Artificial Intelligence HW1

By 405402135 林亞辰

1. 首先，你要說明你針對哪個應用？Data set由哪裡取得？輸入是甚麼？有幾個？輸出是甚麼？有幾個？你想用多少組當訓練資料？用多少組當測試資料？(你的老闆也沒用過，所以要說明一下。) 注意：請詳細說明你所使用之機器軟硬體規格、所用的作業系統等相關資訊以及你為何選擇這樣的規格。另外請提供你的連絡電話，以便不時之需。

ans: (1) 此報告所針對的是圓柱體體積作為測試神經元的測資。

(2) Data Set 為實際用公式：

圓柱體積 = (半徑)² × π × 高

搭配 Python 的 Library：math 所得 π = math.pi

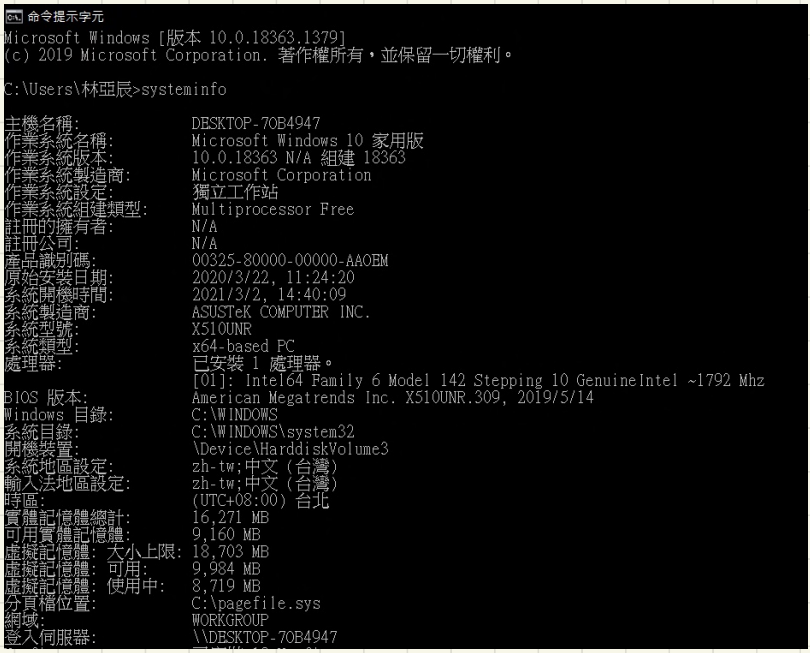
(3,4) 輸入共有3個資料，分別為半徑、高、體積

(5,6) 輸出共有1個布林值，為判斷輸入中的半徑和高所算出來的體是否等於輸入中的體積，若是則輸出1，否則為0。

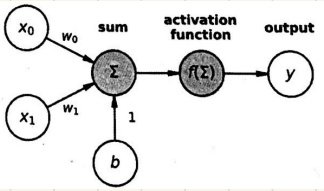
(7) 共用10筆訓練資料，其中半徑和高為Random產生，體積則經前兩項測資實際計算出體積後，奇奴筆資料加乘上-1，偶奴筆資料保持原樣所得(所以輸出結果應為1,0,1,0,…。

(8) 用1組當資試資料，一樣為Random取出，所以可以重複執行確認結果。

(9) 右圖為執行測資的硬體規格：



2. 請用一個神經元(如講義上的，有bias，有activation function)，寫python程式，用來訓練你這個資料集，看花了不同的時間：訓練資料的mean square error結果為何？測試資料的mean square error結果為何？也請說明如何執行你的程式。



此訓練中 Bias 預設為 0（如同老師的 Example）。

實際 training 下：

Training 100 times：

```
**********************************************************
*************  1 Neuron - Sigmoid  ***********************
**********************************************************

============== START  TRAINING ==============
How many times do you want to test the neuron ?
( press 0 to quit ) 100

train_in_volume =
[[ 4.17022005e-01  7.20324493e-01  3.93546449e-02]
 [ 1.14374817e-04  3.02332573e-01 -1.24249783e-09]
 [ 1.46755891e-01  9.23385948e-02  6.24775829e-04]
 [ 1.86260211e-01  3.45560727e-01 -3.76629588e-03]
 [ 3.96767474e-01  5.38816734e-01  2.66479051e-02]
 [ 4.19194514e-01  6.85219500e-01 -3.78277725e-02]
 [ 2.04452250e-01  8.78117436e-01  1.15315122e-02]
 [ 2.73875932e-02  6.70467510e-01 -1.57992091e-04]
 [ 4.17304802e-01  5.58689828e-01  3.05652113e-02]
 [ 1.40386939e-01  1.98101489e-01 -1.22656628e-03]]
want_to_know =
[[0.69232262]
 [0.87638915]
 [0.03926991]]

intial weight =
[[ 0.60148914]
 [ 0.93652315]
 [-0.37315164]]
final weight =
[[ 3.43537468]
 [-1.08586636]
 [ 1.16843301]]

The final prediction is  0.813456856417544
The correct answer is   1
Mean Square Error is  0.017399172208812406
============== END OF TRAINING ==============
```

Training 1000 times：

```
**********************************************************
*************  1 Neuron - Sigmoid  ***********************
**********************************************************

============== START  TRAINING ==============
How many times do you want to test the neuron ?
( press 0 to quit ) 1000

train_in_volume =
[[ 4.17022005e-01  7.20324493e-01  3.93546449e-02]
 [ 1.14374817e-04  3.02332573e-01 -1.24249783e-09]
 [ 1.46755891e-01  9.23385948e-02  6.24775829e-04]
 [ 1.86260211e-01  3.45560727e-01 -3.76629588e-03]
 [ 3.96767474e-01  5.38816734e-01  2.66479051e-02]
 [ 4.19194514e-01  6.85219500e-01 -3.78277725e-02]
 [ 2.04452250e-01  8.78117436e-01  1.15315122e-02]
 [ 2.73875932e-02  6.70467510e-01 -1.57992091e-04]
 [ 4.17304802e-01  5.58689828e-01  3.05652113e-02]
 [ 1.40386939e-01  1.98101489e-01 -1.22656628e-03]]
want_to_know =
[[0.69232262]
 [0.87638915]
 [0.03926991]]

intial weight =
[[ 0.60148914]
 [ 0.93652315]
 [-0.37315164]]
final weight =
[[ 4.32035637]
 [-1.7727213 ]
 [12.82182751]]

The final prediction is  0.87446181895016
The correct answer is   1
Mean Square Error is  0.007879917450651 21
============== END OF TRAINING ==============
```

Training 10000 times：

```
**********************************************************
*************  1 Neuron - Sigmoid  ***********************
**********************************************************

============== START  TRAINING ==============
How many times do you want to test the neuron ?
( press 0 to quit ) 10000

train_in_volume =
[[ 4.17022005e-01  7.20324493e-01  3.93546449e-02]
 [ 1.14374817e-04  3.02332573e-01 -1.24249783e-09]
 [ 1.46755891e-01  9.23385948e-02  6.24775829e-04]
 [ 1.86260211e-01  3.45560727e-01 -3.76629588e-03]
 [ 3.96767474e-01  5.38816734e-01  2.66479051e-02]
 [ 4.19194514e-01  6.85219500e-01 -3.78277725e-02]
 [ 2.04452250e-01  8.78117436e-01  1.15315122e-02]
 [ 2.73875932e-02  6.70467510e-01 -1.57992091e-04]
 [ 4.17304802e-01  5.58689828e-01  3.05652113e-02]
 [ 1.40386939e-01  1.98101489e-01 -1.22656628e-03]]
want_to_know =
[[0.69232262]
 [0.87638915]
 [0.03926991]]

intial weight =
[[ 0.60148914]
 [ 0.93652315]
 [-0.37315164]]
final weight =
[[ 3.89410184]
 [-1.80995068]
 [63.20966777]]

The final prediction is  0.9731948810189608
The correct answer is   1
Mean Square Error is  0.0003592572017938343
============== END OF TRAINING ==============
```

在 READ_ME.txt 中有詳細說明要如何執行此作業的所有程式：

```
============================================================
================= I have the "make" commend ================
============================================================

If your device supports the "make" command, this will be much easier (because my file name is very long...).
You can type "make" in Terminal to see the output of all Python files directly.
You can also type in :
>>>make q2
 # Output the execution result of hw1 question 2

>>>make q3
 # Output the execution result of hw1 question 3

>>>make q5
 # Output the execution result of hw1 question 5

>>>make q6
 # Output the execution result of hw1 question 6

>>>make q8
 # Output the execution result of hw1 question 8

============================================================
================ I do not have the "make" commend ==========
============================================================

If there is no "make" command, you have to enter it one by one :
>>>python HW_1_CylinderVolume_OneNeuron_Sigmoid.py
# Output the execution result of hw1 question 2

>>>python HW_1_CylinderVolume_OneNeuron_Tanh.py
 # Output the execution result of hw1 question 3

>>>python HW_1_CylinderVolume_TwoNeurons_Sigmoid.py
 # Output the execution result of hw1 question 5

>>>python HW_1_CylinderVolume_TwoNeurons_Tanh.py
 # Output the execution result of hw1 question 6

>>>python HW_1_CylinderVolume_ThreeNeurons_Sigmoid.py
 # Output the execution result of hw1 question 8
```

3. 請使用不同的activation functions來做上一項任務，同樣地做說明，並比較有沒有比較好的activation functions？

我嘗試用 tanh（參考 Wikipedia）當 activation function，結果如下：

Training 100 times：

```
**********  1 Neuron - Tanh  **********

============= START  TRAINING =============
How many times do you want to test the neuron ?
( press 0 to quit ) 100

train_in_volume =
[[ 4.17022005e-01  7.20324493e-01  3.93546449e-01]
 [ 1.14374817e-04  3.02332573e-01 -1.24249783e-08]
 [ 1.46755891e-01  9.23385948e-02  6.24775829e-03]
 [ 1.86260211e-01  3.45560727e-01 -3.76629588e-02]
 [ 3.96767474e-01  5.38816734e-01  2.66479051e-01]
 [ 4.19194514e-01  6.85219500e-01 -3.78277725e-01]
 [ 2.04452250e-01  8.78117436e-01  1.15315122e-01]
 [ 2.73875932e-02  6.70467510e-01 -1.57992091e-03]
 [ 4.17304802e-01  5.58689828e-01  3.05652113e-01]
 [ 1.40386939e-01  1.98101489e-01 -1.22656628e-02]]
want_to_know =
[[0.69232262]
 [0.87638915]
 [0.39269908]]

intial weight =
[[0.80074457]
 [0.96826158]
 [0.31342418]]
final weight =
[[ 3.34228189]
 [-0.1176932 ]
 [ 3.68765079]]

The final prediction is  0.998673716953209
The correct answer is   1
Mean Square Error is  8.795133970900969e-07
============= END OF TRAINING =============
```

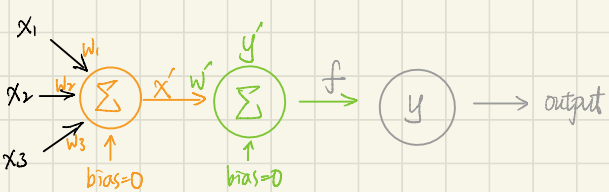
Training 1000 times：

```
**********  1 Neuron - Tanh  **********

============= START  TRAINING =============
How many times do you want to test the neuron ?
( press 0 to quit ) 1000

train_in_volume =
[[ 4.17022005e-01  7.20324493e-01  3.93546449e-01]
 [ 1.14374817e-04  3.02332573e-01 -1.24249783e-08]
 [ 1.46755891e-01  9.23385948e-02  6.24775829e-03]
 [ 1.86260211e-01  3.45560727e-01 -3.76629588e-02]
 [ 3.96767474e-01  5.38816734e-01  2.66479051e-01]
 [ 4.19194514e-01  6.85219500e-01 -3.78277725e-01]
 [ 2.04452250e-01  8.78117436e-01  1.15315122e-01]
 [ 2.73875932e-02  6.70467510e-01 -1.57992091e-03]
 [ 4.17304802e-01  5.58689828e-01  3.05652113e-01]
 [ 1.40386939e-01  1.98101489e-01 -1.22656628e-02]]
want_to_know =
[[0.69232262]
 [0.87638915]
 [0.39269908]]

intial weight =
[[0.80074457]
 [0.96826158]
 [0.31342418]]
final weight =
[[ 3.47891402]
 [-0.13734064]
 [ 3.82114735]]

The final prediction is  0.998976800367583
The correct answer is   1
Mean Square Error is  5.234687438891806e-07
============= END OF TRAINING =============
```

⇒ 由此結果可以發現 Tanh 可以更快速的逼近正確值，但在這之後變化幅度會較小而不易觀察，（從左圖可發現僅為小數點後 7 位之差。）

4. 請設法用兩個神經元 (如講義上的，有bias，有activation function)，來建構類神經網路。請推導backpropagation公式出來。

我所嘗試的方法為：



註：由於 Random 出的值不會為 0，因此直接將 bias 設為 0 較方便。

個人認為此線性的訓練方式可以較容易計算，

而 backpropagation 為：

$$\begin{cases} \frac{dE}{dw_i} = \frac{dE}{dy} \times \frac{dy}{dx'} \times \frac{dx'}{dw_i'} = -(t-y) \times y(1-y) \times x_i' \\ \frac{dE}{dw_i} = \frac{dE}{dy'} \times \frac{dy'}{dx} \times \frac{dx}{dw_i} = -(t-y') \times y'(1-y') \times x_i \end{cases}$$

以 Code 來表示則為：

```
# refine nn_weight
nn_weight_2 += np.dot(np.dot(train_in_volume, nn_weight_1).T, (train_sol_volume -train_out_volume) * dydx(train_out_volume)
train_out_previous = np.dot( train_out_volume, np.linalg.inv(nn_weight_2))
nn_weight_1 += np.dot(train_in_volume.T, (train_sol_volume - train_out_previous) * dydx(train_out_previous))
```

5. 將上一項結論，寫出python程式，來訓練你這個資料集，看花了不同的時間：訓練資料的mean square error結果為何？測試資料的mean square error結果為何？

Training 100 times:

```
**********************************************************
************ 2 Neurons - Sigmoid ******************
**********************************************************


============== START  TRAINING ==============
How many times do you want to test these two neurons ?
( press 0 to quit ) 100

train_in_volume =
[[ 4.17022005e-02  7.20324493e-02  3.93546449e-01]
 [ 1.14374817e-05  3.02332573e-02 -1.24249783e-08]
 [ 1.46755891e-02  9.23385948e-03  6.24775829e-03]
 [ 1.86260211e-02  3.45560727e-02 -3.76629588e-02]
 [ 3.96767474e-02  5.38816734e-02  2.66479051e-01]
 [ 4.19194514e-02  6.85219500e-02 -3.78277725e-01]
 [ 2.04452250e-02  8.78117436e-02  1.15315122e-01]
 [ 2.73875932e-03  6.70467510e-02 -1.57992091e-03]
 [ 4.17304802e-02  5.58689828e-02  3.05652113e-01]
 [ 1.40386939e-02  1.98101489e-02 -1.22656628e-02]]
want_to_know =
[[  0.87638915]
 [  0.89460666]
 [392.6990817 ]]

intial weight 1 =
[[ 0.60148914]
 [ 0.93652315]
 [-0.37315164]]
intial weight 2 =
[[0.38464523]]

final weight 1 =
[[3.5895748 ]
 [6.5036296 ]
 [6.78178184]]
final weight 2 =
[[1.04394646]]
Mean Square Error of training data is  0.07421932792571505

The final prediction is  1.0
The correct answer is  1
Mean Square Error is  0.0
============== END OF TRAINING ==============
```

Training 1000 times:

```
**********************************************************
************ 2 Neurons - Sigmoid ******************
**********************************************************


============== START  TRAINING ==============
How many times do you want to test these two neurons ?
( press 0 to quit ) 1000

train_in_volume =
[[ 4.17022005e-02  7.20324493e-02  3.93546449e-01]
 [ 1.14374817e-05  3.02332573e-02 -1.24249783e-08]
 [ 1.46755891e-02  9.23385948e-03  6.24775829e-03]
 [ 1.86260211e-02  3.45560727e-02 -3.76629588e-02]
 [ 3.96767474e-02  5.38816734e-02  2.66479051e-01]
 [ 4.19194514e-02  6.85219500e-02 -3.78277725e-01]
 [ 2.04452250e-02  8.78117436e-02  1.15315122e-01]
 [ 2.73875932e-03  6.70467510e-02 -1.57992091e-03]
 [ 4.17304802e-02  5.58689828e-02  3.05652113e-01]
 [ 1.40386939e-02  1.98101489e-02 -1.22656628e-02]]
want_to_know =
[[  0.87638915]
 [  0.89460666]
 [392.6990817 ]]

intial weight 1 =
[[ 0.60148914]
 [ 0.93652315]
 [-0.37315164]]
intial weight 2 =
[[0.38464523]]

final weight 1 =
[[ 9.58442382]
 [ 7.04534772]
 [56.29240344]]
final weight 2 =
[[1.30929544]]
Mean Square Error of training data is  0.04862433134797691

The final prediction is  1.0
The correct answer is  1
Mean Square Error is  0.0
============== END OF TRAINING ==============
```

Training 10000 times:

```
**********************************************************
************ 2 Neurons - Sigmoid ******************
**********************************************************


============== START  TRAINING ==============
How many times do you want to test these two neurons ?
( press 0 to quit ) 10000

train_in_volume =
[[ 4.17022005e-02  7.20324493e-02  3.93546449e-01]
 [ 1.14374817e-05  3.02332573e-02 -1.24249783e-08]
 [ 1.46755891e-02  9.23385948e-03  6.24775829e-03]
 [ 1.86260211e-02  3.45560727e-02 -3.76629588e-02]
 [ 3.96767474e-02  5.38816734e-02  2.66479051e-01]
 [ 4.19194514e-02  6.85219500e-02 -3.78277725e-01]
 [ 2.04452250e-02  8.78117436e-02  1.15315122e-01]
 [ 2.73875932e-03  6.70467510e-02 -1.57992091e-03]
 [ 4.17304802e-02  5.58689828e-02  3.05652113e-01]
 [ 1.40386939e-02  1.98101489e-02 -1.22656628e-02]]
want_to_know =
[[  0.87638915]
 [  0.89460666]
 [392.6990817 ]]

intial weight 1 =
[[ 0.60148914]
 [ 0.93652315]
 [-0.37315164]]
intial weight 2 =
[[0.38464523]]

final weight 1 =
[[ 57.93161962]
 [  4.54826527]
 [393.88696883]]
final weight 2 =
[[1.25280727]]
Mean Square Error of training data is  0.024910710275101326

The final prediction is  1.0
The correct answer is  1
Mean Square Error is  0.0
============== END OF TRAINING ==============
```

6. 將上一項結論，使用不同的activation functions，寫出python程式來做上一項任務，同樣地做說明，並比較有沒有比較好的activation functions？

一樣使用 Tanh 作為 activation function.

Training 100 times:

```
**********************************************************
************  2 Neurons -  Tanh  ******************
**********************************************************


============== START  TRAINING ==============
How many times do you want to test these two neurons ?
( press 0 to quit ) 100

train_in_volume =
[[ 4.17022005e-01  7.20324493e-01  3.93546449e-01]
 [ 1.14374817e-04  3.02332573e-01 -1.24249783e-08]
 [ 1.46755891e-01  9.23385948e-02  6.24775829e-03]
 [ 1.86260211e-01  3.45560727e-01 -3.76629588e-02]
 [ 3.96767474e-01  5.38816734e-01  2.66479051e-01]
 [ 4.19194514e-01  6.85219500e-01 -3.78277725e-01]
 [ 2.04452250e-01  8.78117436e-01  1.15315122e-01]
 [ 2.73875932e-02  6.70467510e-01 -1.57992091e-03]
 [ 4.17304802e-01  5.58689828e-01  3.05652113e-01]
 [ 1.40386939e-01  1.98101489e-01 -1.22656628e-02]]
want_to_know =
[[0.87638915]
 [0.89460666]
 [0.39269908]]

intial weight 1 =
[[ 0.60148914]
 [ 0.93652315]
 [-0.37315164]]
intial weight 2 =
[[0.38464523]]

final weight 1 =
[[16.92771155]
 [ 3.22852042]
 [23.7769866 ]]
final weight 2 =
[[1.17473652]]
Mean Square Error of training data is  0.04203932583270931

The final prediction is  0.9965397952091258
The correct answer is  1
Mean Square Error is  5.986508597394266e-06
============== END OF TRAINING ==============
```

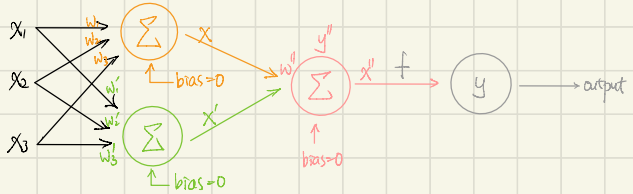
Training 1000 times:

```
**********************************************************
************  2 Neurons -  Tanh  ******************
**********************************************************


============== START  TRAINING ==============
How many times do you want to test these two neurons ?
( press 0 to quit ) 1000

train_in_volume =
[[ 4.17022005e-01  7.20324493e-01  3.93546449e-01]
 [ 1.14374817e-04  3.02332573e-01 -1.24249783e-08]
 [ 1.46755891e-01  9.23385948e-02  6.24775829e-03]
 [ 1.86260211e-01  3.45560727e-01 -3.76629588e-02]
 [ 3.96767474e-01  5.38816734e-01  2.66479051e-01]
 [ 4.19194514e-01  6.85219500e-01 -3.78277725e-01]
 [ 2.04452250e-01  8.78117436e-01  1.15315122e-01]
 [ 2.73875932e-02  6.70467510e-01 -1.57992091e-03]
 [ 4.17304802e-01  5.58689828e-01  3.05652113e-01]
 [ 1.40386939e-01  1.98101489e-01 -1.22656628e-02]]
want_to_know =
[[0.87638915]
 [0.89460666]
 [0.39269908]]

intial weight 1 =
[[ 0.60148914]
 [ 0.93652315]
 [-0.37315164]]
intial weight 2 =
[[0.38464523]]

final weight 1 =
[[34.59924226]
 [-1.33115658]
 [38.05599192]]
final weight 2 =
[[1.00346742]]
Mean Square Error of training data is  0.03337352783789961

The final prediction is  0.9997120665692212
The correct answer is  1
Mean Square Error is  4.145283028003062e-08
============== END OF TRAINING ==============
```

⇒ 從輸入的測資所輸出的 Mean Square Error 可以看到與 Q5 有相同的結論，即它通近的速率更為快速。

7. 請設法用三個神經元(如講義上的，有bias，有activation function)，來建構類神經網路。請推導backprogation公式出來。

這裡我所建構的方式為：



backprogration：

$$\frac{dE}{dw''} = \frac{dE}{dy} \times \frac{dy}{d\begin{bmatrix} x \\ x' \end{bmatrix}} \times \frac{d\begin{bmatrix} x \\ x' \end{bmatrix}}{dw''} = -(t-y) \times y(1-y) \times \begin{bmatrix} x_i \\ x_i' \end{bmatrix}$$

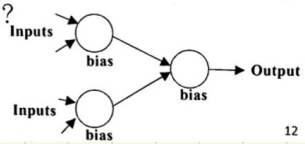
$$\frac{dE}{dw} = \frac{dE}{dy''} \times \frac{dy''}{dx} \times \frac{dx}{dw} = -(t-y'') \times y''(1-y'') \times x_i$$

$$\frac{dE}{dw'} = \frac{dE}{dy''} \times \frac{dy''}{dx} \times \frac{dx}{dw'} = -(t-y'') \times y''(1-y'') \times x_i'$$

若以Code表示則為：

```
# refine nn_weight
nn_weight_3 += np.dot(volumn_hstack.T, (train_sol_volume - train_out_volume) * train_out_volume*(1-train_out_volume))
train_out_previous = np.dot( train_out_volume, np.linalg.pinv(nn_weight_3))
train_out_previous_hsplit1, train_out_previous_hsplit2 = np.hsplit(train_out_previous, [1])
nn_weight_1 += np.dot(train_in_volume.T, (train_sol_volume - train_out_previous_hsplit1) * dydx(train_out_previous_hsplit1) )
nn_weight_2 += np.dot(train_in_volume.T, (train_sol_volume - train_out_previous_hsplit2) * dydx(train_out_previous_hsplit2) )
```

8. 將上一項結論，寫出python程式，來訓練你這個資料集，看花了不同的時間：訓練資料的mean square error結果為何？測試資料的mean square error結果為何？



Traning 100 times：

```
============== START  TRAINING ==============
How many times do you want to test these three neurons ?
( press 0 to quit ) 100

train_in_volume =
[[ 4.17022005e-02  7.20324493e-02  3.93546449e+02]
 [ 1.14374817e-05  3.02332573e-02 -1.24249783e-05]
 [ 1.46755891e-02  9.23385948e-03  6.24775829e+00]
 [ 1.86260211e-02  3.45560727e-02 -3.76629588e+01]
 [ 3.96767474e-02  5.38816734e-02  2.66479051e+02]
 [ 4.19194514e-02  6.85219500e-02 -3.78277725e+02]
 [ 2.04452250e-02  8.78117436e-02  1.15315122e+02]
 [ 2.73875932e-03  6.70467510e-02 -1.57992091e+00]
 [ 4.17304802e-02  5.58689828e-02  3.05652113e+02]
 [ 1.40386939e-02  1.98101489e-02 -1.22656628e+01]]
want_to_know =
[[1.69830420e-01 8.78142503e-01 3.92699082e+05]]

intial weight 1 =
[[ 0.60148914]
 [ 0.93652315]
 [-0.37315164]]
intial weight 2 =
[[0.38464523]
 [0.7527783 ]
 [0.78921333]]
intial weight 3 =
[[-0.82991158]
 [-0.92189043]]

final weight 1 =
[[ 3.46598013e+00]
 [ 8.88645693e+00]
 [-2.80426302e+04]]
final weight 2 =
[[-4.73576347e+01]
 [-5.65931971e+01]
 [-7.59545624e+05]]
final weight 3 =
[[-0.0430398 ]
 [-0.73422778]]
Mean Square Error of training data is  5.900509336206512e-07

The final prediction is  1.0
The correct answer is   1
Mean Square Error is  0.0
============== END OF TRAINING ==============
```

Training 1000 times：

```
============== START  TRAINING ==============
How many times do you want to test these three neurons ?
( press 0 to quit ) 1000

train_in_volume =
[[ 4.17022005e-02  7.20324493e-02  3.93546449e+02]
 [ 1.14374817e-05  3.02332573e-02 -1.24249783e-05]
 [ 1.46755891e-02  9.23385948e-03  6.24775829e+00]
 [ 1.86260211e-02  3.45560727e-02 -3.76629588e+01]
 [ 3.96767474e-02  5.38816734e-02  2.66479051e+02]
 [ 4.19194514e-02  6.85219500e-02 -3.78277725e+02]
 [ 2.04452250e-02  8.78117436e-02  1.15315122e+02]
 [ 2.73875932e-03  6.70467510e-02 -1.57992091e+00]
 [ 4.17304802e-02  5.58689828e-02  3.05652113e+02]
 [ 1.40386939e-02  1.98101489e-02 -1.22656628e+01]]
want_to_know =
[[1.69830420e-01 8.78142503e-01 3.92699082e+05]]

intial weight 1 =
[[ 0.60148914]
 [ 0.93652315]
 [-0.37315164]]
intial weight 2 =
[[0.38464523]
 [0.7527783 ]
 [0.78921333]]
intial weight 3 =
[[-0.82991158]
 [-0.92189043]]

final weight 1 =
[[-9.71453105e+00]
 [-1.43398547e+01]
 [-1.18069296e+05]]
final weight 2 =
[[-1.11626819e+03]
 [-1.94019616e+03]
 [-8.10430367e+06]]
final weight 3 =
[[-0.04316564]
 [-0.73591515]]
Mean Square Error of training data is  6.636071113347954e-29

The final prediction is  1.0
The correct answer is   1
Mean Square Error is  0.0
============== END OF TRAINING ==============
```

Training 10000 times：

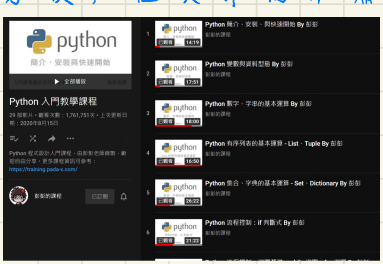
```
============== START  TRAINING ==============
How many times do you want to test these three neurons ?
( press 0 to quit ) 10000

train_in_volume =
[[ 4.17022005e-02  7.20324493e-02  3.93546449e+02]
 [ 1.14374817e-05  3.02332573e-02 -1.24249783e-05]
 [ 1.46755891e-02  9.23385948e-03  6.24775829e+00]
 [ 1.86260211e-02  3.45560727e-02 -3.76629588e+01]
 [ 3.96767474e-02  5.38816734e-02  2.66479051e+02]
 [ 4.19194514e-02  6.85219500e-02 -3.78277725e+02]
 [ 2.04452250e-02  8.78117436e-02  1.15315122e+02]
 [ 2.73875932e-03  6.70467510e-02 -1.57992091e+00]
 [ 4.17304802e-02  5.58689828e-02  3.05652113e+02]
 [ 1.40386939e-02  1.98101489e-02 -1.22656628e+01]]
want_to_know =
[[1.69830420e-01 8.78142503e-01 3.92699082e+05]]

intial weight 1 =
[[ 0.60148914]
 [ 0.93652315]
 [-0.37315164]]
intial weight 2 =
[[0.38464523]
 [0.7527783 ]
 [0.78921333]]
intial weight 3 =
[[-0.82991158]
 [-0.92189043]]

final weight 1 =
[[-1.41513806e+02]
 [-2.46592685e+02]
 [-1.02423585e+06]]
final weight 2 =
[[-1.18042779e+04]
 [-2.07742945e+04]
 [-8.15443539e+07]]
final weight 3 =
[[-0.04316564]
 [-0.73591515]]
Mean Square Error of training data is  2.7883520433767975e-248

The final prediction is  1.0
The correct answer is   1
Mean Square Error is  0.0
============== END OF TRAINING ==============
```

9. 自由申論及發揮：AlphaGo這個人機大賽在推動AI的發展方面，起了甚麼樣的作用？人類被人工智慧打敗了，該怎麼自處？被機器人取代？或如何與人工智慧結合？你覺得你有能力在未來參加嗎？需要甚麼樣的技術或準備？其它相關延伸之研發課題或成果(請自由發揮)

1. 表示 Deeplearning、Artificial Intelligence、Machine Learning... 種種有關和器智慧化的研究已大幅成長且有了實質的成果。
2. 實際上人類並非被人工智慧打敗了，因為它只是建立在快速的計算下能下出獲勝率更大的下棋方式。人類僅在利用和器快速的計算能力，因此人類所能做的就是把制好這個速度，並且讓其有隨時可被遠端終止的手段。
3. 取代的部分肯定极以萬計，例如棋手就算是被取代了，現在的棋士僅是人類在自娛；隨著有駕車的發展，司机也遲早会被取代。
4. 人類現代的生活將和AI密不可分，包含智能家電等等。
5. 以目前來說尚無法勝任設計、研發AI這項任務，但未來若有机会也想參与其中。其實我堂哥正在美國Amazon，研發其重要的AI：Alexa，這令我十分憧憬這個職業。
6. 程設、數學、機器學習、演算法等等。

10. 請說明你做此作業所蹤到的一些狀況及困難。

1. 一開始思考作業要就遇到了最重要的門檻：Python語言的撰寫能力。因此我大概花了四天在網路上學習Python，最一開始是先閱讀一些介紹Python的文章，大致瞭解其撰寫方便，但实作仍不熟悉，因此後來就去觀看Youtube的影片才真正邁入Python新手村。



將影片全部看完後，總算回到此份作業中。

2. 撰寫開頭連要如何取測資都不知道，因為一開始想說可以叫其幫我預測圓週率為何，但沒有考慮到Activation Function：Sigmoid它本身的限制，輸出僅界於0～1，所以不好設計訓練方式。後來思考到0和1正為True or False，因此將訓練方式改為是非問題，經寄信尋問TA後確实可行，

到此大概花我一個禮拜了。

3. 在 One Neuron 的部分幾本上照著老師講義上的思路就可以了 Two Neurons 時才是需要思考的部分。
一開始設計 的方式作訓練，但發現若將兩資料直接取 Sigmoid，似乎像是個別訓練兩個 Neuron 罷了，並沒有 1+1>2 的效果。而另一方面，若將兩結果取優值合併再取 Sigmoid，如此一來看似可以成功訓練 Neuron（畢竟有擇優計算），但實際上 backprogration 的部分並不好計算，主要是因擇優後的數據並不好直接代入當的計算中，因此最後才變成 →○→○→ 的串接式訓練方式。

4. 另外，還有一點是非本科的難題，即自己電腦程設環境架設的問題。因為要寫 Python 但又想要同時可以撰寫 C，因此需要一個強大且方便的程式編譯環境，後來選擇了很大眾的 Visual Studio Code 來進行撰寫。但架設途中也是歷經波折，因為網路上每個人的方式都不一樣，有些在自己的電腦上執行就是会出包，最後也大概花了兩三天才搞定。

11. 請列出你的參考文獻(含網站)來源，並請說明參考了那些部份用於作業中。

1. 我幾本上是照著老師上課的講義構思。
2. Wikipedia 尋找 Activation Function：https://en.m.wikipedia.org/wiki/Activation_function
3. 思考 Multiple Neurons 的部分：
https://towardsdatascience.com/multi-layer-neural-networks-with-sigmoid-function-deep-learning-for-rookies-2-bf464f09eb7f
4. Python Language Learning：https://youtube.com/playlist?list=PL-g0fdC5RMboYEyt6QS2iLb_1m7QcgfHk