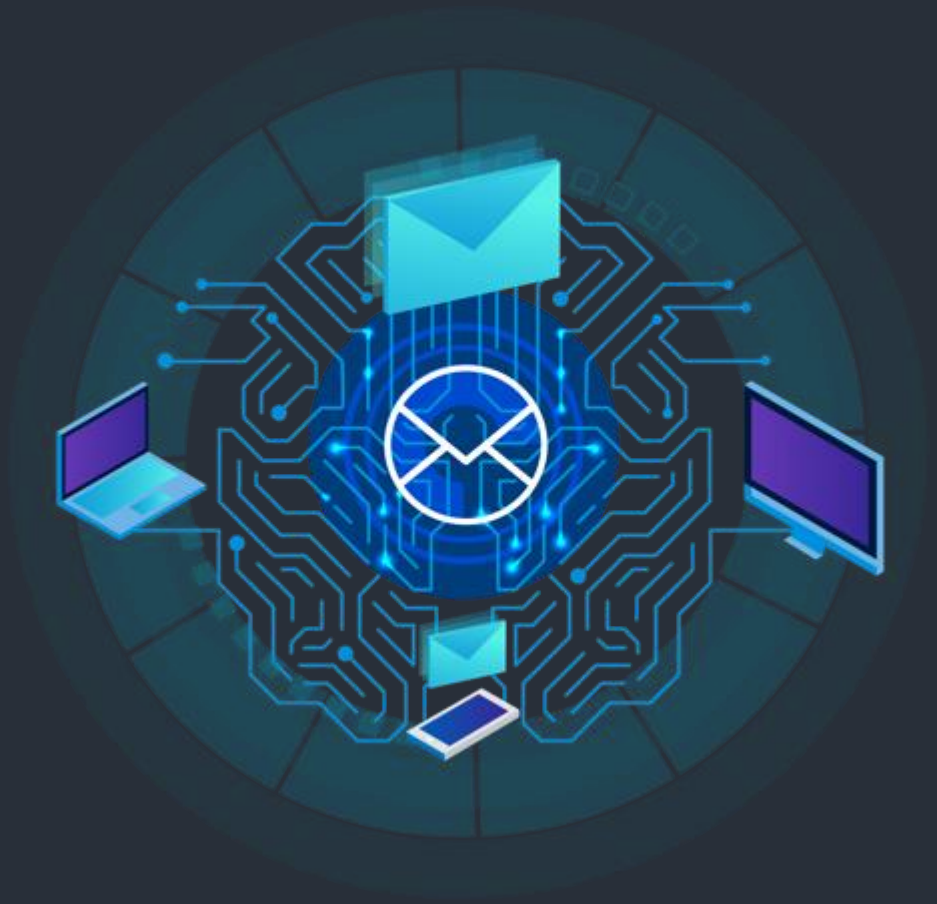
多層感知器 Multilayer perceptron

國立東華大學電機工程學系楊哲旻

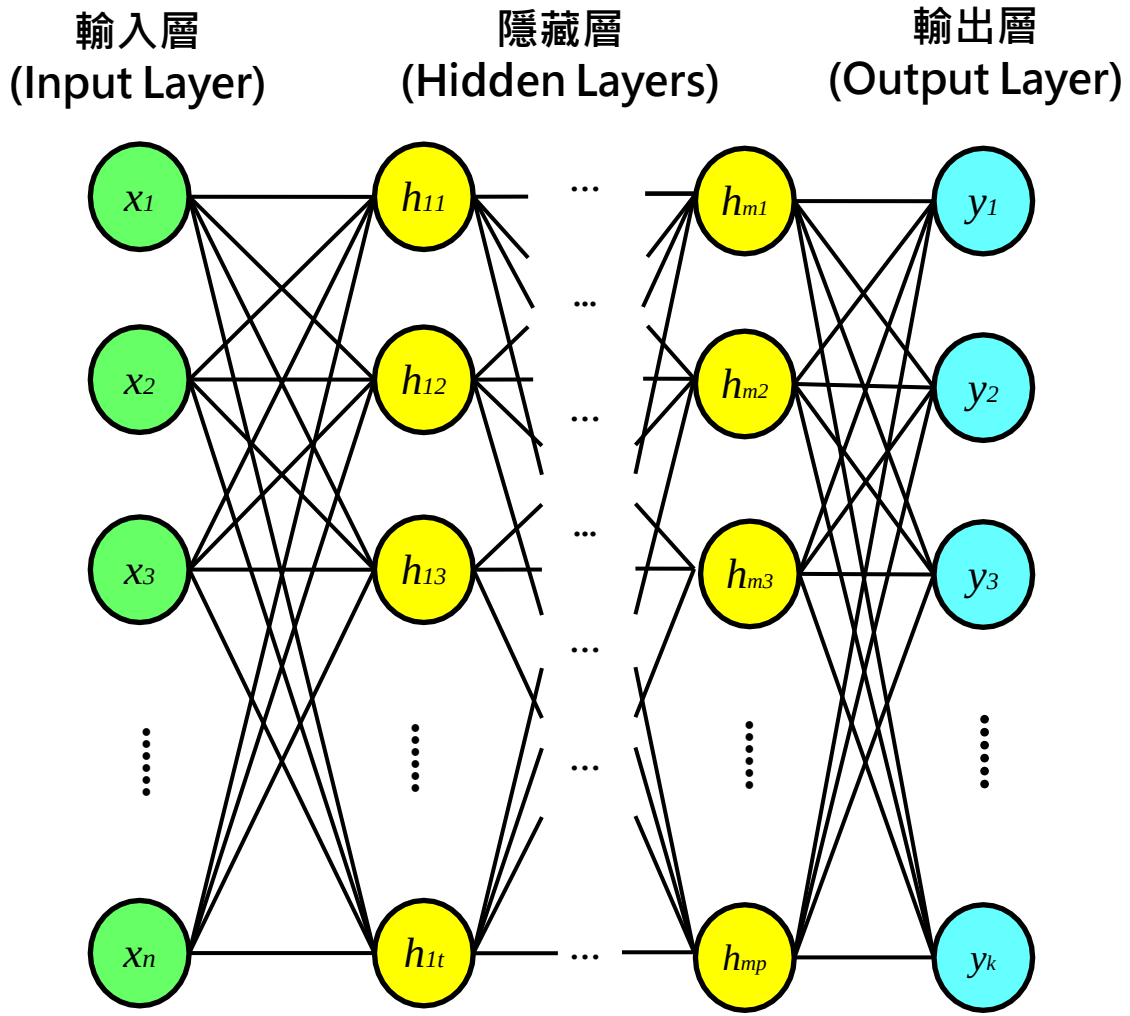
Outline



1. 模型架構
2. 前向傳播
3. 反向傳播
4. 線性可分數據
5. 線性不可分數據
6. 激勵函數
7. 丟棄法
8. 超參數
8. 多層感知器實作

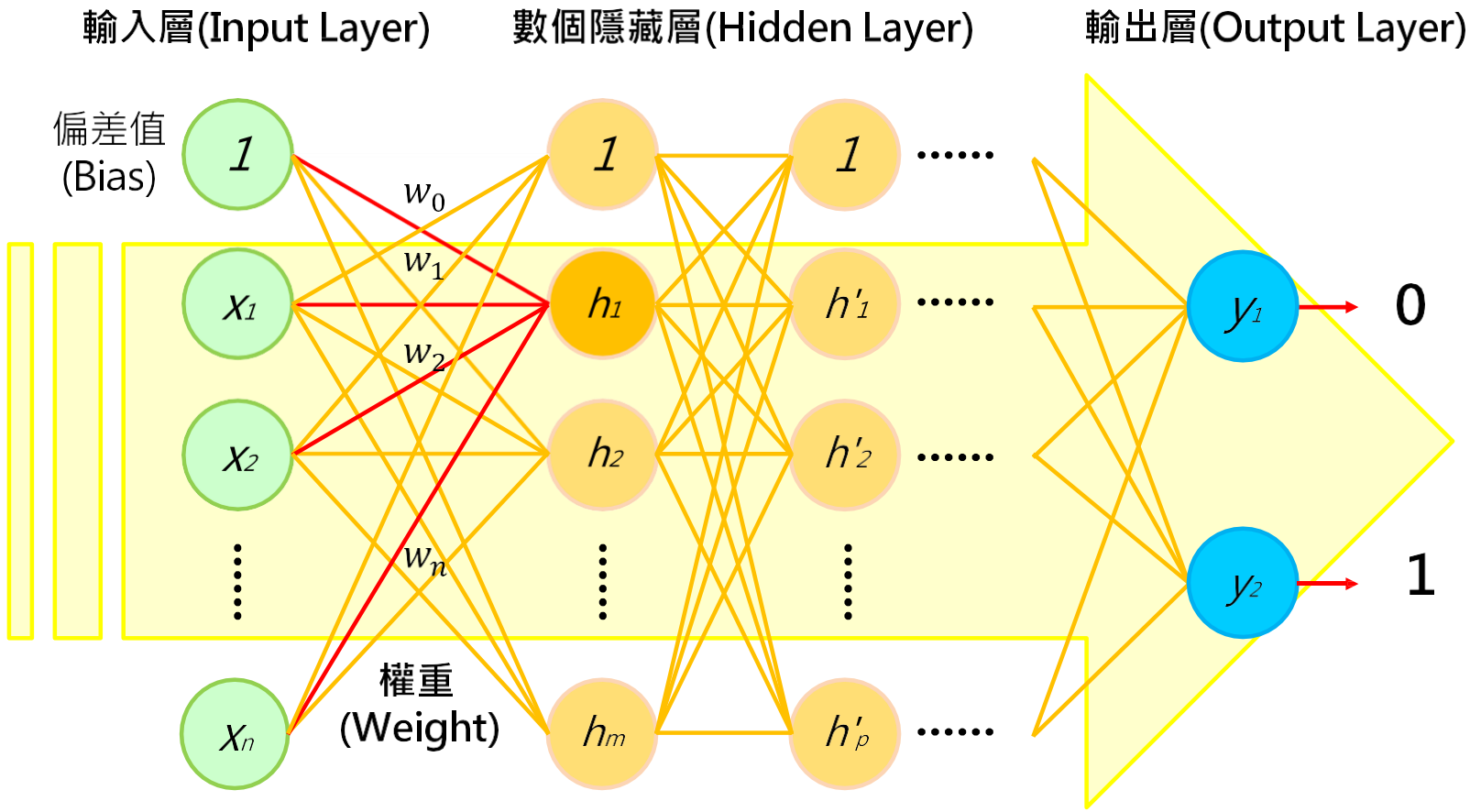
深度學習 – 多層感知器 01. 模型架構

多層感知器為深度學習的分類模型，架構為一輸入層、數個隱藏層與一輸出層所組成。訓練過程是從訓練集中以梯度下降法來確定權重與偏差。



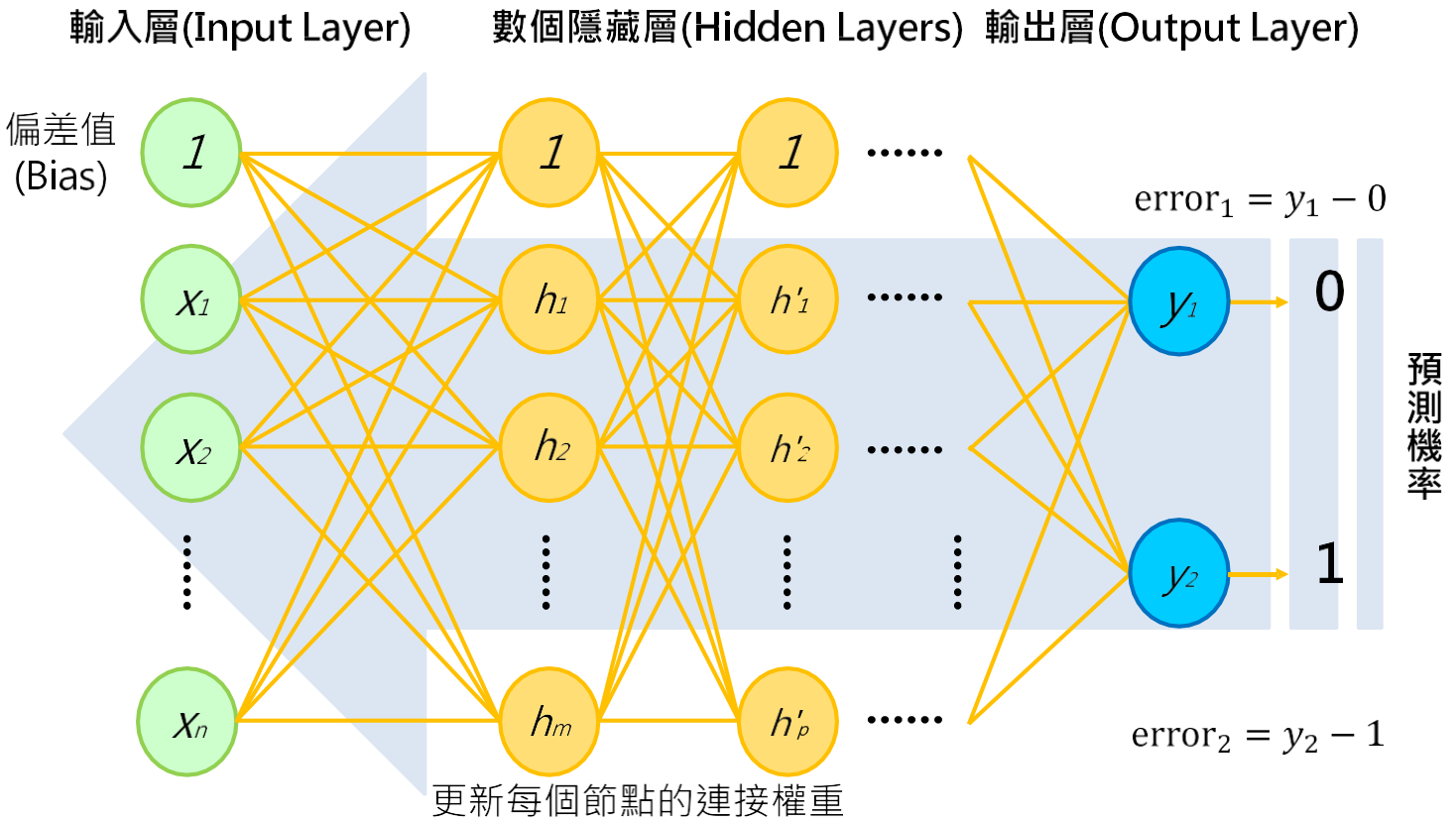
深度學習 – 多層感知器 02. 前向傳播

前向傳播 (Forward-Propagation, FP)



深度學習 – 多層感知器 03. 反向傳播

反向傳播 (Back-Propagation, BP)

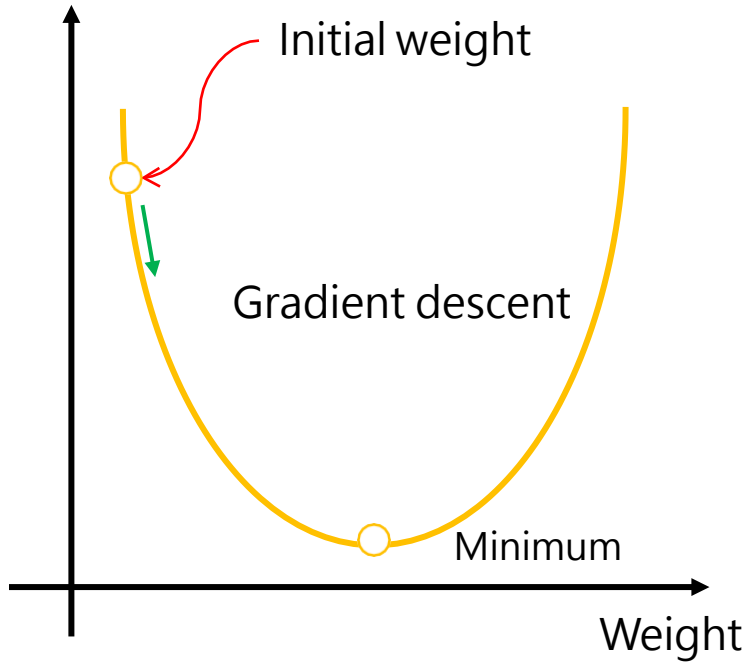


總誤差(Square error)：$\frac{1}{2}error_1{}^2 + \frac{1}{2}error_2{}^2$

1.鏈式求導法則(Chain rule)
2.梯度下降法(Gradient descent)

線性可分的數據

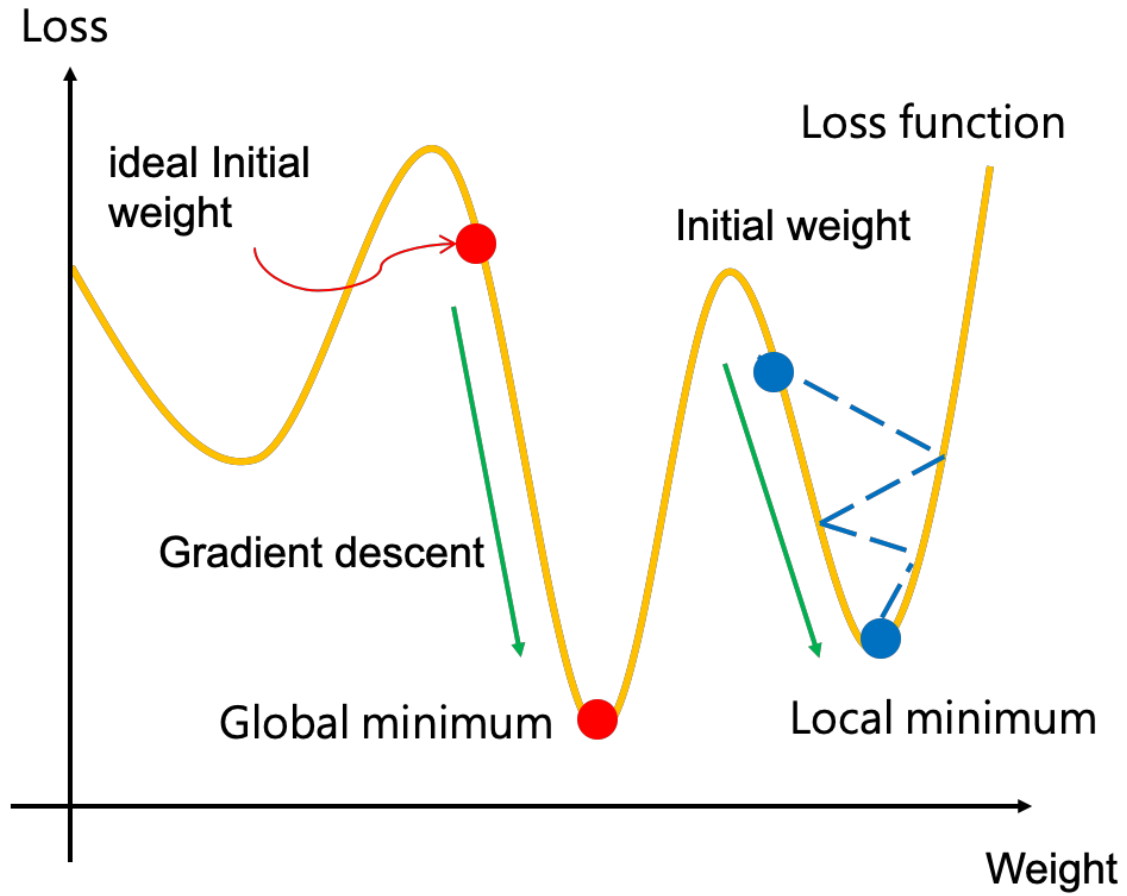
線性可分的數據其損失函數為凸函數，隨機的初始權重必然下降到全域最小值(Global Minimum)



05. 線性不可分的數據

線性不可分的數據

線性不可分的數據其損失函數為非凸函數，如果不是理想的初始權重時，在梯度下降法運算中會陷入超淺的局部為小值(Local minimum)，且會隨著隱藏層的擴大更加嚴重，這問題為「梯度消失(Vanishing)」



線性不可分的數據

神經元引入非線性的激勵函數，從而使神經網絡能夠解決線性不可分的問題

1.線性函數(Linear) (不能解決線性不可分的數據)

$$f(z) = z$$

2.乙狀、邏輯函數(Sigmoid)

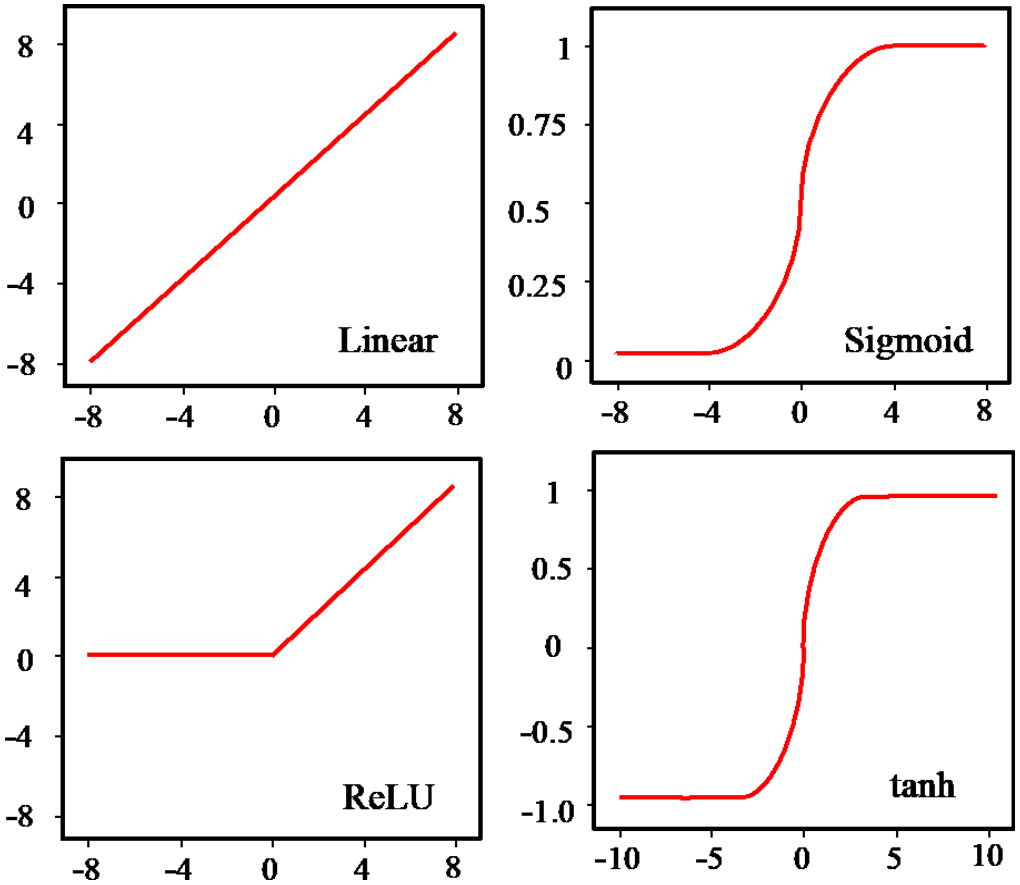
$$f(z) = \frac{1}{1 + e^{-z}}$$

3.線性整流函數(Rectified Linear Unit, ReLU)

$$f(z) = \begin{cases} z, & z \geq 0 \\ 0, & z < 0 \end{cases}$$

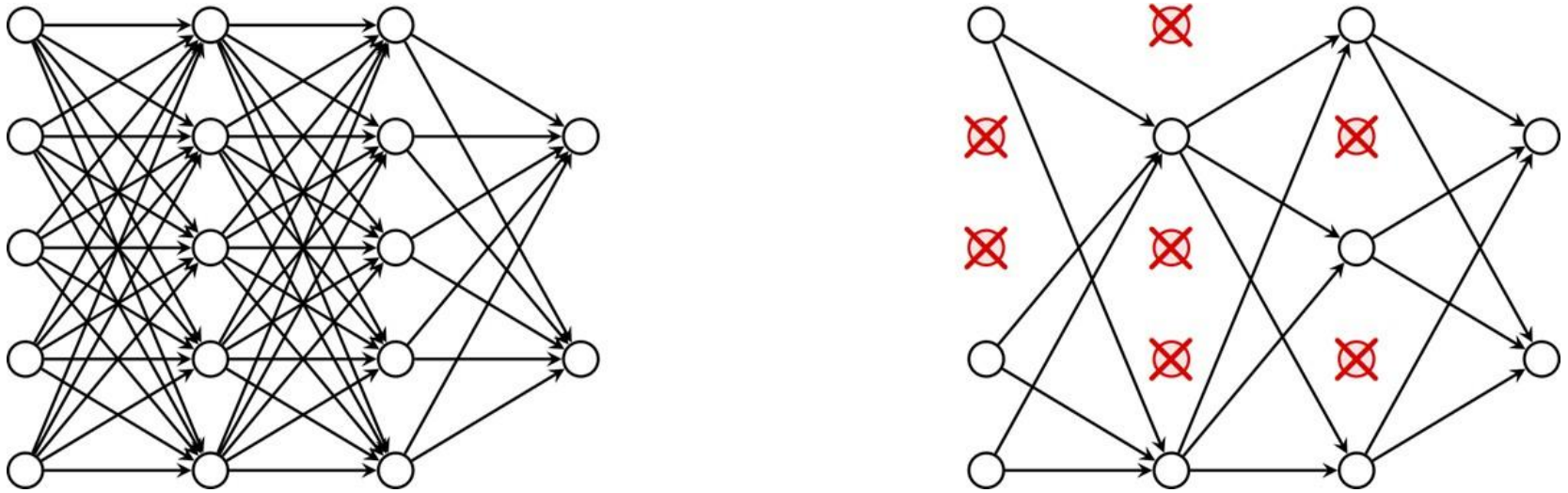
4.雙曲正切函數 (tanh)

$$f(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$



丟棄法(Dropout)

在訓練時每一次的迭代皆以一定的機率p丟棄隱藏層神經元，而被丟棄的神經元不會傳遞訊息，丟棄法可以降低過度擬和



深度學習 – 多層感知器 08. 超參數

多層感知器的超參數

- 學習速率 (Learning Rate)
- 批量 (Batch)
- 迭代次數 (Number of iterations)
- 正則化懲罰係數 (Regularization)
- 隱藏層層數與神經元數量
- 丟棄法 (Dropout)

多層感知器－實作

多層感知器 – 實作

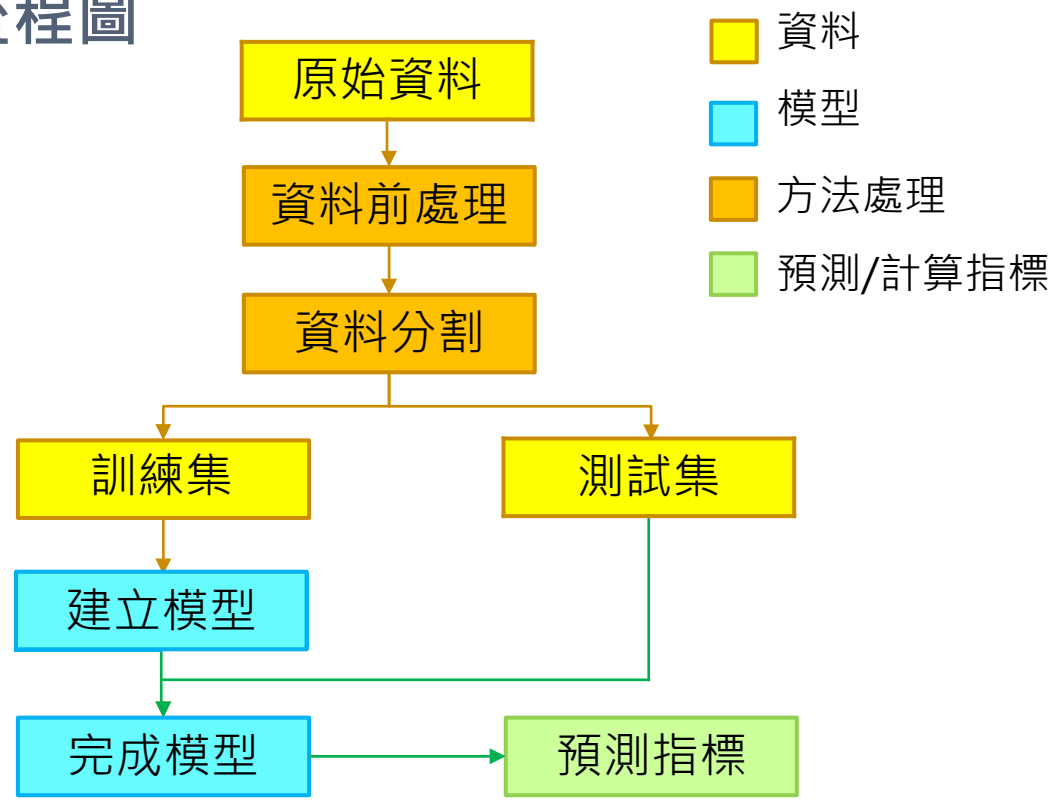
Dogs vs. Cats Database

Kaggle Dataset

https://www.kaggle.com/c/dogs-vs-cats/data

- 數據集包含25000張貓狗影像，訓練集與測試集各12500張，標籤以.csv檔儲存，其中1為狗，0為貓

流程圖



多層感知器 – 實作

Dogs vs. Cats Database

問題

1. 如果影像的資料集太小，該如何解決呢？
2. 隱藏層與其神經元數量等的超參數該如何選擇？
3. 三維的彩色影像該如何做訓練呢？
4. 若一張影像為非貓非狗的影像，該分類器是否還是擇一做輸出？

   另外，為有貓狗的影像，該分類器如何辨識？

作業一

1. 請使用資料增量方式以提高模型的預測能力
2. 嘗試以Scikit-learn、Keras or Tensorflow來實踐貓狗分類器