C1 :用 Keras 開啟深度學習的 Hello World

主講人 : 陳俊豪、謝長潤

時間 : 2016 – 11 – 24

Outline

- 深度學習概述
  - 深度學習基本介紹
  - 梯度下降與反向傳播
- Keras 基礎教學
  - 30 秒內開啟 Hello World
  - 圖像識別模型
    - mnist 手寫辨識資料集
    - 使用 CNN 完成手寫辨識模型

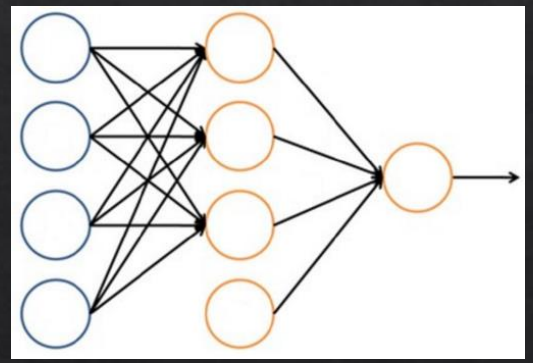
Outline

- 深度學習概述
  - 深度學習基本介紹
  - 梯度下降與反向傳播
- Keras 基礎教學
  - 30 秒內開啟 Hello World
  - 圖像識別模型
    - mnist 手寫辨識資料集
    - 使用 CNN 完成手寫辨識模型

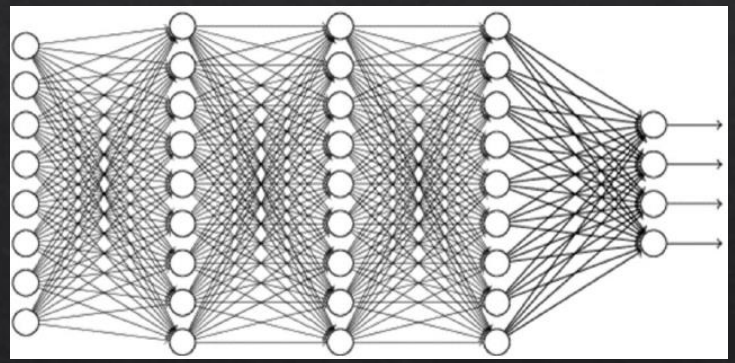
深度學習概述：深度學習基本介紹

如果要只用一句話不是十分精確地說明什麼是深度學習，可以把深度學習形容成一種**「比較深」的類神經網路**，並搭配了各式各樣特別的類神經網路階層，如卷積神經網路、遞歸神經網路等等。所以一些深度學習架構也常被稱為深度神經網路(Deep neural network, DNN)。

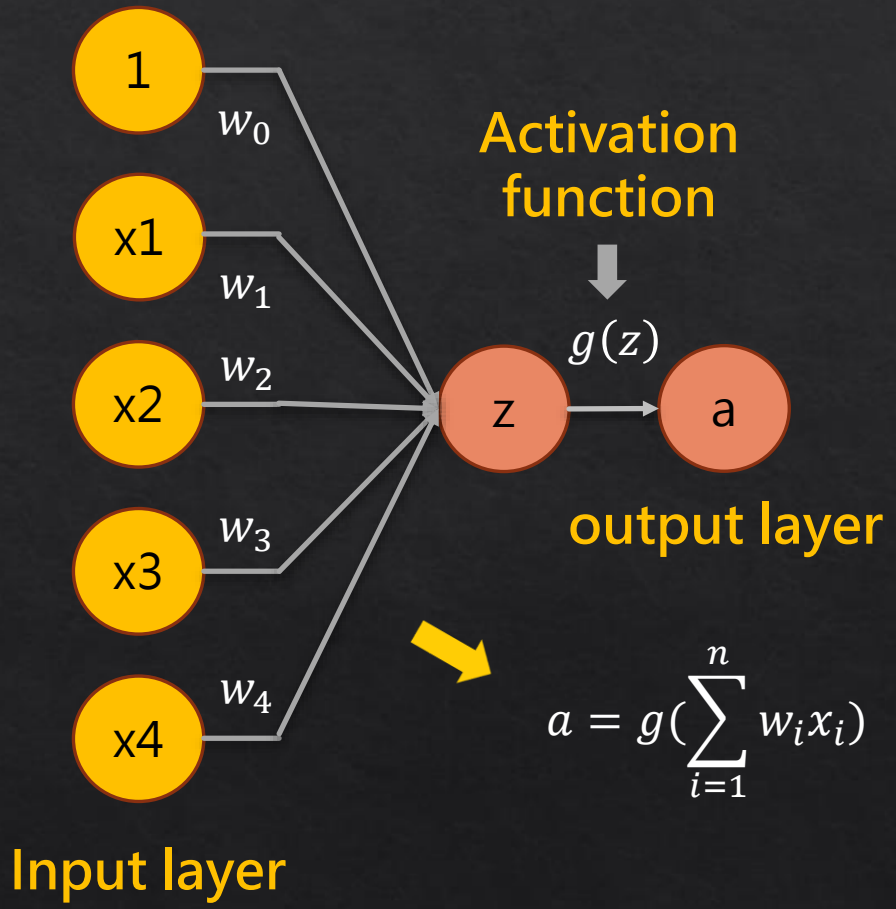
淺層神經網路



深層神經網路 (DNN)



深度學習概述：深度學習基本介紹

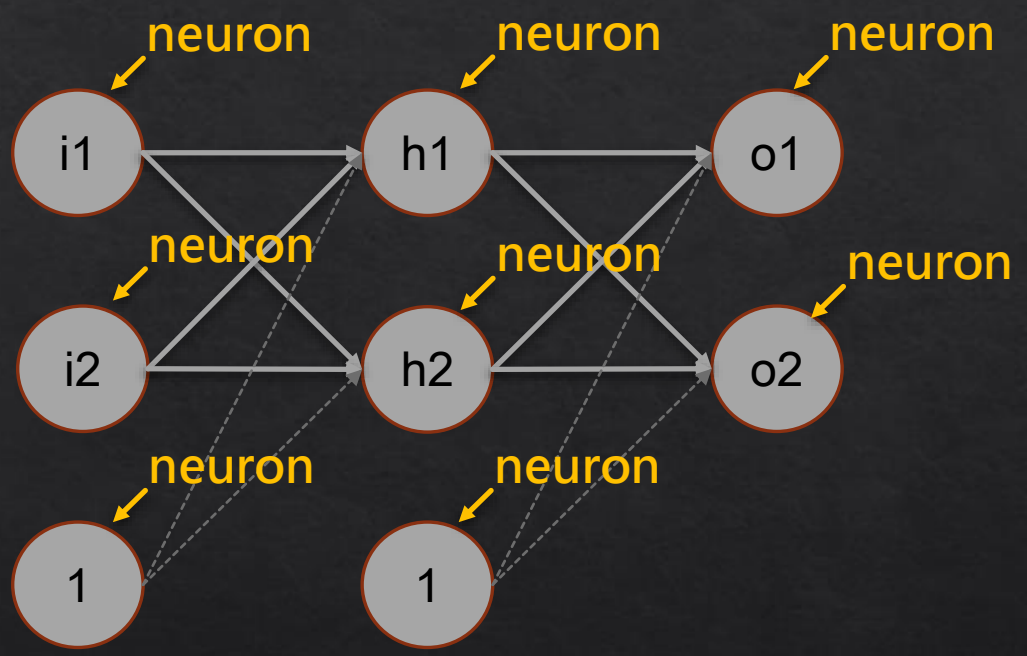


類神經網路是一種模仿生物神經系統的數學模型。在類神經網路中，通常會有數個階層，每個階層中會有數十到數百個神經元(neuron)，神經元會將上一層神經元的輸入加總後，進行活化函數(Activation function)的轉換，當成神經元的輸出。每個神經元會跟下一層的神經元有特殊的連接關係，使上一層神經元的輸出值經過權重計算(weight)後傳遞給下一層的神經元。

深度學習概述：梯度下降與反向傳播

- 神經網絡模型簡介：

神經元 (neural) 為神經網路的基本元素，神經網絡模型就是由大量的神經元相互聯結，並通過數學的學習方法（Learning Method）進行學習優化。



深度學習概述：梯度下降與反向傳播

- 基本神經網絡結構介紹：

  第一層 - input layer : i1、i2

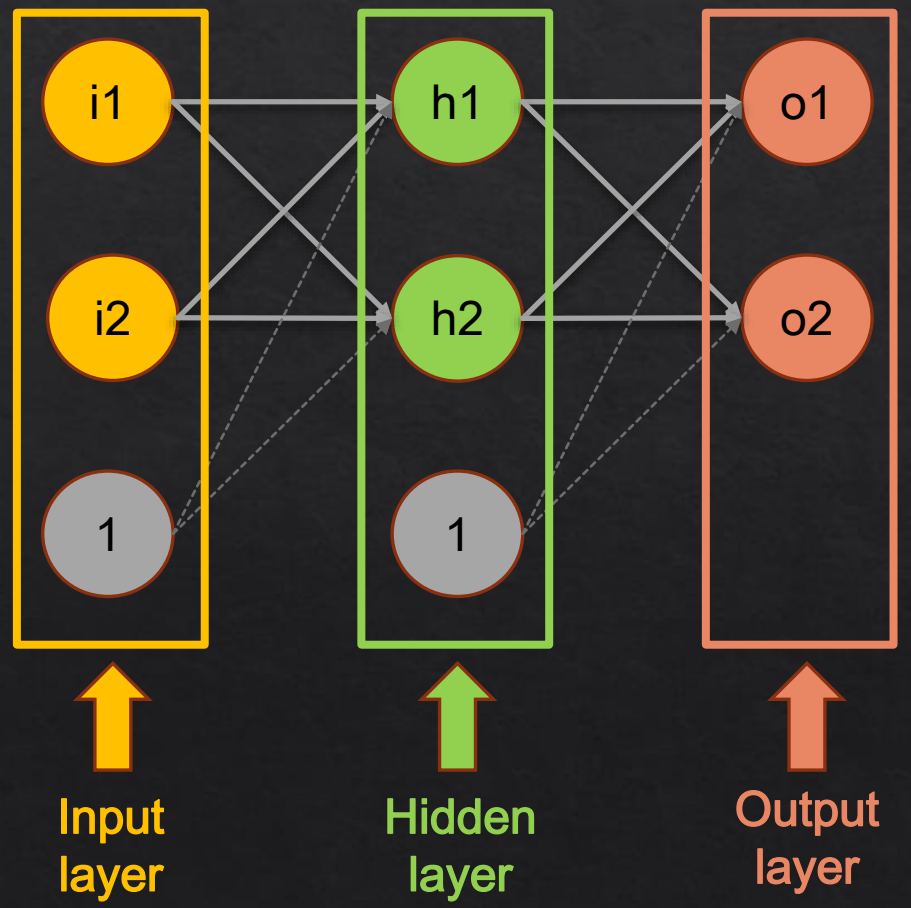
  第二層 - hidden layer : h1、h2

  第三層 - output layer : h1、h2

  w : 權重

  b : 截距項

  Activation function : 使用 sigmoid



深度學習概述：梯度下降與反向傳播

- 基本神經網絡結構介紹：

  第一層 - input layer : i1、i2

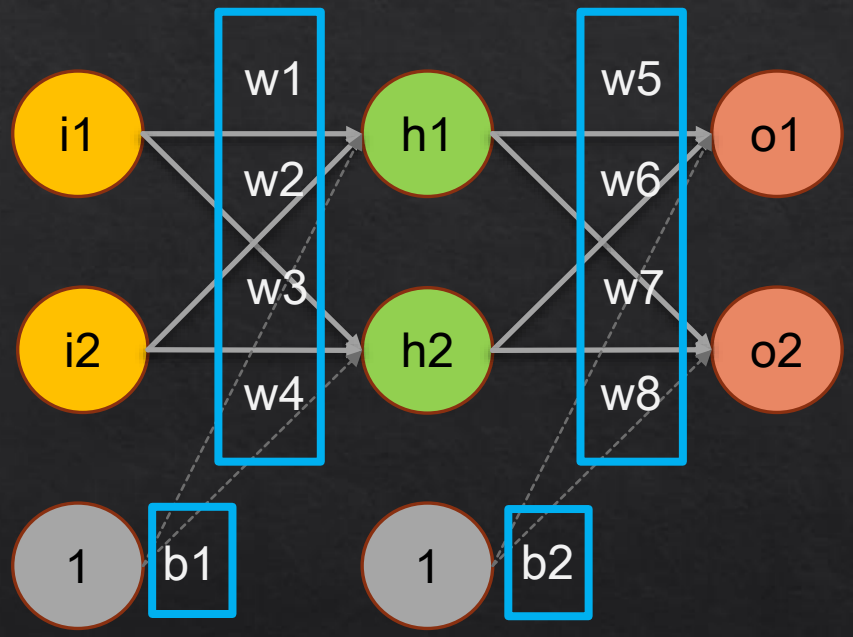
  第二層 - hidden layer : h1、h2

  第三層 - output layer : h1、h2

  w : 權重

  b : 截距項

  Activation function : 使用 sigmoid



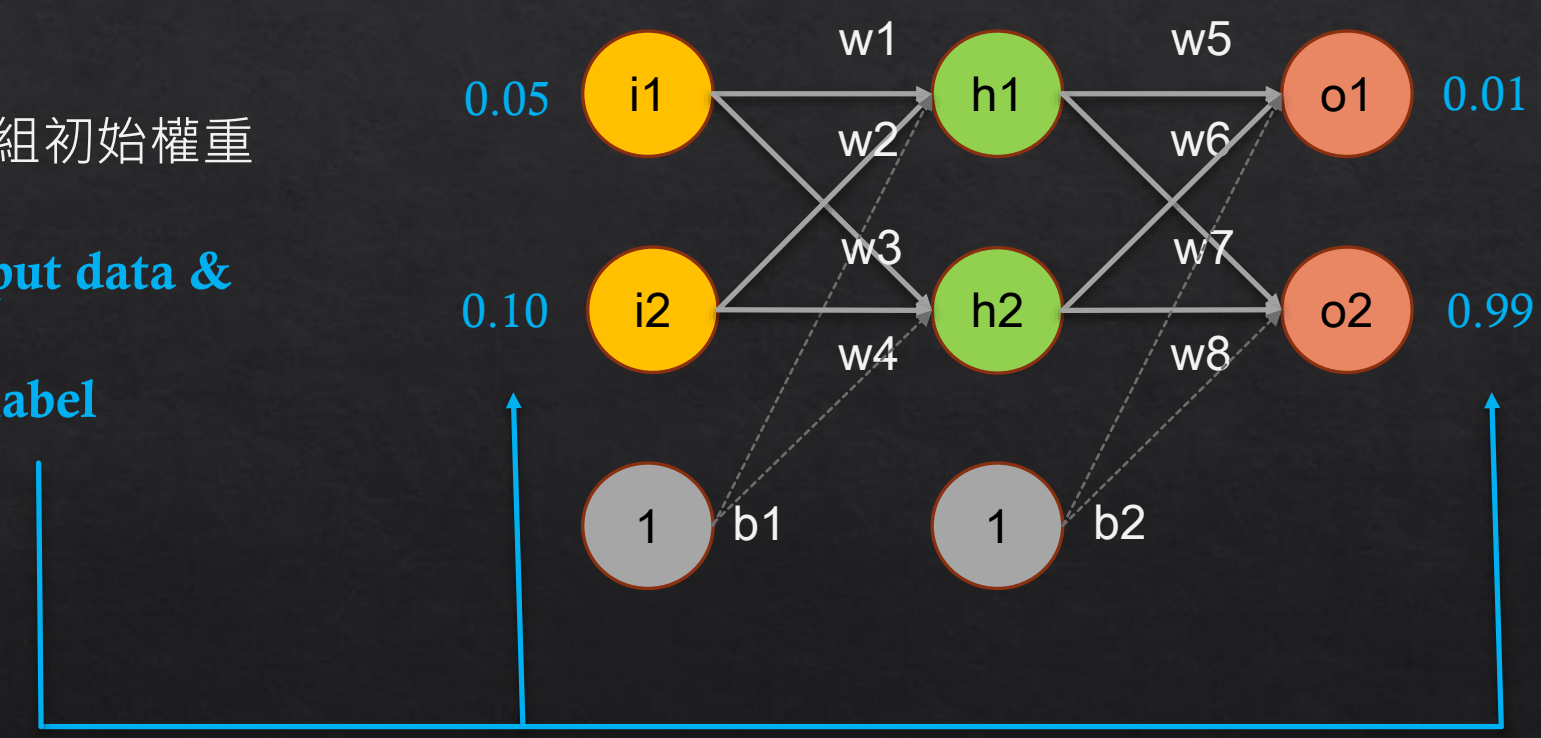
深度學習概述：梯度下降與反向傳播

- 運作流程：

Step 1. 給定一組初始權重

給定 input data &

output label



深度學習概述：梯度下降與反向傳播

- 運作流程：

Step 1. 給定一組初始權重

給定 input data &

output label

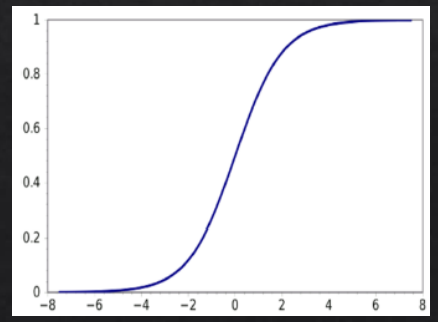
深度學習概述：梯度下降與反向傳播

- 運作流程：

Step 2. 開始進行正向傳播

(Activation 使用 sigmoid 函數)

$$a = g(z) = \frac{1}{1 + e^{-z}}$$



深度學習概述：梯度下降與反向傳播

- 運作流程：

Step 2-1. 開始進行正向傳播

(Input layer → hidden layer)

$$h1 = 0.05 * 0.15 + 0.10 * 0.20 + 1 * 0.35$$

$$= 0.3775$$

深度學習概述：梯度下降與反向傳播

- 運作流程：

Step 2-1. 開始進行正向傳播

(Input layer → hidden layer)

$$h1 = 0.05 * 0.15 + 0.10 * 0.20 + 1 * 0.35$$

$$= 0.3775$$

$$\boldsymbol{h1_a} = \frac{1}{1 + e^{-0.3775}} = \mathbf{0.593269992}$$

深度學習概述：梯度下降與反向傳播

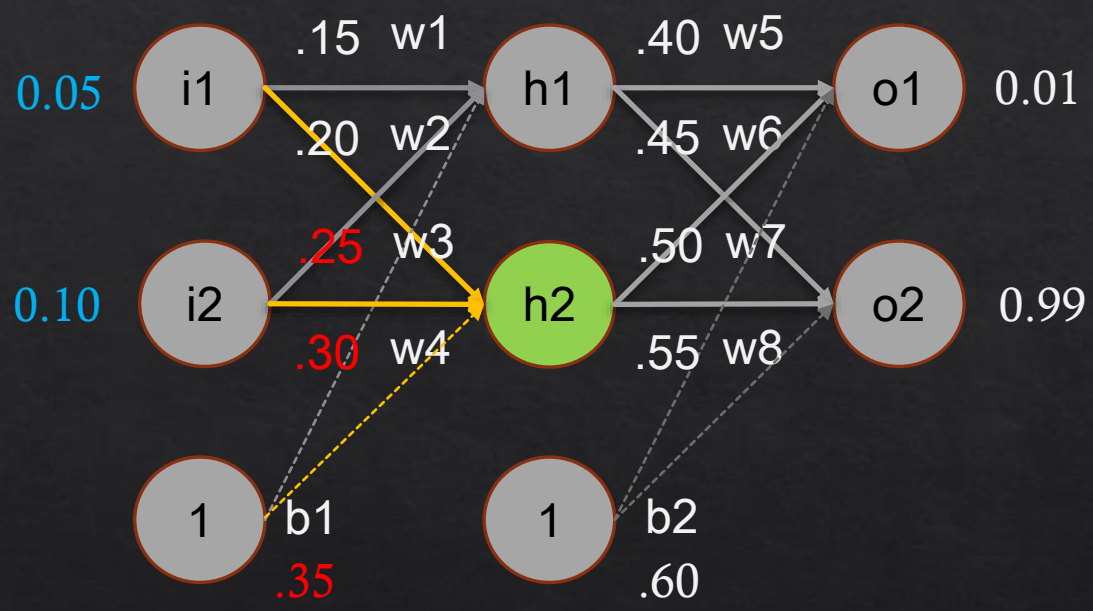
- 運作流程：

Step 2-1. 開始進行正向傳播

(Input layer → hidden layer)

$$h2 = 0.05 * 0.25 + 0.10 * 0.30 + 1 * 0.35$$

$$= 0.3925$$



深度學習概述：梯度下降與反向傳播

- 運作流程：

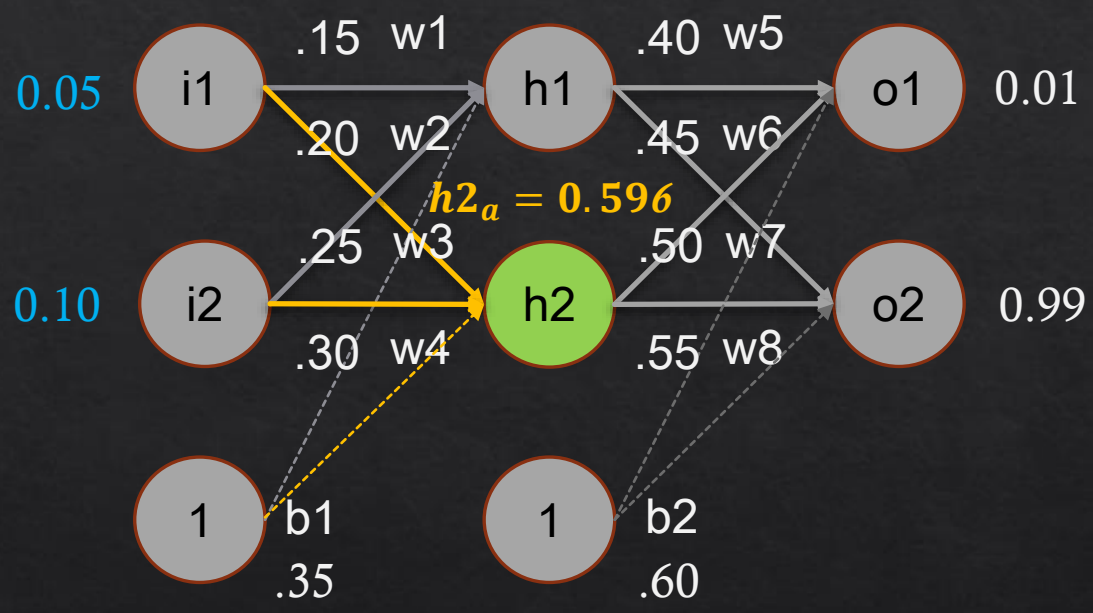
Step 2-1. 開始進行正向傳播

(Input layer → hidden layer)

$$h2 = 0.05 * 0.25 + 0.10 * 0.30 + 1 * 0.35$$

$$= 0.3925$$

$$\boldsymbol{h2_a} = \frac{1}{1 + e^{-0.3925}} = \mathbf{0.596884378}$$

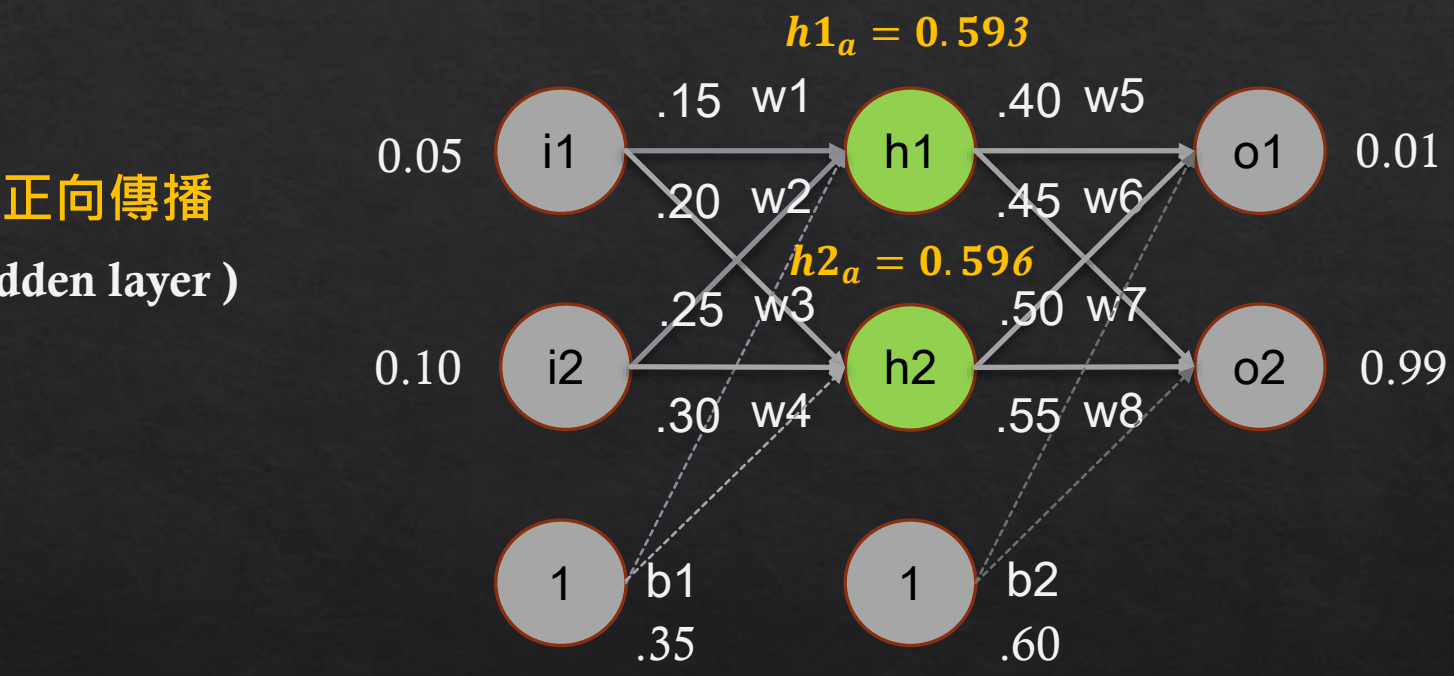


深度學習概述：梯度下降與反向傳播

- 運作流程：

Step 2-1. 開始進行正向傳播

(Input layer → hidden layer)

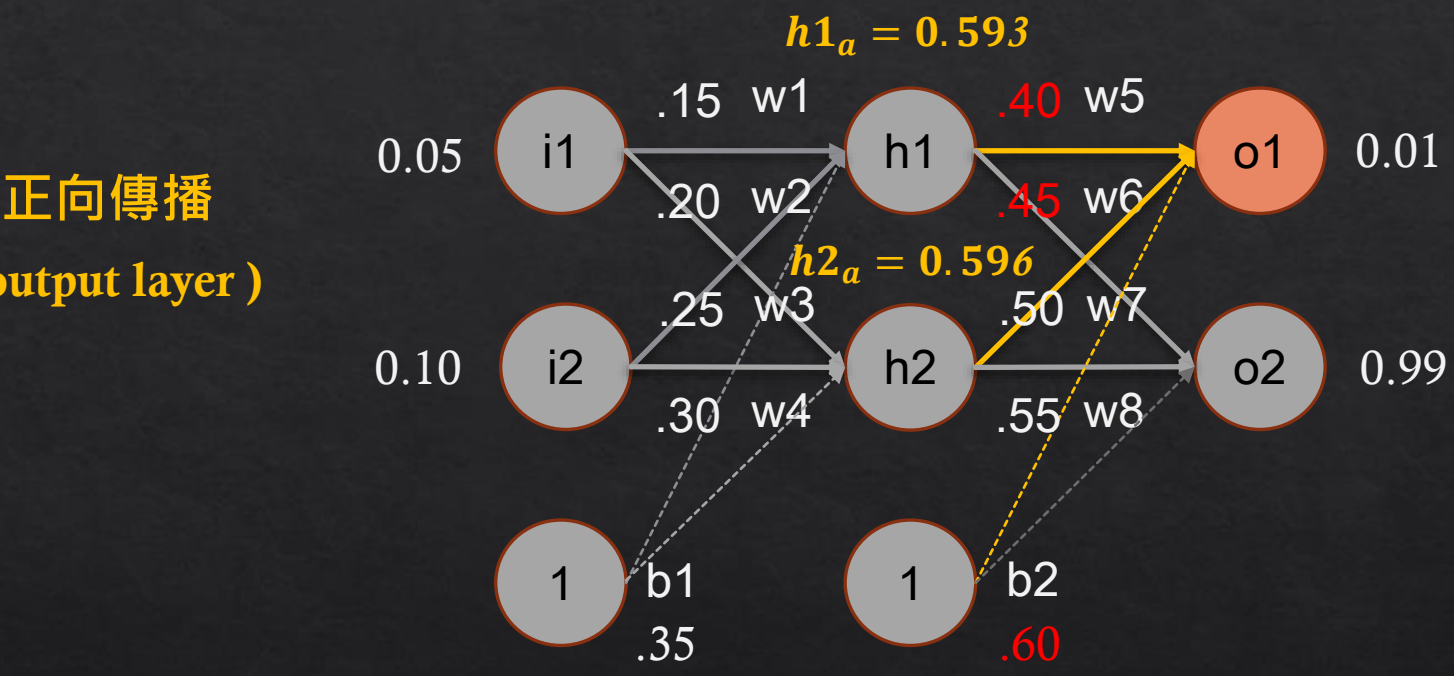


深度學習概述：梯度下降與反向傳播

- 運作流程：

Step 2-2. 開始進行正向傳播

(hidden layer → output layer)



深度學習概述：梯度下降與反向傳播

- 運作流程：

Step 2-2. 開始進行正向傳播

(hidden layer → output layer)

$o1 = 0.593 * 0.40 + 0.596 * 0.45 + 1 * 0.60$

$= 1.105905967$

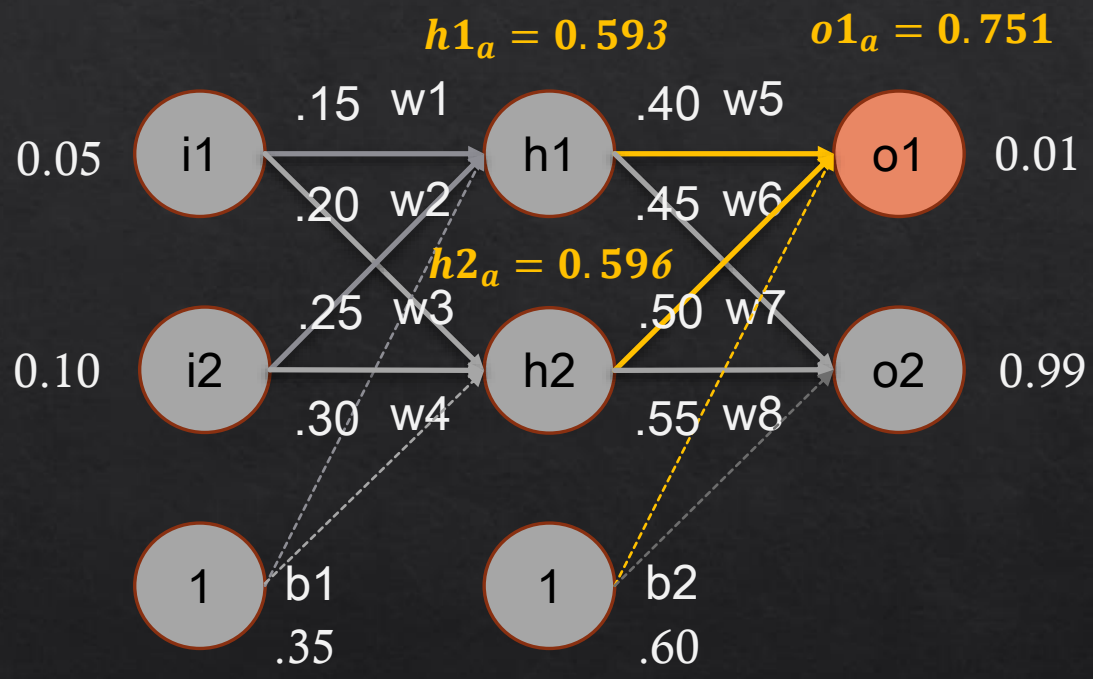
深度學習概述：梯度下降與反向傳播

- 運作流程：

Step 2-2. 開始進行正向傳播

(hidden layer → output layer)

$$o1 = 0.593 * 0.40 + 0.596 * 0.45 + 1 * 0.60$$
$$= 1.105905967$$
$$o1_a = \frac{1}{1 + e^{-1.105905967}} = 0.75136507$$



深度學習概述：梯度下降與反向傳播

- 運作流程：

Step 2-2. 開始進行正向傳播

(hidden layer → output layer)

$o2 = 0.593 * 0.50 + 0.596 * 0.55 + 1 * 0.60$

$= 1.224921$

深度學習概述：梯度下降與反向傳播

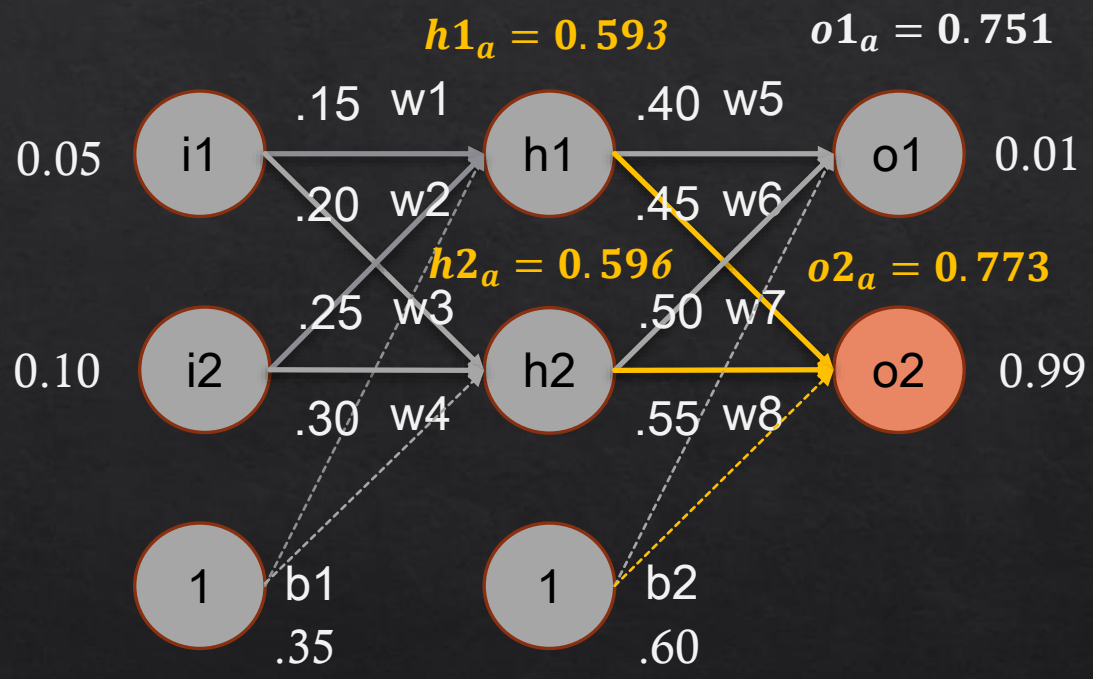
- 運作流程：

Step 2-2. 開始進行正向傳播

(hidden layer → output layer)

$$o2 = 0.593 * 0.50 + 0.596 * 0.55 + 1 * 0.60$$
$$= 1.224921$$

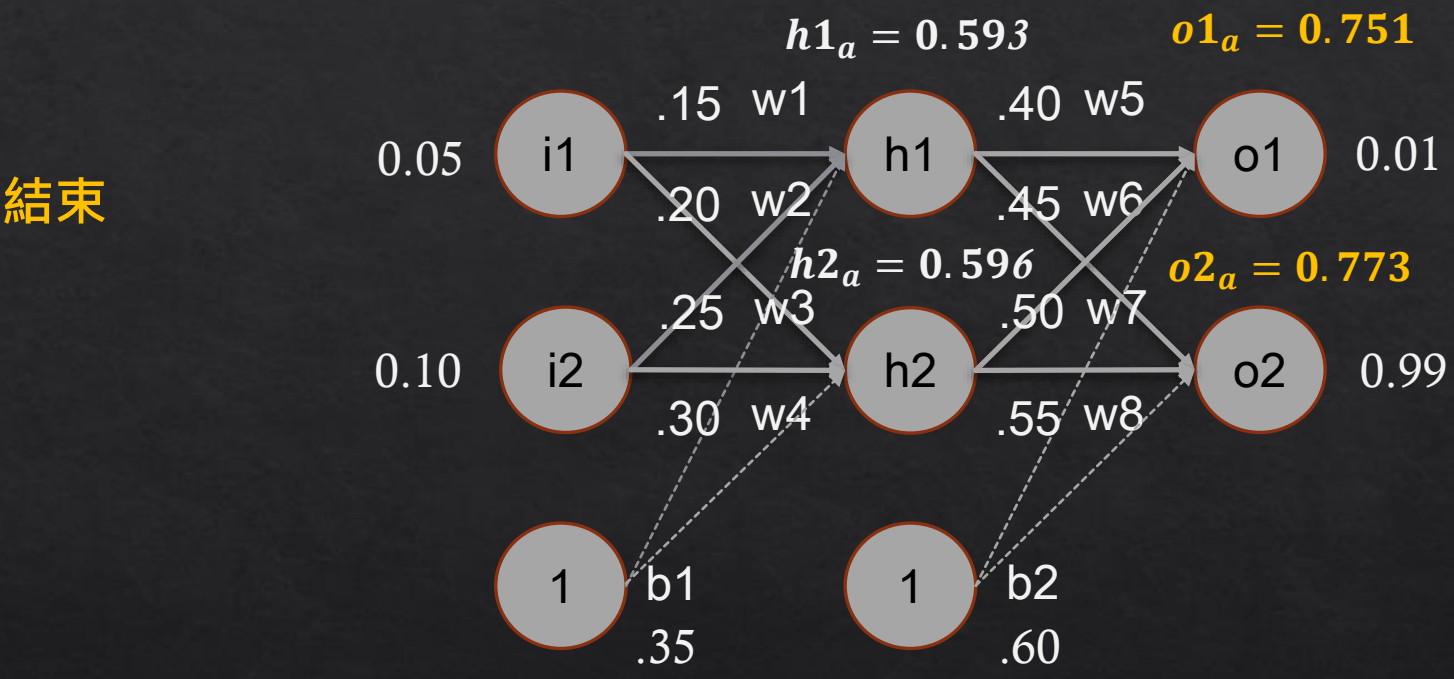
$$o2_a = \frac{1}{1 + e^{-1.224921}} = 0.772928465$$



深度學習概述：梯度下降與反向傳播

- 運作流程：

Step 2-2. 正向傳播結束

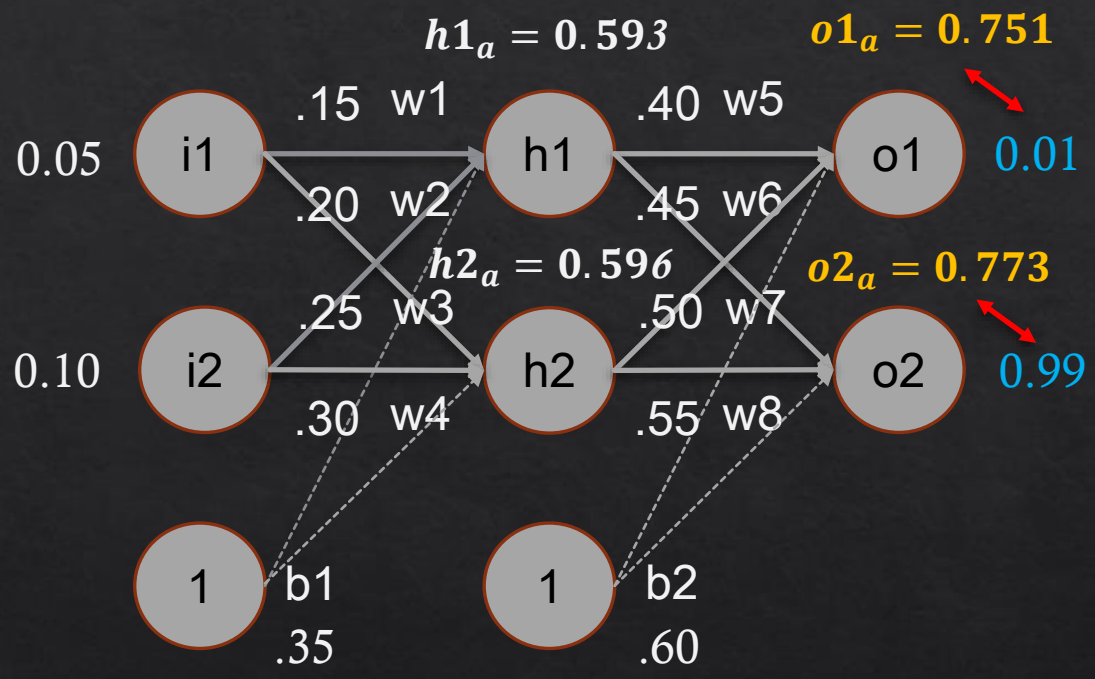


深度學習概述：梯度下降與反向傳播

- 運作流程：

Step 3. 開始計算誤差

很明顯可以看出我們使用初始 weight 計算出的最後 output 結果與實際的結果相去甚遠。



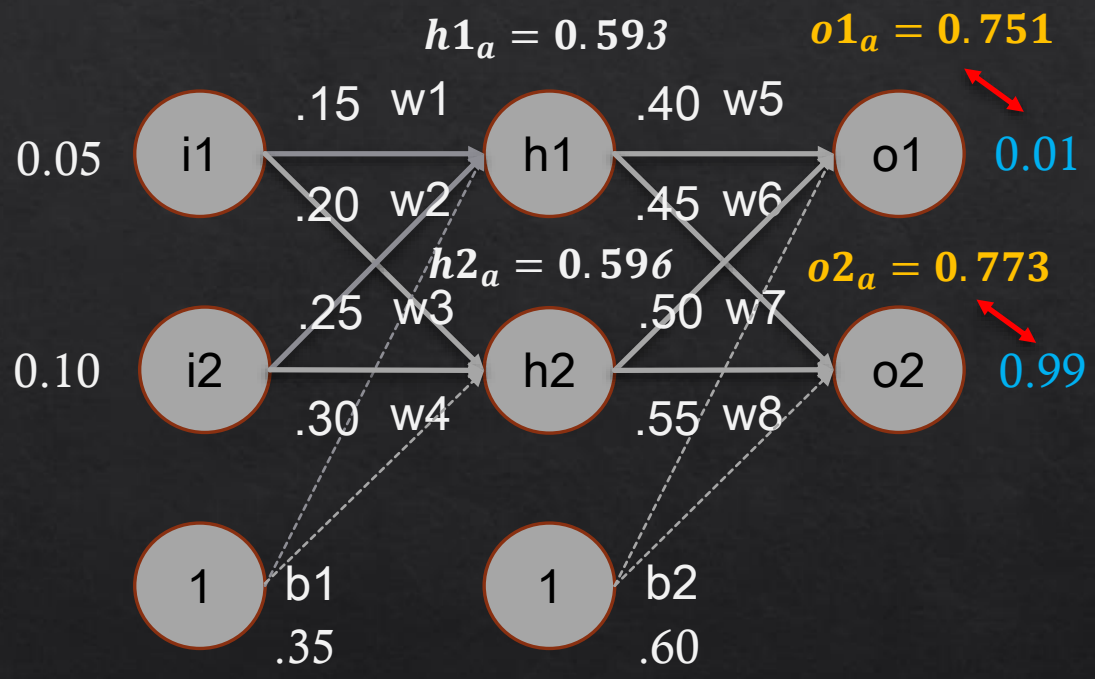
深度學習概述：梯度下降與反向傳播

- 運作流程：

Step 3. 開始計算誤差

(使用 square error 作為 cost function)

$$E_{total} = \sum \frac{1}{2}(\text{actual} - \text{output})^2$$



深度學習概述：梯度下降與反向傳播

- 運作流程：

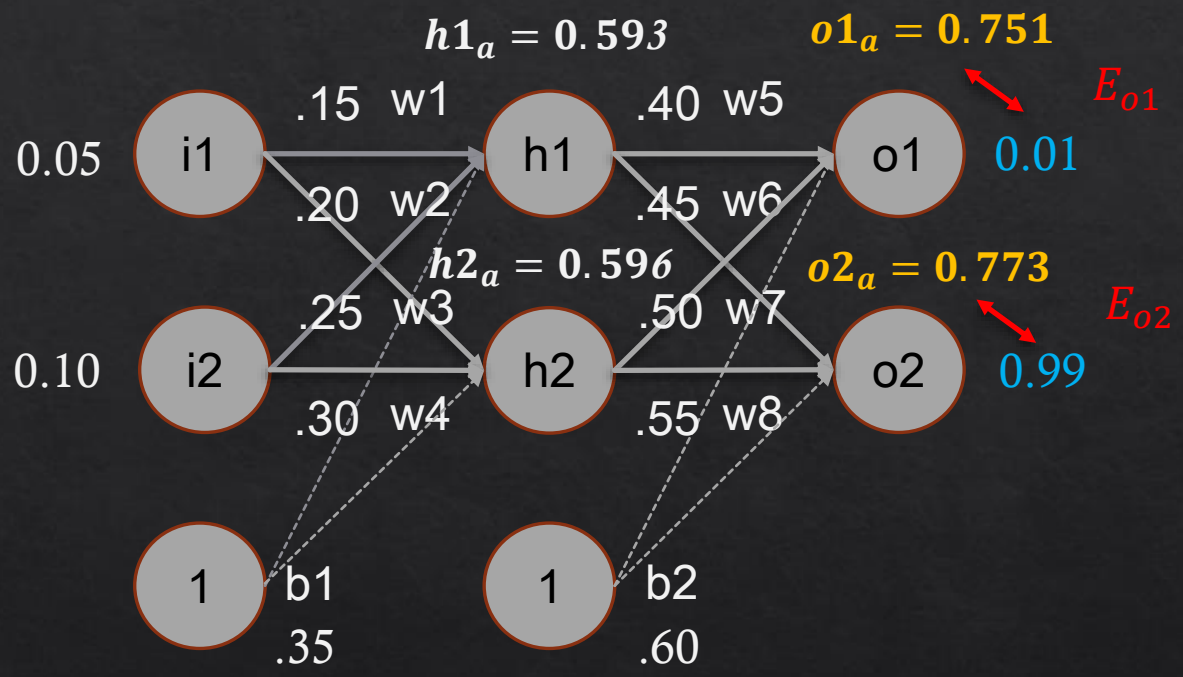
Step 3. 開始計算誤差

(使用 square error 作為 cost function)

$$E_{o1} = \sum \frac{1}{2}(0.01 - 0.751)^2 = 0.274811083$$

$$E_{o2} = \sum \frac{1}{2}(0.99 - 0.773)^2 = 0.023560026$$

$$E_{total} = E_{o1} + E_{o2} = 0.298371109$$



深度學習概述：梯度下降與反向傳播

- 運作流程：

Step 3. 開始計算誤差

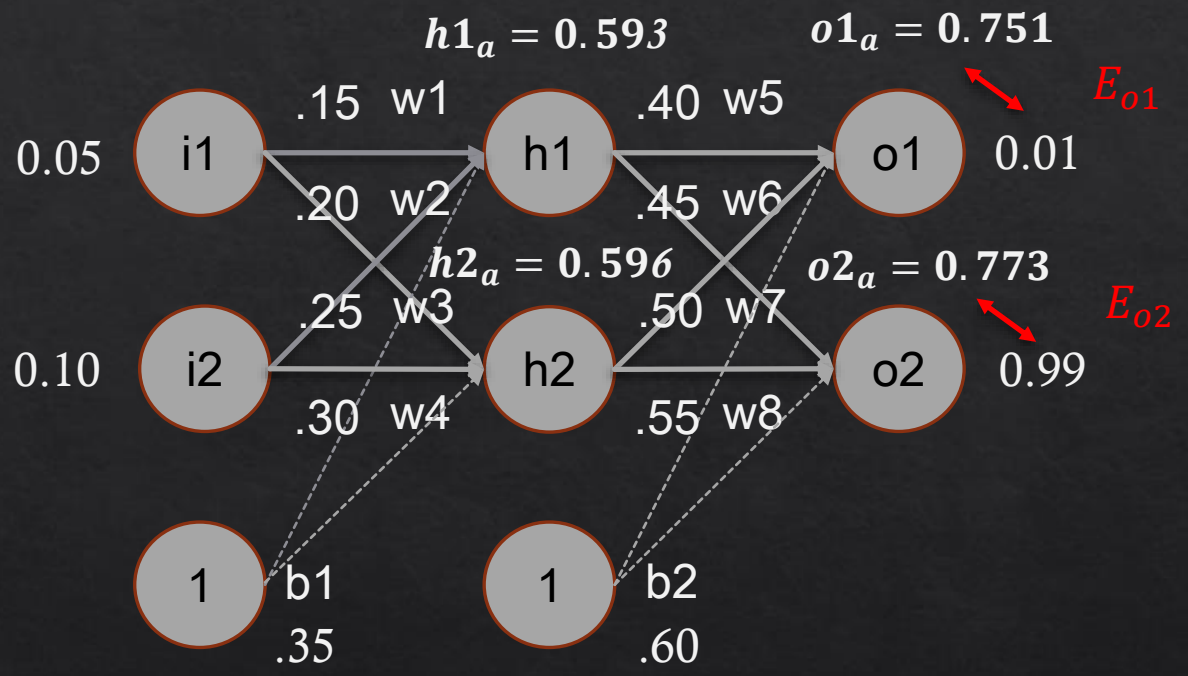
(使用 square error 作為 cost function)

$$E_{o1} = \sum \frac{1}{2}(0.01 - 0.751)^2 = 0.274811083$$

$$E_{o2} = \sum \frac{1}{2}(0.99 - 0.773)^2 = 0.023560026$$

$$E_{total} = E_{o1} + E_{o2} = 0.298371109$$

目標：找到能使整體誤差呈現最小的 weight



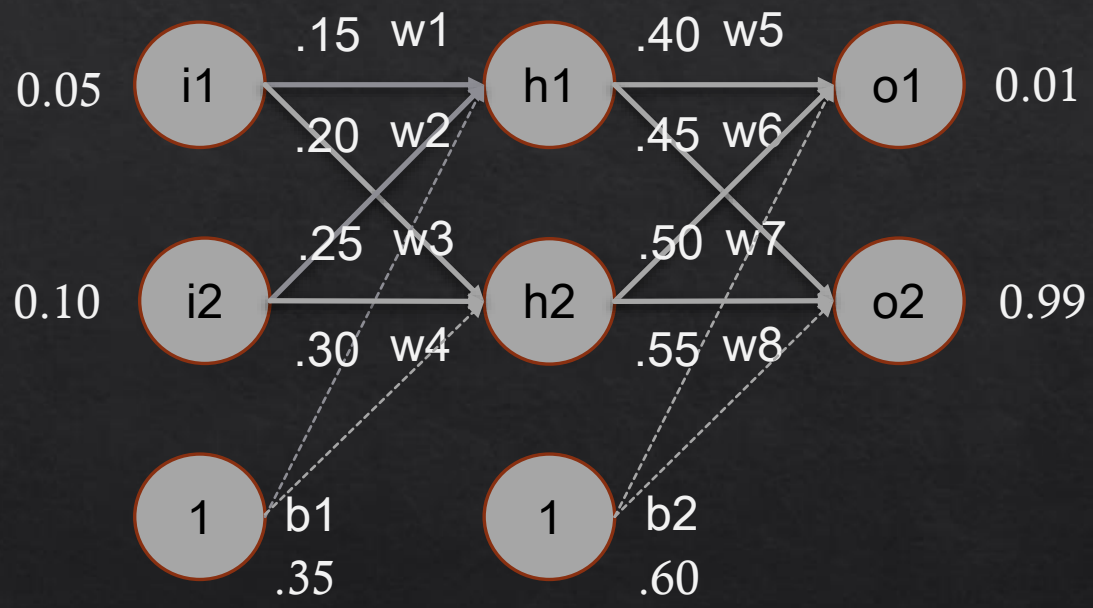
深度學習概述：梯度下降與反向傳播

- 運作流程：

Step 4. 透過反向傳播與梯度下降調整權重

- **梯度下降法調整觀念：**

  梯度方向為函數最快上升的方向，故朝著梯度的 **反方向 (負梯度方向)** 前進，能使成本函數 (cost function) 朝著下降的方向前進。



深度學習概述：梯度下降與反向傳播

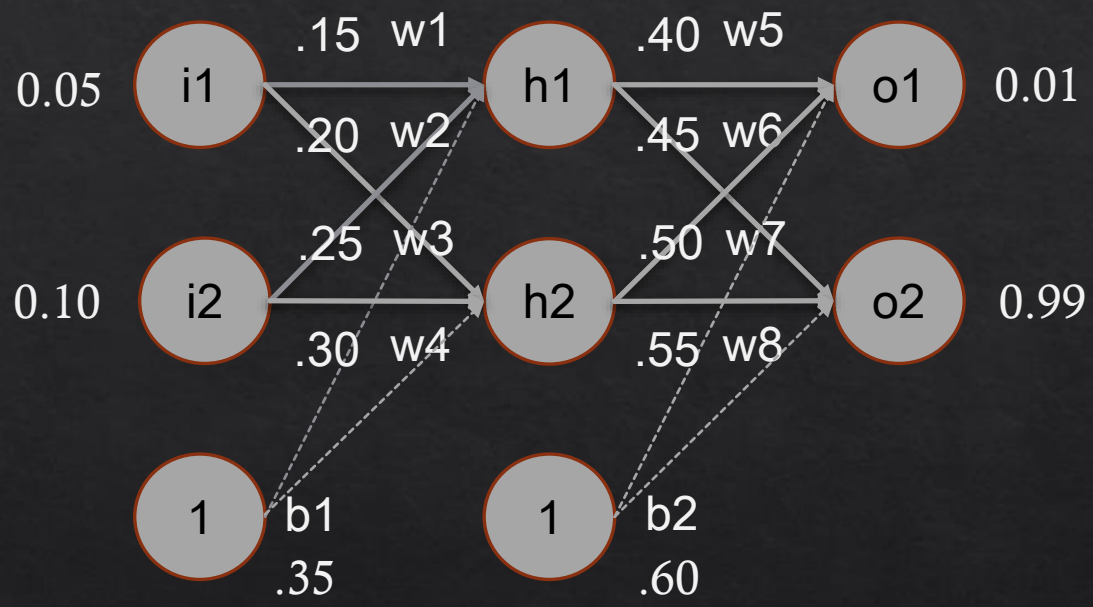
- 運作流程：

Step 4. 透過反向傳播與梯度下降調整權重

- **梯度下降調整過程為：**

$$w_i^{new} = w_i - \eta * \frac{\partial E_{total}}{\partial w_i}$$

朝著負梯度調整權重



深度學習概述：梯度下降與反向傳播

- 運作流程：

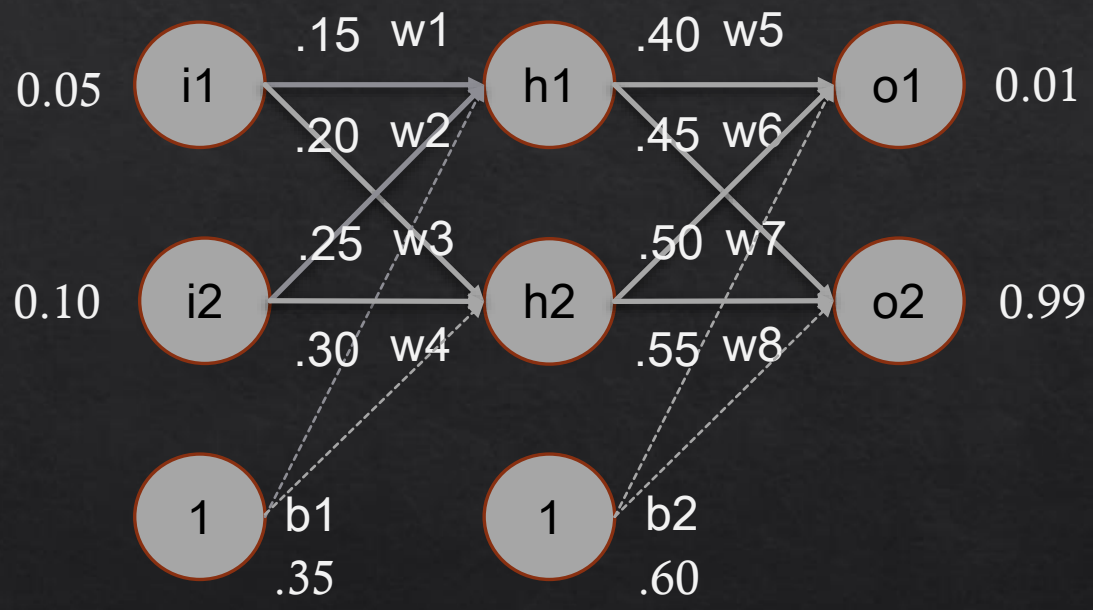
Step 4. 透過反向傳播與梯度下降調整權重

- **梯度下降調整過程為：**

$$w_i^{new} = w_i - \eta * \frac{\partial E_{total}}{\partial w_i}$$

步長，又稱學習速度

- 步長小 → 調整過程慢、易陷入局部最適。
- 步長大 → 不容易收斂。

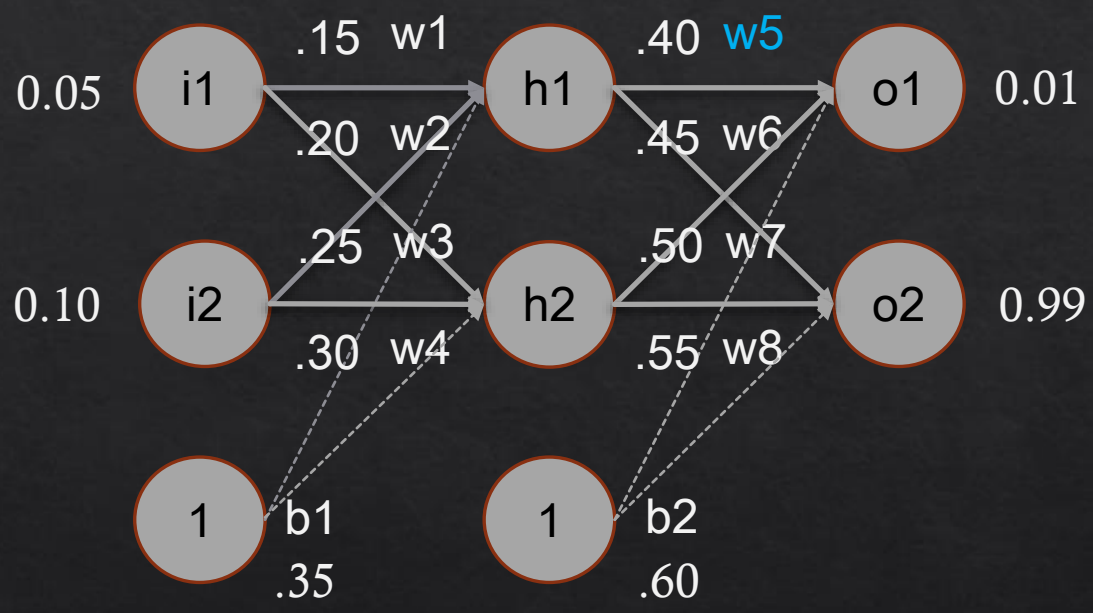


深度學習概述：梯度下降與反向傳播

- 運作流程：

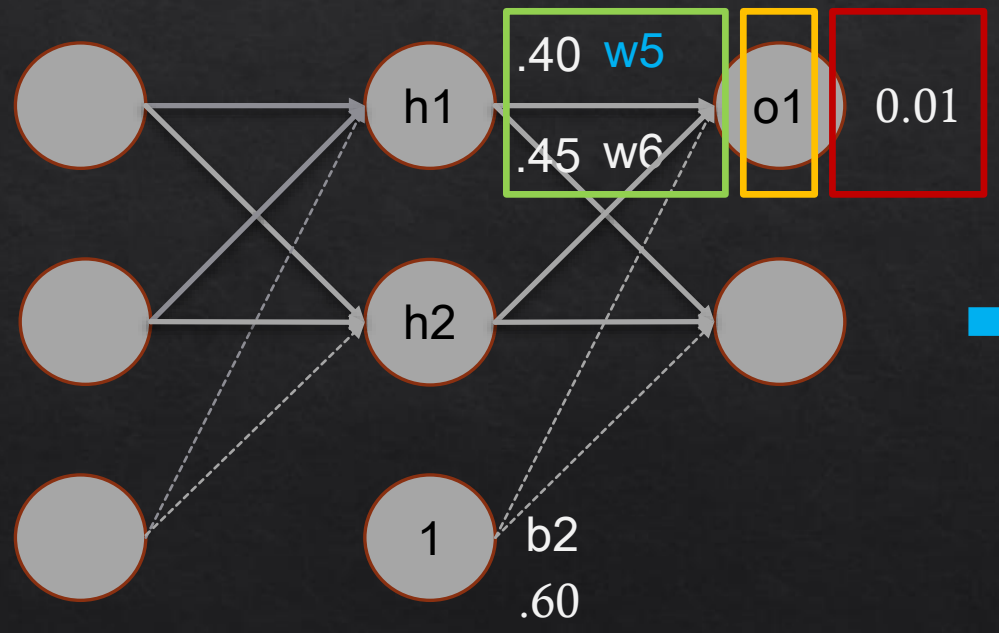
Step 4-1. 透過反向傳播求出偏導數

（僅以 w5 為例）



深度學習概述：梯度下降與反向傳播

- 反求 w5 過程可以拆解成：(Chain rule)



$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial o1} * \frac{\partial o1}{\partial o1_a} * \frac{\partial o1_a}{\partial w_5}$$

整體成本函數為各權重的函數

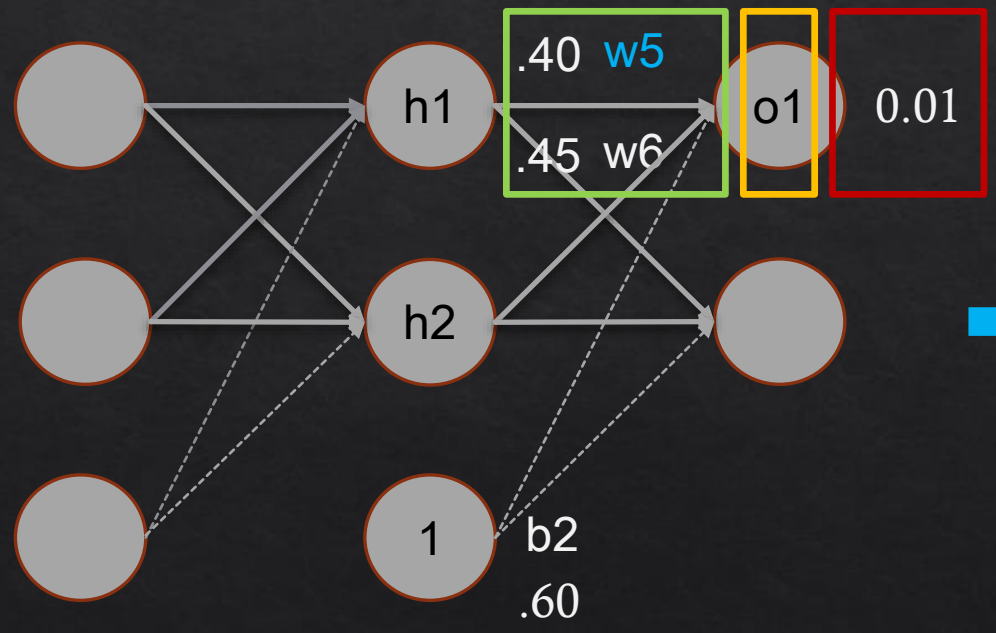
$$E_{total} = E_{o1} + E_{o2} = \frac{1}{2}(actual_{o1} - output_{o1})^2 + \frac{1}{2}(actual_{o2} - output_{o2})^2$$

$$o1_a = \frac{1}{1 + e^{-o1}}$$

$$o1 = w_5 * h1_a + w_6 * h2_a + b2 * 1$$

深度學習概述：梯度下降與反向傳播

- 反求 w5 過程可以拆解成：



1. 處理成本函數：

$$E_{total} = E_{o1} + E_{o2}$$

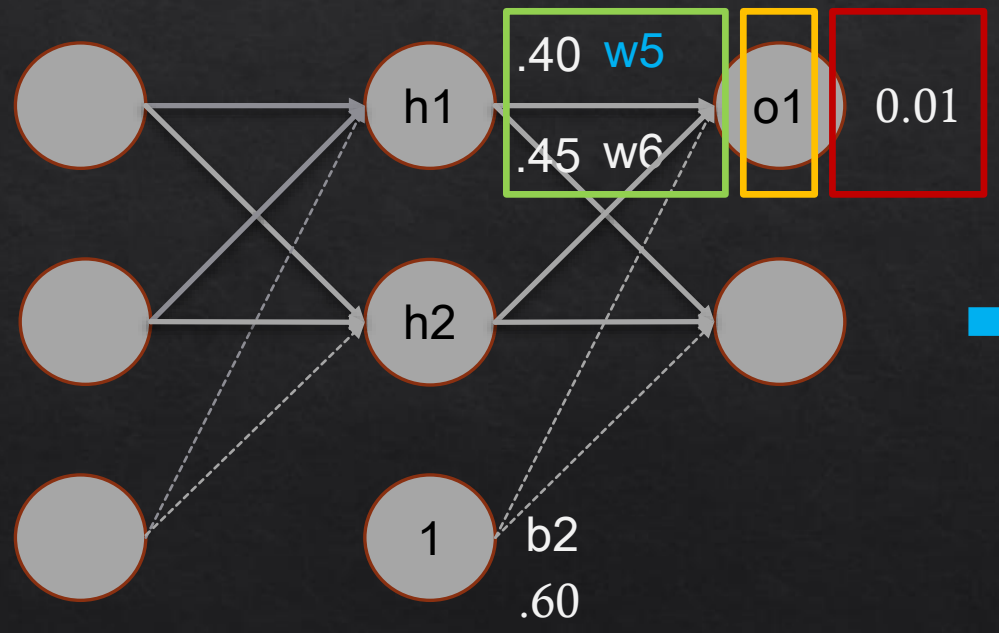
$$= \frac{1}{2}(actual_{o1} - output_{o1})^{2} + \frac{1}{2}(actual_{o2} - output_{o2})^{2}$$

$$\frac{\partial E_{total}}{\partial o1} = 2 * \frac{1}{2}(actual_{o1} - output_{o1})^{2-1} * (-1)$$

$$= 0.74136507$$

深度學習概述：梯度下降與反向傳播

- 反求 w5 過程可以拆解成：



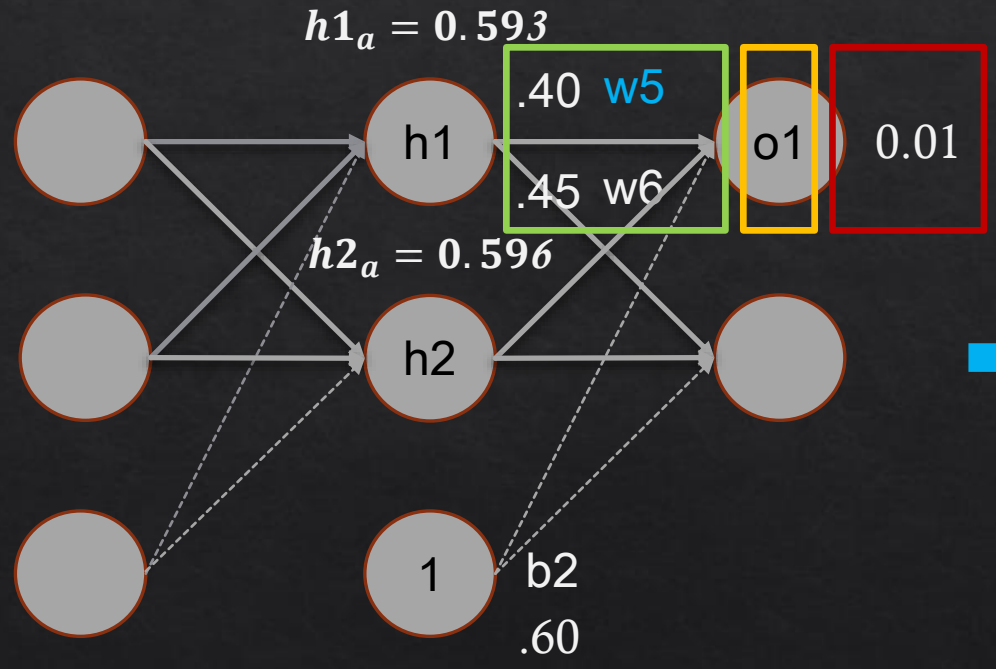
2. 處理 Activation fun. :

$$o1_a = \frac{1}{1 + e^{-o1}}$$

$$\frac{\partial o1_a}{\partial o1} = o1_a(1 - o1_a) = 0.186815602$$

深度學習概述：梯度下降與反向傳播

- 反求 w5 過程可以拆解成：



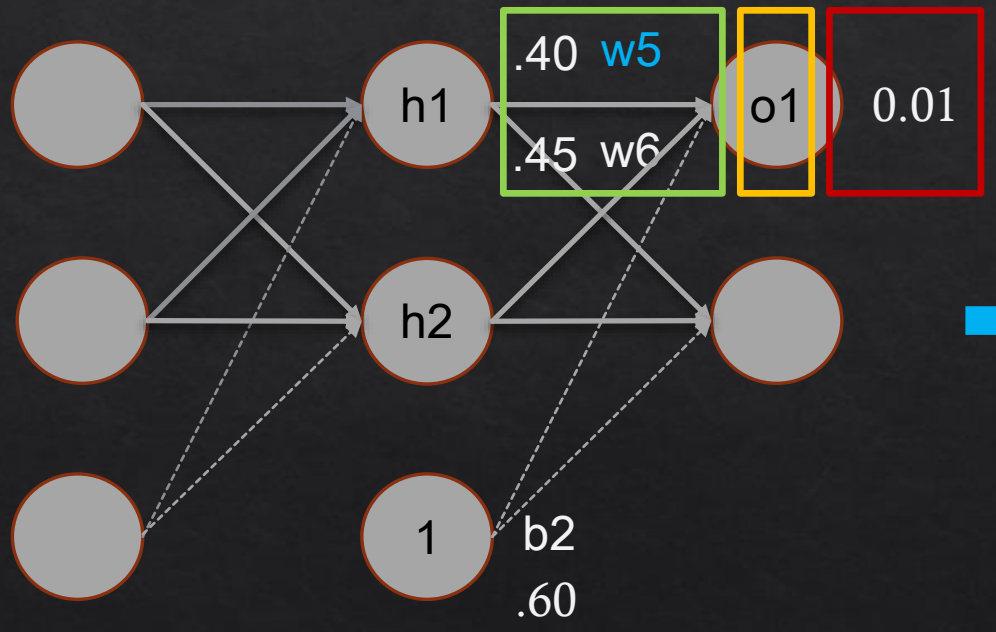
3. 處理 w5：

$$o1 = w_5 * h1_a + w_6 * h2_a + b2 * 1$$

$$\frac{\partial o1}{\partial w_5} = h1_a = 0.593269992$$

深度學習概述：梯度下降與反向傳播

- 反求 w5 過程可以拆解成：



$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial o1} * \frac{\partial o1}{\partial o1_a} * \frac{\partial o1_a}{\partial w_5}$$

$$= 0.082167041$$

深度學習概述：梯度下降與反向傳播

- 運作流程：

Step 4-2. 進行梯度下降調整權重

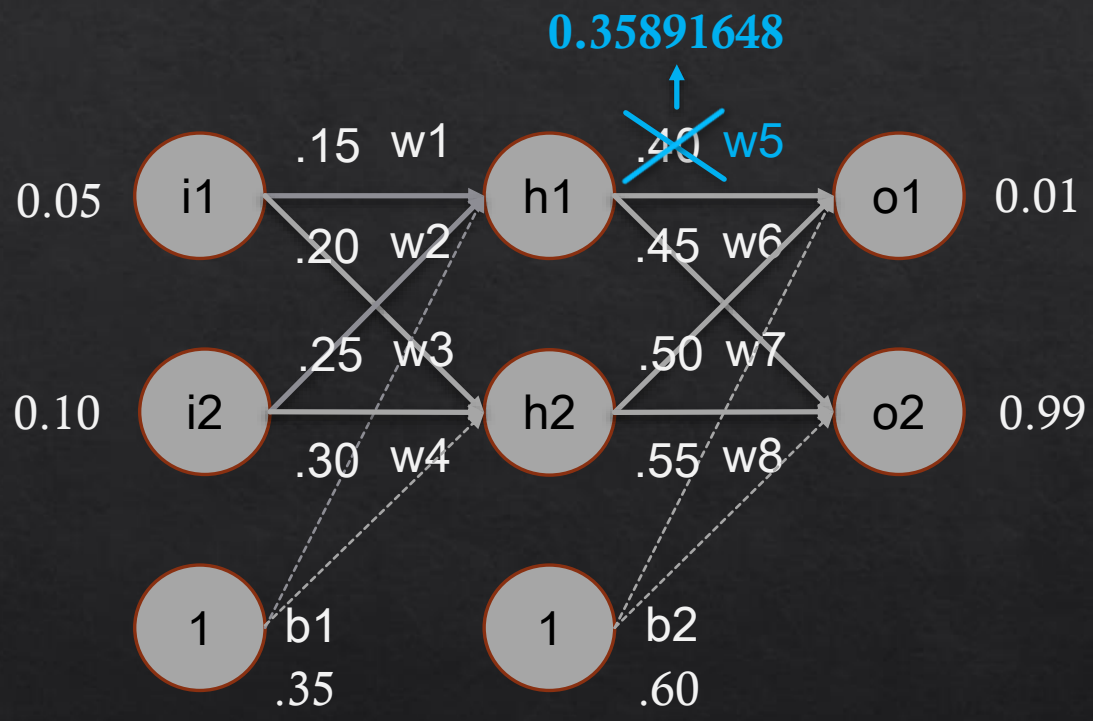
（僅以 w5 為例）

$$w_5^{new} = w_5 - \eta * \frac{\partial E_{total}}{\partial w_5}$$

$$= 0.4 - 0.5 * 0.082167041$$

$$= 0.35891648$$

（此處步伐使用 0.5）

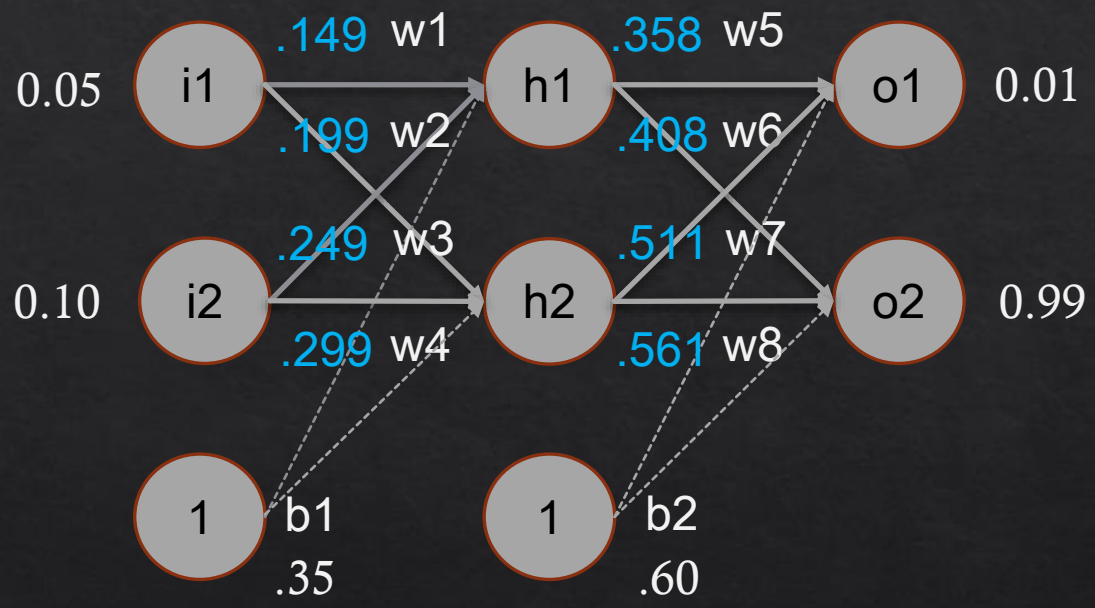


深度學習概述：梯度下降與反向傳播

- 運作流程：

Step 4-2. 依此類推修正所有權種

所有權種修改完後就完成了第一遍的反向傳播。

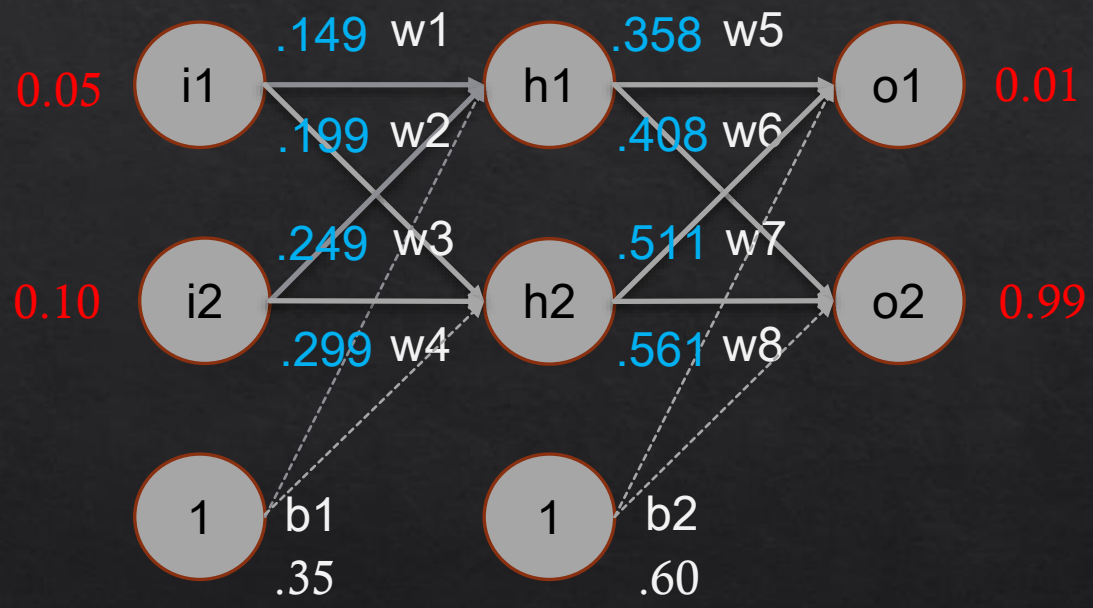


深度學習概述：梯度下降與反向傳播

- 運作流程：

Step 5. 再次進行正向傳播

不斷重覆 Step1. 到 Step5. 直到找到一組最接近真實值 (0.01, 0.99) 的 (o1, o2) ，即找到一組能使成本函數為最小的權重 weight。



Outline

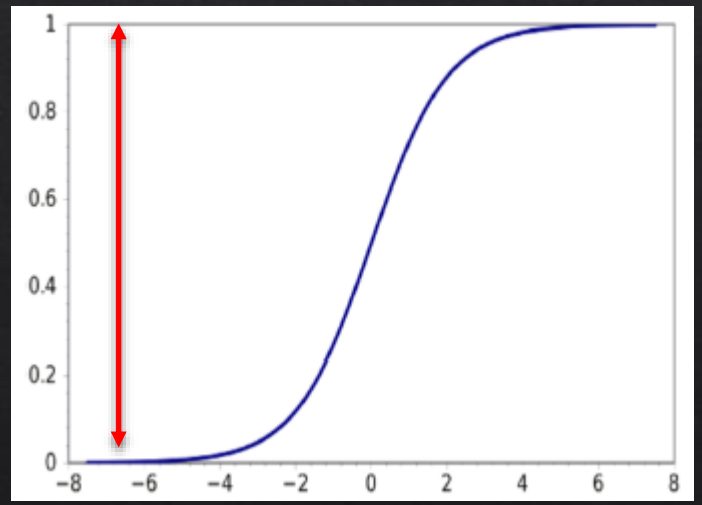
- 深度學習概述
  - 深度學習基本介紹
  - 梯度下降與反向傳播
- Keras 基礎教學
  - 30 秒內開啟 Hello World
  - 圖像識別模型
    - mnist 手寫辨識資料集
    - 使用 CNN 完成手寫辨識模型

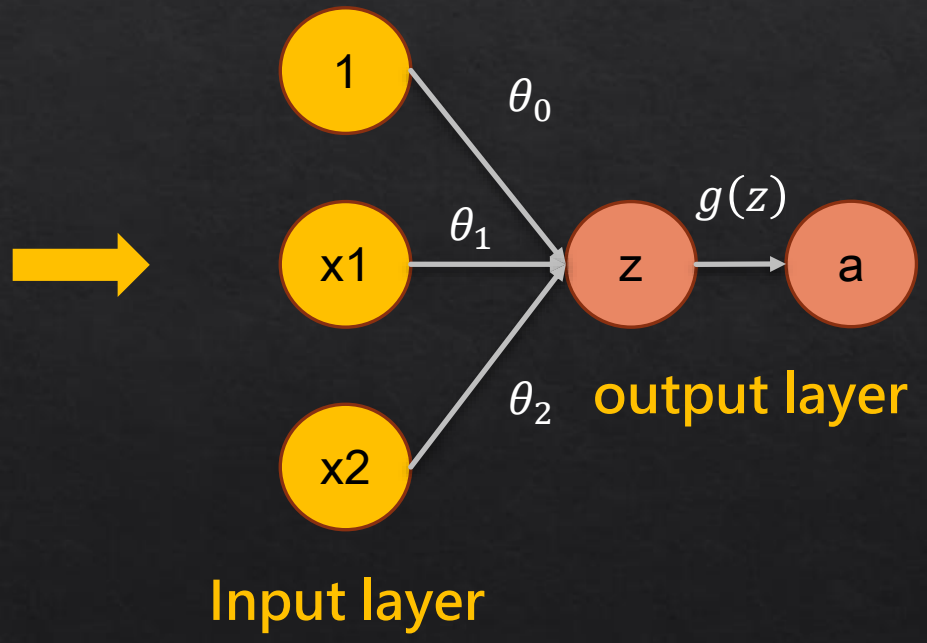
利用 Keras 在 30 秒內開啟深度學習的 Hello World

- **使用神經網路實現 Logistic regression :**

$$z = \theta_0 + \theta_1 X_1 + \theta_2 X_2$$

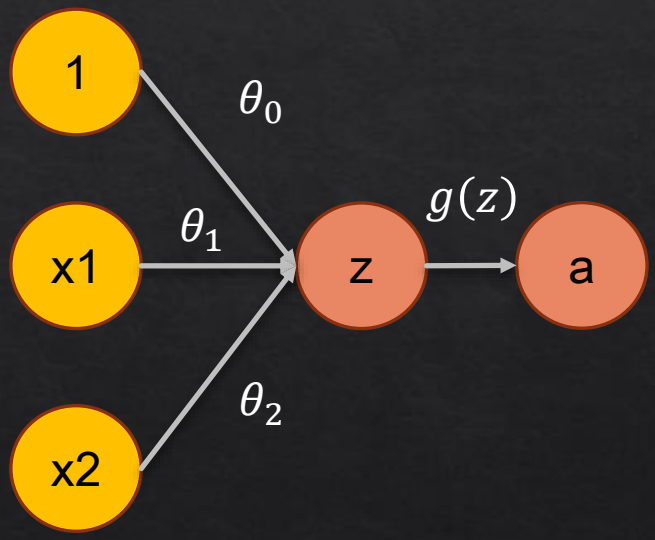
$$a = g(z) = \frac{1}{1 + e^{-z}}$$





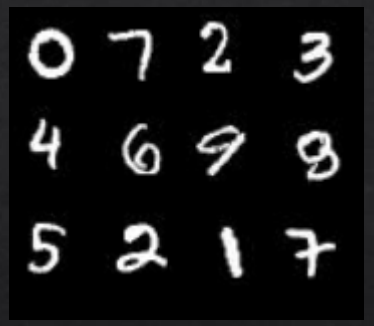
利用 Keras 在 30 秒內開啟深度學習的 Hello World

```
import os
os.environ['KERAS_BACKEND']='theano'
import numpy as np
from keras.models import Sequential
from keras.layers import Dense
from keras.utils import np_utils

data = np.array([[1, 160, 58],
                 [1, 172, 65],
                 [1, 152, 52],
                 [1, 148, 60]])
label = np.array([1,0,0,1])

model = Sequential()
model.add(Dense(output_dim=1, input_dim=3, activation='sigmoid'))
model.compile(loss='mse', optimizer='sgd', metrics=['accuracy'])
```

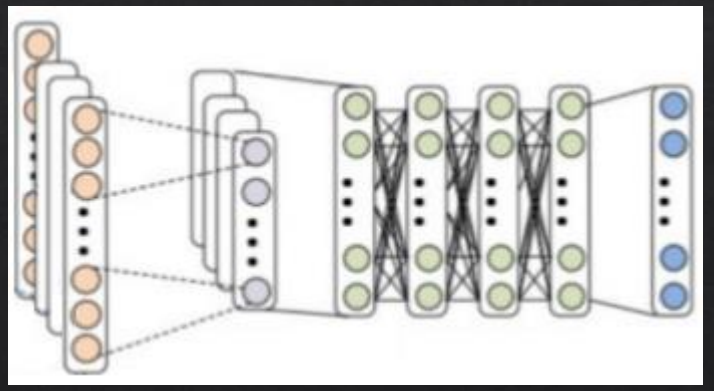


圖像識別模型：使用 CNN 完成手寫辨識模型

Input data : mnist 資料集



CNN model



label

0
7
2
3
1
2
5

圖像識別模型：使用 CNN 完成手寫辨識模型

- Import 必要的 modules

```
import os
os.environ['KERAS_BACKEND']='theano'
from PIL import Image
import numpy as np
from __future__ import absolute_import
from __future__ import print_function
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation, Flatten
from keras.layers.advanced_activations import PReLU
from keras.layers.convolutional import Convolution2D, MaxPooling2D
from keras.optimizers import SGD, Adadelta, Adagrad
from keras.utils import np_utils, generic_utils
from six.moves import range
import random
np.random.seed(1024)
```

由於 Keras 的 backend 有分成 theano 和 tensorflow 兩種，這邊我們使用 theano 作為 backend，所以需要數入這兩行來進行控制。

圖像識別模型：使用 CNN 完成手寫辨識模型

- Import 必要的 modules

```
import os
os.environ['KERAS_BACKEND']='theano'
from PIL import Image
import numpy as np
from __future__ import absolute_import
from __future__ import print_function
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation, Flatten
from keras.layers.advanced_activations import PReLU
from keras.layers.convolutional import Convolution2D, MaxPooling2D
from keras.optimizers import SGD, Adadelta, Adagrad
from keras.utils import np_utils, generic_utils
from six.moves import range
import random
np.random.seed(1024)
```

PIL (sudo pip install Pillow) 包為 python 模組中在處理圖像上常用的 module。

numpy 提供了 python 中最常見的 array 形式，而 array 形式也是 Keras 要求的 input 格式。

圖像識別模型：使用 CNN 完成手寫辨識模型

- Import 必要的 modules

```
import os
os.environ['KERAS_BACKEND']='theano'
from PIL import Image
import numpy as np
from __future__ import absolute_import
from __future__ import print_function
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation, Flatten
from keras.layers.advanced_activations import PReLU
from keras.layers.convolutional import Convolution2D, MaxPooling2D
from keras.optimizers import SGD, Adadelta, Adagrad
from keras.utils import np_utils, generic_utils
from six.moves import range
import random
np.random.seed(1024)
```

CNN 會使用到的 modules，這次先不細項討論 CNN 的結構，僅帶大家嘗試使用 CNN。

圖像識別模型：使用 CNN 完成手寫辨識模型

- **讀入圖片檔：**

```
def load_data():
    data = np.empty((42000,1,28,28),dtype="float32")
    label = np.empty((42000,),dtype="uint8")

    imgs = os.listdir("/home/bigmoumou/Desktop/keras/mnist")
    num = len(imgs)  # 42000
    for i in range(num):
        img = Image.open("/home/bigmoumou/Desktop/keras/mnist/"+imgs[i])
        arr = np.asarray(img,dtype="float32")
        data[i,:,:,:] = arr
        label[i] = int(imgs[i].split('.')[0])
    return data,label
```

創建一個 load_data( ) 的 function，後續呼叫此 function 時會執行內部的所有 code。

圖像識別模型：使用 CNN 完成手寫辨識模型

- 讀入圖片檔：

```
def load_data():
    data = np.empty((42000,1,28,28),dtype="float32")
    label = np.empty((42000,),dtype="uint8")

    imgs = os.listdir("/home/bigmoumou/Desktop/keras/mnist")
    num = len(imgs)  # 42000
    for i in range(num):
        img = Image.open("/home/bigmoumou/Desktop/keras/mnist/"+imgs[i])
        arr = np.asarray(img,dtype="float32")
        data[i,:,:,:] = arr
        label[i] = int(imgs[i].split('.')[0])
    return data,label
```

創建空的 array 用於放置 42,000 筆樣本的 data 和 label，data 中的每個圖像都是 28 * 28，label 則是 0 – 9 的數字。

圖像識別模型：使用 CNN 完成手寫辨識模型

- **讀入圖片檔：**

```
def load_data():
    data = np.empty((42000,1,28,28),dtype="float32")
    label = np.empty((42000,),dtype="uint8")

    imgs = os.listdir("/home/bigmoumou/Desktop/keras/mnist")
    num = len(imgs)  # 42000
    for i in range(num):
        img = Image.open("/home/bigmoumou/Desktop/keras/mnist/"+imgs[i])
        arr = np.asarray(img,dtype="float32")
        data[i,:,:,:] = arr
        label[i] = int(imgs[i].split('.')[0])
    return data,label
```

列出該路徑下的資料夾內的所有檔案名稱。

圖像識別模型：使用 CNN 完成手寫辨識模型

- 讀入圖片檔：

```
def load_data():
    data = np.empty((42000,1,28,28),dtype="float32")
    label = np.empty((42000,),dtype="uint8")

    imgs = os.listdir("/home/bigmoumou/Desktop/keras/mnist")
    num = len(imgs)  # 42000
    for i in range(num):
        img = Image.open("/home/bigmoumou/Desktop/keras/mnist/"+imgs[i])
        arr = np.asarray(img,dtype="float32")
        data[i,:,:,:] = arr
        label[i] = int(imgs[i].split('.')[0])
    return data,label
```

len() 為計算有幾個個數，用在這就是該資料夾下有幾個檔案。

圖像識別模型：使用 CNN 完成手寫辨識模型

- **讀入圖片檔：**

```
def load_data():
    data = np.empty((42000,1,28,28),dtype="float32")
    label = np.empty((42000,),dtype="uint8")

    imgs = os.listdir("/home/bigmoumou/Desktop/keras/mnist")
    num = len(imgs)  # 42000
    for i in range(num):
        img = Image.open("/home/bigmoumou/Desktop/keras/mnist/"+imgs[i])
        arr = np.asarray(img,dtype="float32")
        data[i,:,:,:] = arr
        label[i] = int(imgs[i].split('.')[0])
    return data,label
```

建立一個迴圈，把 range(42000) 從頭到尾疊代一遍，即從 0 到 42000 - 1。

圖像識別模型：使用 CNN 完成手寫辨識模型

- **讀入圖片檔：**

```
def load_data():
    data = np.empty((42000,1,28,28),dtype="float32")
    label = np.empty((42000,),dtype="uint8")

    imgs = os.listdir("/home/bigmoumou/Desktop/keras/mnist")
    num = len(imgs)  # 42000
    for i in range(num):
        img = Image.open("/home/bigmoumou/Desktop/keras/mnist/"+imgs[i])
        arr = np.asarray(img,dtype="float32")
        data[i,:,:,:] = arr
        label[i] = int(imgs[i].split('.')[0])
    return data,label
```

利用迴圈，該資料夾中的檔案各別讀取進來，imgs[i] 就是第 i 個檔案名稱。

圖像識別模型：使用 CNN 完成手寫辨識模型

- **讀入圖片檔：**

```
def load_data():
    data = np.empty((42000,1,28,28),dtype="float32")
    label = np.empty((42000,),dtype="uint8")

    imgs = os.listdir("/home/bigmoumou/Desktop/keras/mnist")
    num = len(imgs)  # 42000
    for i in range(num):
        img = Image.open("/home/bigmoumou/Desktop/keras/mnist/"+imgs[i])
        arr = np.asarray(img,dtype="float32")
        data[i,:,:,:] = arr
        label[i] = int(imgs[i].split('.')[0])
    return data,label
```

把讀入的圖片資料轉成 numpy 的 array 型態。

圖像識別模型：使用 CNN 完成手寫辨識模型

- **讀入圖片檔：**

```
def load_data():
    data = np.empty((42000,1,28,28),dtype="float32")
    label = np.empty((42000,),dtype="uint8")

    imgs = os.listdir("/home/bigmoumou/Desktop/keras/mnist")
    num = len(imgs)  # 42000
    for i in range(num):
        img = Image.open("/home/bigmoumou/Desktop/keras/mnist/"+imgs[i])
        arr = np.asarray(img,dtype="float32")
        data[i,:,:,:] = arr
        label[i] = int(imgs[i].split('.')[0])
    return data,label
```

再把已轉成 array 型態的圖片資料一筆筆放到一開始建立的空 array 之中。

圖像識別模型：使用 CNN 完成手寫辨識模型

- **讀入圖片檔：**

```
def load_data():
    data = np.empty((42000,1,28,28),dtype="float32")
    label = np.empty((42000,),dtype="uint8")

    imgs = os.listdir("/home/bigmoumou/Desktop/keras/mnist")
    num = len(imgs)  # 42000
    for i in range(num):
        img = Image.open("/home/bigmoumou/Desktop/keras/mnist/"+imgs[i])
        arr = np.asarray(img,dtype="float32")
        data[i,:,:,:] = arr
        label[i] = int(imgs[i].split('.')[0])
    return data,label
```

由於檔案名稱已故意取成「label.第幾張.jpg」，ex 0.46.jpg，故該行程式碼利用“.”去切分檔名，並僅擷取第一部份的“label”。

圖像識別模型：使用 CNN 完成手寫辨識模型

- 正式讀入圖檔並將資料打散：

```
X_data, Y_data  = load_data()
# 打亂資料
index = [i for i in range(len(X_data))]
random.shuffle(index)
X_data = X_data[index]
Y_data = Y_data[index]
```

使用剛建立好的 load_data()，並將分別存成 X_data 和 Y_data。

圖像識別模型：使用 CNN 完成手寫辨識模型

- 正式讀入圖檔並將資料打散：

```
X_data, Y_data  = load_data()
# 打亂資料
index = [i for i in range(len(X_data))]
random.shuffle(index)
X_data = X_data[index]
Y_data = Y_data[index]
```

製造一筆由 0 – 41999 的 index，並使用 shuffle 函數將其隨機打亂，再讓 X_data 和 Y_data 照著這個已打亂的 index 排序完成數據打亂。

```
X_train, X_test = X_data[:30000,:,:,:], X_data[30000:,:,:,:]
Y_train, Y_test = Y_data[:30000,], Y_data[30000:,]
```

取前 30000 筆資料作為訓練集，後 12000 做為訓練集。

圖像識別模型：使用 CNN 完成手寫辨識模型

- 將資料轉成 Keras 要求的 input 格式：

```
# input image dimensions
X_train = X_train.reshape(X_train.shape[0], 28, 28, 1 )
input_shape = (28, 28, 1)
```

將 X_train 從原本的 (42000, 1, 28, 28) 轉成 (42000, 28, 28, 1)

```
Y_train = np_utils.to_categorical(Y_train, 10)
```

利用 Keras 內建的 np_utils.to_categorical

把分類資料的 label 轉成 one-hot

圖像識別模型：使用 CNN 完成手寫辨識模型

- 使用 Keras 官方文檔的 CNN model：

```
model = Sequential()
model.add(Convolution2D(4, 5, 5, border_mode='valid',input_shape = input_shape))
model.add(Activation('tanh'))
model.add(Convolution2D(8, 3, 3, border_mode='valid'))
model.add(Activation('tanh'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Convolution2D(16, 3, 3, border_mode='valid'))
model.add(Activation('tanh'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(128, init='normal'))
model.add(Activation('tanh'))
model.add(Dense(10, init='normal'))
model.add(Activation('softmax'))
```

圖像識別模型：使用 CNN 完成手寫辨識模型

- 開始訓練 CNN model：

```
model.fit(X_train, Y_train, batch_size=100, nb_epoch=10,shuffle=True,verbose=1)
```

```
Epoch 1/10
30000/30000 [==============================] - 46s - loss: 0.0238 - acc: 0.9936
Epoch 2/10
30000/30000 [==============================] - 47s - loss: 0.0205 - acc: 0.9936
Epoch 3/10
30000/30000 [==============================] - 49s - loss: 0.0171 - acc: 0.9955
Epoch 4/10
30000/30000 [==============================] - 50s - loss: 0.0216 - acc: 0.9940
Epoch 5/10
30000/30000 [==============================] - 48s - loss: 0.0233 - acc: 0.9926
Epoch 6/10
30000/30000 [==============================] - 52s - loss: 0.0233 - acc: 0.9924
Epoch 7/10
30000/30000 [==============================] - 50s - loss: 0.0229 - acc: 0.9918
Epoch 8/10
30000/30000 [==============================] - 48s - loss: 0.0367 - acc: 0.9873
Epoch 9/10
30000/30000 [==============================] - 47s - loss: 0.0234 - acc: 0.9919
Epoch 10/10
30000/30000 [==============================] - 49s - loss: 0.0120 - acc: 0.9967
```

圖像識別模型：使用 CNN 完成手寫辨識模型

- 使用測試集評估 model 的準確度：

```
score = model.evaluate(X_test, Y_test)
```

```
12000/12000 [==============================] - 15s
```

```
print("Total Loss on Testing Set:", score[0])
print("Accuracy of Testing Set:", score[1])
```

```
Total Loss on Testing Set: 0.0554579609113
Accuracy of Testing Set: 0.983833333333
```

使用 30,000 個訓練集訓練出的 CNN 模型有 99.6% 的整體準確度，而使用 12,000 個測試集去做 predict 出來的結果也有 98.38% 的整體準確度。