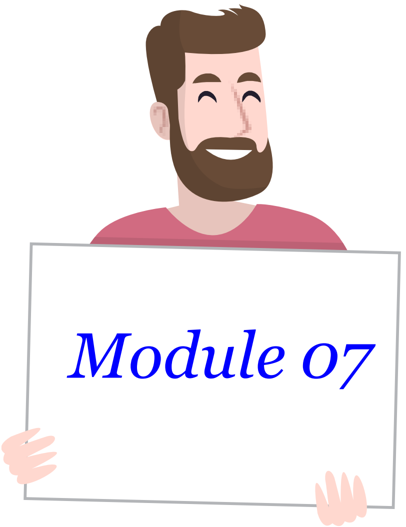
優化原理及神經網路驗證



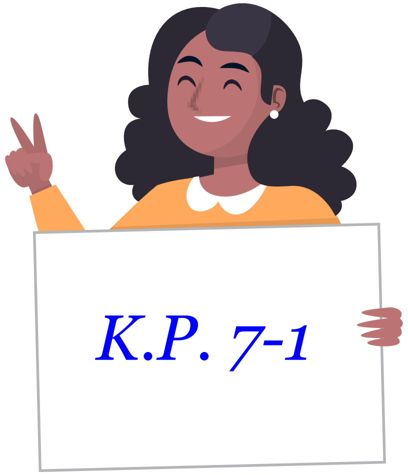
Estimated time:
45 min.

學習目標

- **7-1: 優化原理與Backpropagation**
- **7-2: 驗證神經網路**
- **7-3: DNN神經網路數值範例**

7-1:優化原理與Backpropagation

- 優化原理
- Backpropagation



優化原理

- 神經網路會使用優化器將網路內的參數優化到適當的數值
  – 例如使用Gradient Descent, Adagrad, ……
- 但在使用這些優化器時，我們常需要針對損失函數做微分，由於神經網路函數非常複雜，所以如何對網路微分是一件很難的事情
  – 為了要解決這個問題，於是有人提出了Backpropagation這樣的演算法

隨機選取 θ_1 當起始點

計算 $\frac{dC(\theta_1)}{d\theta}$

$$\theta_2 \leftarrow \theta_1 - \eta \boxed{\frac{dC(\theta_1)}{d\theta}}$$

如何求神經網路的微分，是一件困難的事

Backpropagation

- **Backpropagation是為了解決將神經網路微分所提出來的演算法**
  - **Backpropagation可以快速有效求出損失函數對神經網路任一參數之微分的方法**

$$\frac{dC(\theta_1)}{d\theta}$$

Backpropagation證明

- 將DNN神經網路運算式子拆解如下：

$$y = f(x) = \sigma(W^L \ldots \sigma(W^2\sigma(W^1X + b^1) + b^2) \ldots + b^L)$$

$$\Downarrow$$

$$z^1 = W^1X + b^1$$
$$a^1 = \sigma(z^1)$$
$$z^2 = W^2a^1 + b^2$$
$$a^2 = \sigma(z^2)$$
$$\vdots$$
$$z^l = W^l a^{l-1} + b^l$$
$$a^l = \sigma(z^l)$$

Backpropagation證明

- 目標就是要求出損失函數對神經網路任一參數之微分
  - 根據微積分的連鎖率，可以拆解成兩項相乘

$$\frac{\partial C}{\partial w_{ij}^{l}} = \frac{\partial C}{\partial z_i^{l}} * \frac{\partial z_i^{l}}{\partial w_{ij}^{l}}$$

1. Calculate $\frac{\partial z_i^{l}}{\partial w_{ij}^{l}}$

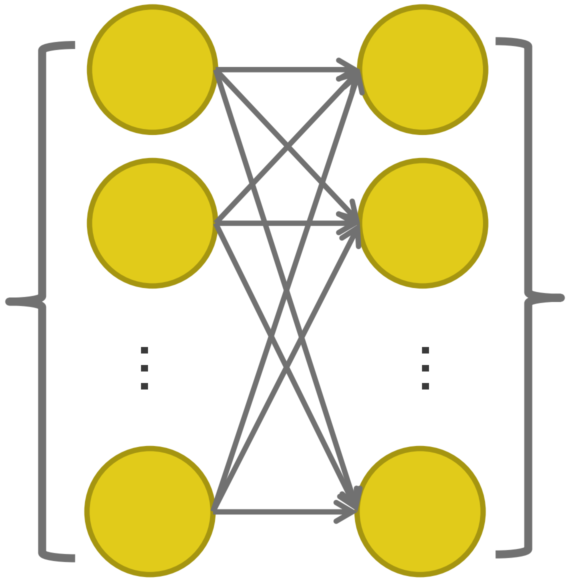
2. Calculate $\frac{\partial C}{\partial z_i^{l}}$ (**error signal**)

Backpropagation證明

- **計算第一項** $\frac{\partial z_i^l}{\partial w_{ij}^l}$

隱藏層L-1 隱藏層L

L-1層的第j個神經元



L層的第i個神經元

$if\ l = 1$ (輸入層 → 隱藏層1)

$$z_i^1 = \sum_j w_{ij}^1 x_j^r + b_i^1 \qquad \frac{\partial z_i^1}{\partial w_{ij}^1} = x_j^{\text{[redacted]}}$$

$if\ l > 1$(隱藏層$L-1$ → 隱藏層L)

$$z_i^l = \sum_j w_{ij}^l a_j^{l-1} + b_i^l \qquad \frac{\partial z_i^l}{\partial w_{ij}^l} = a_j^{l-1}$$

Backpropagation證明

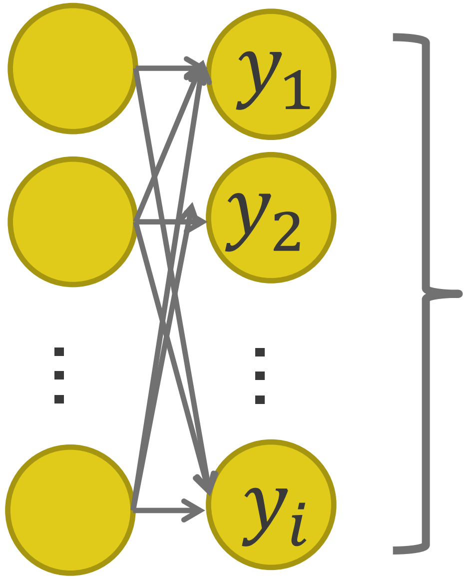
- **計算第二項 $\frac{\partial C}{\partial z_i^l}$ (又叫做error signal，δ_i^l)**
  - **要計算這項比較複雜，需要分成兩個步驟**

1. 計算L層的error signal δ^L
2. 計算兩相鄰層 δ^l 及 δ^{l+1}關係

Backpropagation證明

- **計算δ_i^l步驟1：**

隱藏層L　　輸出層



輸出層的第i個神經元

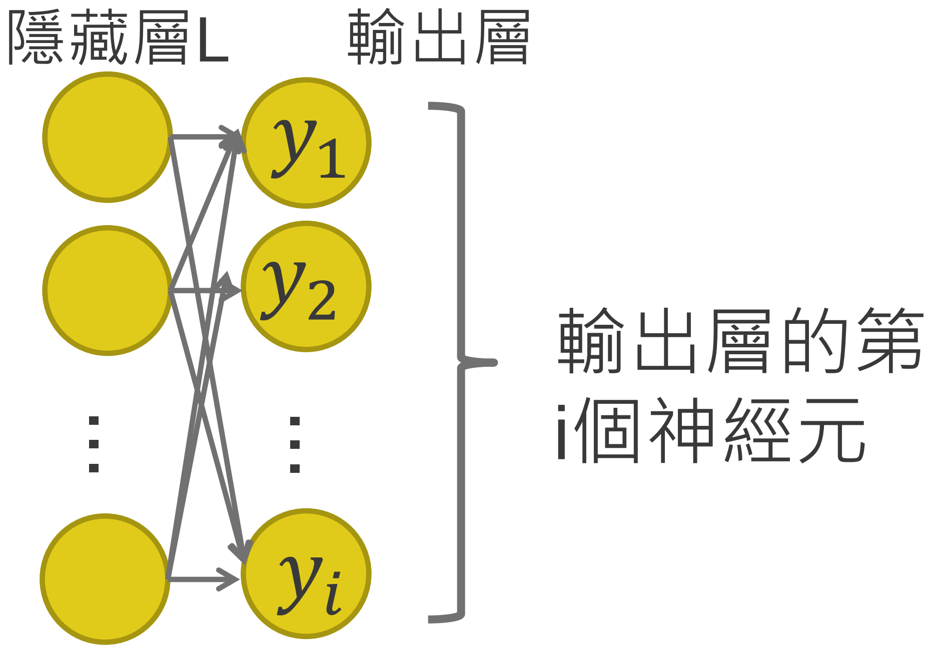
$$\frac{\partial C}{\partial z_i^l} \quad (\text{error signal} \quad \delta_i^l \quad)$$

1. 計算隱藏層L δ^L ：

$$\delta_i^L = \frac{\partial C}{\partial z_i^L} = \frac{\partial y_i}{\partial z_i^L} * \frac{\partial C}{\partial y_i} = \sigma'(z_i^L) * \frac{\partial C}{\partial y_i}$$

Backpropagation證明

- **計算δ_i^l步驟1：**



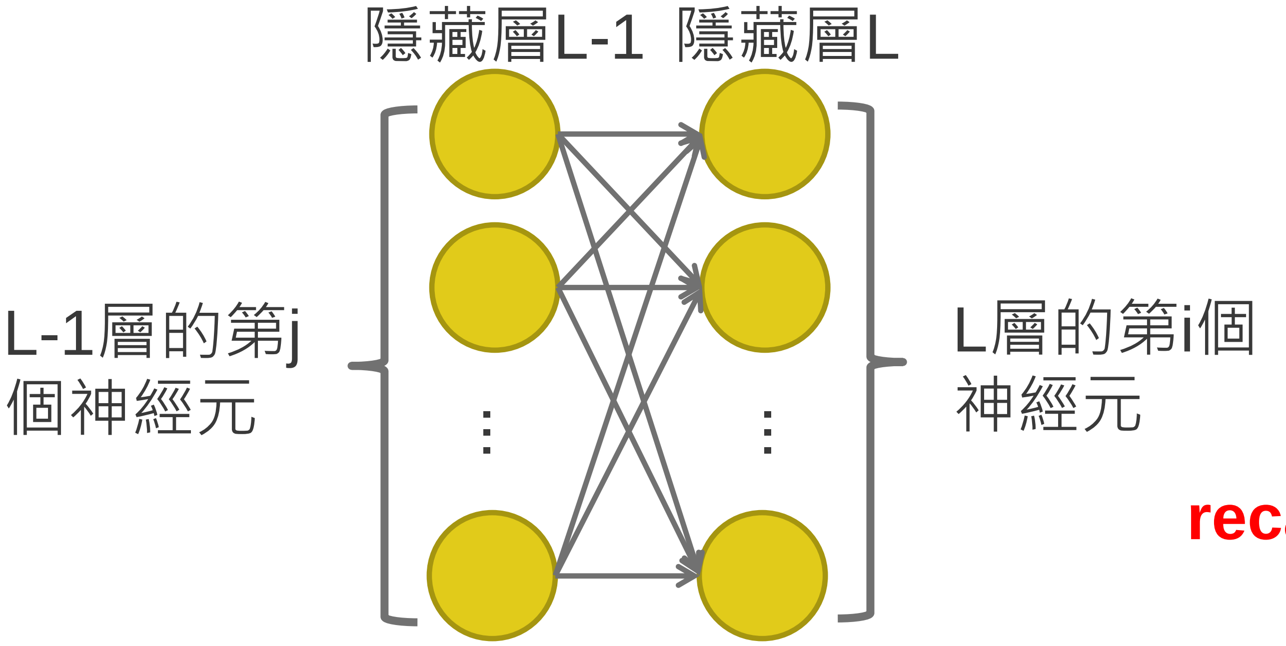
1. 計算隱藏層L δ^L(矩陣形式)：

$$\delta^L = \sigma'(z^L) \bullet \nabla C(y)$$

$$\sigma'(z^L) = \begin{bmatrix} \sigma'(z_1^L) \\ \sigma'(z_2^L) \\ \vdots \\ \sigma'(z_n^L) \end{bmatrix} \qquad \nabla C'(y) = \begin{bmatrix} \frac{\partial C}{\partial y_1} \\ \frac{\partial C}{\partial y_2} \\ \vdots \\ \frac{\partial C}{\partial y_n} \end{bmatrix}$$

Backpropagation證明

- 計算兩相鄰層 δ^l 及 δ^{l+1}關係



$$\frac{\partial C}{\partial z_i^l} \quad (\text{error signal} \quad \delta_i^l \;)$$

2.計算兩相鄰層 δ^l 及 δ^{l+1}關係

$$\delta_i^l = \frac{\partial C}{\partial z_i^l} = \frac{\partial a_i^l}{\partial z_i^l} \sum_k \frac{\partial z_k^{l+1}}{\partial a_i^l} \frac{\partial C}{\partial z_k^{l+1}}$$

recall

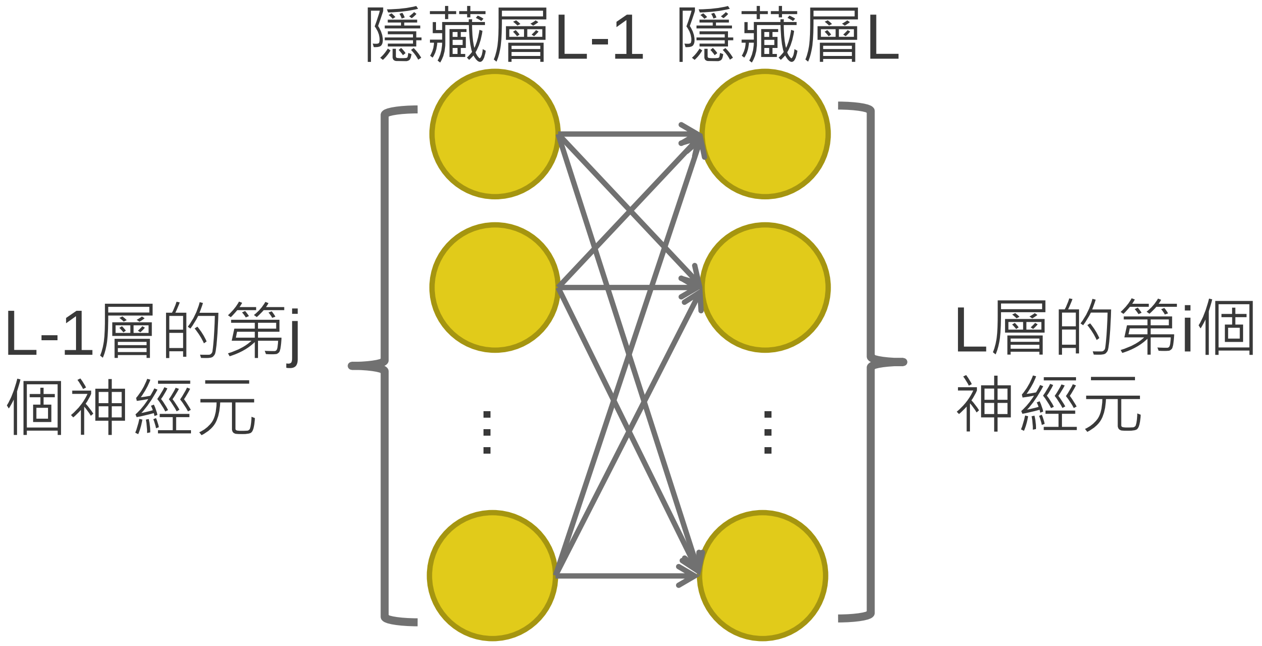
$$\boxed{\frac{\partial a_i^l}{\partial z_i^l} = \sigma'(z_i^l), \quad \frac{\partial z_k^{l+1}}{\partial a_i^l} = w_{ki}^{l+1}, \quad \frac{\partial C}{\partial z_k^{l+1}} = \delta_k^{l+1}}$$

$$\delta_i^l = \sigma'(z_i^l) \sum_k w_{ki}^{l+1} \delta_k^{l+1}$$

Backpropagation證明

- **計算兩相鄰層 $\boldsymbol{\delta^l}$ 及 $\boldsymbol{\delta^{l+1}}$關係**

$$\frac{\partial C}{\partial z_i^l} \quad (\text{error signal} \quad \delta_i^l \ )$$



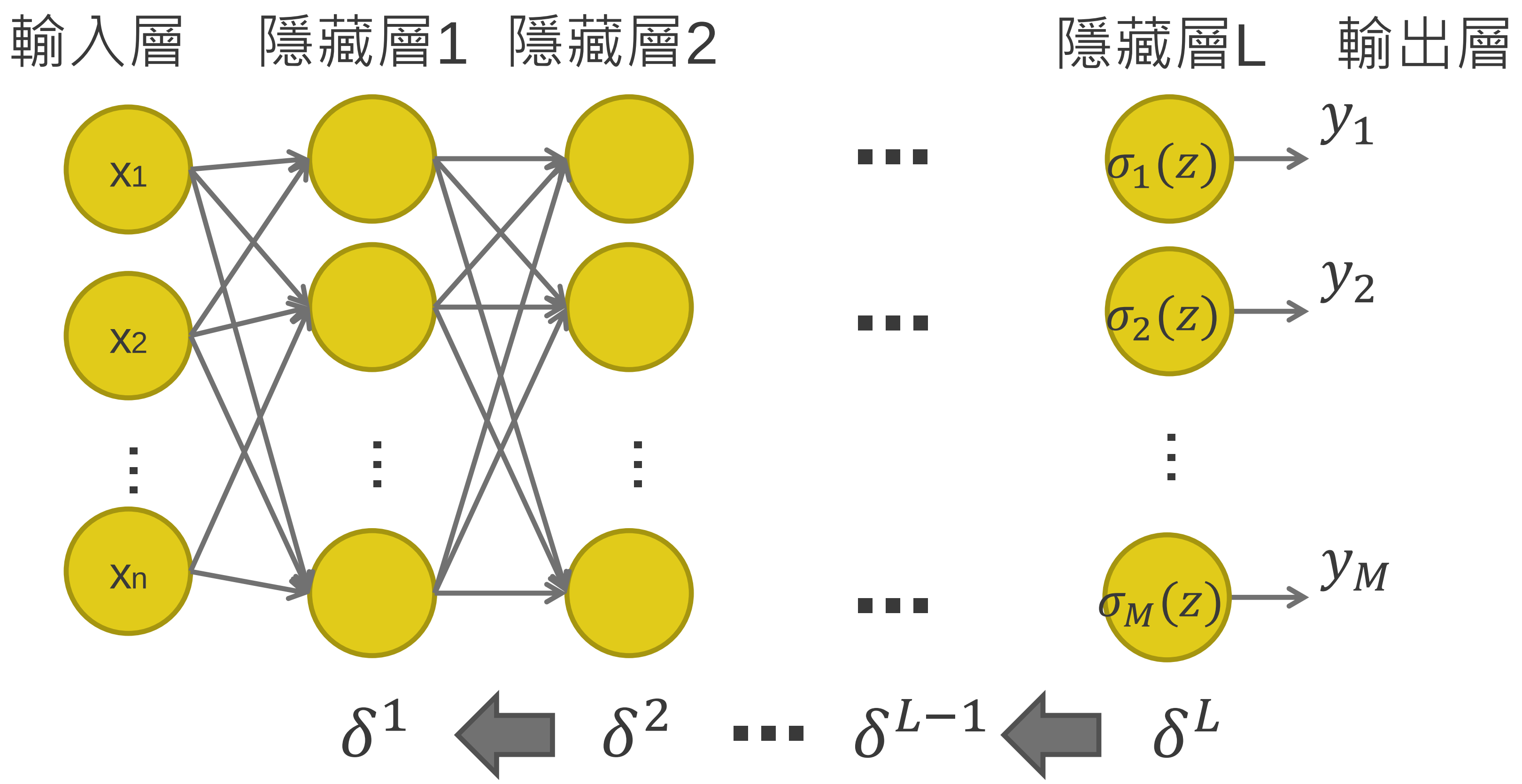
2.計算兩相鄰層 $\boldsymbol{\delta^l}$ 及 $\boldsymbol{\delta^{l+1}}$關係(矩陣形式)

$$\delta^l = \sigma'(z^l) \bullet (W^{l+1})^T \delta^{l+1}$$

Backpropagation證明

- 計算兩相鄰層 δ^l 及 δ^{l+1}關係

$$\begin{cases} \delta^L = \sigma'(z^L) \bullet \nabla C(y) \\ \delta^l = \sigma'(z^l) \bullet (W^{l+1})^T \delta^{l+1} \end{cases}$$



Backpropagation總結

- **Backpropagation利用微積分的連鎖率，將優化器所需要的微分項拆成兩部分，並利用數學技巧去分別求出結果在相乘**

$$\frac{\partial C}{\partial w_{ij}^l} = \frac{\partial C}{\partial z_i^l} * \frac{\partial z_i^l}{\partial w_{ij}^l}$$

$$\delta^L = \sigma'(z^L) \bullet \nabla C(y)$$

$$\delta^{L-1} = \sigma'(z^{L-1}) \bullet (W^L)^T \delta^L$$

$$\vdots$$

$$\delta^{l-1} = \sigma'(z^{l-1}) \bullet (W^l)^T \delta^l$$

$$\vdots$$

if L = 1 (輸入層 → 隱藏層1)

$$z_i^1 = \sum_j w_{ij}^1 x_j^r + b_i^1 \qquad \frac{\partial z_i^1}{\partial w_{ij}^1} = x_j^r$$

if L > 1(隱藏層L − 1 → 隱藏層L)

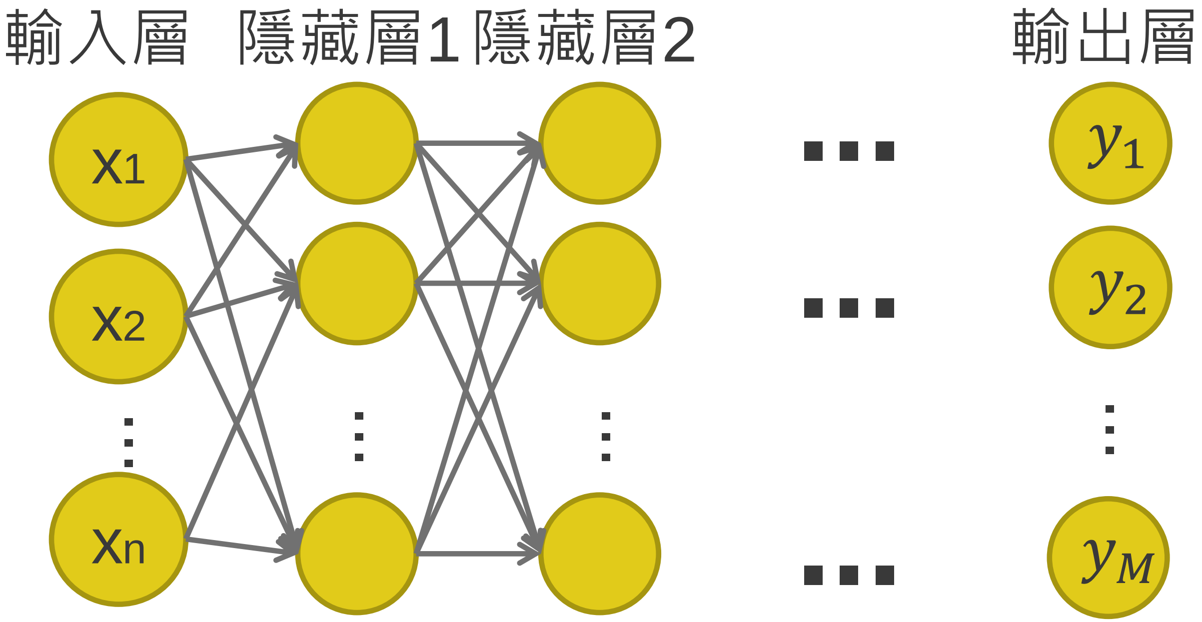
$$z_i^l = \sum_j w_{ij}^l a_j^{l-1} + b_i^l \qquad \frac{\partial z_i^l}{\partial w_{ij}^l} = a_j^{l-1}$$

7-2:驗證神經網路

- 驗證神經網路準確度
- Overfitting問題

驗證神經網路準確度

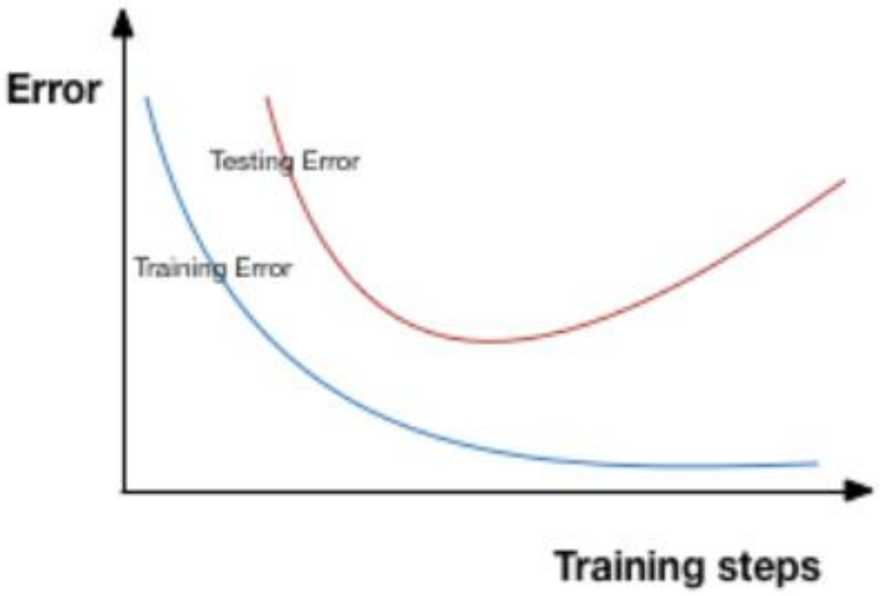
- "*" 代表經由優化器所找到之最佳化參數
- 可以將測試資料拿出來去驗證模型準度
  - 準度 = 猜對多少筆測試資料 / 測試資料總筆數



$$\theta = \{W^{1*}, b^{1*}, W^{2*}, b^{2*}, \dots, W^{L*}, b^{L*}\}$$

Overfitting

- 網路在訓練時，正常情況下，訓練錯誤與測試錯誤的值會一起往下
- 如果訓練到後期發現訓練錯誤往下，但測試錯誤卻開始上升，那就是發生所謂Overfitting

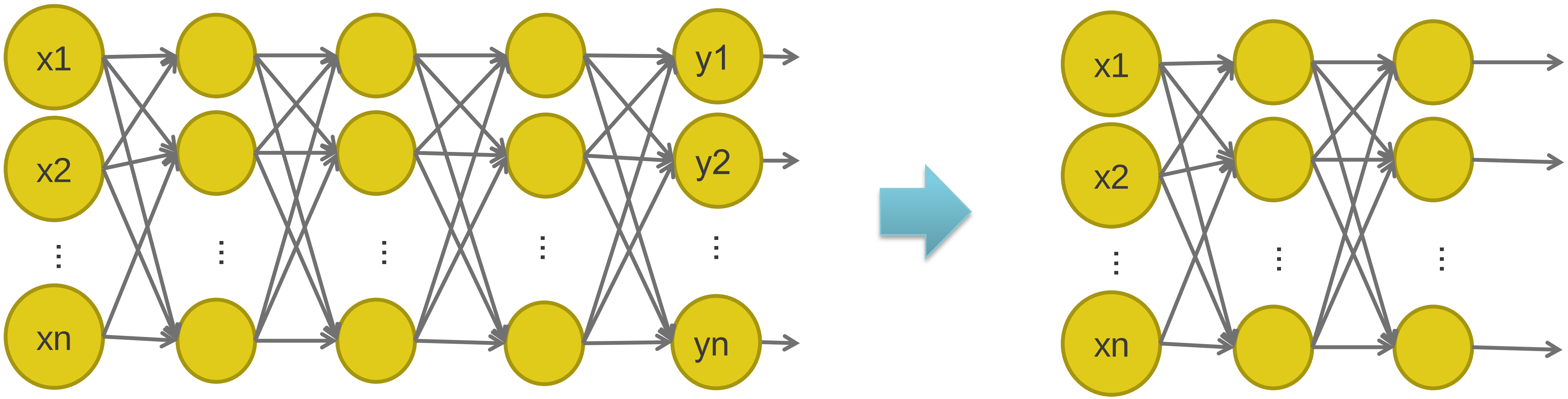


Overfitting

- 減緩Overfitting的方法
  - 減少模型複雜度
  - 使用更多訓練資料
  - 使用正則化(後面章節會教)
  - 做資料增強(後面章節會教)

減少模型複雜度

- 減少模型複雜度指的是讓模型的參數變少
  - 這樣模型就沒有能力產生太過陡峭的曲線，減緩overfitting現象
  - 在神經網路裡，最簡單的方式就是讓層數減少

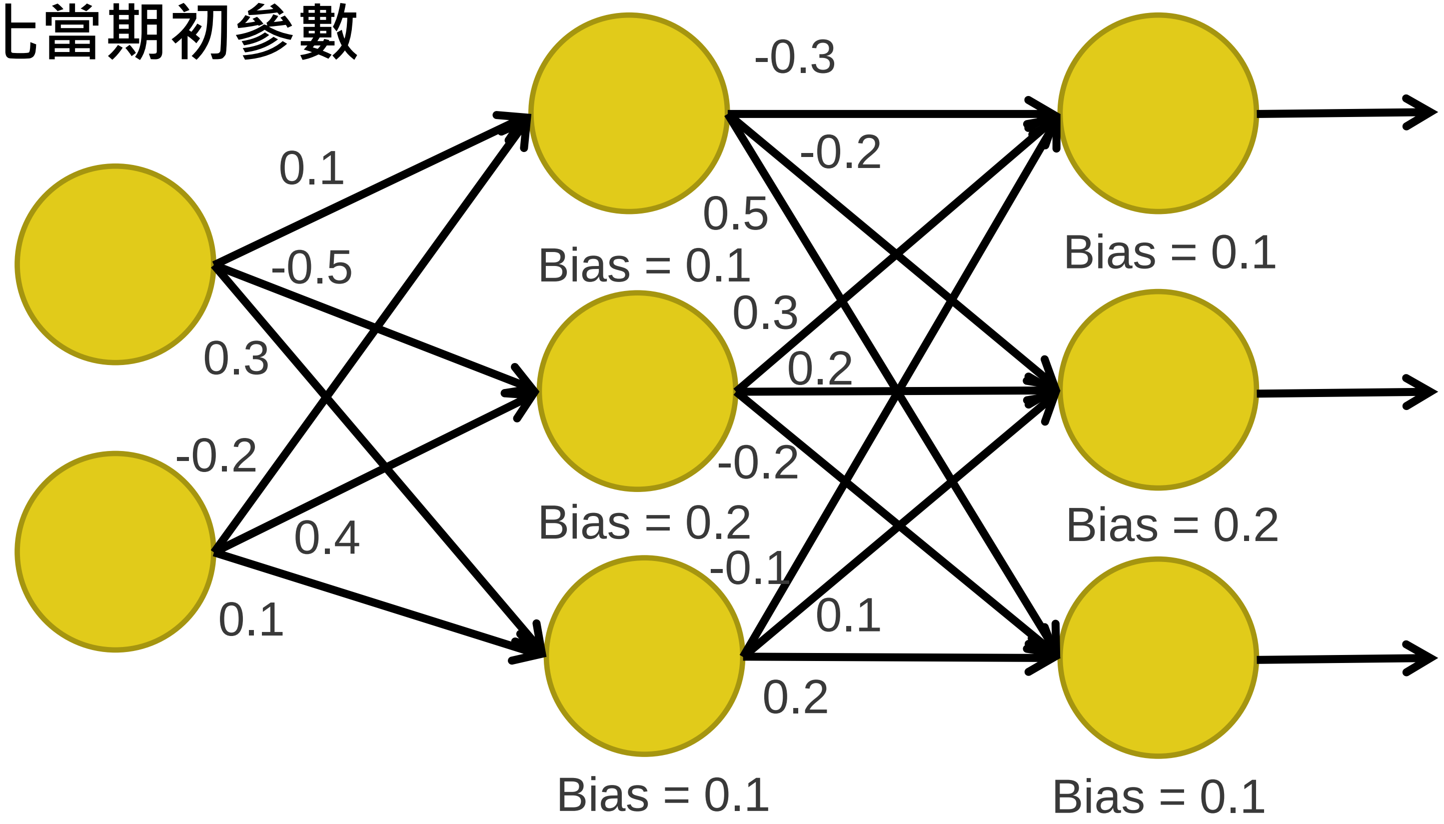


7-3:DNN神經網路數值範例

- **DNN神經網路數值範例**

DNN神經網路數值範例

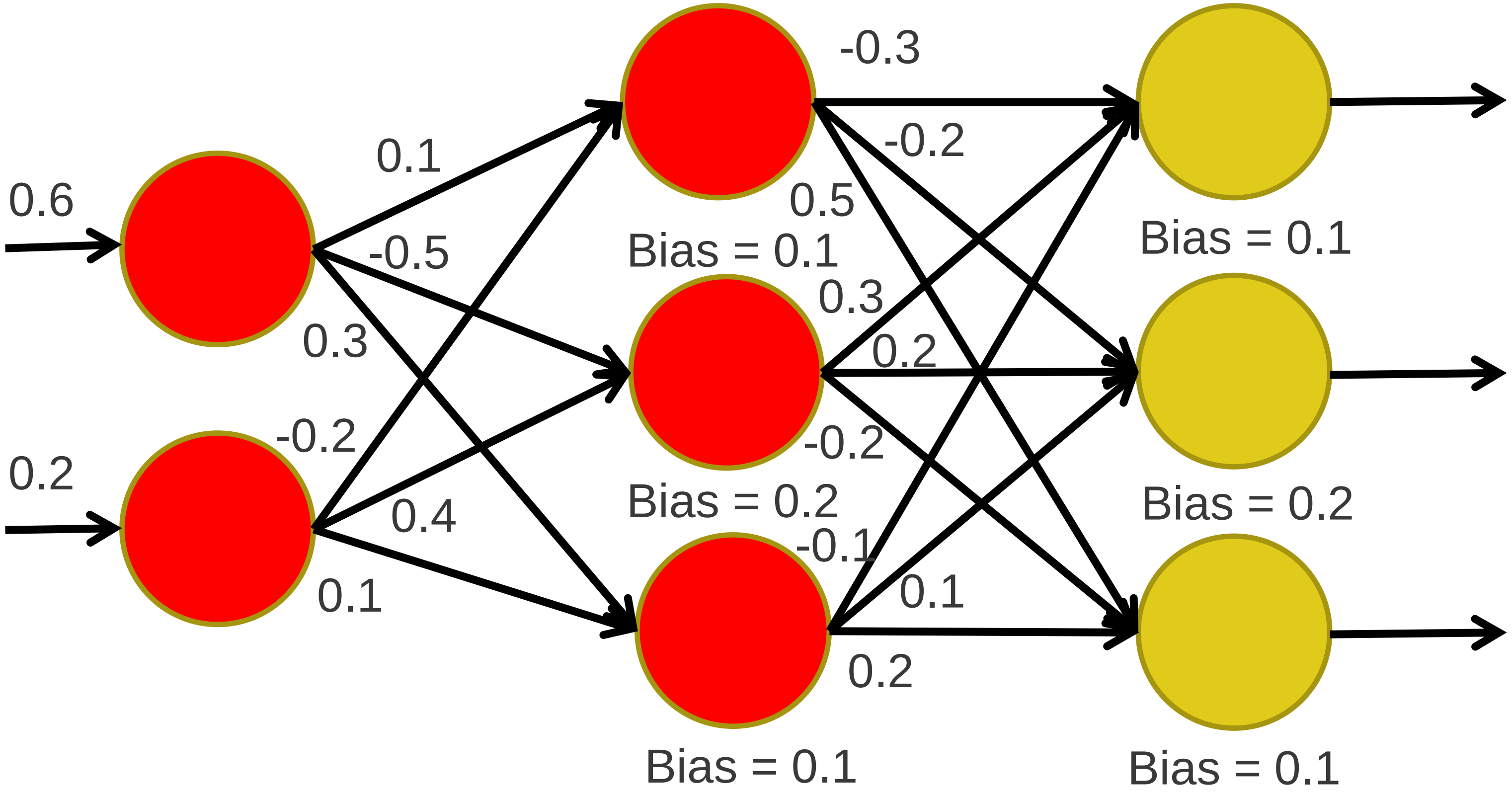
- 假設我們建構一個DNN神經網路如下：
  - 其有輸入為2D向量、輸出為3D向量、網路總參數為21
  - 隨機初始化當期初參數



網路參數為21個

DNN神經網路數值範例

- 輸入一筆為(0.6, 0.2)的資料
- 計算輸入層到隱藏層1



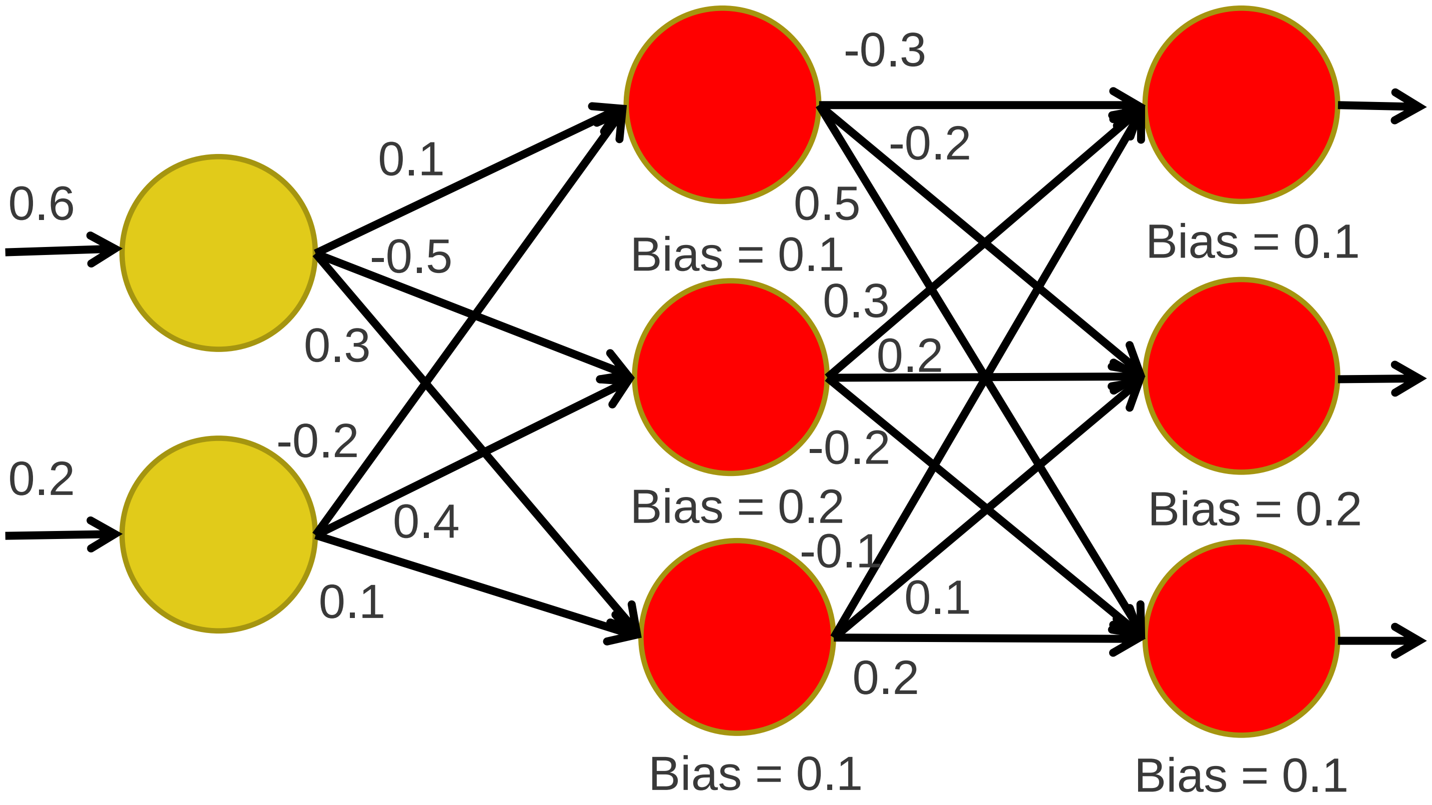
$$relu\left(\begin{bmatrix}0.1 & -0.2\\ -0.5 & 0.4\\ 0.3 & 0.1\end{bmatrix}\begin{bmatrix}0.6\\ 0.2\end{bmatrix}+\begin{bmatrix}0.1\\ 0.2\\ 0.1\end{bmatrix}\right)$$

$$= relu\left(\begin{bmatrix}0.12\\ -0.02\\ 0.3\end{bmatrix}\right)$$

$$= \begin{bmatrix}0.12\\ 0\\ 0.3\end{bmatrix}$$

DNN神經網路數值範例

- **計算隱藏層1到輸出層**



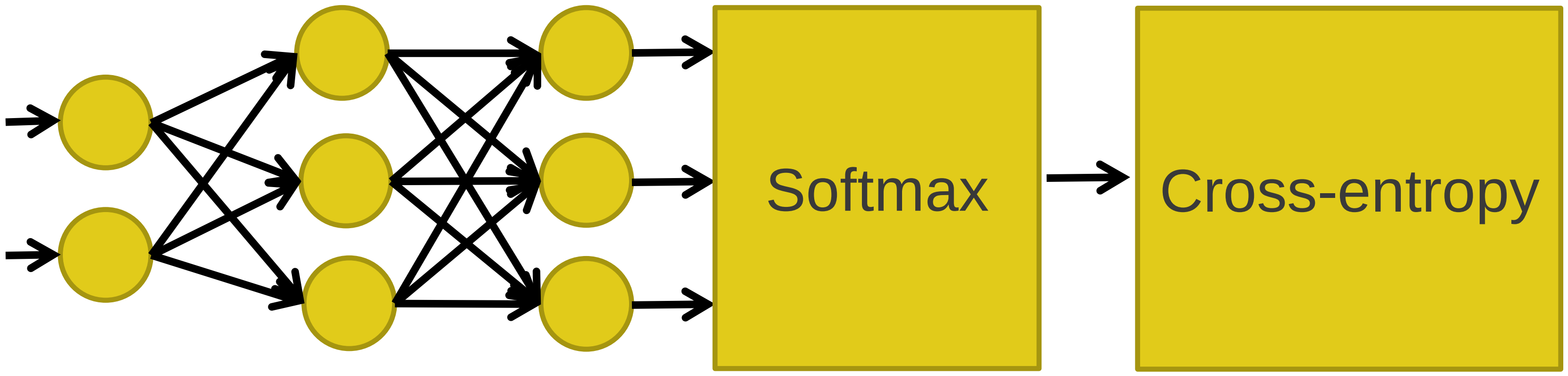
$$relu\left(\begin{bmatrix} -0.3 & 0.3 & -0.1 \\ -0.2 & 0.2 & 0.1 \\ 0.5 & -0.2 & 0.2 \end{bmatrix}\begin{bmatrix} 0.12 \\ 0 \\ 0.3 \end{bmatrix} + \begin{bmatrix} 0.1 \\ 0.2 \\ 0.1 \end{bmatrix}\right)$$

$$= relu\left(\begin{bmatrix} -0.066 \\ 0.006 \\ 0.12 \end{bmatrix}\right)$$

$$= \begin{bmatrix} 0 \\ 0.006 \\ 0.12 \end{bmatrix}$$

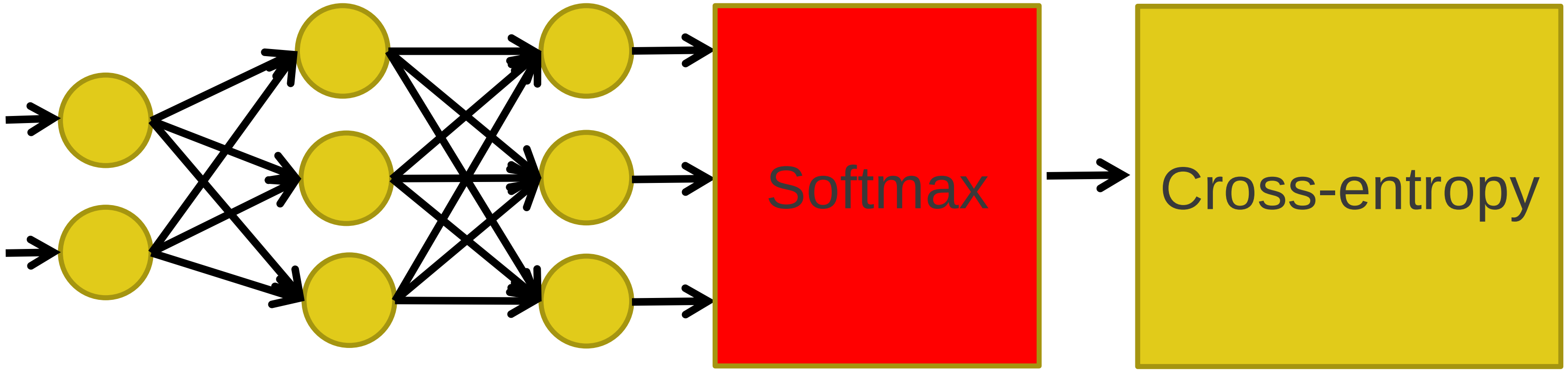
DNN神經網路數值範例

- 假設使用Cross-Entropy來當損失函數
  - 記得Cross-Entropy之前會需要加Softmax層



DNN神經網路數值範例

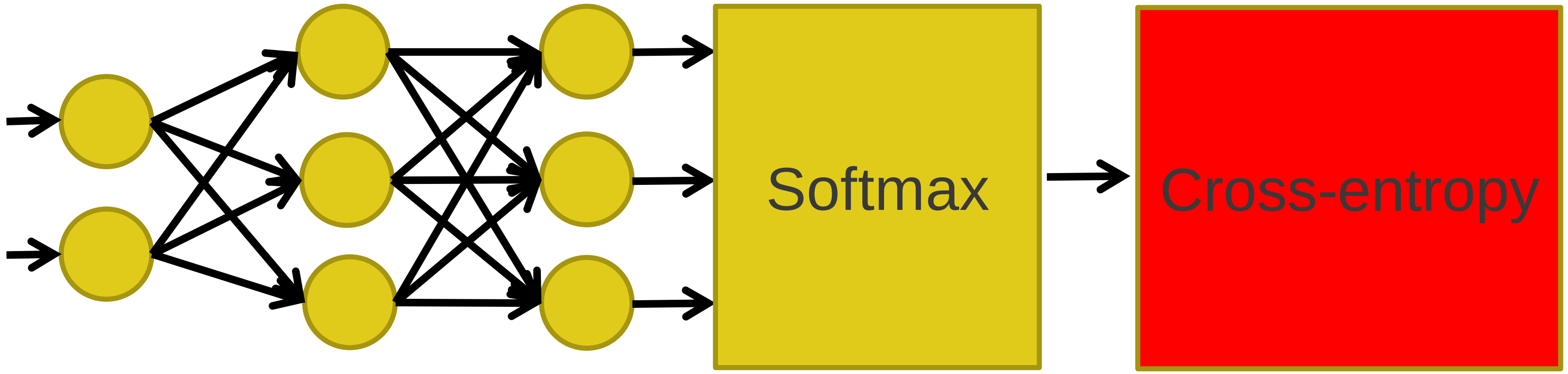
- **Softmax層的目的就是把向量壓成機率分布**



$$\text{Softmax}\left(\begin{bmatrix} 0 \\ 0.006 \\ 0.12 \end{bmatrix}\right) = \begin{bmatrix} 0.319 \\ 0.321 \\ 0.36 \end{bmatrix}$$

DNN神經網路數值範例

- 上述步驟我們得到了一個預測向量，而假設期望向量為[0, 1, 0]
  - 由於期望向量第二個地方為1，可以推估此筆資料應該屬於第二個類別



期望向量

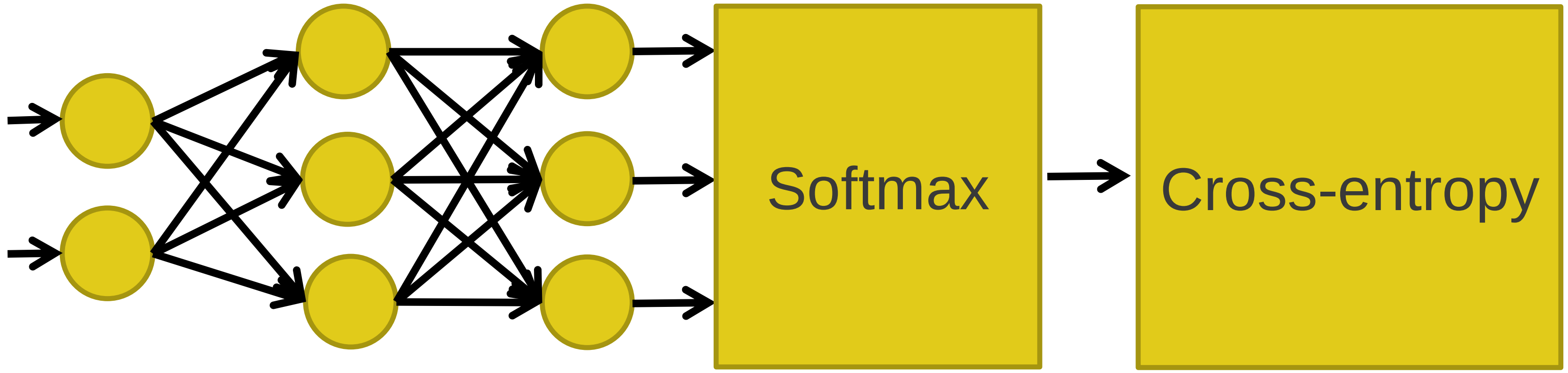
$$\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

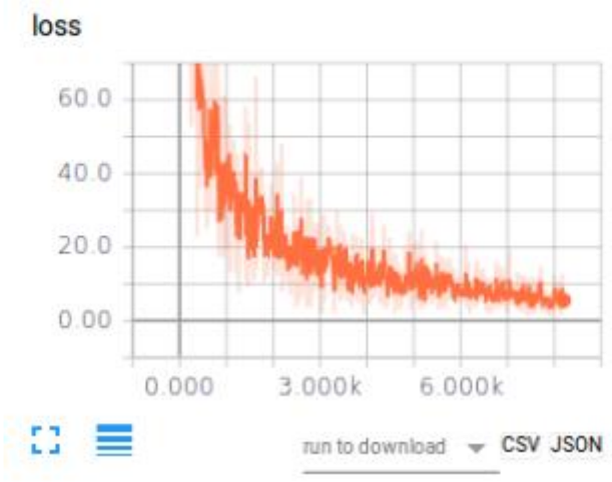
- 0 * ln(0.319) – 1 * ln(0.321) – 0 * ln(0.36)
= 1.1363

$$-\sum_{i=0}^{class\,\#} \hat{y}_i \ln(y_i)$$

DNN神經網路數值範例

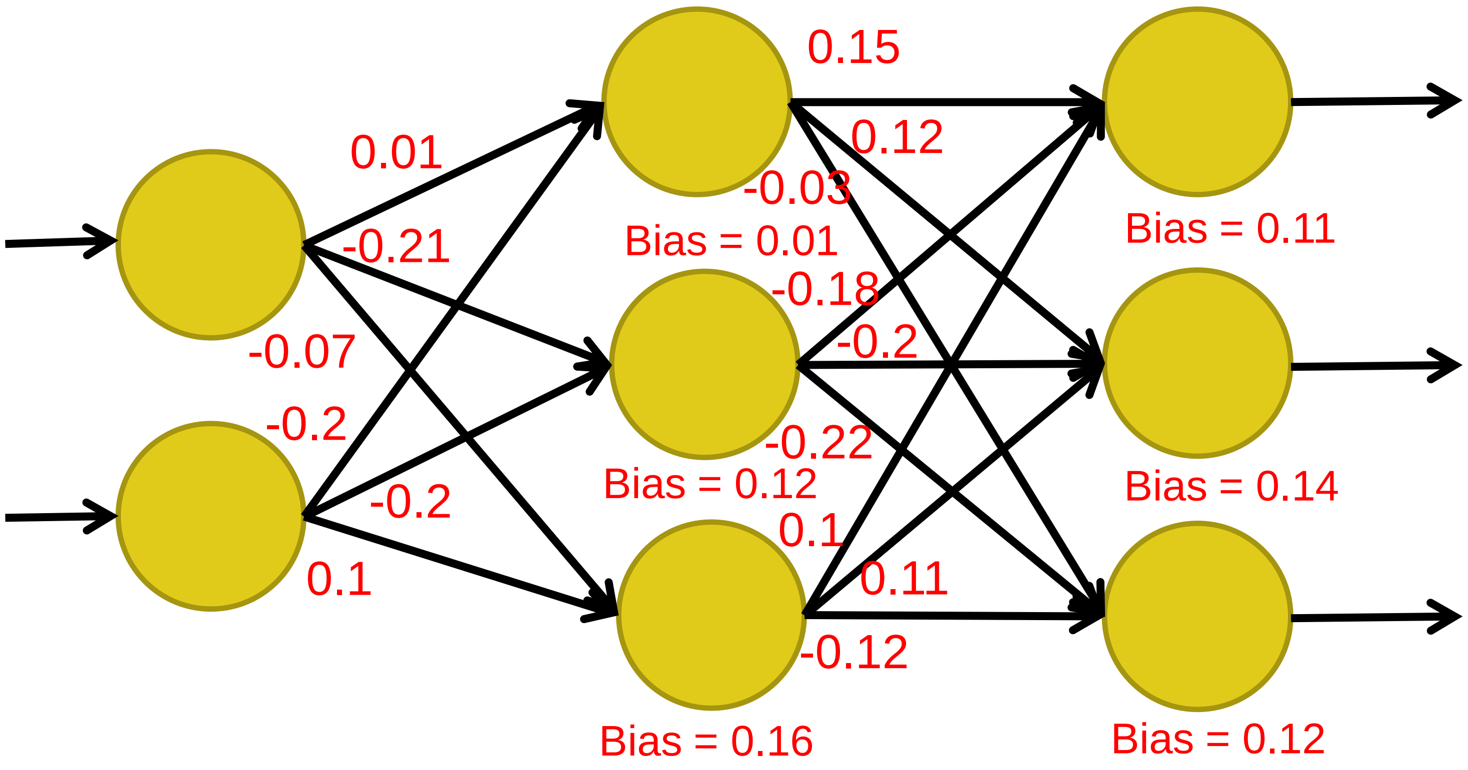
- 使用優化器去幫我們調整網路內參數的數值
  - 調整參數過程中，損失函數越來越小





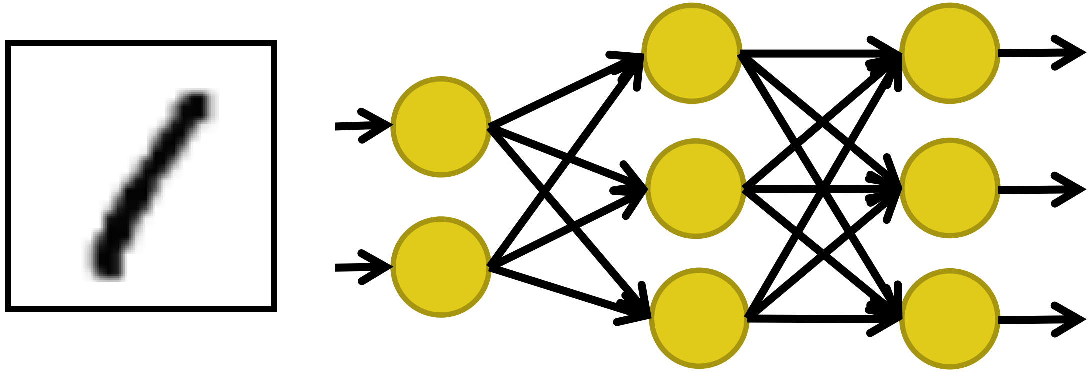
DNN神經網路數值範例

- 當經過漫長的訓練，我們最終會得到一組非常好的參數
  - 這組參數可以幫我們把此分類的問題做得很好



DNN神經網路數值範例

- 當模型訓練結束後，我們可以把測試資料拿來做驗證
  - 計算模型是否表現得很好
  - 預測向量最大值的位置，即是網路所猜之類別



$$\begin{bmatrix} 0.1 \\ 0.5 \\ 0.4 \end{bmatrix}$$

模型猜測為類別2

Demo 7-3

- 開啟Demo_7-3.ipynb
- DNN神經網路訓練及驗證
- 調整神經網路參數

線上Corelab

- 題目1：DNN的準確率(基礎)
  - 將以下DNN網路改成5層，並將MNIST資料集倒入網路訓練，並輸出訓練準確率
  - 請以EPOCH=20、batch size=300為單位訓練、優化器請選用Adam，learning_rate=0.02
- 題目2：DNN的準確率(中等)
  - 將以下DNN網路改成5層，並將MNIST資料集倒入網路訓練，並輸出訓練準確率
  - 請以EPOCH=20、batch size=640為單位訓練、優化器請選用Adam，learning_rate=0.02
- 題目3：DNN的準確率(進階)
  - 將以下DNN網路改成5層，並將MNIST資料集倒入網路訓練，並輸出訓練準確率
  - EPOCH與batch size請適當給予
  - learning_rate請設定0.01與0.001並觀察這兩者之間準確率的關係

本章重點精華回顧

- 優化原理與Backpropagation
- 驗證神經網路
- DNN神經網路計算流程

Lab:Python 簡介

- **Lab01: DNN神經網路訓練及驗證**
- **Lab02: 調整批次大小**
- **Lab03: 調整學習率與初始化方法**

Estimated time:

20 minutes


THANK YOU