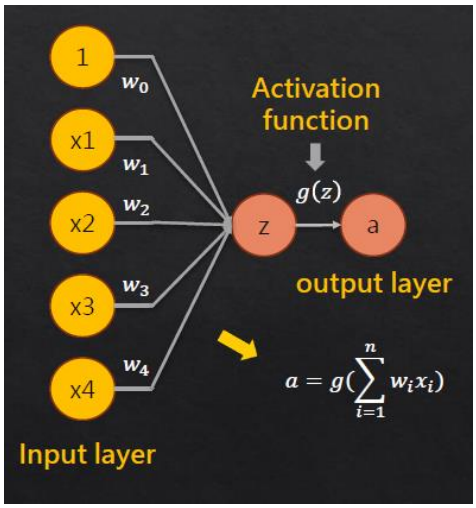
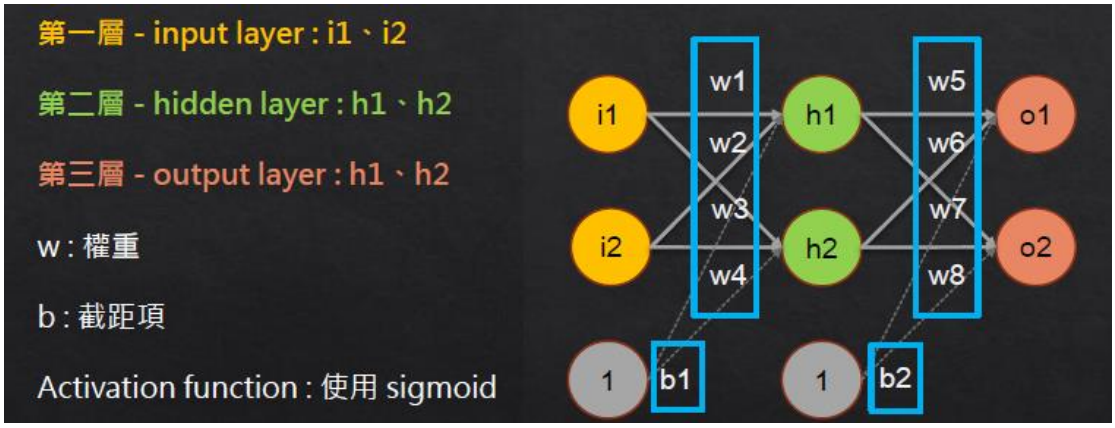
類神經網路 (Neural Network)

一、 原理

1. 類神經網路

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。

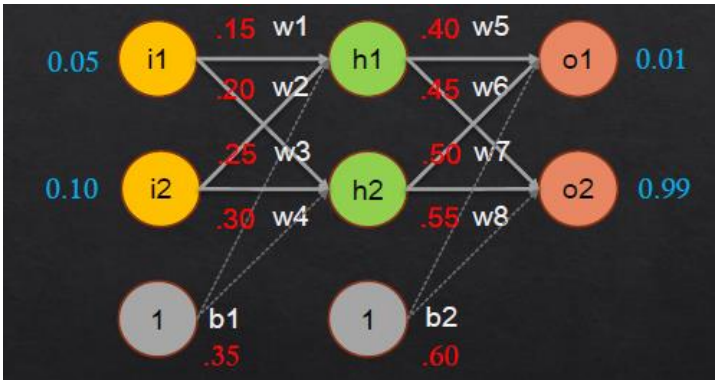




2. 類神經網路前向傳遞(Forward propagation)與倒傳遞（Backward propagation) 運作流程

(多層感知機 Multilayer perceptron, MLP)

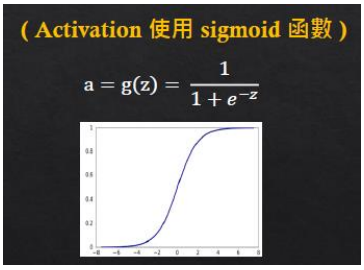
A. 給定一組初始權重，給定 input data & output label

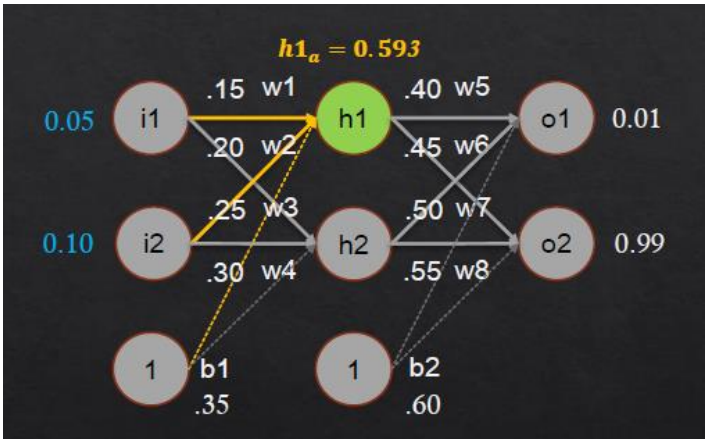


B. 進行正向傳播（Input layer > hidden layer）

$h1 = 0.05 * 0.15 + 0.10 * 0.20 + 1 * 0.35 = 0.3775$

$$\boldsymbol{h1_a} = \frac{1}{11 + e{-0.3775}} = \mathbf{0.593269992}$$

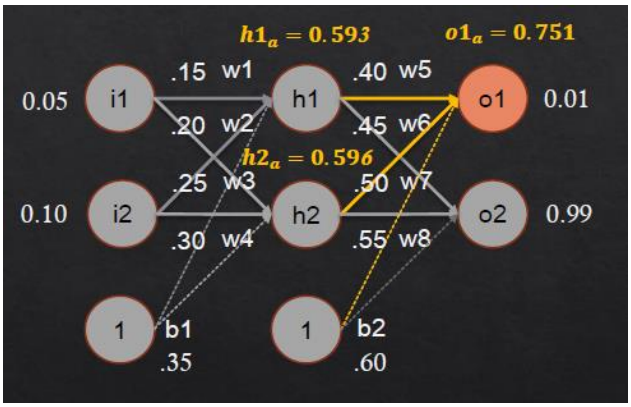




C. 進行正向傳播（hidden layer > output layer）

o1 = 0.593∗0.40+0.596∗0.45+1∗0.60 =1.105905967

$\boldsymbol{o1_a}$ = 11 + e−1.105905967 = **0.75136507**



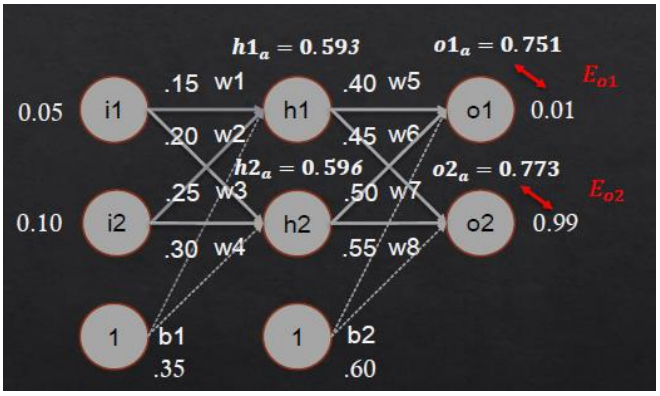
D. 開始計算誤差

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。使用 square error 作為 cost function。

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

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

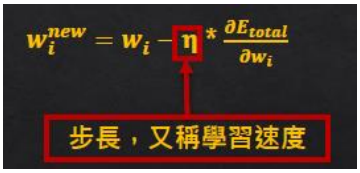
$$E_{total} = E_{o1} + E_{o2} = 0.298371109$$



E. 透過反向傳播與梯度下降調整權重

- 梯度下降法調整觀念:

梯度方向為成本函數上升的方向，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。



(朝著負梯度方向調整權重)

(步長小　調整過程慢、易陷入局部最適。步長大　不容易收斂。)

- 透過反向傳播求出偏導數 (僅以 w5 為例)

最佳化的目的就是讓「所有樣本的誤差均方和」越小越好，所以目標是 $\frac{\partial E_{total}}{\partial w_i}$

以 w5 為例:

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

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

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

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

處理成本函數:

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

$$\frac{\partial E_{total}}{\partial o1} = 2 * \frac{1}{2}(actual_{o1} - output_{o1})^{2-1} * (-1) = 0.74136507$$

處理 Activation fun:

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

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

處理 w5 :

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

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

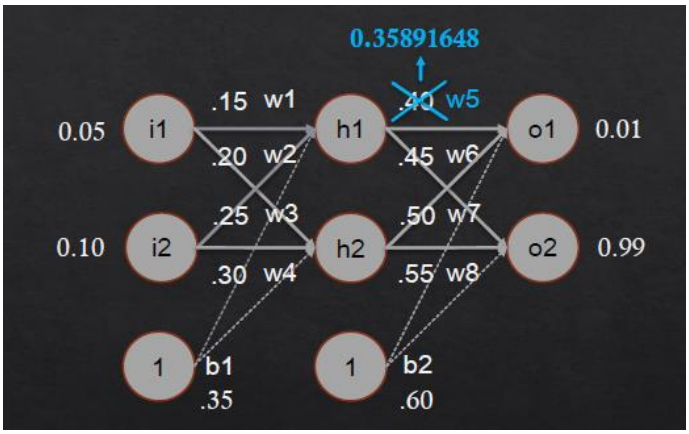
$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial o1} * \frac{\partial o1}{\partial o1_a} * \frac{\partial o1_a}{\partial w_5} = 0.082167041$$

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

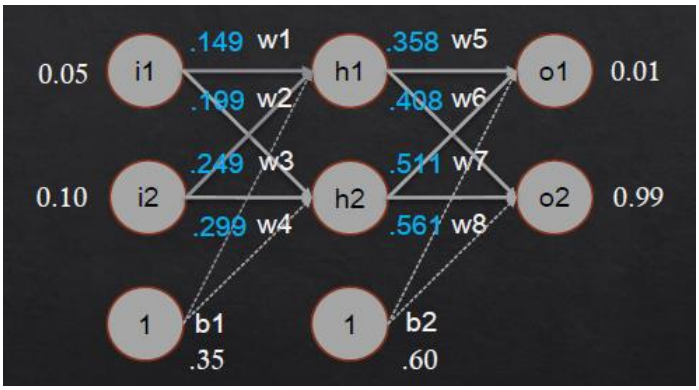
$$= 0.4 - 0.5 * 0.082167041$$

$$= 0.35891648$$

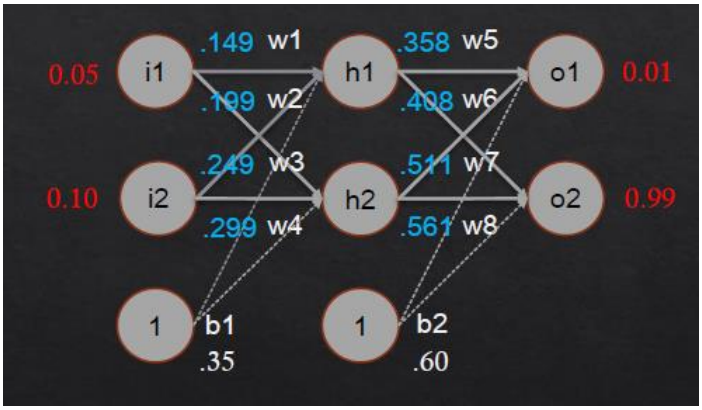
(此處步伐使用 0.5)



- 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。



- 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。



3. 討論

- 通常隱藏層處理單元數目愈多收斂愈慢，但可達到更小的誤差值，特別是訓練範例的誤差。但是，超過一定數目之後，再增加則對降低測試範例誤差幾乎沒有幫助，徒然增加執行時間。
- 隱藏層處理單元數目的選取原則:
  隱藏層處理單元數目= (輸入層單元數目 + 輸出層單元數) / 2
  隱藏層處理單元數目= (輸入層單元數目 × 輸出層單元數)$^{1/2}$
- 問題的雜訊高，隱藏層處理單元數目宜少
- 問題的複雜性高，隱藏層處理單元數目宜多
- 測試範例誤差遠高於訓練範例誤差(overfitting)，隱藏層處理單元數目宜減少；反之，宜增加。
- 通常隱藏層的層數為一層到二層時有最好的收斂性質，太多層或是太少層其收斂結果均較差。
- 一般問題可取一層隱藏層，較複雜的問題則取二層隱藏層。
- 先用無隱藏層架構做做看。
- 通常學習率太大或太小對網路的收斂性質均不利。依據經驗取 0.5，或 0.1 到 1 的值作為學習速率的值，大都可以得到良好的收斂性。

二、 參考資料

1. Class Handout, Lee, Chia-Jung professor, MDM64001, School of Big Data Management, Soochow University。

2. 用 Keras 開啟深度學習的 Hello World，陳俊豪、謝長潤，2016/11/24。

3. 機器學習- 神經網路(多層感知機 Multilayer perceptron, MLP) 含倒傳遞(Backward propagation)詳細推導

   https://medium.com/@chih.sheng.huang821/%E6%A9%9F%E5%99%A8%E5%AD%B8%E7%BF%92-%E7%A5%9E%E7%B6%93%E7%B6%B2%E8%B7%AF-%E5%A4%9A%E5%B1%A4%E6%84%9F%E7%9F%A5%E6%A9%9F-multilayer-perceptron-mlp-%E5%90%AB%E8%A9%B3%E7%B4%B0%E6%8E%A8%E5%B0%8E-ee4f3d5d1b41