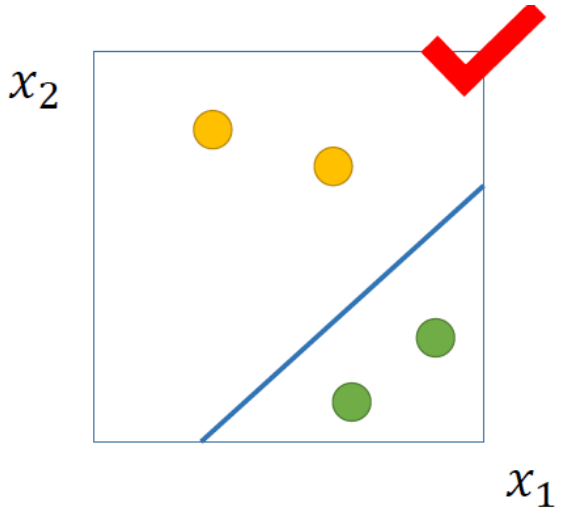
第三周

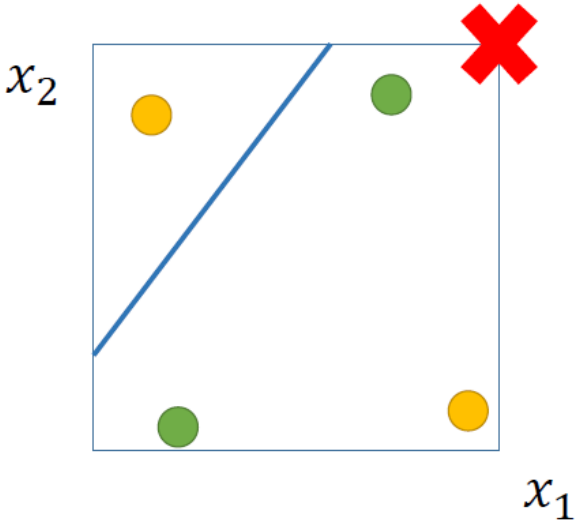
第三周的課程正式進入到了神經網路(Neural Network)的範圍，並帶我們了解一些神經網路的形式、基礎概念等。

1. 人工神經網路
   ① 起源：人工神經網路為模擬生物神經的一種計算模型。
   ② 神經元
   線性模型為最簡單的線性神經元，而線性模型為將多個資料彙整成一個函數 $f(x)$ ➡ 概念和神經網路相同(神經網路為多個神經元構成)。
   一個神經元的用途：

可以進行線性可分之二元分類任務：



但若非單個直線可分之分類任務，則不行：



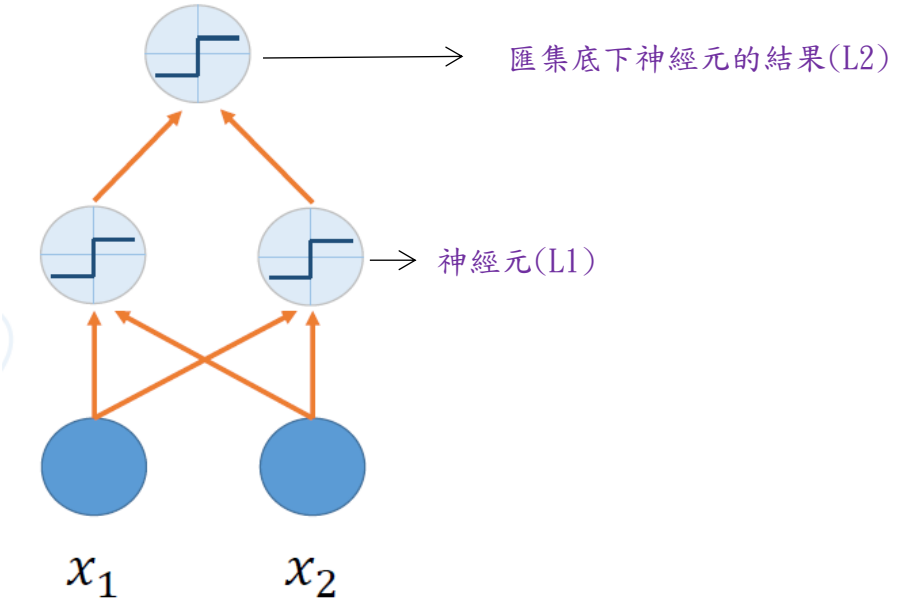
③多層神經網路

為了解決上述單個神經元的問題，我們將多個神經元組合成多層神經網路。

當定好神經網路的結構之後，便會形成一個假設集合 ➡ 結構、神經元的組成皆會影響此假設集合。

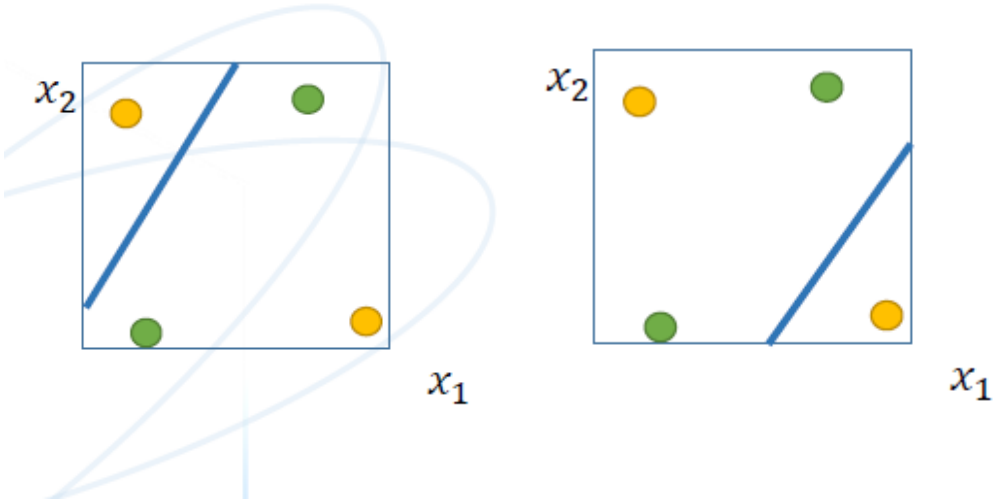
當有許多神經元，便可切分出近似於任何圖形的函式。

1. 神經網路的模式

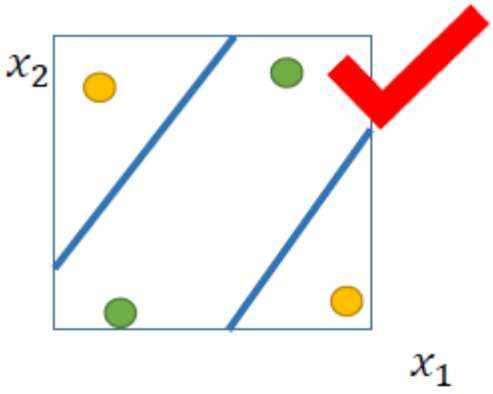


以上圖的神經網路為例，我們再次切分上一個單個神經元無法解決的問題：

在 L1 層有兩個神經元，他們分別接收點的 X_1 和 X_2 資料，接著，他們分別給出它們的分割結果，如下兩圖。



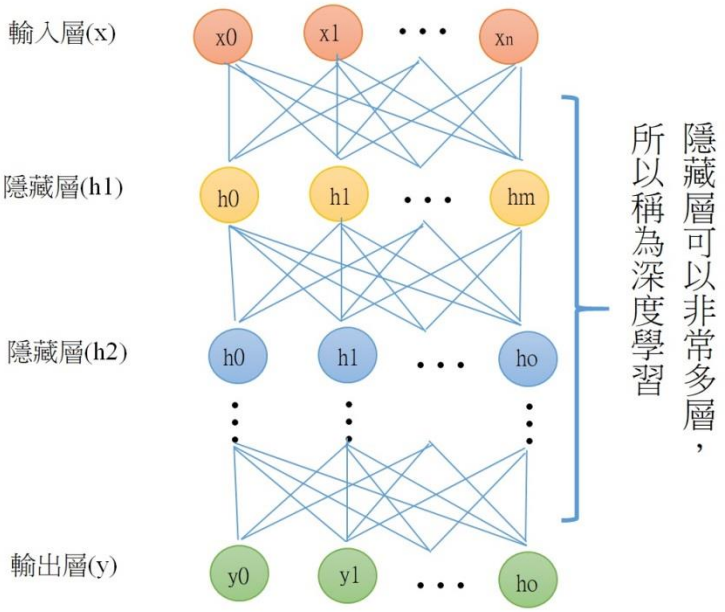
但是以上兩圖都分別不能解決問題，因此，當圖片傳遞到下一層(L2層)時，神經元將兩圖彙整成一個分割結果：



這樣就解決了這個分割問題。由上述我們可以發現，神經網路是潛力無窮的，因為當我們不停地增加神經網路規模(先不論硬體的限制)，模型就可以生成更多更為複雜的概念。

2. 神經網路的計算過程

各層將前一層的輸出，變成輸入的資料。



上圖中，神經元和神經元之間的每一個箭頭皆代表傳遞的過程，而傳遞過程中包含了一些運算，以輸入層(x)和隱藏層(h1)之間的過程為例(假設此模型為線性模型)：

$$Z_{h1}=W_{h1}X+b_{h1}$$

$$h_1=g(Z_{h1})$$，g 為激活函數

而每一層的結果皆可匯集成一可變矩陣。

△激活函數(Activation function)

一個單純的線性模型，往往不能靈活的表示各種多樣的特徵，因此常常會有 Underfitting 的情形發生。假如我們加上一個激活函數 $g(x)$，將此線性模型變為一個非線性的神經元，則可增加複雜度，更可模擬各種情形。

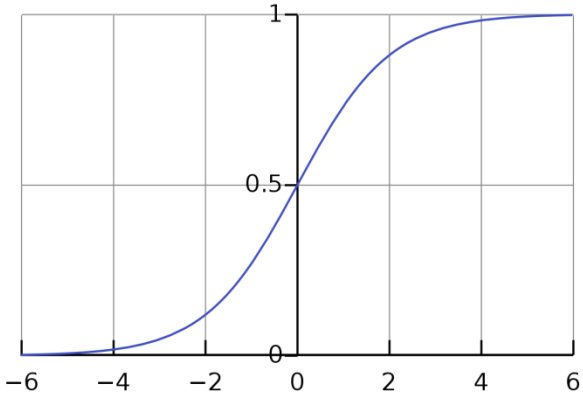
數學式如下：

$$f(x) = g(W_1X_1+W_2X_2+\cdots\cdots+W_nX_n-b)$$

激活函數的種類：

現今有許多種類的激活函數來幫助我們對應各種各樣的問題，這裡會舉出幾種較常使用的。

1. Sigmoid



數學式：$Sigmoid(x) = \frac{1}{1+e^{-x}}$

特徵：會輸出 0-1 之間的數，類似於機率分布。

用途：常應用在多標籤分類問題

優點：

① 輸出範圍限制在 0-1 之間，優化穩定，常用於輸出層。

② 為可微函數

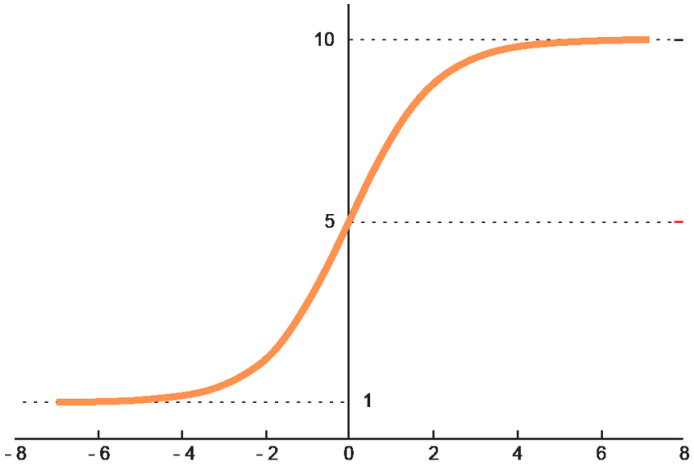
缺點：

① 由於函數極值附近微分相當小，很可能造成梯度消失問題。

② 函數輸出皆大於零，意味著並非零均值，上一層的輸入會受到非零均值的影響。

③ 為指數型式，計算複雜度高

2. Softmax



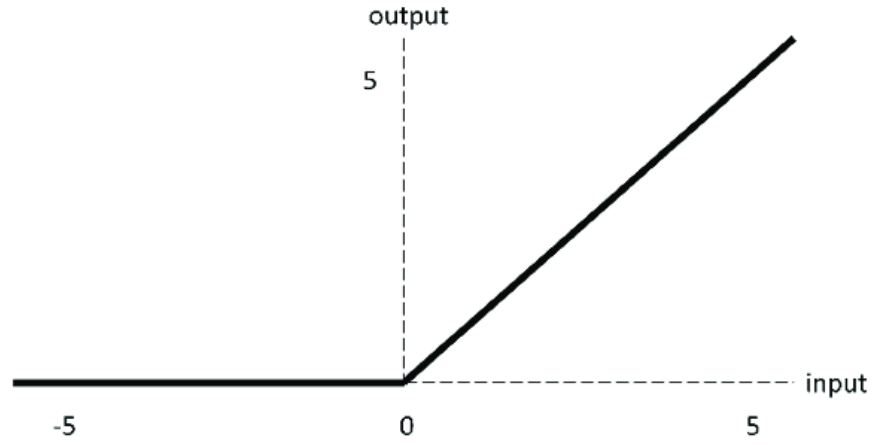
數學式：$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$，其中 z 為一有 K 個元素的陣列。

特徵：也同樣為一機率分布，但和 Sigmoid 最大的不同，便是 Softmax 的輸出結果中所有元素的總合為 1，引此可以看出輸入陣列 z 之中 K 個元素之間的概率分布。而 Sigmoid 的結果則是機率各項各自獨立，總和不一定為 1。

用途：常用在多類別分類問題(如數字辨識)

缺點：在零點時不可微，意味著輸入為負時的梯度為零，可能造成死亡的神經元(反向傳播時權重不會被更新)。

3. Relu



數學式：$\sigma(x) = \begin{cases} max(0,x), x \geq 0 \\ 0, x < 0 \end{cases}$

特徵：輸入大於等於零時，$f(x)=x$；輸入小於零時，輸出 0。

用途：常作為隱藏層的激活函數

優點：

① 當輸入大於零時，因為輸入值不會改變，不會造成梯度消失問題。

② 計算簡單。

缺點：

① 當輸入小於零，梯度為零。因此在反向傳播的過程中，若輸入小於零，可能導致死亡神經元出現。

②並非零均值函數。

2. 計算流程圖

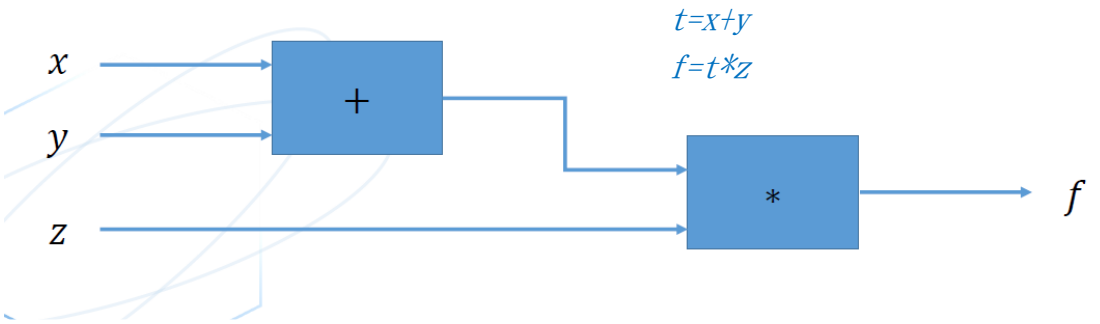
計算流程圖即是將一個計算流程用圖的方式表達，是現今 Tensorflow 和 Pytorch 等深度學習模組最底層的概念。

①為何使用計算流程圖?

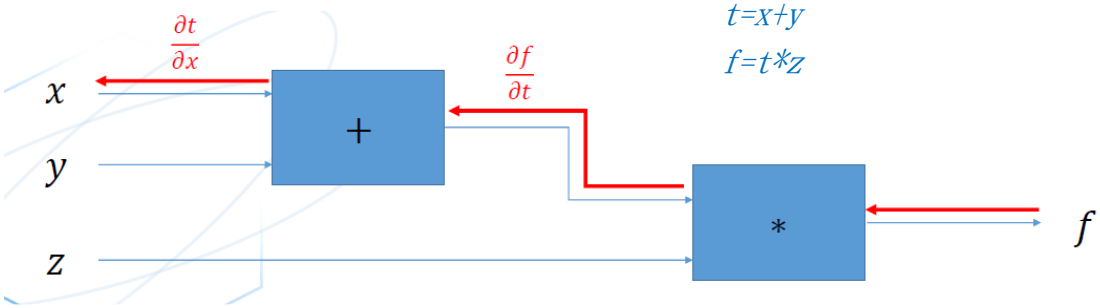
計算流程圖對我們人類來說或許是非必要的，畢竟我們將算式列出即可。但是當今天整體的計算變得複雜或參數變多時，計算流程圖往往可以相當清楚的表示計算過程和每一個計算之間的關係。無論是減輕人類定義計算模型的難度或是讓機器更好理解計算，計算流程圖有它的重要性在。另外，計算流程圖的反向傳遞可以計算微分值，對於更新權重來說也是相當方便。

②計算流程圖的形式

先舉一個最基礎的運算：$f=(x+y)*z$ 為例：



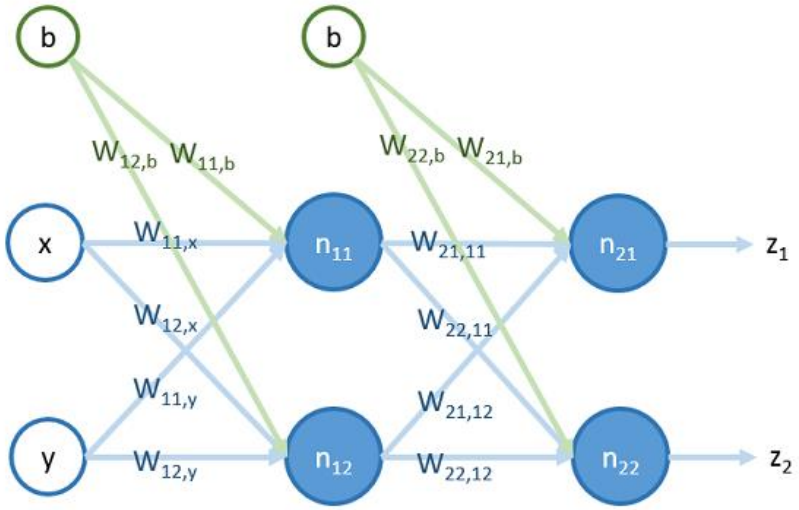
前面有提到，計算圖的反向傳播可以計算梯度：



$$\frac{\partial f}{\partial x}=\frac{\partial f}{\partial t}\frac{\partial t}{\partial x}$$

$$\frac{\partial f}{\partial y}=\frac{\partial f}{\partial t}\frac{\partial t}{\partial y}$$

③神經網路的計算流程圖



詳細的推導和計算過程，會放在第四周的部分。

參考資料

1. 深度學習的種類

https://ithelp.ithome.com.tw/articles/10217967