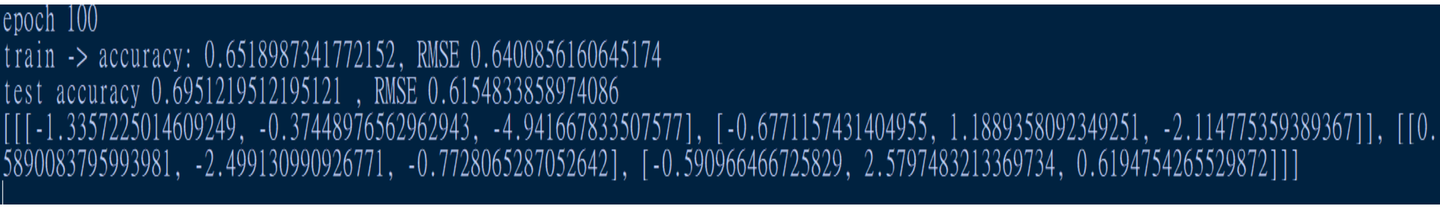
  4. 按下button“轉換到textview”，就會看到目前資料輸出資料就換下載資料。



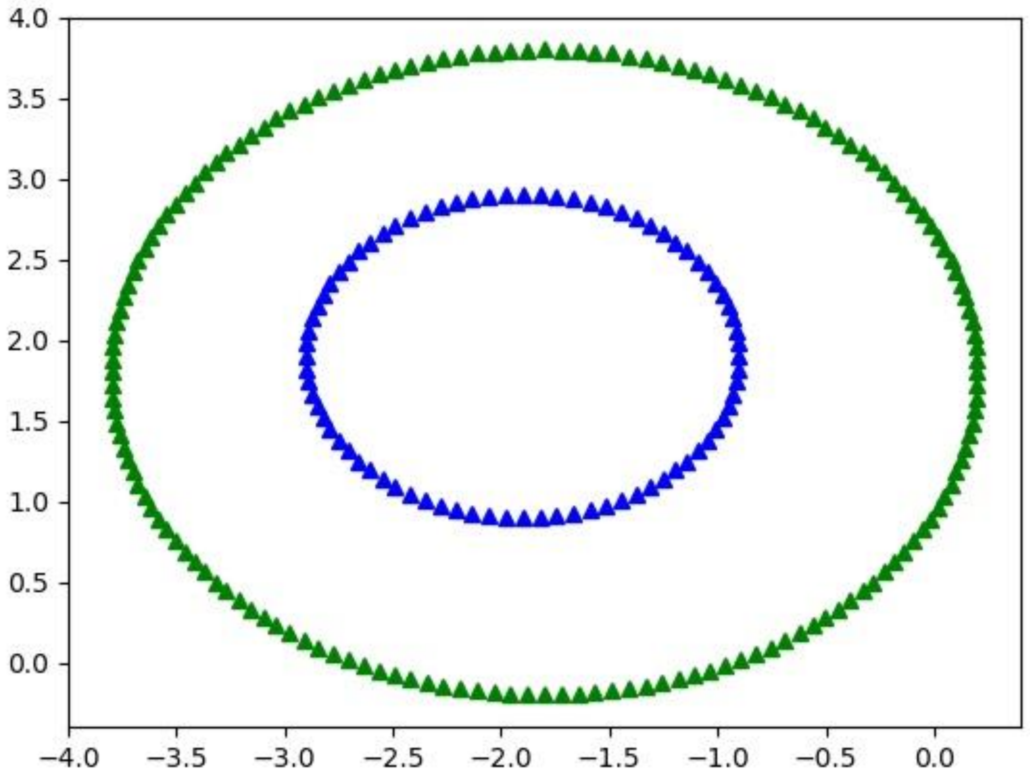
```
0 1 1 1 1 0 0 0 0 1 0 0 1 1 1 0 0 0 0 1 0 1 1 1 1 3
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 9
```



- 實驗結果
  - ✓ dataset/2Ccircle1.txt

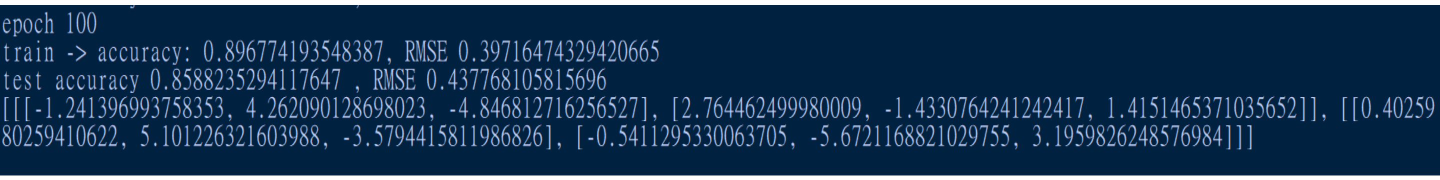


```
epoch 100
train -> accuracy: 0.6518987341772152, RMSE 0.6400856160645174
test accuracy 0.6951219512195121 , RMSE 0.6154833858974086
[[[-1.3357225014609249, -0.37448976562962943, -4.941667833507577], [-0.6771157431404955, 1.1889358092349251, -2.114775359389367]], [[0.
5890083795993981, -2.499130990926771, -0.7728065287052642], [-0.590966466725829, 2.5797483213369734, 0.6194754265529872]]]
```

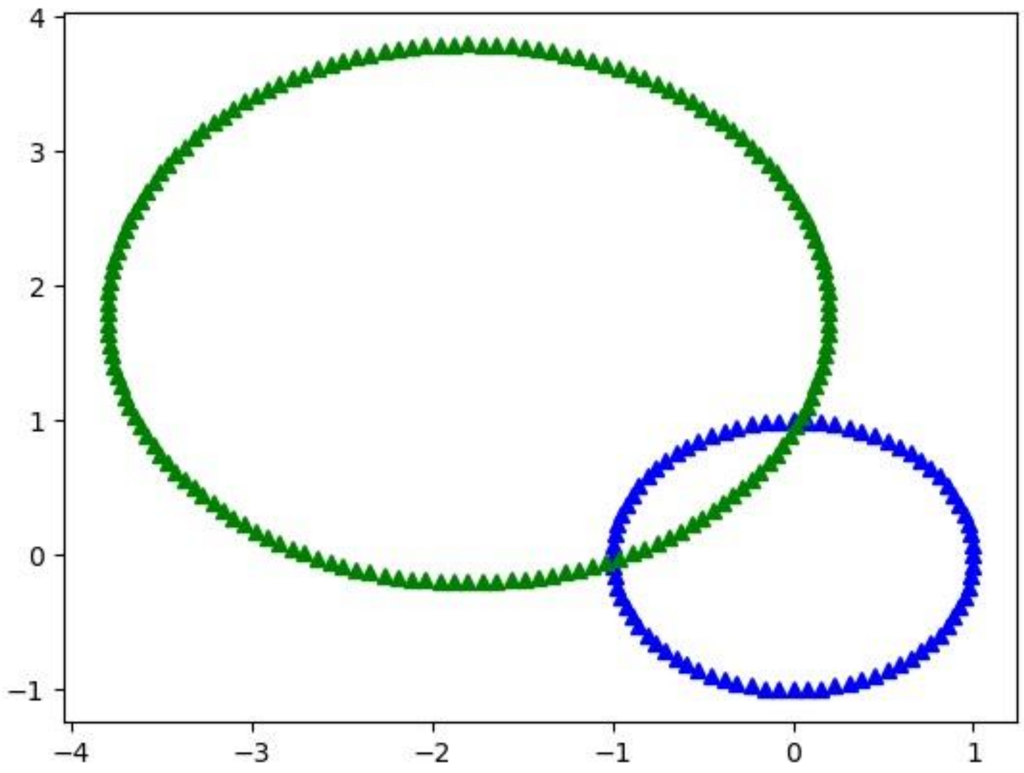


分析:分2類 速度非常慢 rmse下降緩慢 準確率提升至0.69

- ✓ dataset/2Circle1.txt

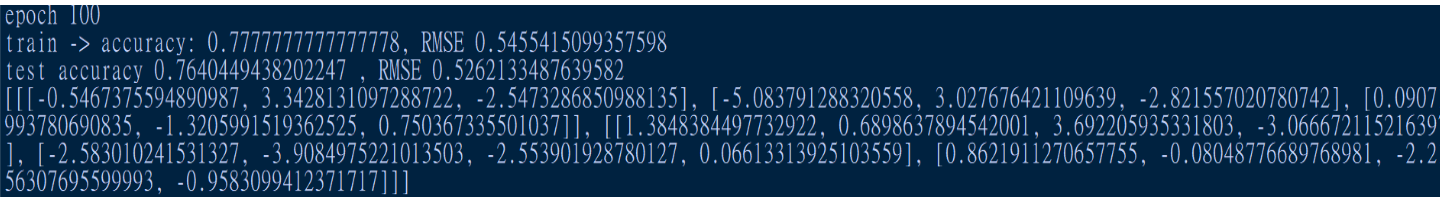


```
epoch 100
train -> accuracy: 0.896774193548387, RMSE 0.39716474329420665
test accuracy 0.8588235294117647 , RMSE 0.437768105815696
[[[-1.241396993758353, 4.262090128698023, -4.846812716256527], [2.764462499980009, -1.4330764241242417, 1.4151465371035652]], [[0.40259
80259410622, 5.101226321603988, -3.5794415811986826], [-0.5411295330063705, -5.6721168821029755, 3.1959826248576984]]]
```

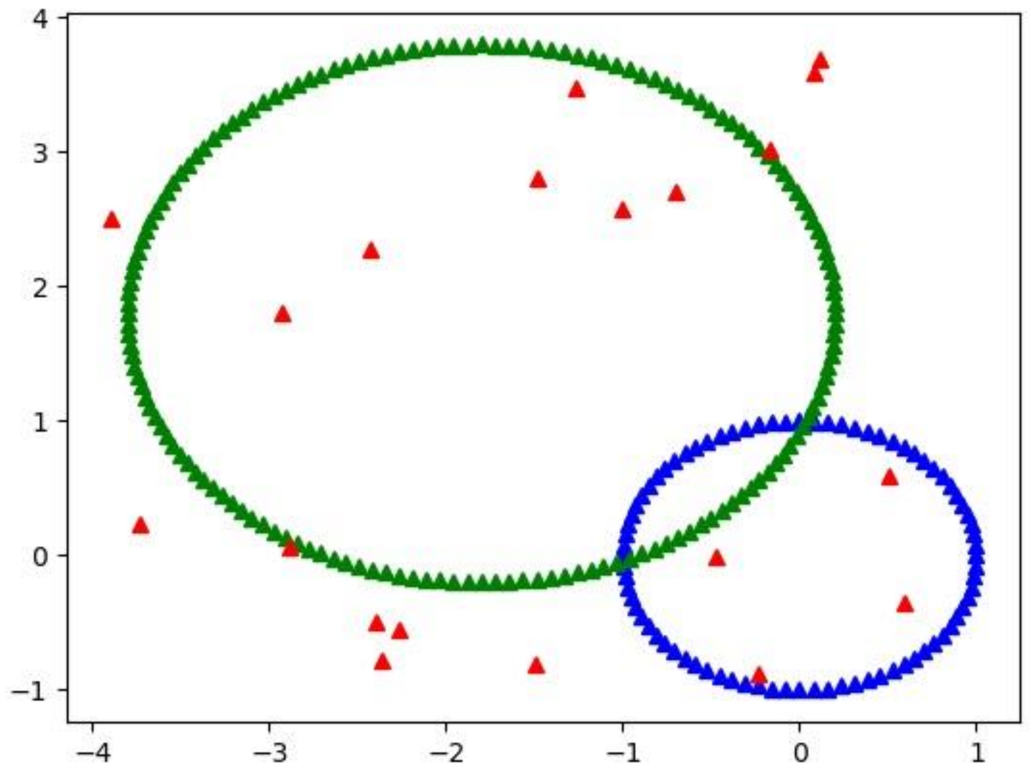


分析:分2類 準確率提升至0.85

✓ dataset/2Circle2.txt

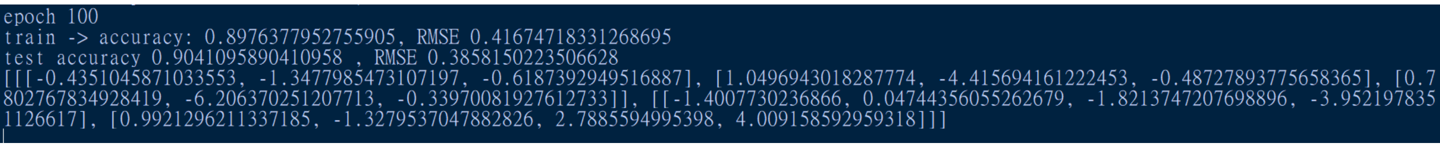


```
epoch 100
train -> accuracy: 0.7777777777777778, RMSE 0.5455415099357598
test accuracy 0.7640449438202247 , RMSE 0.5262133487639582
[[[-0.5467375594890987, 3.3428131097288722, -2.5473286850988135], [-5.083791288320558, 3.027676421109639, -2.821557020780742], [0.0907
993780690835, -1.3205991519362525, 0.7503673355010371]], [[1.3848384497732922, 0.6898637894542001, 3.692205935331803, -3.06667211521639
], [-2.583010241531327, -3.9084975221013503, -2.553901928780127, 0.06613313925103559], [0.8621911270657755, -0.08048776689768981, -2.2
56307695599993, -0.9583099412371717]]]
```

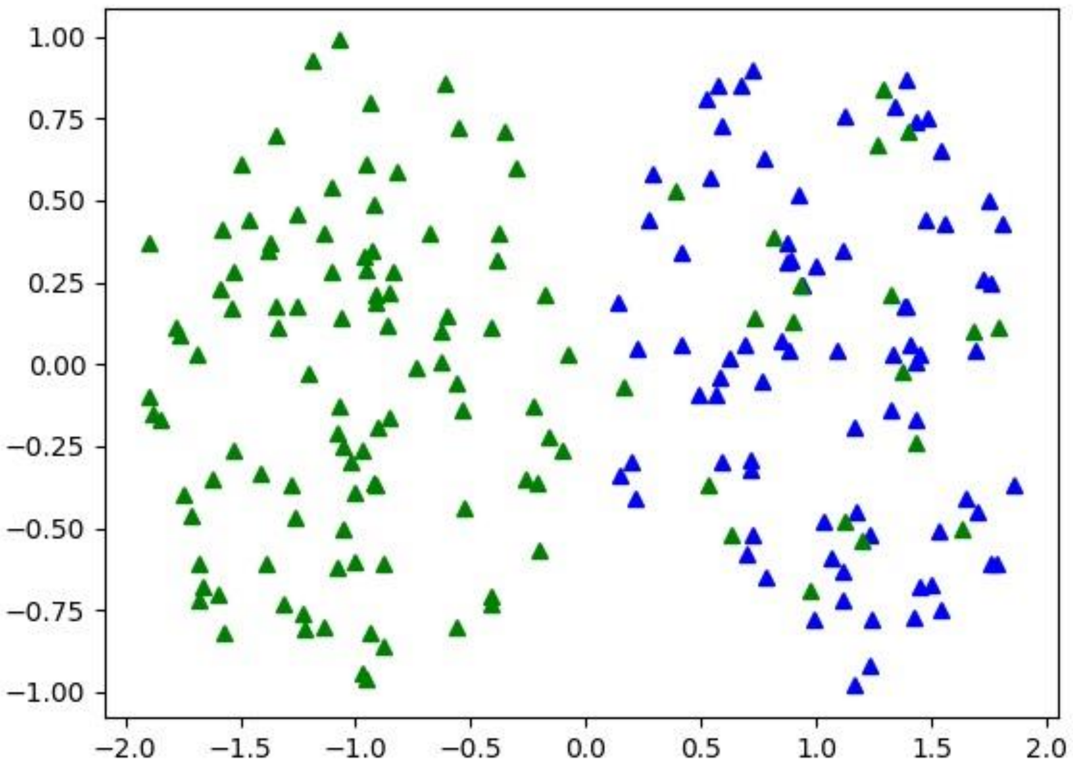


分析:分3類 準確率由0.23提升至0.86下降至0.76

- ✓ dataset/2CloseS.txt

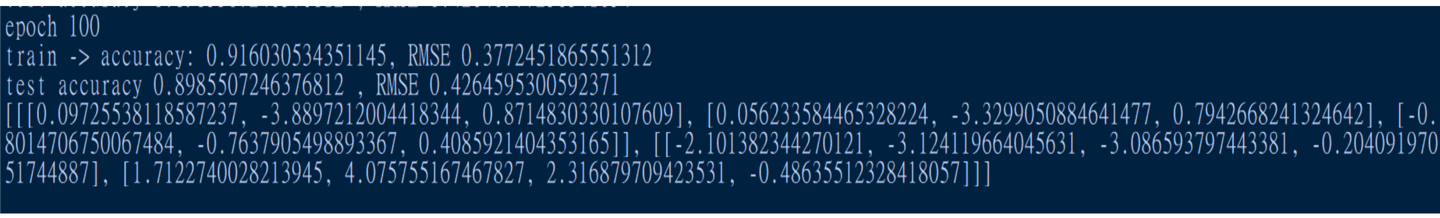


```
epoch 100
train -> accuracy: 0.8976377952755905, RMSE 0.41674718331268695
test accuracy 0.9041095890410958 , RMSE 0.3858150223506628
[[[-0.4351045871033553, -1.3477985473107197, -0.6187392949516887], [1.0496943018287774, -4.415694161222453, -0.48727893775658365], [0.7
802767834928419, -6.206370251207713, -0.33970081927612733]], [[-1.4007730236866, 0.04744356055262679, -1.8213747207698896, -3.952197835
1126617], [0.9921296211337185, -1.3279537047882826, 2.7885594995398, 4.009158592959318]]]
```

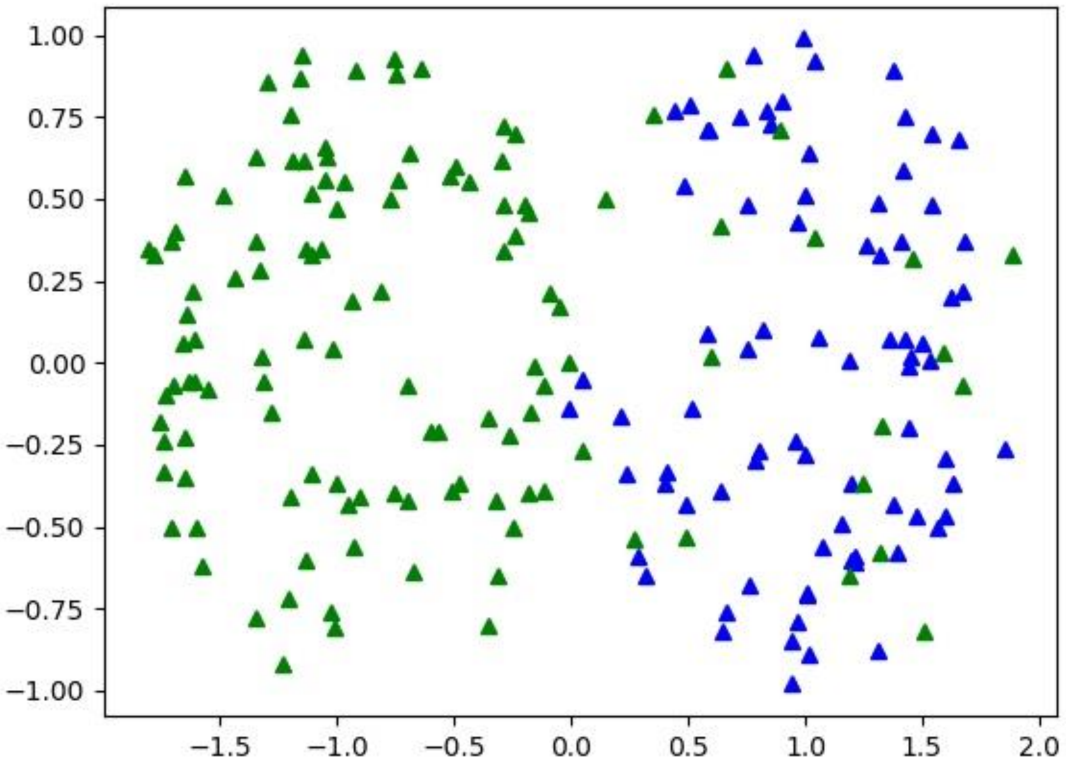


分析:分2類 準確率提升至0.90

- ✓ dataset/2CloseS2.txt

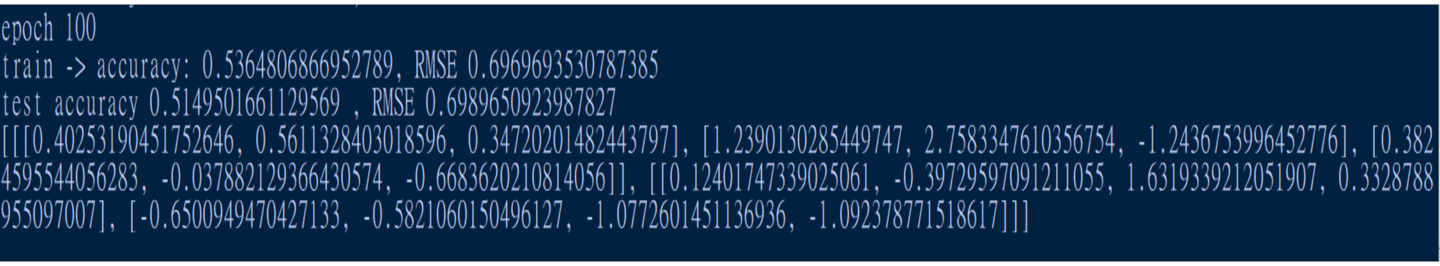


```
epoch 100
train -> accuracy: 0.916030534351145, RMSE 0.3772451865551312
test accuracy 0.8985507246376812 , RMSE 0.4264595300592371
[[[0.09725538118587237, -3.8897212004418344, 0.8714830330107609], [0.056233584465328224, -3.3299050884641477, 0.7942668241324642], [-0.
8014706750067484, -0.7637905498893367, 0.4085921404353165]], [[-2.101382344270121, -3.124119664045631, -3.086593797443381, -0.204091970
51744887], [1.7122740028213945, 4.075755167467827, 2.316879709423531, -0.48635512328418057]]]
```

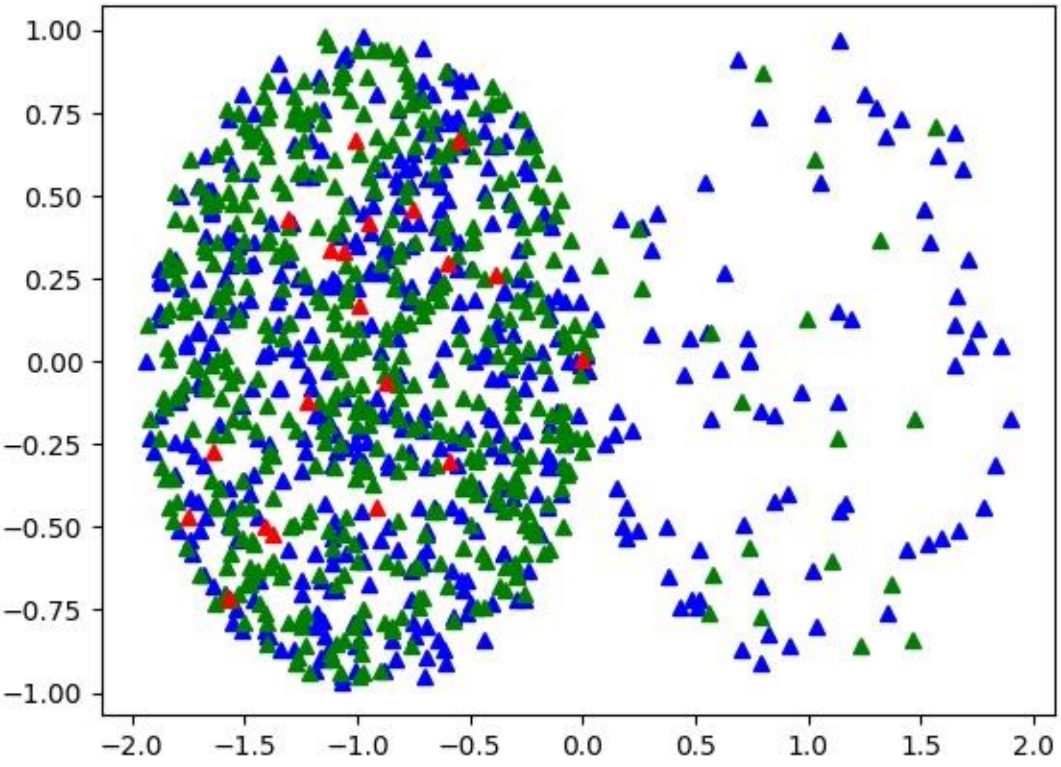


分析:分2類 準確率提升至0.89

✓ dataset/2CloseS3.txt

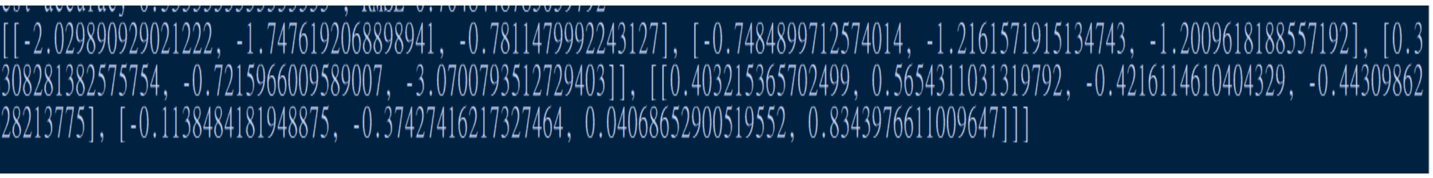


```
epoch 100
train -> accuracy: 0.5364806866952789, RMSE 0.6969693530787385
test accuracy 0.5149501661129569 , RMSE 0.6989650923987827
[[[0.40253190451752646, 0.5611328403018596, 0.34720201482443797], [1.2390130285449747, 2.7583347610356754, -1.2436753996452776], [0.382
4595544056283, -0.037882129366430574, -0.6683620210814056]], [[0.12401747339025061, -0.39729597091211055, 1.6319339212051907, 0.3328788
955097007], [-0.6500949470427133, -0.5821060150496127, -1.0772601451136936, -1.092378771518617]]]
```

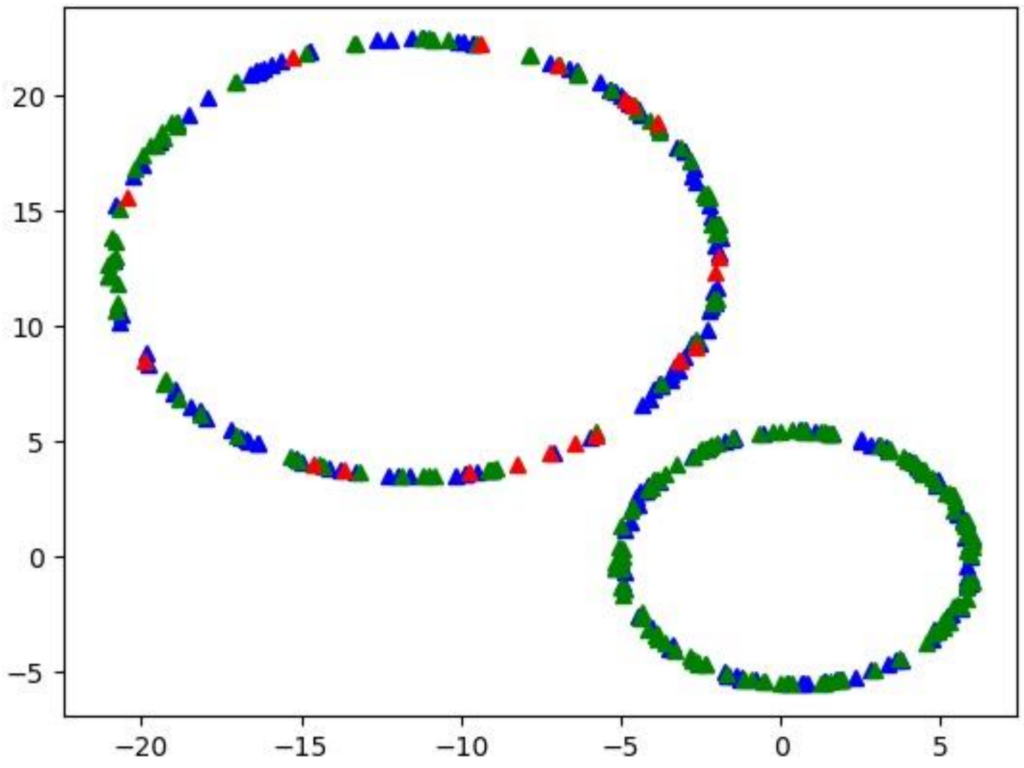


分析:分3類 rmse持續下降 但準確度無法有效上升

- ✓ dataset/2cring.txt

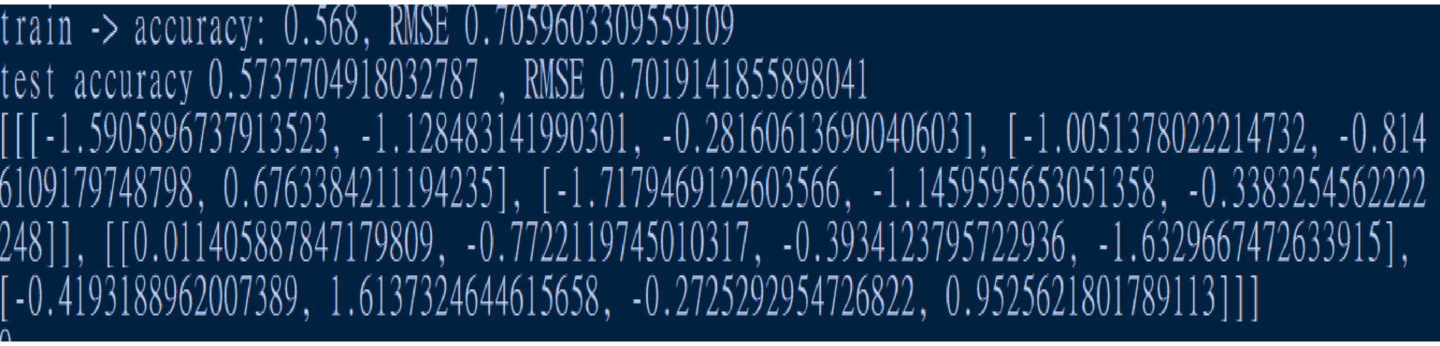


```
[illegible]
[[-2.029890929021222, -1.7476192068898941, -0.7811479992243127], [-0.7484899712574014, -1.2161571915134743, -1.2009618188557192], [0.3
308281382575754, -0.7215966009589007, -3.0700793512729403]], [[0.403215365702499, 0.5654311031319792, -0.4216114610404329, -0.44309862
28213775], [-0.1138484181948875, -0.37427416217327464, 0.04068652900519552, 0.8343976611009647]]]
```

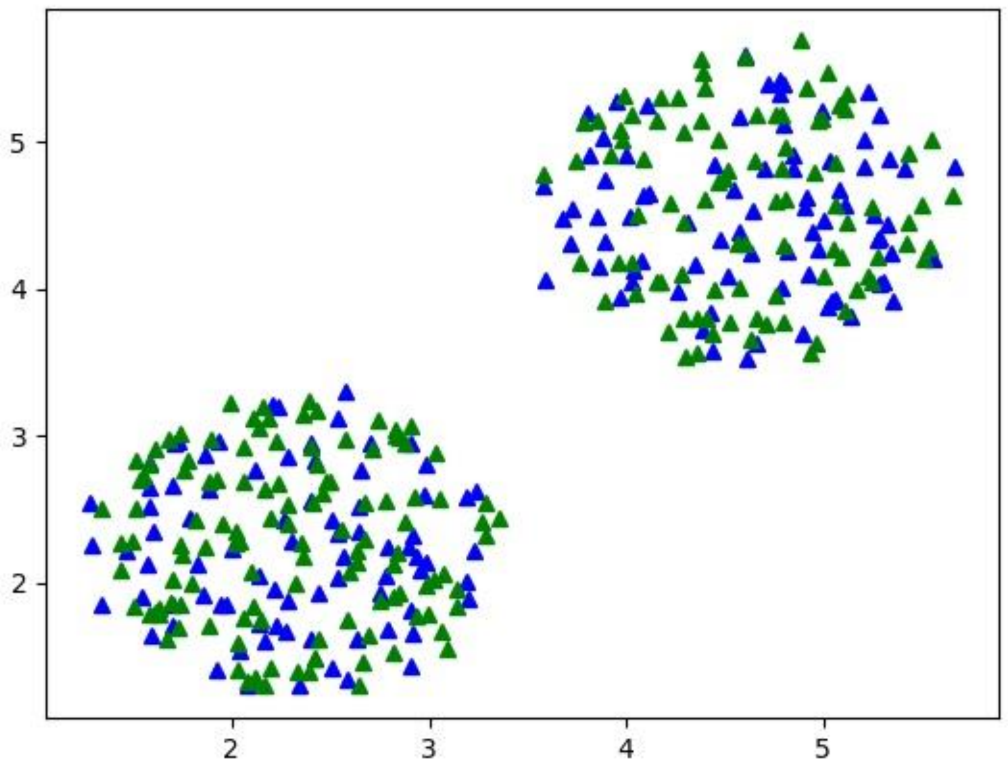


分析:分3類 權重初始化重要 準確度
在0.53上上下下

- ✓ dataset/2cring.txt

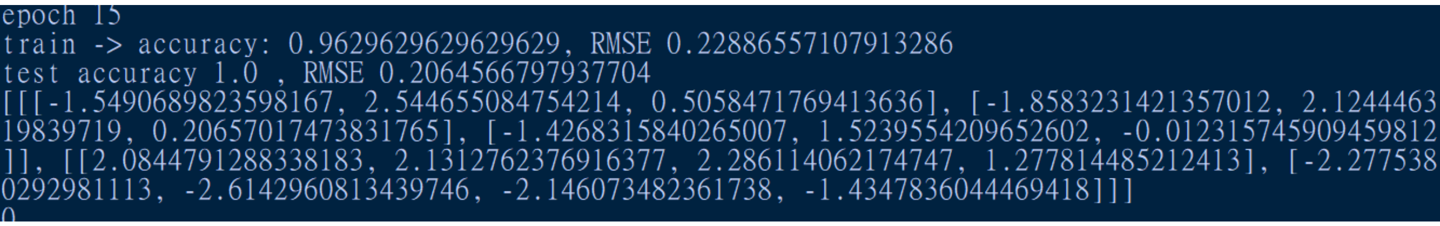


```
train -> accuracy: 0.568, RMSE 0.7059603309559109
test accuracy 0.5737704918032787 , RMSE 0.7019141855898041
[[[-1.5905896737913523, -1.128483141990301, -0.28160613690040603], [-1.0051378022214732, -0.814
6109179748798, 0.6763384211194235], [-1.7179469122603566, -1.1459595653051358, -0.3383254562222
248]], [[0.011405887847179809, -0.7722119745010317, -0.3934123795722936, -1.632966747263391 5],
[-0.4193188962007389, 1.6137324644615658, -0.2725292954726822, 0.9525621801789113]]]
```

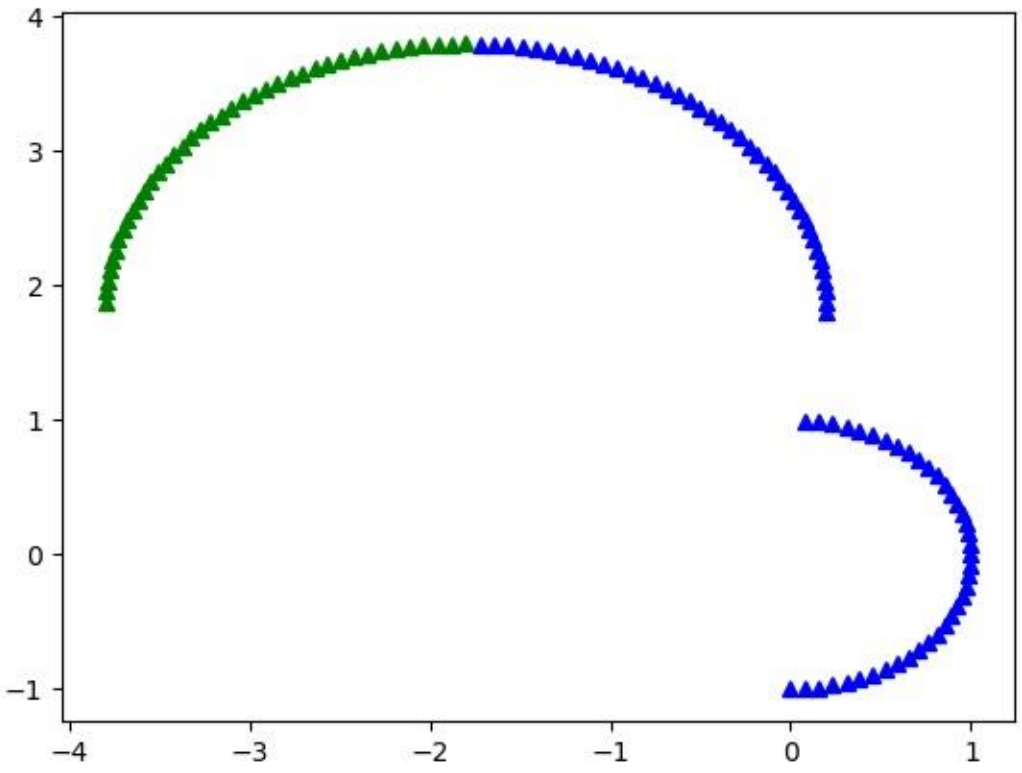


分析:分2類 rmse持續下降 但準確度無法有效上升

- ✓ dataset/2Hcircle1.txt

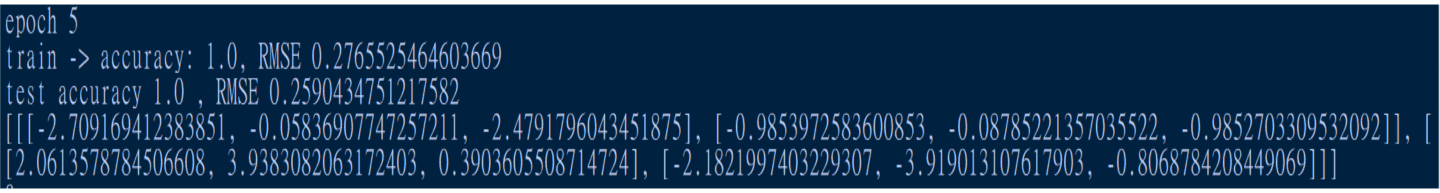


```
epoch 15
train -> accuracy: 0.9629629629629629, RMSE 0.22886557107913286
test accuracy 1.0 , RMSE 0.2064566797937704
[[[-1.5490689823598167, 2.544655084754214, 0.5058471769413636], [-1.8583231421357012, 2.1244463
19839719, 0.206570174738317 65], [-1.4268315840265007, 1.5239554209652602, -0.012315745909459812
]], [[2.0844791288338183, 2.1312762376916377, 2.286114062174747, 1.277814485212413], [-2.277538
0292981113, -2.6142960813439746, -2.146073482361738, -1.4347836044469418]]]
0
```

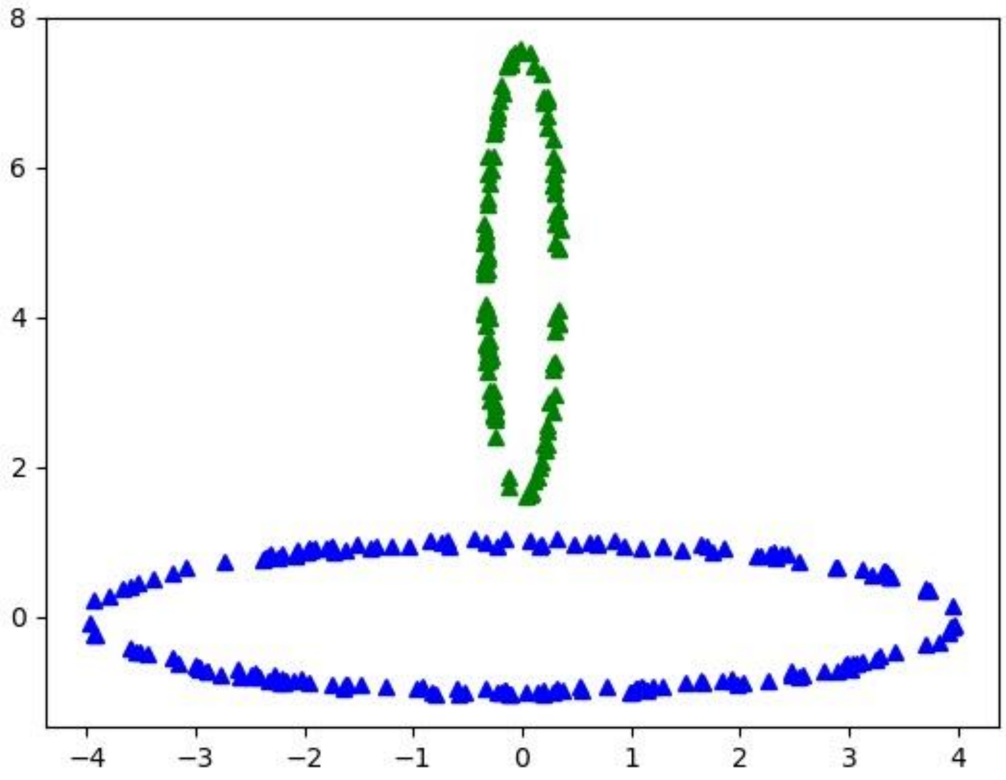


分析:分2類 準確率從0.47
上升至1 epoch15就結束
了

- ✓ dataset/2ring.txt

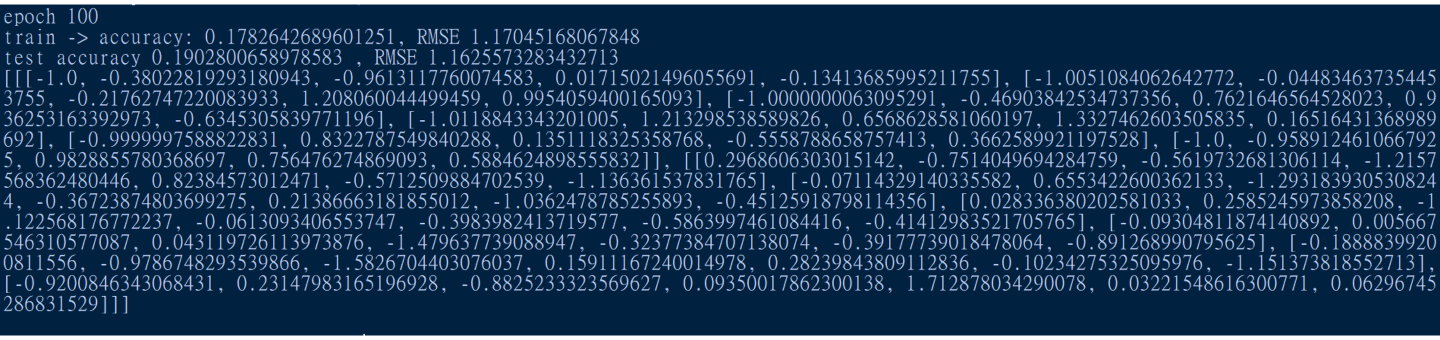


```
epoch 5
train -> accuracy: 1.0, RMSE 0.2765525464603669
test accuracy 1.0 , RMSE 0.2590434751217582
[[[-2.709169412383851, -0.05836907747257211, -2.4791796043451875], [-0.9853972583600853, -0.08785221357035522, -0.9852703309532092]], [
[2.0613578784506608, 3.9383082063172403, 0.3903605508714724], [-2.1821997403229307, -3.919013107617903, -0.8068784208449069]]]
```



分析:分2類 準確率從0.42 上升至1 epoch5就結束了

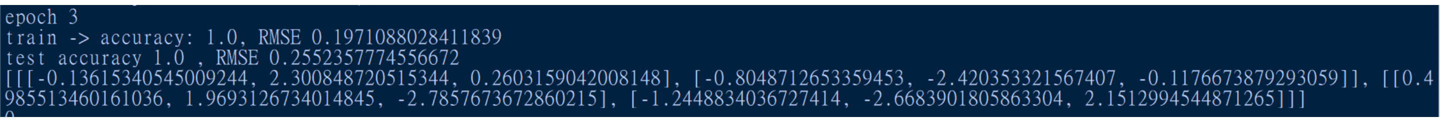
- ✓ dataset/4satellite-6.txt



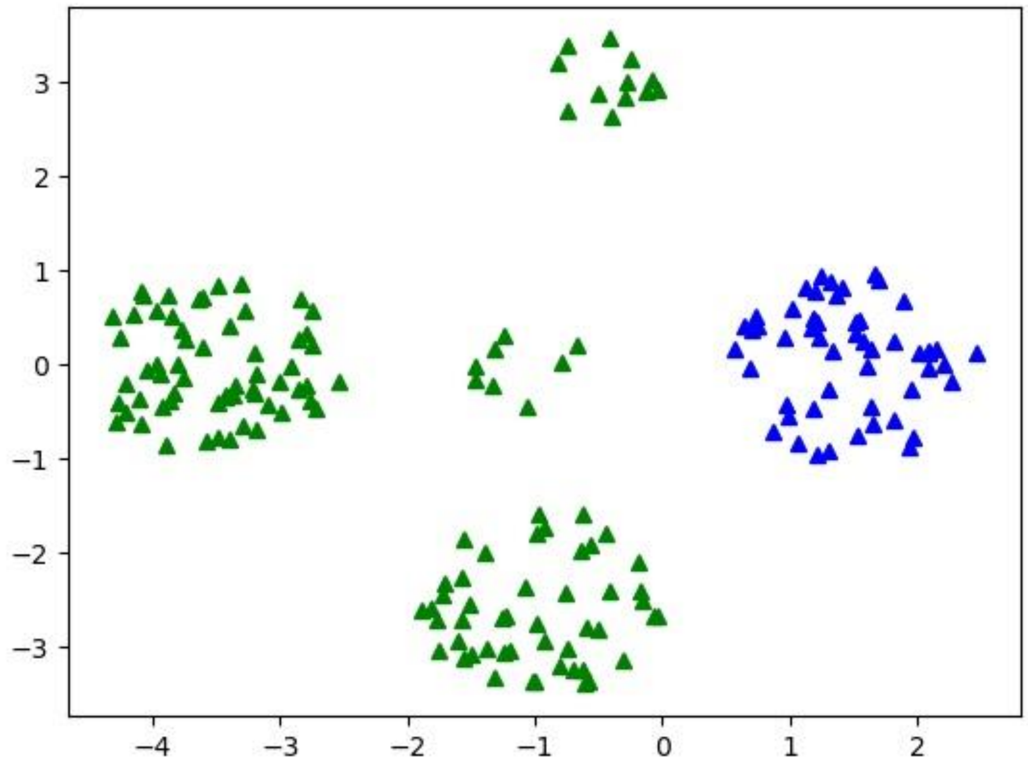
```
epoch 100
train -> accuracy: 0.1782642689601251, RMSE 1.17045168067848
test accuracy 0.1902800658978583 , RMSE 1.1625573283432713
[[[-1.0, -0.38022819293180943, -0.9613117760074583, 0.01715021496055691, -0.13413685995211755], [-1.0051084062642772, -0.04483463735445
3755, -0.21762747220083933, 1.208060044499459, 0.9954059400165093], [-1.0000000063095291, -0.46903842534737356, 0.7621646564528023, 0.9
36253163392973, -0.6345305839771196], [-1.0118843343201005, 1.213298538589826, 0.6568628581060197, 1.3327462603505835, 0.16516431368989
692], [-0.9999997588822831, 0.8322787549840288, 0.1351118325358768, -0.5558788658757413, 0.3662589921197528], [-1.0, -0.958912461066792
5, 0.9828855780368697, 0.756476274869093, 0.5884624898555832]], [[0.2968606303015142, -0.7514049694284759, -0.5619732681306114, -1.2157
568362480446, 0.82384573012471, -0.5712509884702539, -1.1363615378317765], [-0.07114329140335582, 0.6553422600362133, -1.293183930530824
4, -0.36723874803699275, 0.2138666318185501 2, -1.0362478785255893, -0.45125918798114356], [0.02833638020258103 3, 0.2585245973858208, -1
.122568176772237, -0.0613093406553747, -0.3983982413719577, -0.5863997461084416, -0.41412983521705765], [-0.09304811874140892, 0.005667
546310577087, 0.04311972611397387 6, -1.479637739088947, -0.32377384707138074, -0.39177739018478064, -0.891268990795625], [-0.1888839920
0811556, -0.9786748293539866, -1.5826704403076037, 0.15911167240014978, 0.28239843809112836, -0.10234275325095976, -1.151373818552713],
[-0.9200846343068431, 0.23147983165196928, -0.8825233323569627, 0.09350017862300138, 1.712878034290078, 0.03221548616300771, 0.06296745
286831529]]]
```

分析:分6類 速度非常慢 rmse下降緩慢
準確率無提升

- ✓ dataset/2ring.txt

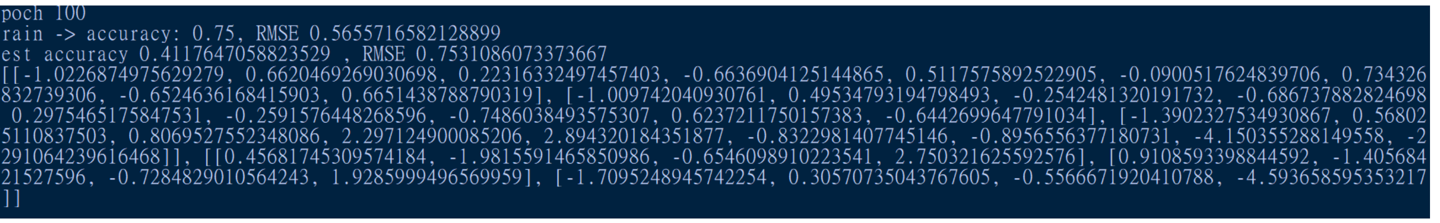


```
epoch 3
train -> accuracy: 1.0, RMSE 0.1971088028411839
test accuracy 1.0 , RMSE 0.2552357774556672
[[[-0.13615340545009244, 2.300848720515344, 0.2603159042008148], [-0.8048712653359453, -2.420353321567407, -0.1176673879293059]], [[0.4
985513460161036, 1.9693126734014845, -2.7857673672860215], [-1.2448834036727414, -2.6683901805863304, 2.1512994544871265]]]
```



分析:分2類 準確率從0.66 上升至1
epoch3就結束了

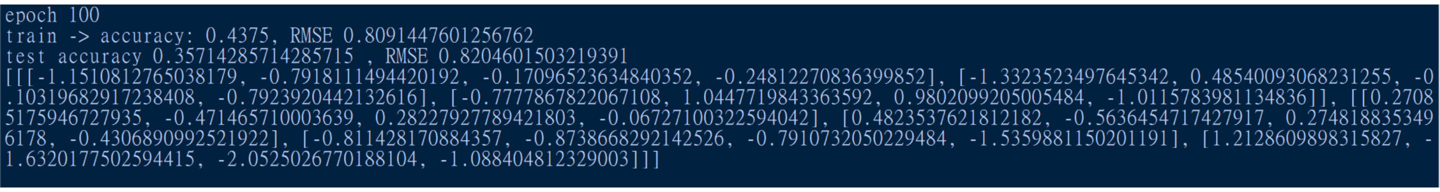
- ✓ dataset/8OX.txt



```
poch 100
rain -> accuracy: 0.75, RMSE 0.5655716582128899
est accuracy 0.4117647058823529 , RMSE 0.7531086073373667
[[-1.0226874975629279, 0.6620469269030698, 0.22316332497457403, -0.6636904125144865, 0.5117575892522905, -0.0900517624839706, 0.734326
832739306, -0.6524636168415903, 0.6651438788790319], [-1.009742040930761, 0.49534793194798493, -0.2542481320191732, -0.6867378824698
 0.2975465175847531, -0.2591576448268596, -0.7486038493575307, 0.6237211750157383, -0.6442699647791034], [-1.3902327534930867, 0.56802
5110837503, 0.8069527552348086, 2.297124900085206, 2.894320184351877, -0.8322981407745146, -0.8956556377180731, -4.150355288149558, -2
291064239616468]], [[0.45681745309574184, -1.9815591465850986, -0.6546098910223541, 2.750321625592576], [0.9108593398844592, -1.405684
21527596, -0.7284829010564243, 1.9285999496569959], [-1.7095248945742254, 0.30570735043767605, -0.5566671920410788, -4.593658595353217
]]
```

分析:rmse持續下降 但準確度無法有效上升 只上升至0.41

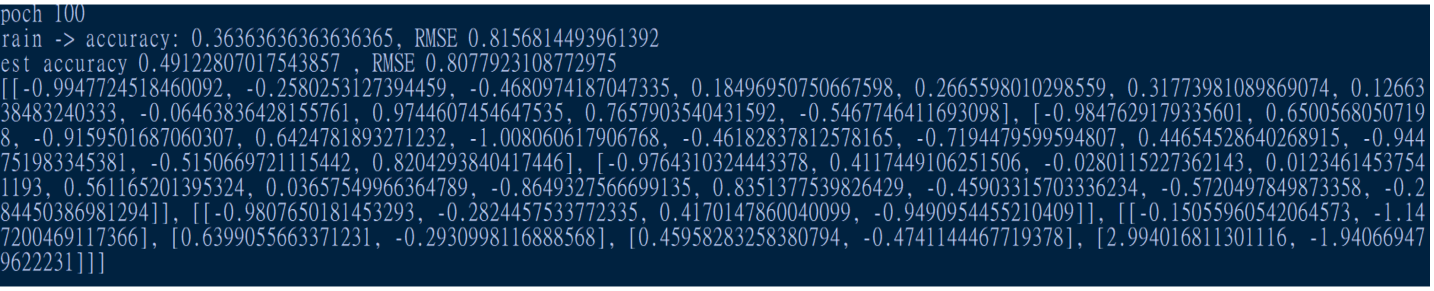
- ✓ dataset/C3D.txt



```
epoch 100
train -> accuracy: 0.4375, RMSE 0.8091447601256762
test accuracy 0.35714285714285715 , RMSE 0.8204601503219391
[[[-1.1510812765038179, -0.7918111494420192, -0.17096523634840352, -0.24812270836399852], [-1.3323523497645342, 0.48540093068231255, -0
.10319682917238408, -0.7923920442132616], [-0.7777867822067108, 1.0447719843363592, 0.9802099205005484, -1.0115783981134836]], [[0.2708
5175946727935, -0.471465710003639, 0.28227927789421803, -0.06727100322594042], [0.4823537621812182, -0.5636454717427917, 0.274818835349
6178, -0.4306890992521922], [-0.811428170884357, -0.8738668292142526, -0.7910732050229484, -1.5359881150201191], [1.2128609898315827, -
1.6320177502594415, -2.0525026770188104, -1.088404812329003]]]
```

分析:rmse上上下下 且準確度無法有效上升 只上升至0.357

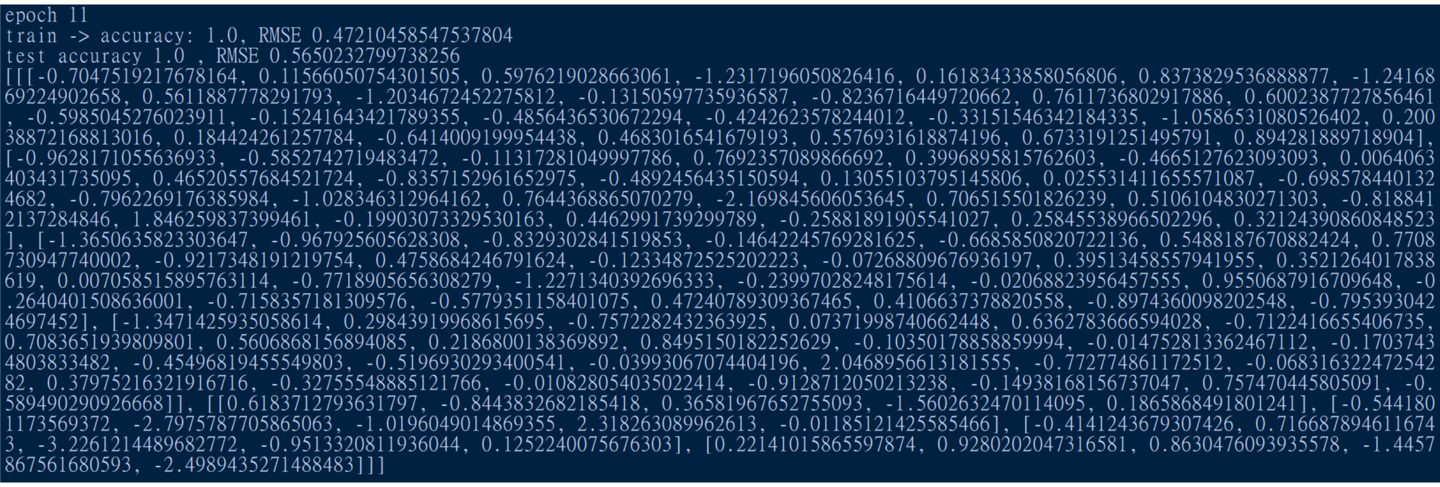
- ✓ dataset/C10D.txt



```
poch 100
rain -> accuracy: 0.36363636363636365, RMSE 0.8156814493961392
est accuracy 0.49122807017543857 , RMSE 0.8077923108772975
[[-0.9947724518460092, -0.2580253127394459, -0.4680974187047335, 0.18496950750667598, 0.2665598010298559, 0.31773981089869074, 0.12663
38483240333, -0.06463836428155761, 0.9744607454647535, 0.7657903540431592, -0.5467746411693098], [-0.9847629179335601, 0.6500568050719
8, -0.9159501687060307, 0.6424781893271232, -1.008060617906768, -0.46182837812578165, -0.7194479599594807, 0.44654528640268915, -0.944
751983345381, -0.5150669721115442, 0.8204293840417446], [-0.9764310324443378, 0.4117449106251506, -0.0280115227362143, 0.0123461453754
1193, 0.561165201395324, 0.03657549966364789, -0.8649327566699135, 0.8351377539826429, -0.45903315703336234, -0.5720497849873358, -0.2
84450386981294]], [[-0.9807650181453293, -0.2824457533772335, 0.4170147860040099, -0.9490954455210409]], [[-0.15055960542064573, -1.14
7200469117366], [0.6399055663371231, -0.2930998116888568], [0.45958283258380794, -0.4741144467719378], [2.994016811301116, -1.94066947
9622231]]]
```

分析:分4類 rmse持四下降 但準確度無法有效上升 只上升至0.49

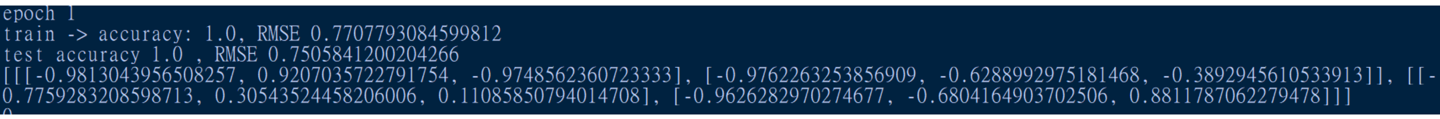
✓ dataset/Number.txt



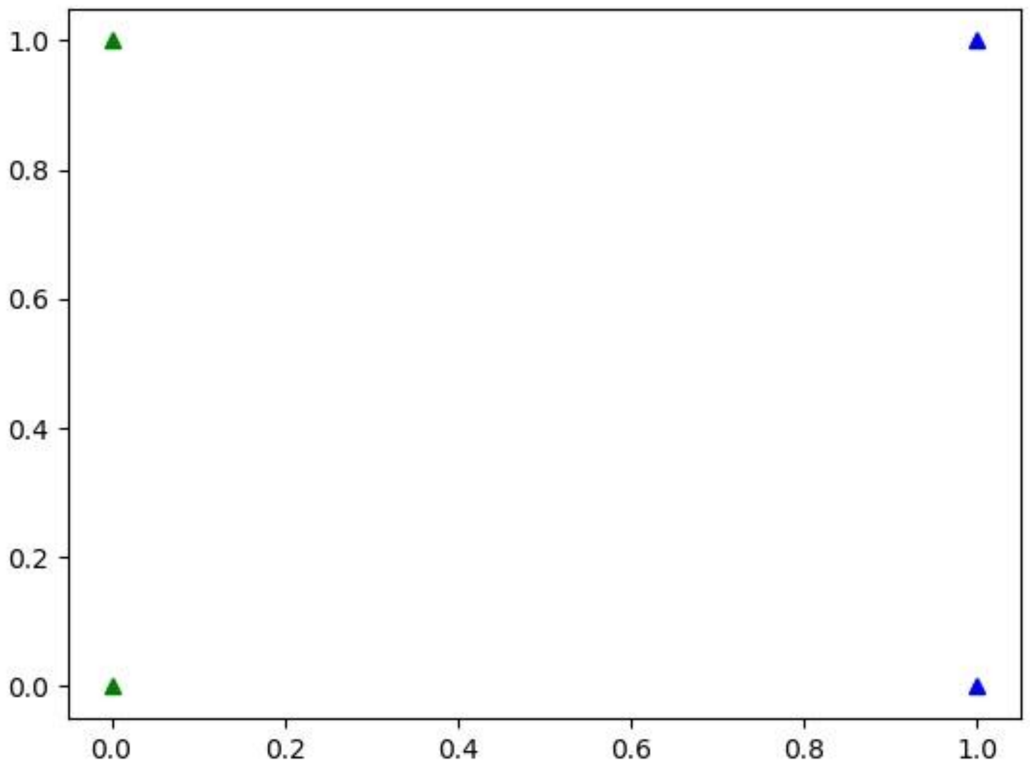
```
epoch 11
train -> accuracy: 1.0, RMSE 0.47210458547537804
test accuracy 1.0 , RMSE 0.5650232799738256
[[[-0.7047519217678164, 0.11566050754301505, 0.5976219028663061, -1.2317196050826416, 0.16183433858056806, 0.8373829536888877, -1.24168
69224902658, 0.5611887778291793, -1.2034672452275812, -0.13150597735936587, -0.8236716449720662, 0.7611736802917886, 0.6002387727856461
, -0.5985045276023911, -0.15241643421789355, -0.4856436530672294, -0.4242623578244012, -0.33151546342184335, -1.0586531080526402, 0.200
38872168813016, 0.184424261257784, -0.6414009199954438, 0.4683016541679193, 0.5576931618874196, 0.6733191251495791, 0.894281889718904],
[-0.9628171055636933, -0.5852742719483472, -0.11317281049997786, 0.7692357089866692, 0.3996895815762603, -0.4665127623093093, 0.0064063
403431735095, 0.46520557684521724, -0.8357152961652975, -0.4892456435150594, 0.13055103795145806, 0.025531411655571087, -0.698578440132
4682, -0.7962269176385984, -1.028346312964162, 0.7644368865070279, -2.169845606053645, 0.706515501826239, 0.5106104830271303, -0.818841
2137284846, 1.846259837399461, -0.19903073329530163, 0.4462991739299789, -0.25881891905541027, 0.25845538966502296, 0.32124390860848523
], [-1.3650635823303647, -0.967925605628308, -0.8329302841519853, -0.14642245769281625, -0.6685850820722136, 0.5488187670882424, 0.7708
730947740002, -0.9217348191219754, 0.4758684246791624, -0.12334872525202223, -0.07268809676936197, 0.39513458557941955, 0.3521264017838
619, 0.007058515895763114, -0.7718905656308279, -1.2271340392696333, -0.23997028248175614, -0.02068823956457555, 0.9550687916709648, -0
.2640401508636001, -0.7158357181309576, -0.5779351158401075, 0.47240789309367465, 0.4106637378820558, -0.8974360098202548, -0.795393042
4697452], [-1.3471425935058614, 0.29843919968615695, -0.7572282432363925, 0.07371998740662448, 0.6362783666594028, -0.7122416655406735,
0.7083651939809801, 0.5606868156894085, 0.2186800138369892, 0.8495150182252629, -0.10350178858859994, -0.014752813362467112, -0.1703743
4803833482, -0.45496819455549803, -0.5196930293400541, -0.03993067074404196, 2.0468956613181555, -0.772774861172512, -0.068316322472542
82, 0.37975216321916716, -0.32755548885121766, -0.010828054035022414, -0.9128712050213238, -0.14938168156737047, 0.757470445805091, -0.
589490290926668]], [[0.6183712793631797, -0.8443832682185418, 0.36581967652755093, -1.5602632470114095, 0.1865868491801241], [-0.544180
1173569372, -2.7975787705865063, -1.0196049014869355, 2.318263089962613, -0.011851214255855466], [-0.4141243679307426, 0.716687894611674
3, -3.2261214489682772, -0.9513320811936044, 0.1252240075676303], [0.22141015865597874, 0.9280202047316581, 0.8630476093935578, -1.4457
867561680593, -2.4989435271488483]]]
```

分析:分4類 複製Number.txt多筆資料
準確率從0.23 上升至1 epoch11就結束
了

- ✓ dataset/perceptron1.txt

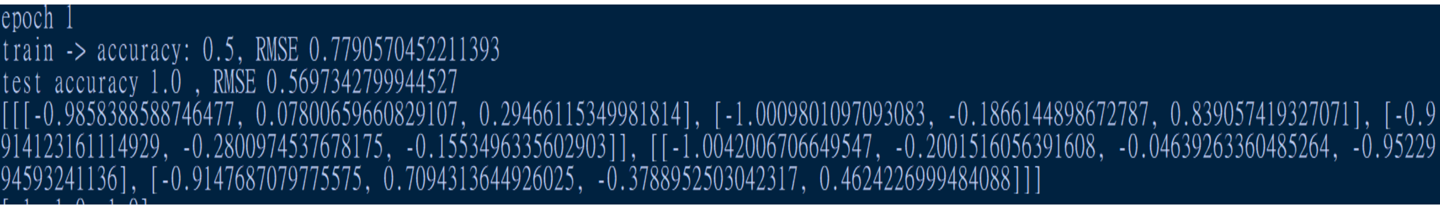


```
epoch 1
train -> accuracy: 1.0, RMSE 0.7707793084599812
test accuracy 1.0 , RMSE 0.7505841200204266
[[[-0.9813043956508257, 0.9207035722791754, -0.9748562360723333], [-0.9762263253856909, -0.6288992975181468, -0.3892945610533913]], [[-
0.7759283208598713, 0.30543524458206006, 0.11085850794014708], [-0.9626282970274677, -0.6804164903702506, 0.8811787062279478]]]
```

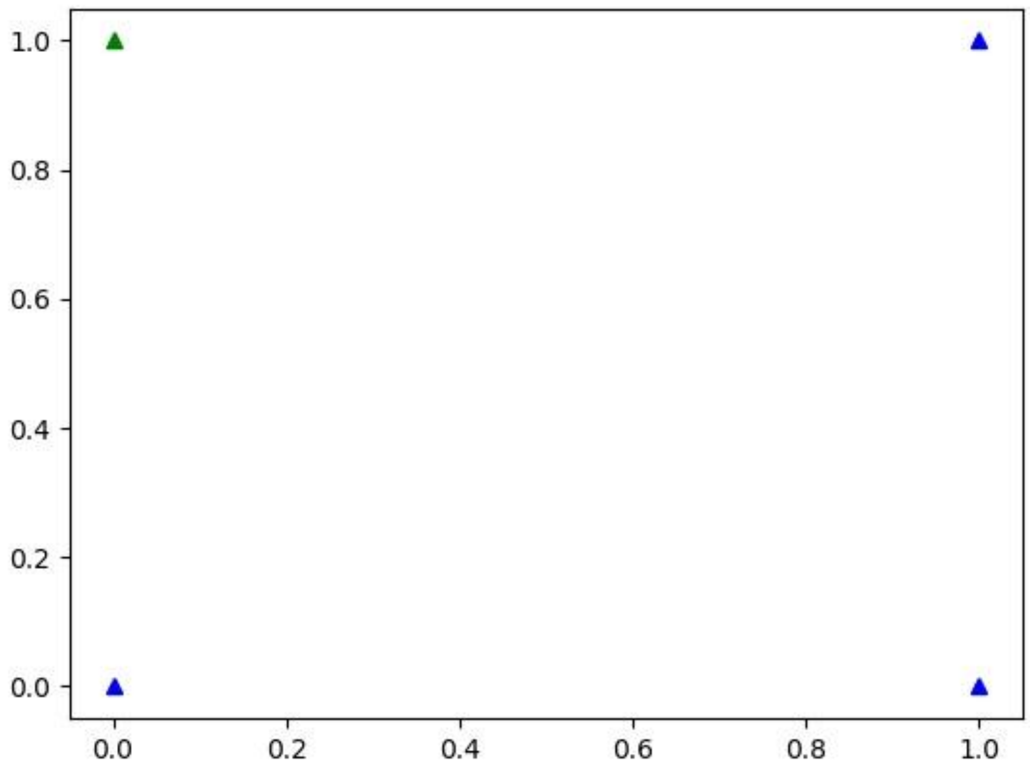


分析:分2類 epoch1準確率就是1

- ✓ dataset/perceptron2.txt

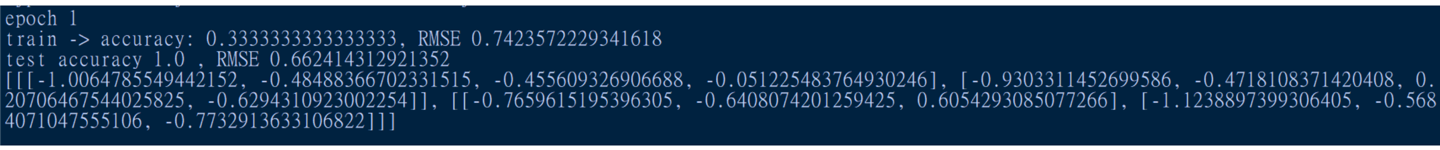


```
epoch 1
train -> accuracy: 0.5, RMSE 0.7790570452211393
test accuracy 1.0 , RMSE 0.5697342799944527
[[[-0.9858388588746477, 0.07800659660829107, 0.29466115349981814], [-1.0009801097093083, -0.1866144898672787, 0.8390574193270711], [-0.9
914123161114929, -0.2800974537678175, -0.1553496335602903]], [[-1.0042006706649547, -0.2001516056391608, -0.04639263360485264, -0.95229
94593241136], [-0.9147687079775575, 0.7094313644926025, -0.3788952503042317, 0.4624226999484088]]]
```



分析:分2類 資料分類很重要

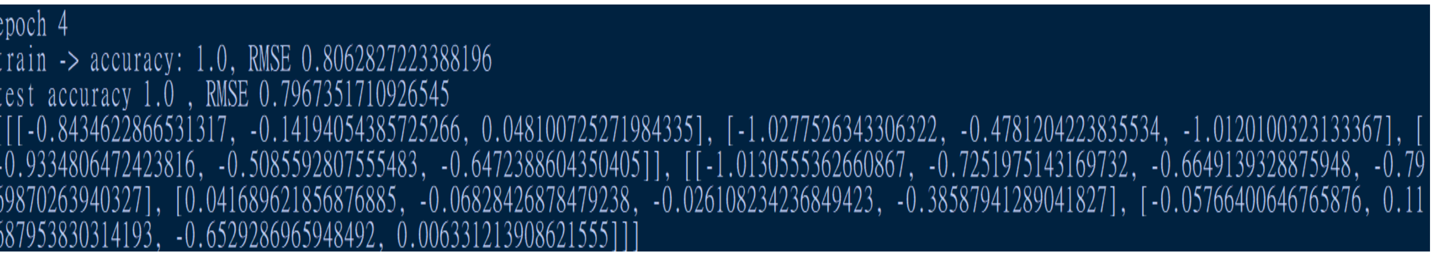
- ✓ dataset/perceptron3.txt



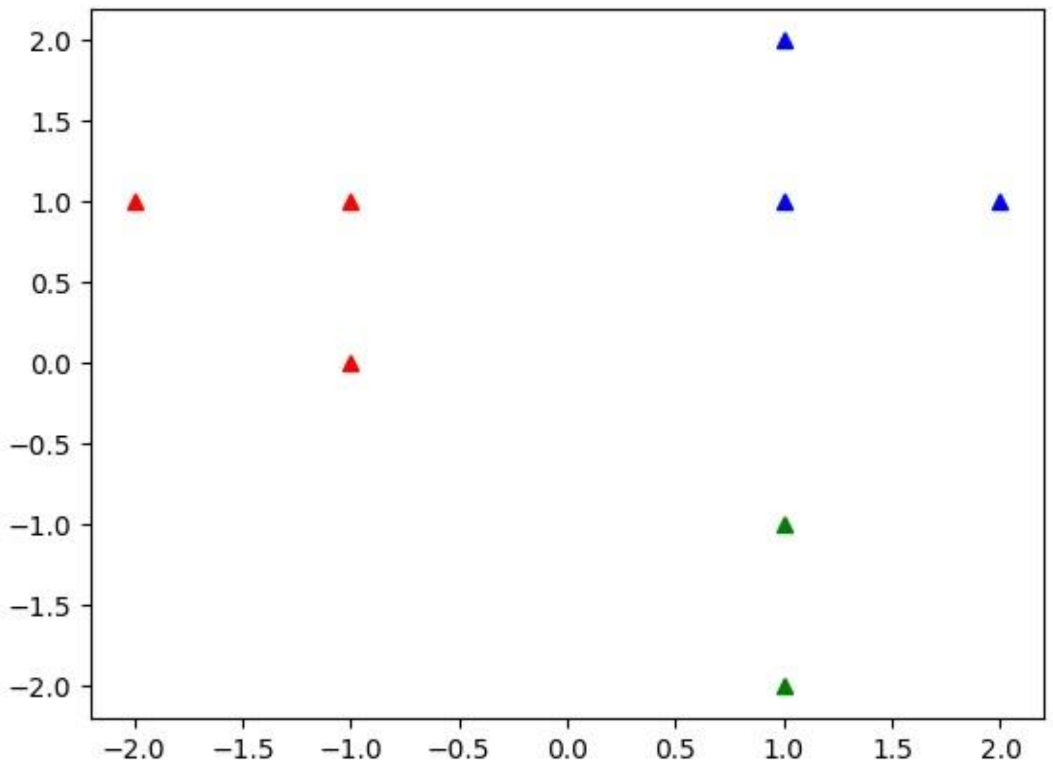
```
epoch 1
train -> accuracy: 0.3333333333333333, RMSE 0.7423572229341618
test accuracy 1.0 , RMSE 0.662414312921352
[[[-1.0064785549442152, -0.48488366702331515, -0.455609326906688, -0.051225483764930246], [-0.9303311452699586, -0.4718108371420408, 0.
20706467544025825, -0.6294310923002254]], [[-0.7659615195396305, -0.6408074201259425, 0.6054293085077266], [-1.1238897399306405, -0.568
4071047555106, -0.7732913633106822]]]
```

分析:分2類 資料分類很重要 epoch1 test 準確率就是1

- ✓ dataset/perceptron4.txt

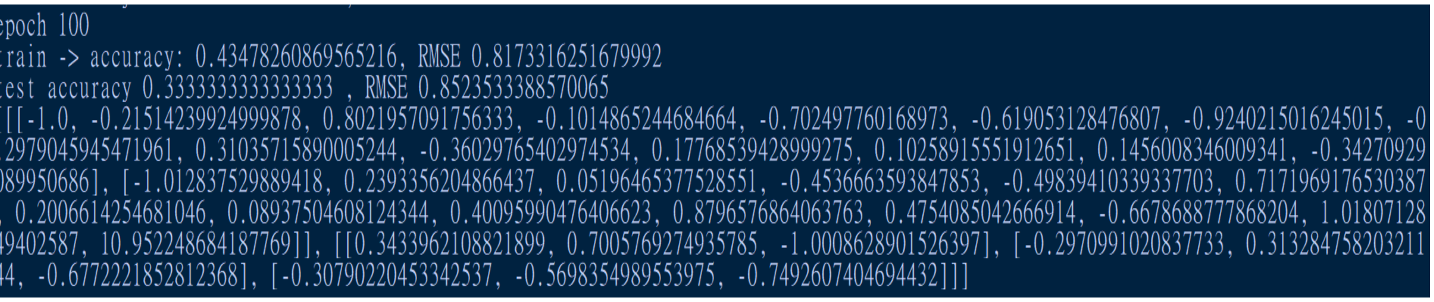


```
epoch 4
train -> accuracy: 1.0, RMSE 0.8062827223388196
test accuracy 1.0 , RMSE 0.7967351710926545
[[[-0.8434622866531317, -0.14194054385725266, 0.048100725271984335], [-1.0277526343306322, -0.4781204223835534, -1.0120100323133367], [
-0.9334806472423816, -0.5085592807555483, -0.6472388604350405]], [[-1.0130555362660867, -0.7251975143169732, -0.6649139328875948, -0.79
59870263940327], [0.041689621856876885, -0.06828426878479238, -0.026108234236849423, -0.38587941289041827], [-0.05766400646765876, 0.11
687953830314193, -0.6529286965948492, 0.006331213908621555]]]
```



分析:分3類 資料分類很重要 epoch1
test 準確率就是1

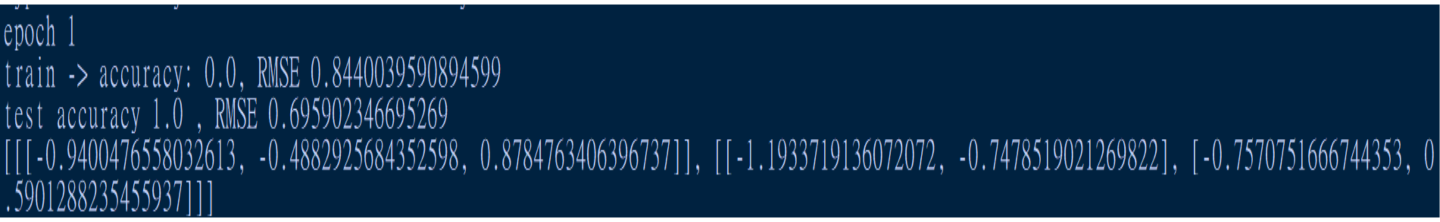
- ✓ dataset/wine.txt



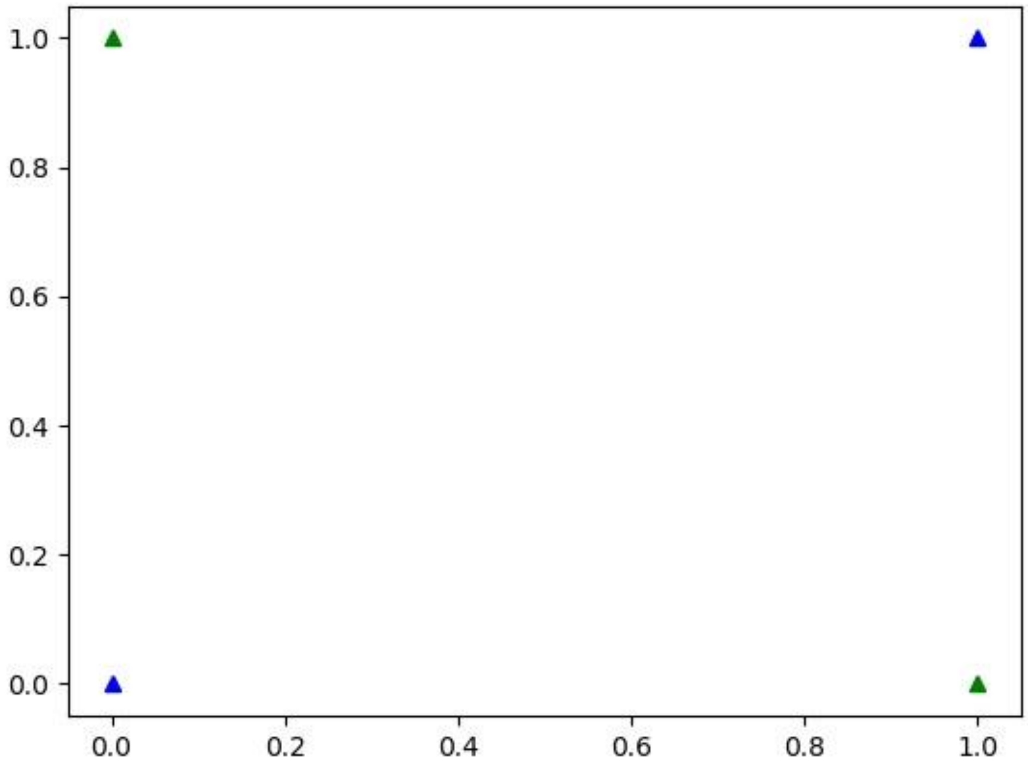
```
epoch 100
train -> accuracy: 0.43478260869565216, RMSE 0.8173316251679992
test accuracy 0.3333333333333333 , RMSE 0.8523533388570065
[[[-1.0, -0.21514239924999878, 0.8021957091756333, -0.1014865244684664, -0.702497760168973, -0.619053128476807, -0.9240215016245015, -0
.2979045945471961, 0.31035715890005244, -0.36029765402974534, 0.17768539428999275, 0.10258915551912651, 0.1456008346009341, -0.34270929
089950686], [-1.012837529889418, 0.2393356204866437, 0.05196465377528551, -0.4536663593847853, -0.49839410339337703, 0.7171969176530387
, 0.2006614254681046, 0.08937504608124344, 0.40095990476406623, 0.8796576864063763, 0.4754085042666914, -0.6678688777868204, 1.01807128
49402587, 10.952248684187769]], [[0.3433962108821899, 0.7005769274935785, -1.0008628901526397], [-0.2970991020837733, 0.313284758203211
44, -0.6772221852812368], [-0.30790220453342537, -0.5698354989553975, -0.7492607404694432]]]
```

分析: rmse持四下降 但準確度無法有效上升 只上升至0.33

- ✓ dataset/xor.txt



```
epoch 1
train -> accuracy: 0.0, RMSE 0.8440039590894599
test accuracy 1.0 , RMSE 0.695902346695269
[[[-0.9400476558032613, -0.4882925684352598, 0.8784763406396737]], [[-1.1933719136072072, -0.7478519021269822], [-0.7570751666744353, 0
.5901288235455937]]]
```



分析:分2類 資料分類很重要 epoch1
test 準確率就是1

- 實驗分析
  - ✓ 從以上實驗結果可得知
    1. 銓重初始化與資料分割十分重要
    2. 有些資料增加神經元也無法增加準確度
    3. 有些資料增加隱藏層也無法增加準確度
    4. 有些資料rmse下降，準確度無法有效上升

- 加分項目
  1. 能夠處理多維資料
  2. 能夠處理多群資料
  3. 隱藏層層數可設定
  4. 隱藏層的神經元個數可設定
  5. Demo 影片