實作架構：hopfield網路

一、程式執行說明

1. 打開程式執行檔

2. 按下選擇訓練資料按鈕並選擇訓練資料(必須先按此按鈕再按測試資料按鈕)

3. 按下選擇測試資料按鈕並選擇測試資料

4. 測試測資的最後回想回顯示在右方

二、程式簡介

1. 這是選擇訓練資料按鈕的觸發函數

def select\_train():

    flag = True

    global w,o

    w = np.zeros((108,108))

    templists1 = []

    templists2 = []

    a = np.zeros((1,108))

    o = np.zeros((108,1))

    linenumber = 0

    trainnumber = 0

    path = easygui.fileopenbox()

    file = open(path, mode="r")

    for line in file:

        templists1 += line

        linenumber += 1

        if(linenumber == 12):

//將訓練資料以12列為單位拆開

            trainnumber += 1

            for i in range(len(templists1)):

                if(templists1[i] != '\n'):

                    templists2 += templists1[i]

            for i in range(len(templists2)):

                if(templists2[i] == ' '):

                    templists2[i] = '-1'

            for i in range(len(templists2)):

                a[0][i]+=int(templists2[i])

            b = a[0].reshape(a[0].shape[0],1)

//b為a的轉置矩陣即是有108個元素的列矩陣

            c = b.dot(a)

//將所有訓練測資的矩陣與自己的轉置矩陣相乘成為108\*108的矩陣並加到w中

            w = w+c

            templists1 = []

            templists2 = []

            a = np.zeros((1,108))

            linenumber = -1

    I = np.identity(108)

//這步驟是最重要的將w經過以下轉換變為最後鍵結值(108是維度,trainnumber是訓練測資數量)

    w =  1/108\*w - (trainnumber/108\*I)

//o則是要與之後仄是資料比對看看應該要變為1 or -1 或維持原本數值不變的比較矩陣

    for i in range(108):

        for j in range(108):

            o[i][0] += w[i][j]

2. 這是選擇測試資料按鈕的觸發函數

def select\_test():

    global w,o

    templists1 = []

    templists2 = []

    linenumber = 0

    a = np.zeros((1,108))

    path = easygui.fileopenbox()

    file = open(path, mode="r")

    for line in file:

        templists1 += line

        linenumber += 1

        if(linenumber == 12):

            for i in range(len(templists1)):

                if(templists1[i] != '\n'):

                    templists2 += templists1[i]

            for i in range(len(templists2)):

                if(templists2[i] == ' '):

                    templists2[i] = '-1'

            for i in range(len(templists2)):

                a[0][i]+=int(templists2[i])

            b = a[0].reshape(a[0].shape[0],1)

            a = a[0].reshape(a[0].shape[0],1)

            number = 0

//印出再回想前的測試資料模樣

            for i in range(12):

                for j in range(9):

                    if(a[number][0]==1):

                        text.insert("insert","1")

                    else:

                        text.insert("insert"," ")

                    number+=1

                text.insert("insert",'\n')

            number = 0

            c = w.dot(b)

            while(1):

                falsenumber = True

//一直做回想的動作直到前一個回想出得值當前回想出的值相同

                for i in range(108):

//回想出的值去跟o陣列做比較如果比矩陣o的對應值小則c對應的值改成-1，如果較大則改成1，如果不便則維持跟b矩陣對應項相同

                    if(c[i][0]<o[i][0]):

                        c[i][0] = -1

                    elif(c[i][0]>o[i][0]):

                        c[i][0] = 1

                    elif(c[i][0] == o[i][0]):

                        c[i][0] = b[i][0]

                for i in range(108):

                    if(b[i][0] != c[i][0]):

                        falsenumber = False

                        break

                if(falsenumber):

                    break

                else:

                    for i in range(108):

                        b[i][0] = c[i][0]

            number = 0

            text.insert("insert","\n")

            text.insert("insert","  回想到\n")

            text.insert("insert","\n")

//印出回想收斂後的結果

            for i in range(12):

                for j in range(9):

                    if(c[number][0]==1):

                        text.insert("insert","1")

                    else:

                        text.insert("insert"," ")

                    number+=1

                text.insert("insert",'\n')

            text.insert("insert","\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

            text.insert("insert",'\n')

            templists1 = []

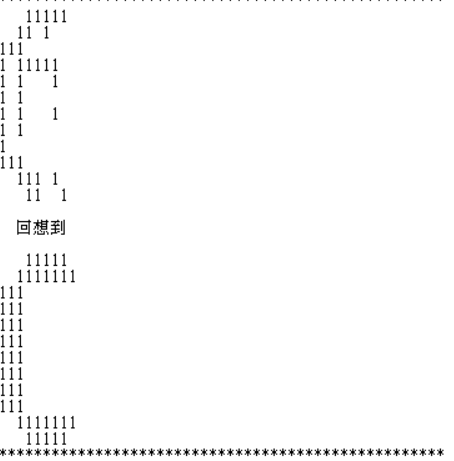
            templists2 = []

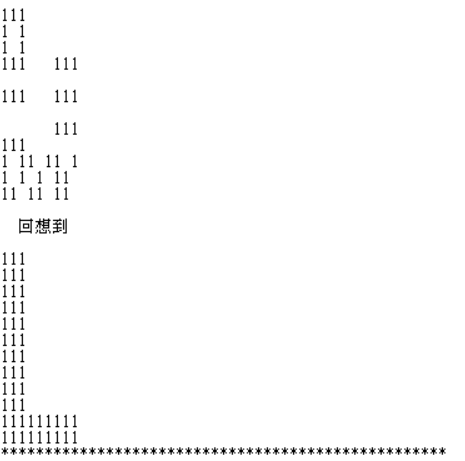
            a = np.zeros((1,108))

            linenumber = -1

三、實驗結果





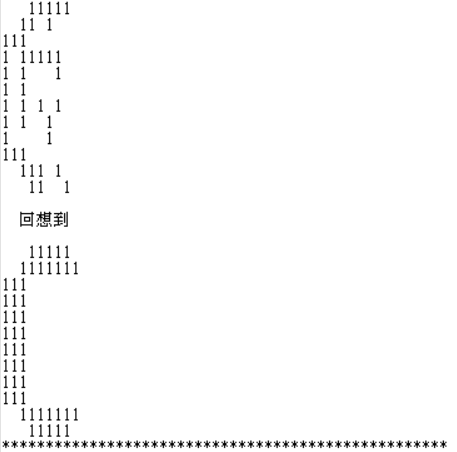


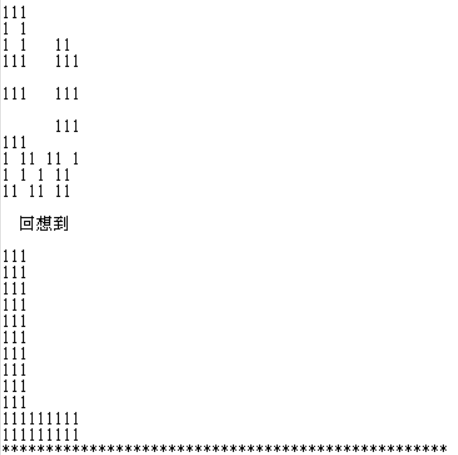
四、實驗結果分析及討論

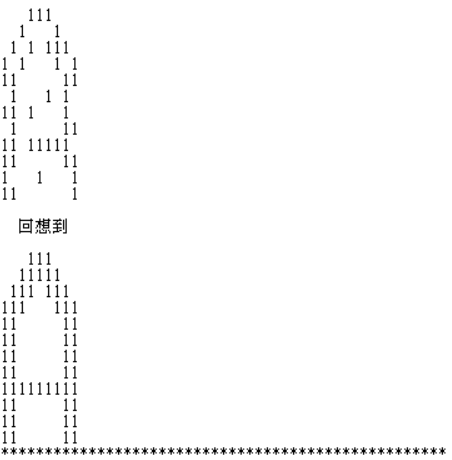
訓練測資的字母序列跟測試測資的回想結果相同(都是A->C->L)，我在放上一個測試測資序列是(C->L->A)的來測試看看結果也是能回想到與訓練資料相符的字母。

五、額外

1. Basic測資加入雜訊







都能正確回想到訓練測資

2. Bonus

因為這是10\*10要用100維的陣列，所以我又寫了另一個程式碼去跑，這次的結果明顯與訓練資料都沒有類似的，但收斂到錯誤的局部極小值（local minimum），而非全局極小（global minimum）的情況也可能發生，bonus的部分測資所發生的情形就是收斂在局部極小值中。

