

Day 72

激活函數

激活函數的介紹與應用




本日知識點目標



目標
知識點

了解激活函數

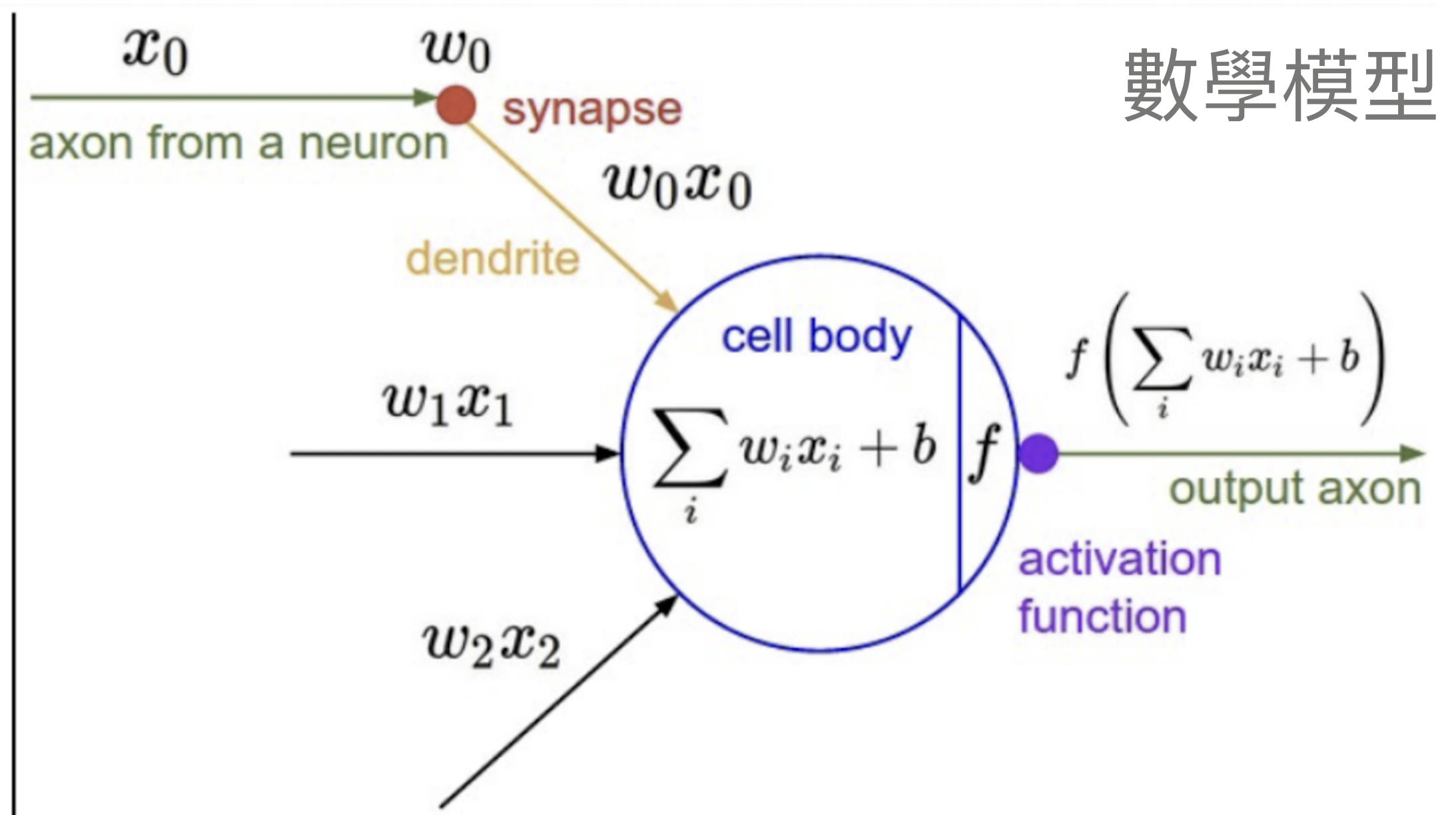
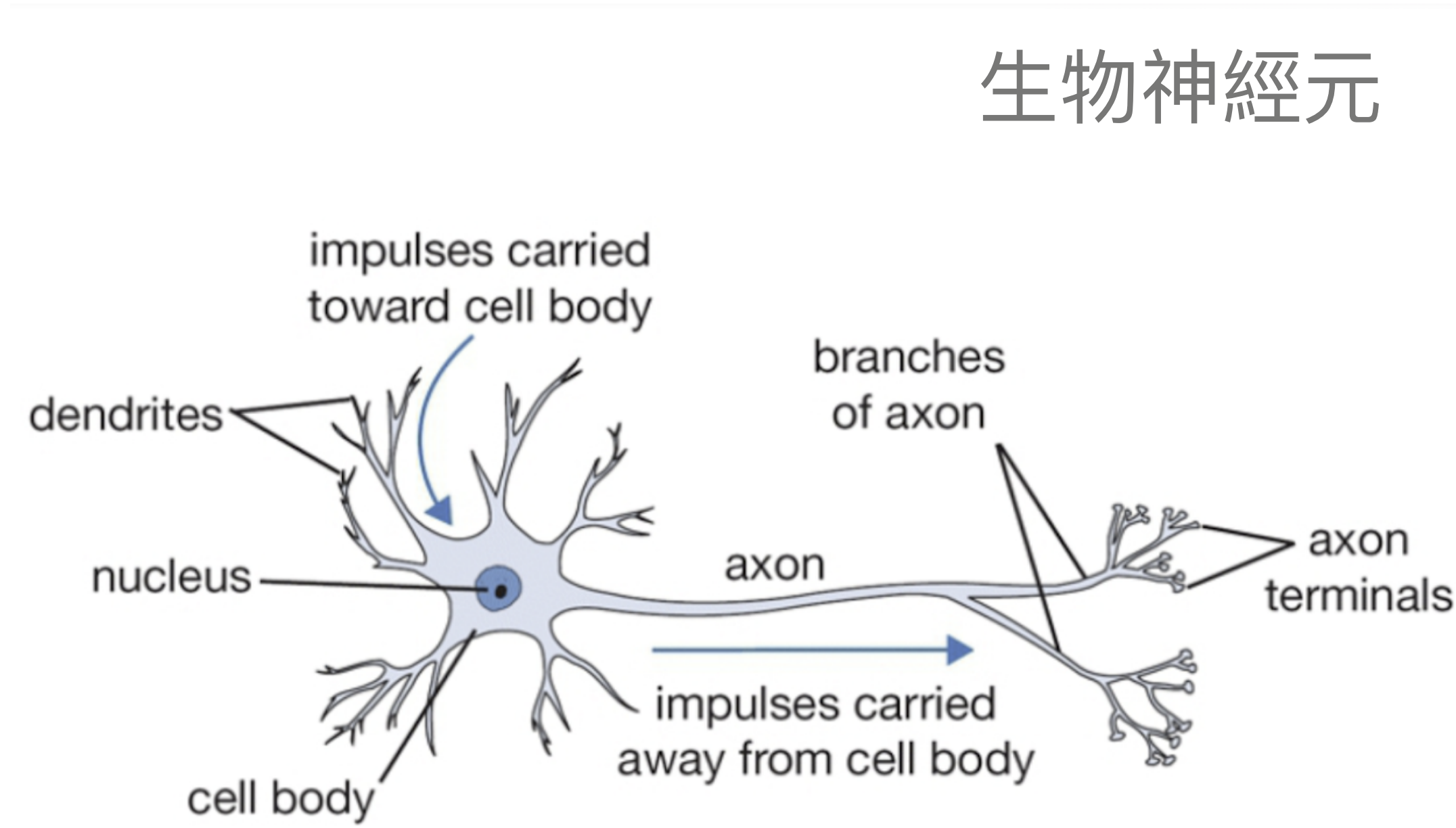


獲得
知識點

完成今日課程後你應該可以了解

- 針對不同的問題使用合適的激活函數

何謂激活函數(I)



圖片來源：blog.zenika

激活函數定義了每個節點（神經元）的輸出和輸入關係的函數為神經元提供規模化非線性化能力，讓神經網絡具備強大的擬合能力

何謂激活函數(II)

- 輸出值的範圍

- 當激活函數輸出值是**有限**的時候，基於梯度的優化方法會更加**穩定**，因為特徵的表示受有限權值的影響更顯著
- 當激活函數的輸出是**無限**的時候，模型的訓練會更加**高效**

激活函數的作用

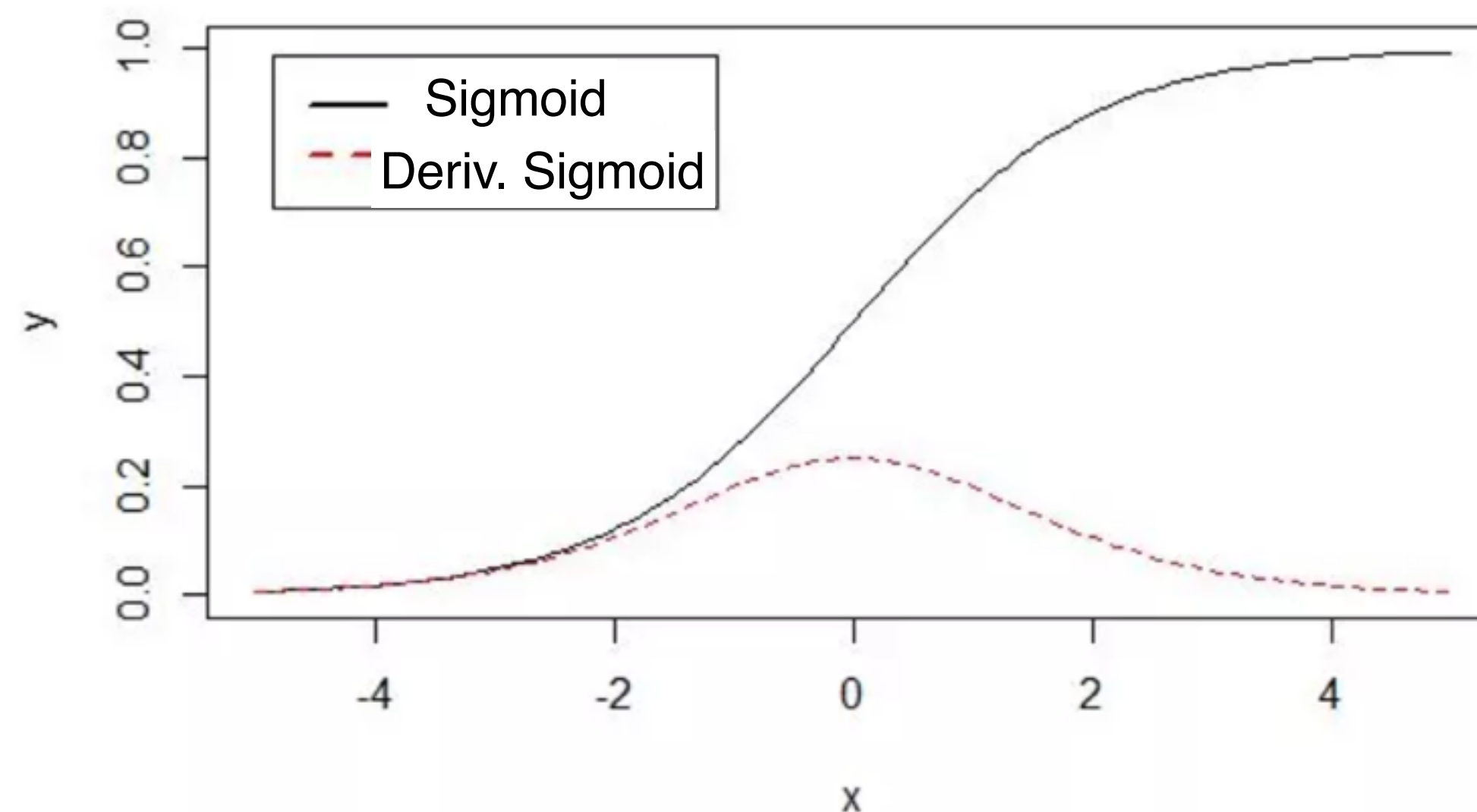
- 深度學習的基本原理是基於人工神經網絡，信號從一個神經元進入，經過**非線性的** activation function
 - 如此循環往復，直到輸出層。正是由於這些非線性函數的反复疊加，才使得神經網絡有足夠的 capacity 來抓取複雜的 pattern
- 激活函數的最大作用就是非線性化
 - 如果不用激活函數的話，無論神經網絡有多少層，輸出都是輸入的線性組合
- 激活函數的另一個重要特徵是：它應該是可以區分
 - 以便在網絡中向後推進以計算相對於權重的誤差（丟失）梯度時執行反向優化策略，然後相應地使用梯度下降或任何其他優化技術優化權重以減少誤差

常用激活函數介紹：Sigmoid

- 特點是會把輸出限定在 0~1 之間，在 $x < 0$ ，輸出就是 0，在 $x > 0$ ，輸出就是 1，這樣使得數據在傳遞過程中不容易發散
- 兩個主要缺點
 - 一是 Sigmoid 容易過飽和，丟失梯度。這樣在反向傳播時，很容易出現梯度消失的情況，導致訓練無法完整
 - 二是 Sigmoid 的輸出均值不是 0

原函數及導數圖如下：

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

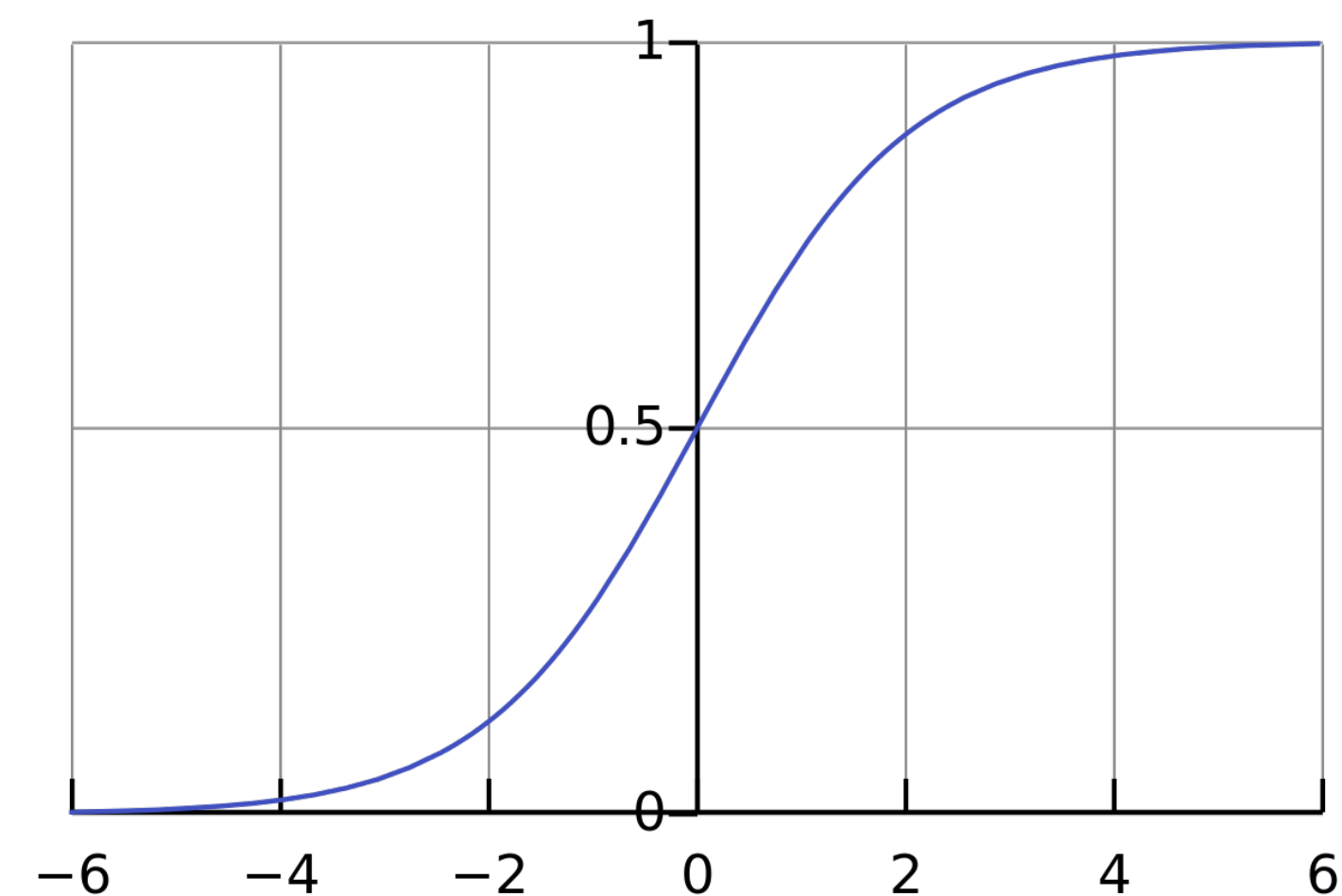


常用激活函數介紹：Softmax

- Softmax 把一個 k 維的 real value 向量 (a1,a2,a3,a4....) 映射成一個 (b1,b2,b3,b4....) 其中 bi 是一個 0~1 的常數，輸出神經元之和為1.0，所以可以拿來做多分類的機率預測
- 為什麼要取指數
 - 第一個原因是要模擬 max 的行為，所以要讓大的更大。
 - 第二個原因是需要一個可導的函數

原函數及導數圖如下：

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}}$$



常用激活函數介紹：Tanh

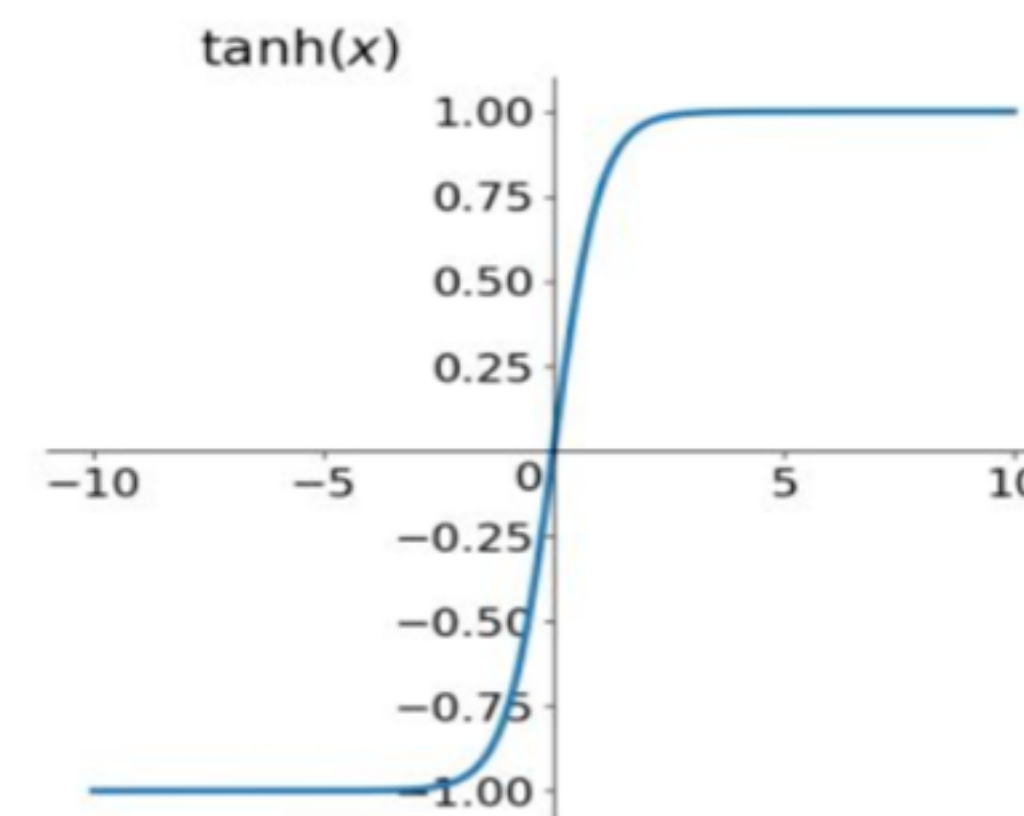
- tanh 讀作 Hyperbolic Tangent
- tanh 也稱為雙切正切函數，取值範圍為 $[-1,1]$ 。

tanh 在特徵相差明顯時的效果會很好，在循環過程中會不斷擴大特徵效果

原函數及導數圖如下：

$$\tanh(x) = 2\sigma(2x) - 1$$

$$\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

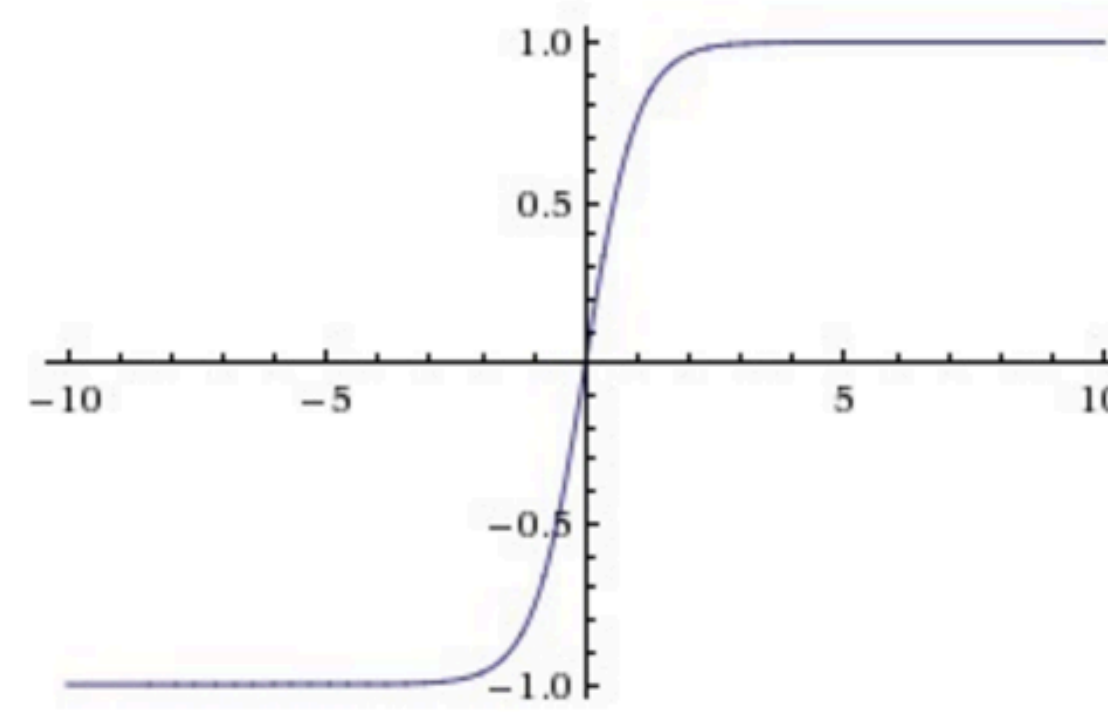
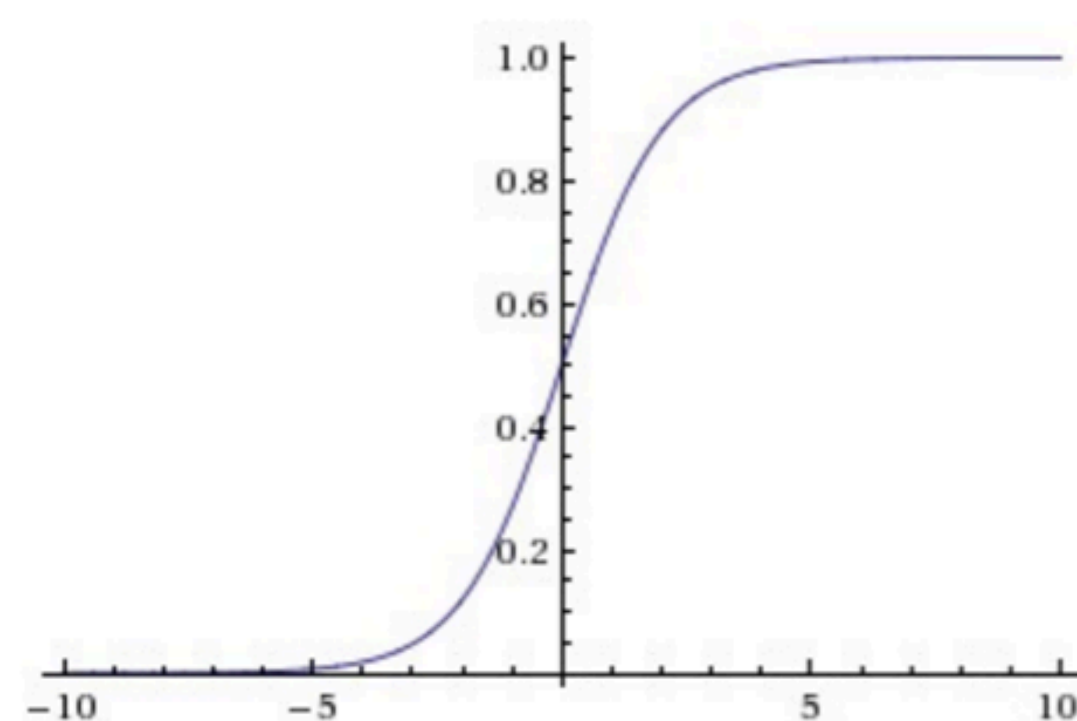


常用激活函數介紹：Sigmoid vs Softmax

- Sigmoid 將一個 real value 映射到 $(0,1)$ 的區間，用來做二分類
- Softmax 把一個 k 維的 real value 向量 $(a_1, a_2, a_3, a_4, \dots)$ 映射成一個 $(b_1, b_2, b_3, b_4, \dots)$ 其中 b_i 是一個 $0 \sim 1$ 的常數，輸出神經元之和為 1.0，所以可以拿來做多分類的機率預測
- 二分類問題時 sigmoid 和 softmax 是一樣的，求的都是 cross entropy loss

常用激活函數介紹: Sigmoid vs Tanh

- tanh 函數將輸入值壓縮到 $-1 \sim 1$ 的範圍，因此它是 0 均值的，解決了 Sigmoid 函數的非 zero-centered 問題，但是它也存在梯度消失和冪運算的問題。
- 其實 $\tanh(x) = 2\text{sigmoid}(2x) - 1$



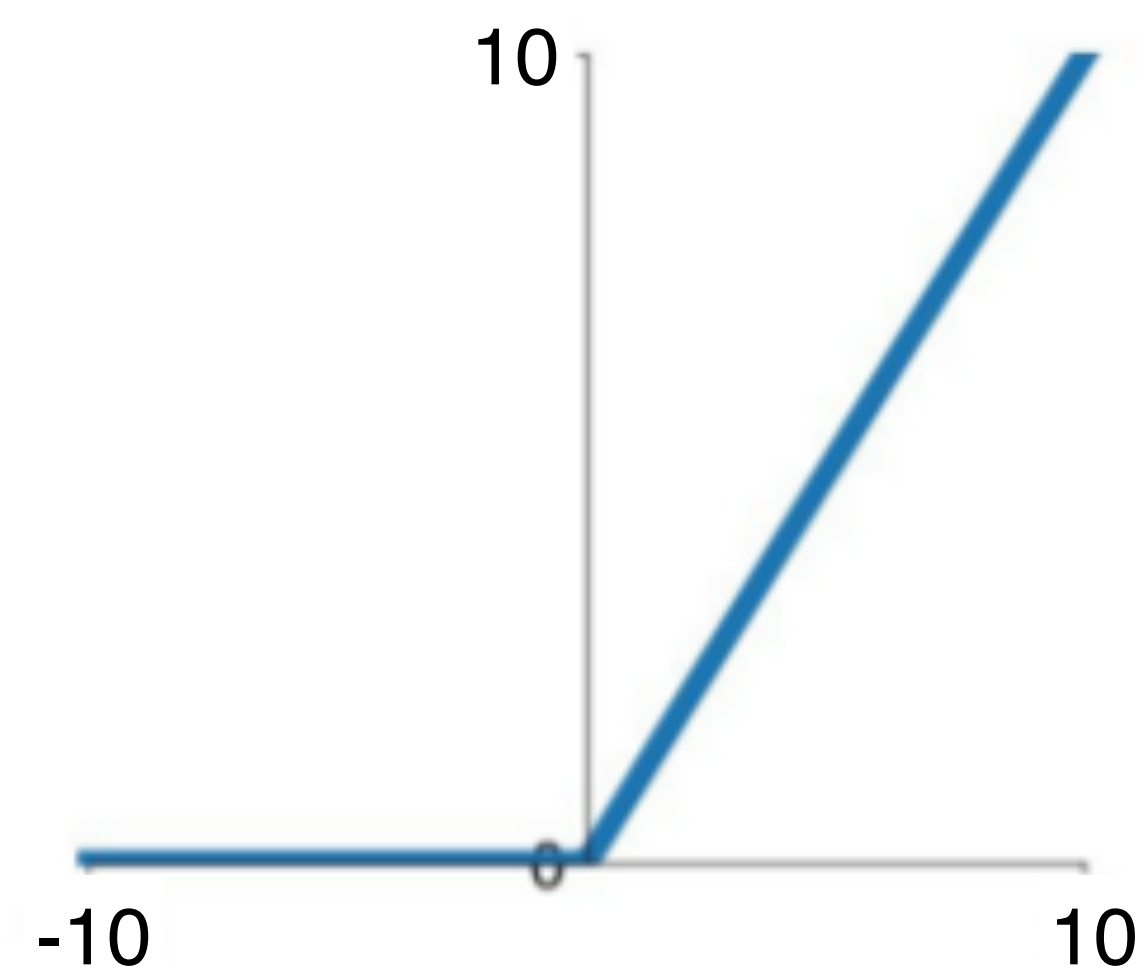
左邊是 Sigmoid 非線性函數，將實數壓縮到 $[0, 1]$ 之間。右邊是 Tanh 函數，將實數壓縮到 $[-1, 1]$ 。

常用激活函數介紹：ReLU

- 修正線性單元（Rectified linear unit，ReLU）
 - 在 $x > 0$ 時導數恆為 1
 - 對於 $x < 0$ ，其梯度恆為 0，這時候它也會出現飽和的現象，甚至使神經元直接無效，從而其權重無法得到更新（在這種情況下通常稱為 dying ReLU）
 - Leaky ReLU 和 PReLU 的提出正是為了解決這一問題

ReLU 函數公式和曲線如下：

$$f(x) = \max(0, x)$$

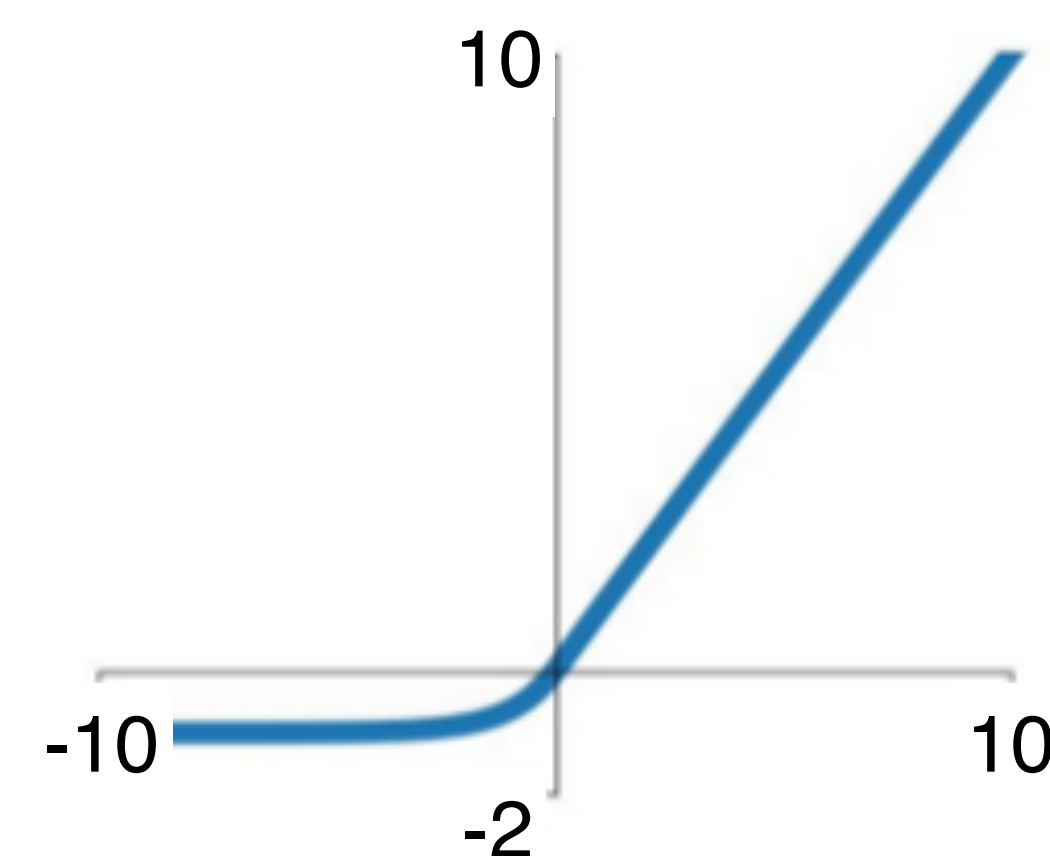


常用激活函數介紹：ReLU (II)

ELU 函數公式和曲線如下：

- ELU 函數是針對 ReLU 函數的一個改進型，相比於 ReLU 函數，在輸入為負數的情況下，是有一定的輸出的
- 這樣可以消除 ReLU 死掉的問題
- 還是有梯度飽和和指數運算的問題

$$f(x) = \begin{cases} x & , x > 0 \\ a(e^x - 1) & , x \leq 0 \end{cases}$$

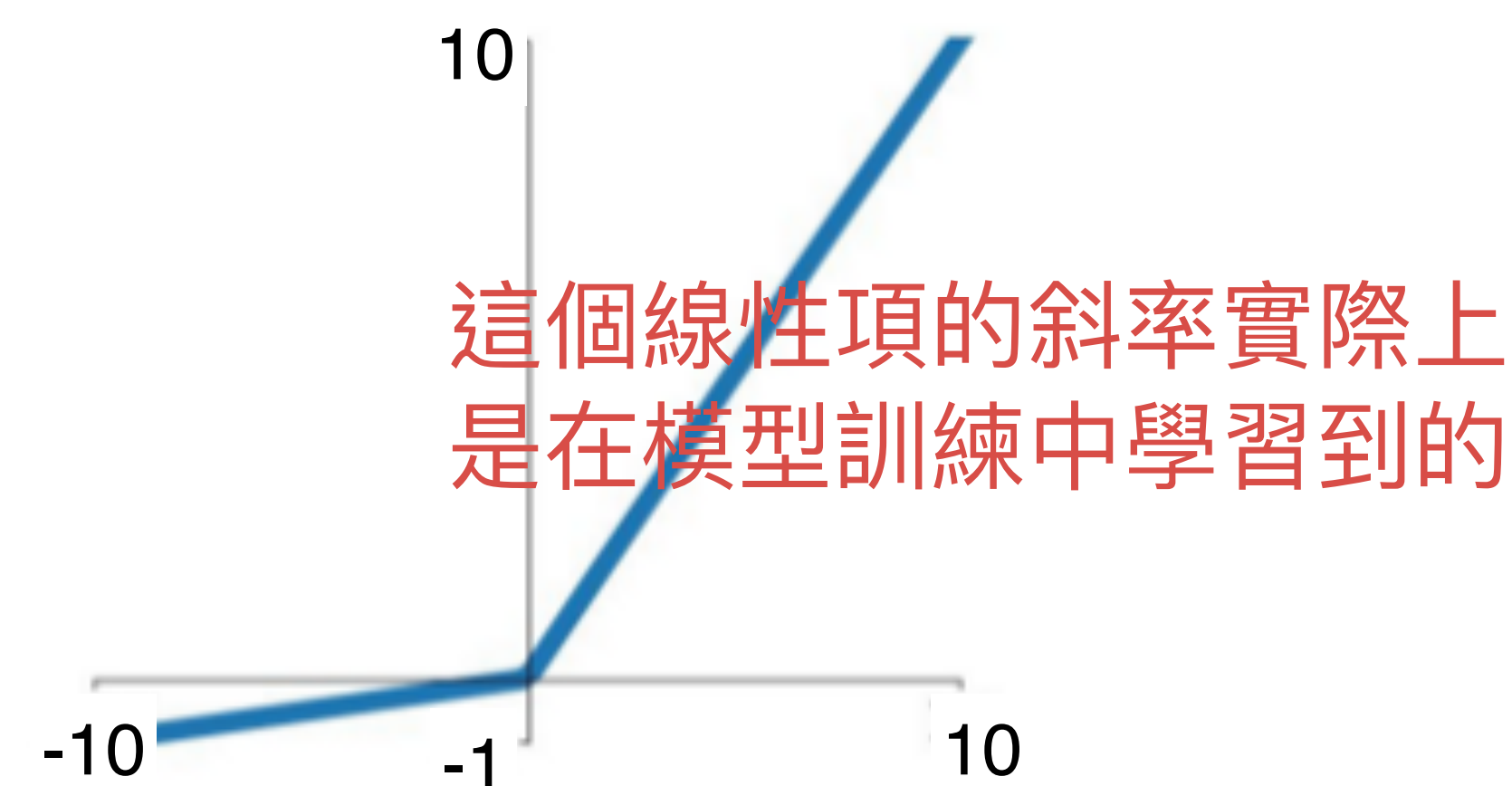


常用激活函數介紹：ReLU (III)

- PReLU
 - 參數化修正線性單元 (Parameteric Rectified Linear Unit, PReLU) 屬於 ReLU 修正類激活函數的一員。
- Leaky ReLU
 - 當 $\alpha=0.1$ 時，我們叫 PReLU 為 Leaky ReLU，算是 PReLU 的一種特殊情況
- RReLU 以及 Leaky ReLU 有一些共同點，即為負值輸入添加了一個線性項。

PReLU 函數公式和曲線如下：

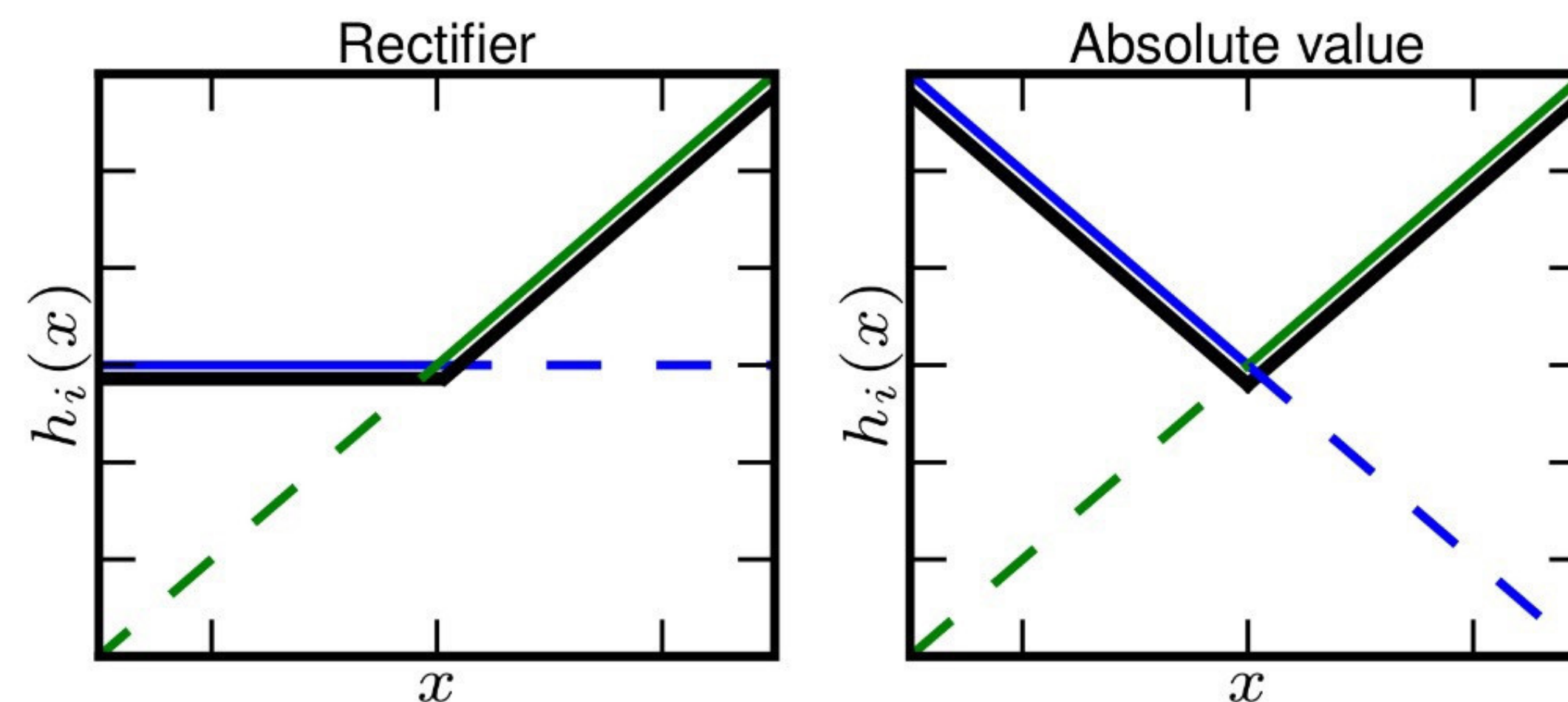
$$f(x) = \max(ax, x)$$



常用激活函數介紹: Maxout

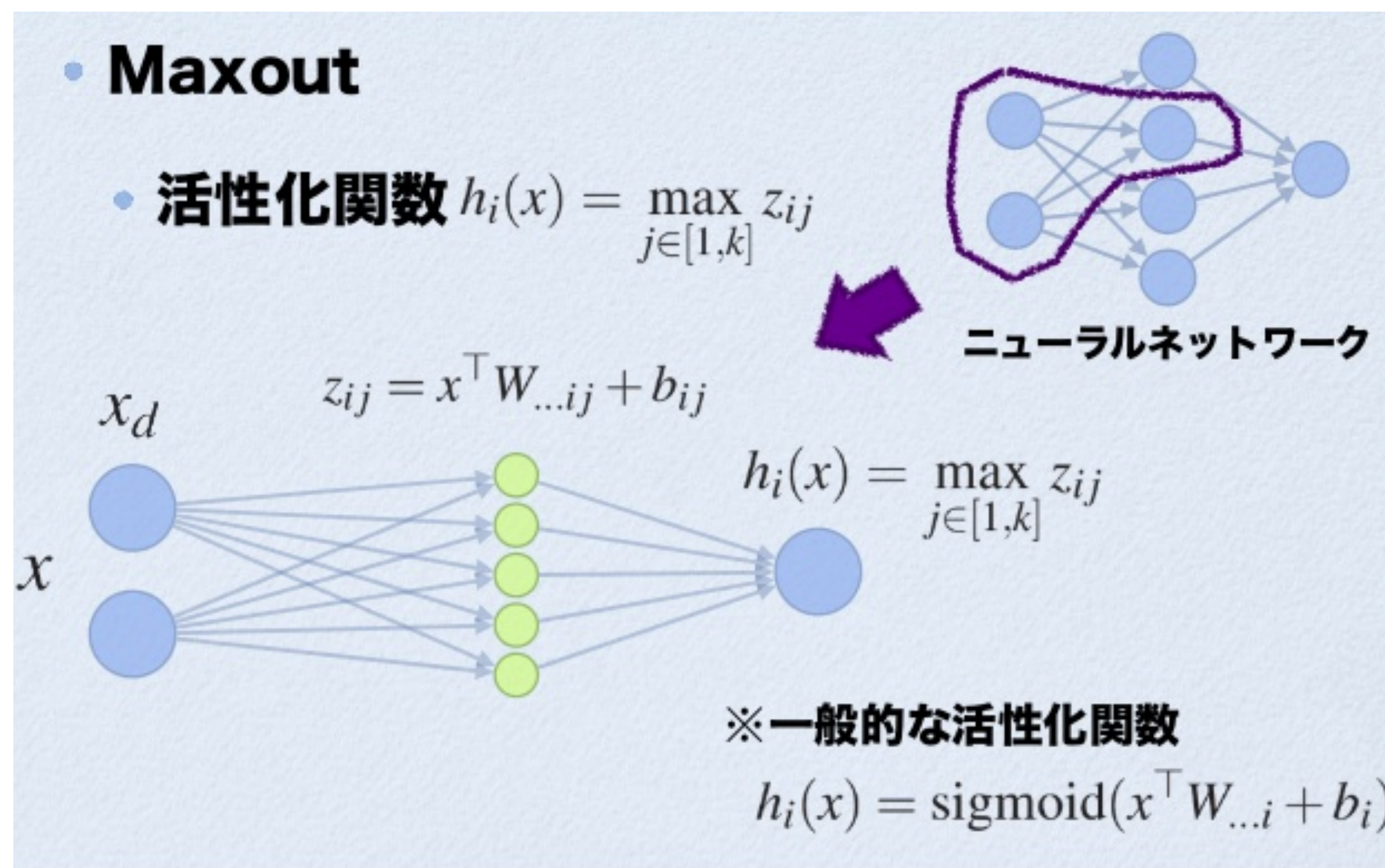
- Maxout 是深度學習網絡中的一層網絡，就像池化層、卷積層一樣，可以看成是網絡的激活函數層
- Maxout 神經元的激活函數是取得所有這些「函數層」中的最大值
- Maxout 的擬合能力是非常強的，優點是計算簡單，不會過飽和，同時又沒有 ReLU 的缺點
- Maxout 的缺點是過程參數相當於多了一倍

$$f(x) = \max(wT1x+b1, wT2x+b2)$$



常用激活函數介紹: Maxout (II)

$$f(x) = \max(wT1x+b1, wT2x+b2)$$



如何選擇正確的激活函數

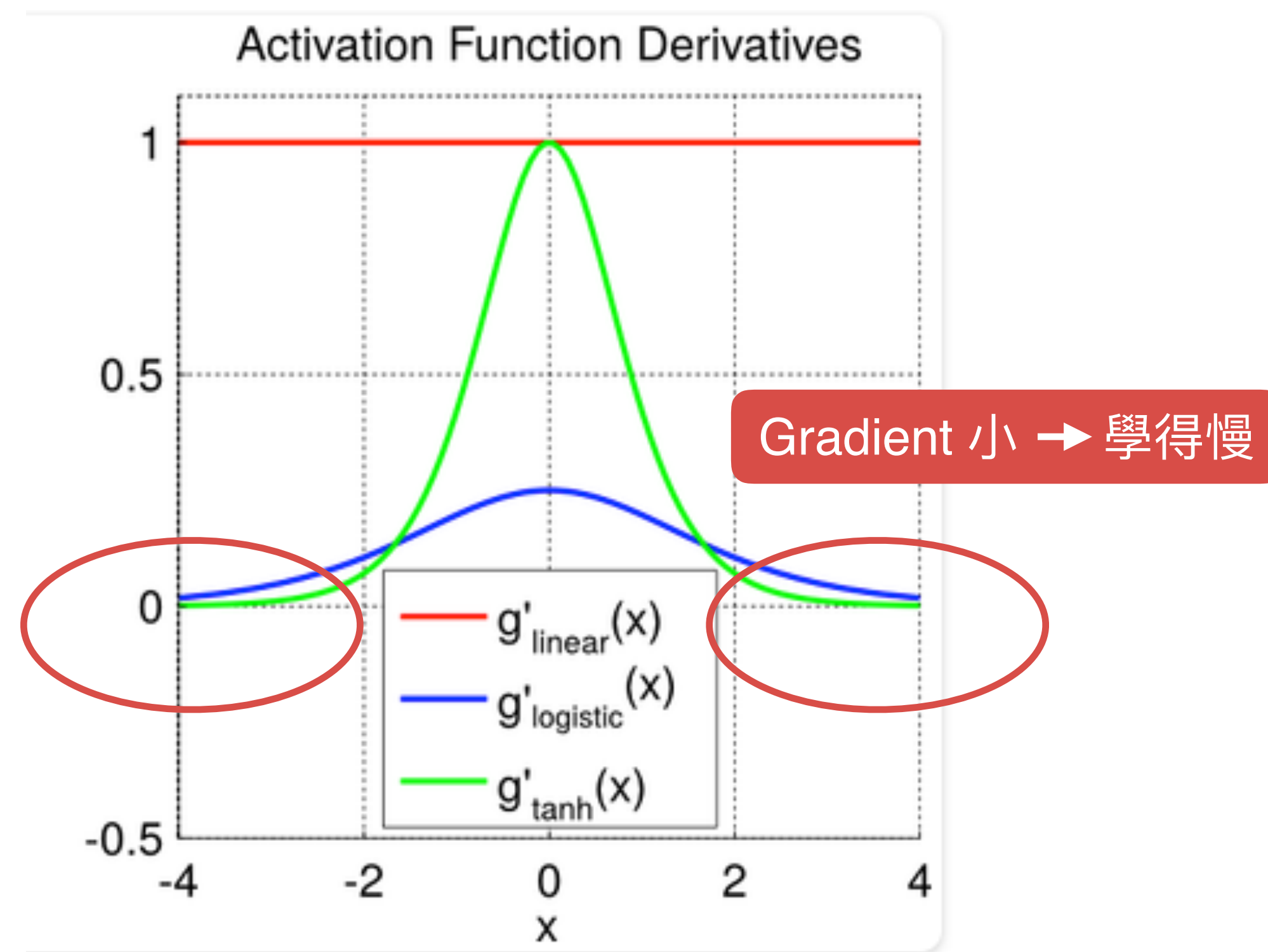
- 根據各個函數的優缺點來配置
 - 如果使用 ReLU，要小心設置 learning rate，注意不要讓網絡出現很多「dead」神經元，如果不好解決，可以試試 Leaky ReLU、PReLU 或者 Maxout
- 根據問題的性質
 - 用於分類器時，Sigmoid 函數及其組合通常效果更好
 - 由於梯度消失問題，有時要避免使用 sigmoid 和 tanh 函數。ReLU 函數是一個通用的激活函數，目前在大多數情況下使用
 - 如果神經網絡中出現死神經元，那麼 PReLU 函數就是最好的選擇
 - ReLU 函數建議只能在隱藏層中使用

如何選擇正確的激活函數 (II)

- 考慮 DNN 損失函數和激活函數
 - 如果使用 sigmoid 激活函數，則交叉熵損失函數一般肯定比均方差損失函數好；
 - 如果是 DNN 用於分類，則一般在輸出層使用 softmax 激活函數
 - ReLU 激活函數對梯度消失問題有一定程度的解決，尤其是在 CNN 模型中。

如何選擇正確的激活函數 (III)

- 梯度消失 Vanishing gradient problem
- 原因：前面的層比後面的層梯度變化更小，故變化更慢
- 結果：Output 變化慢 \rightarrow Gradient 小 \rightarrow 學得慢
- Sigmoid, Tanh 都有這樣特性
- 不適合用在 Layers 多的DNN 架構



圖片來源：[the clever machine](#)

前述流程 / python程式 對照

```
import numpy as np
from numpy import *
import matplotlib.pyplot as plt
%matplotlib inline

def ReLU(x):
    return abs(x) * (x > 0)

def dReLU(x):
    return (1 * (x > 0))

# linspace generate an array from start and stop value
# with requested number of elements.
x = plt.linspace(-10,10,100)

# prepare the plot, associate the color r(ed) or b(lue) and the label
plt.plot(x, ReLU(x), 'r')
plt.plot(x, dReLU(x), 'b')
```

前述流程 / python程式 對照

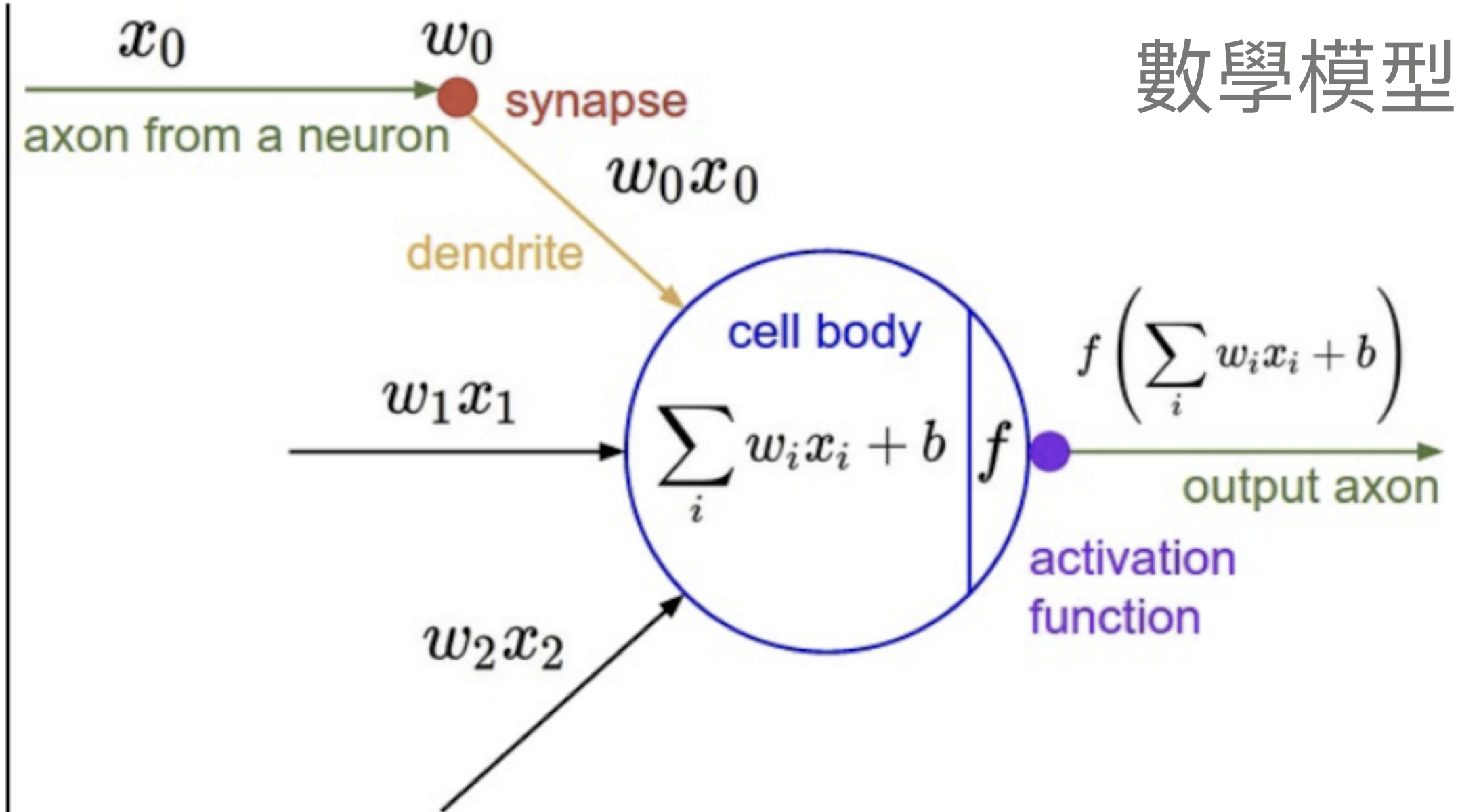
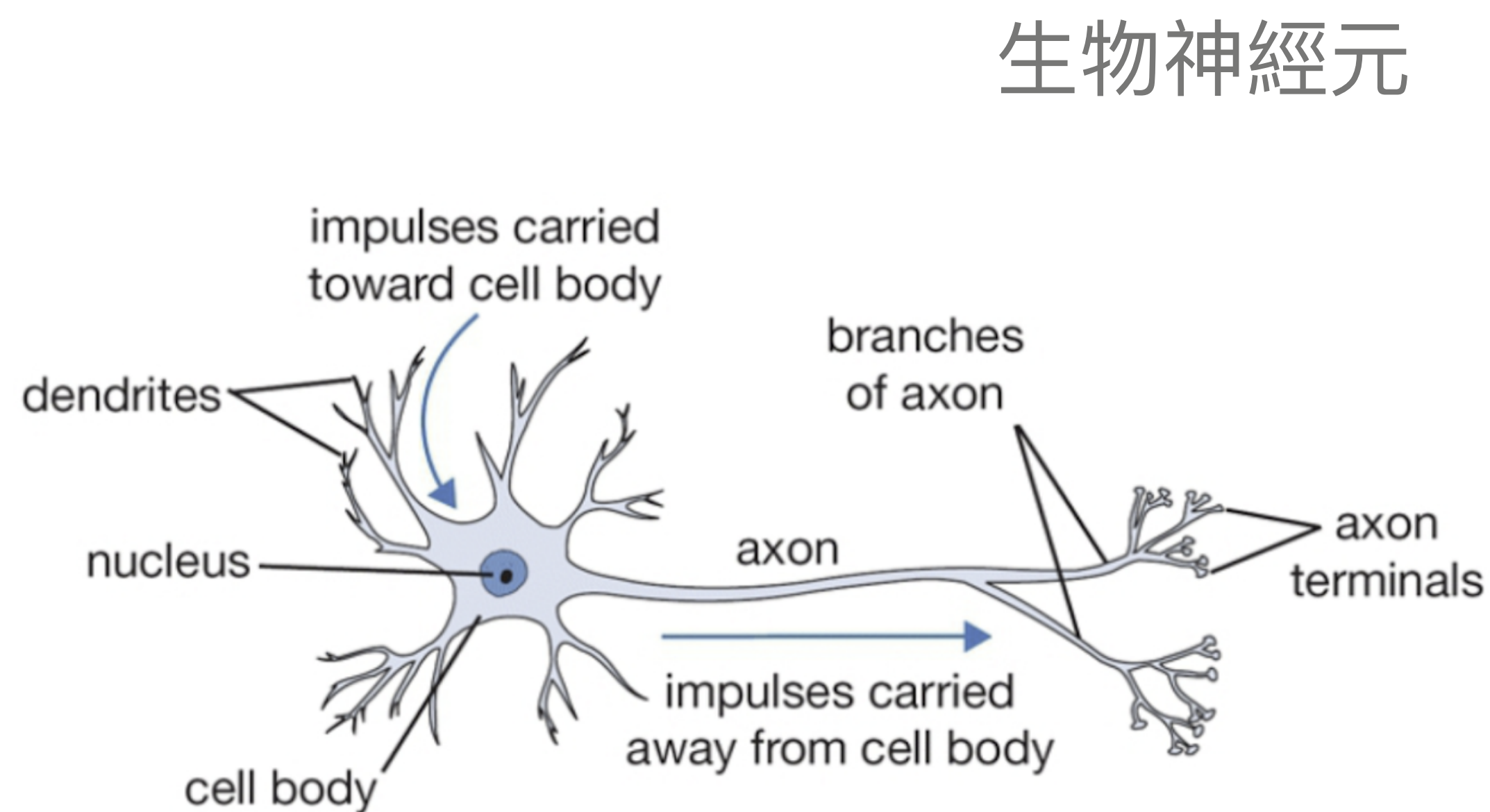
激活函數可以通過設置單獨的激活層實現，可以也。在構造層對象時通過傳遞 activation 參數實現：

```
from keras.layers import Activation,Dense  
model.add(Dense(64,activation='tanh'))
```

考慮不同Backend support，通過傳遞一個元素運算的Theano / TensorFlow / CNTK函數來作為激活函數：

```
from keras import backend as k  
model.add(Dense(64,activation=k.tanh))  
model.add( Activation(k.tanh))
```


重要知識點複習：何謂激活函數



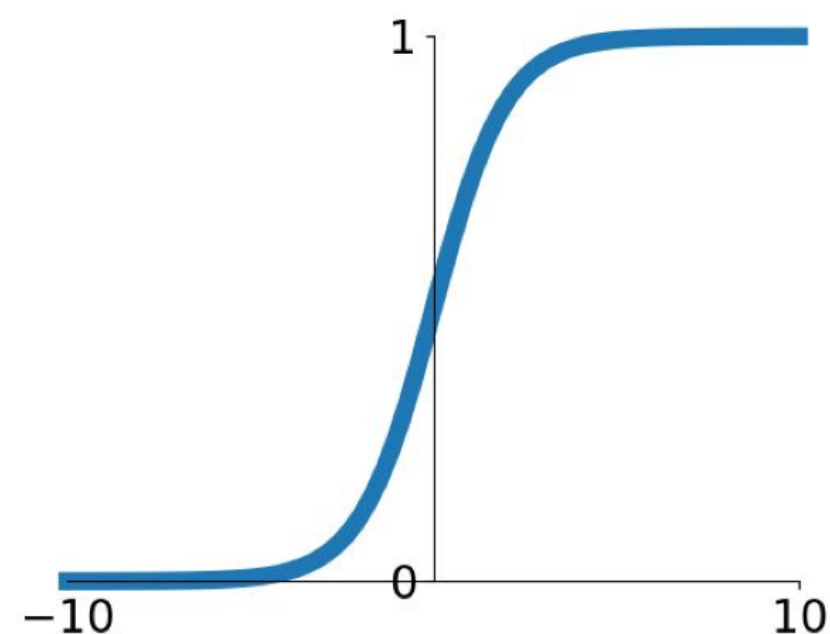
圖片來源：blog.zenika

激活函數定義了每個節點（神經元）的輸出和輸入關係的函數為神經元提供規模化非線性化能力，讓神經網絡具備強大的擬合能力

重要知識點複習：圖示說明

Sigmoid

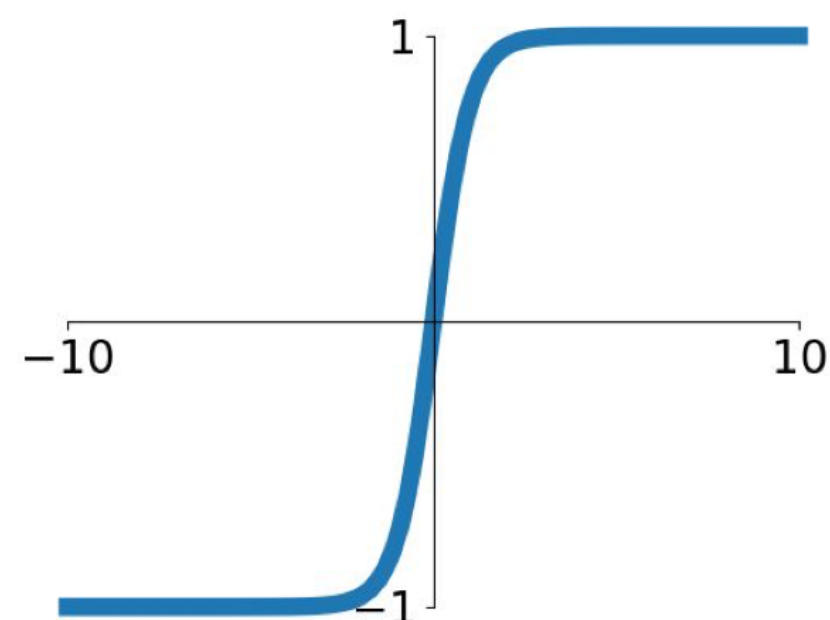
$$\sigma(x) = \frac{1}{1+e^{-x}}$$



如果是 DNN 用於分類，則一般在輸出層使用 softmax 激活函數

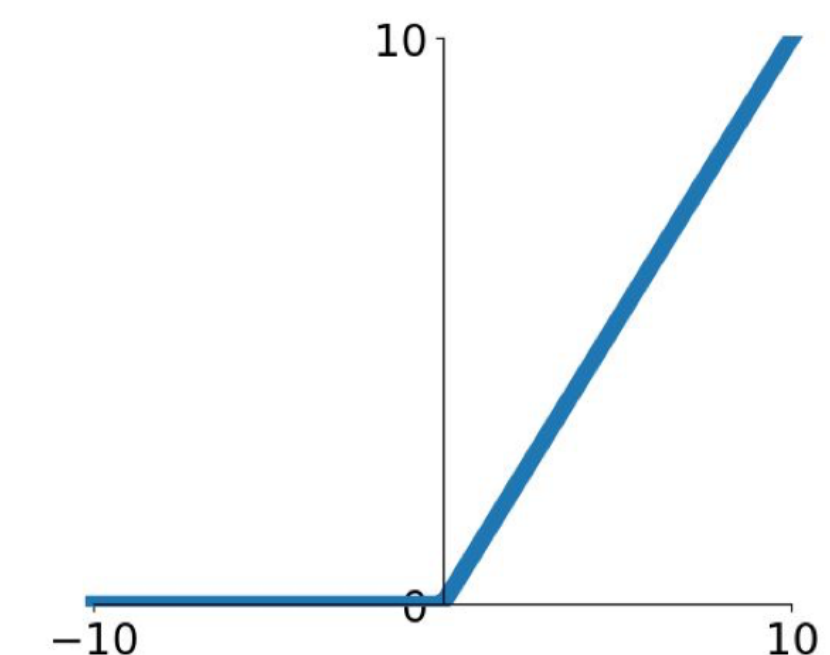
tanh

$$\tanh(x)$$



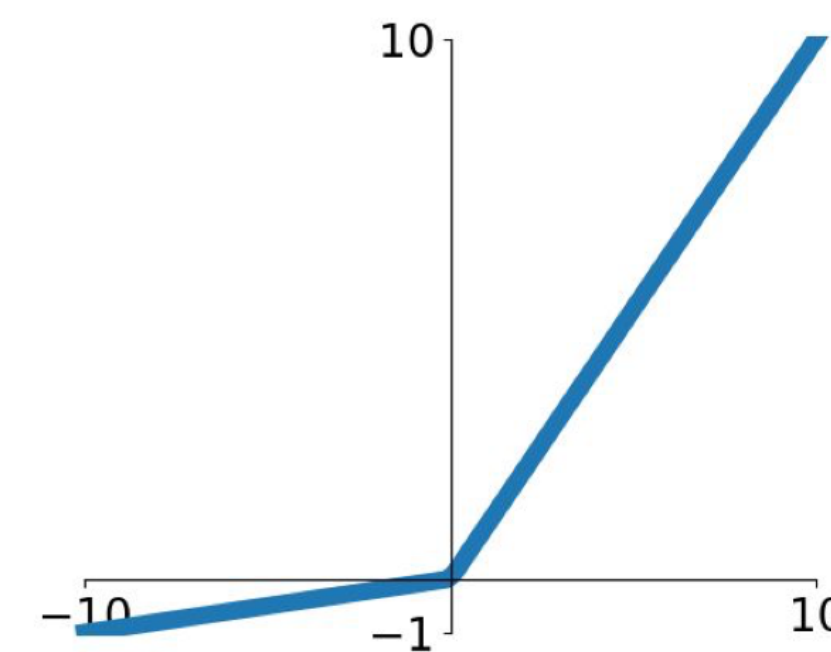
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

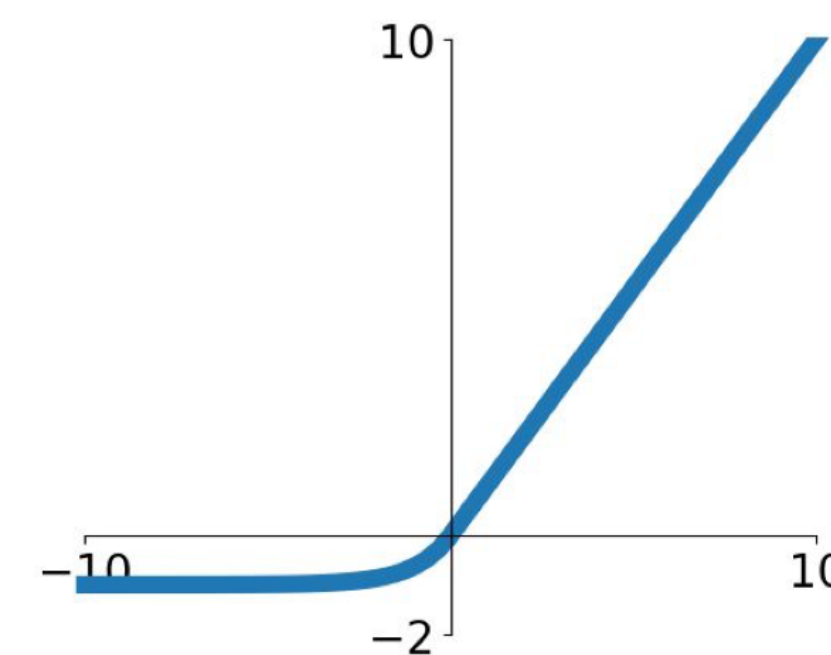


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



重要知識點複習：如何選擇正確的激活函數

- 根據各個函數的優缺點來配置
 - 如果使用 ReLU，要小心設置 learning rate，注意不要讓網絡出現很多「dead」神經元，如果不好解決，可以試試 Leaky ReLU、PReLU 或者 Maxout
- 根據問題的性質
 - 用於分類器時，Sigmoid 函數及其組合通常效果更好
 - 由於梯度消失問題，有時要避免使用 sigmoid 和 tanh 函數。ReLU 函數是一個通用的激活函數，目前在大多數情況下使用
 - 如果神經網絡中出現死神經元，那麼 PReLU 函數就是最好的選擇
 - ReLU 函數建議只能在隱藏層中使用

解題時間 It's Your Turn

請跳出PDF至官網Sample Code & 作業
開始解題

