Artificial Neural Network

國立宜蘭大學資訊工程系

吳政瑋 助理教授

wucw@niu.edu.tw

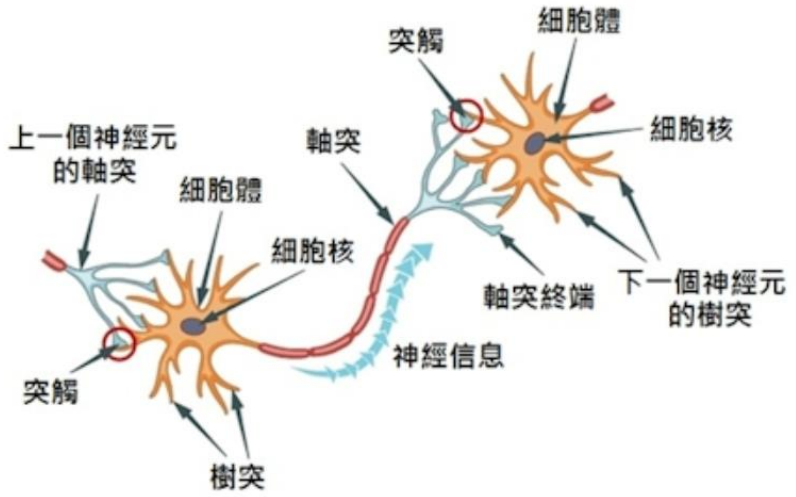
為什麼需要類神經網路？

- **傳統方法的限制**
  - 規則需人工設計(if–else)
  - 難以處理非線性與高維資料
  - 對雜訊與複雜模式適應性差

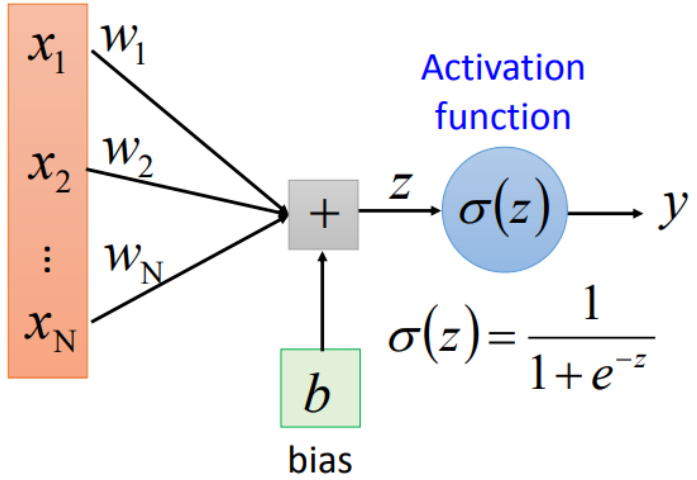
- **類神經網路的優勢**
  - 可從資料中自動學習特徵
  - 擅長處理非線性問題
  - 可應用於影像、語音、文字等複雜任務

Biological Neurons vs. Artificial Neurons

- **Biological Neurons** (生物神經元)
  - 樹突(Dendrite)：接收訊號
  - 突觸 (Synapse) ：決定訊號傳遞強弱程
  - 細胞體(Soma)：整合訊號
  - 軸突(Axon)：輸出訊號

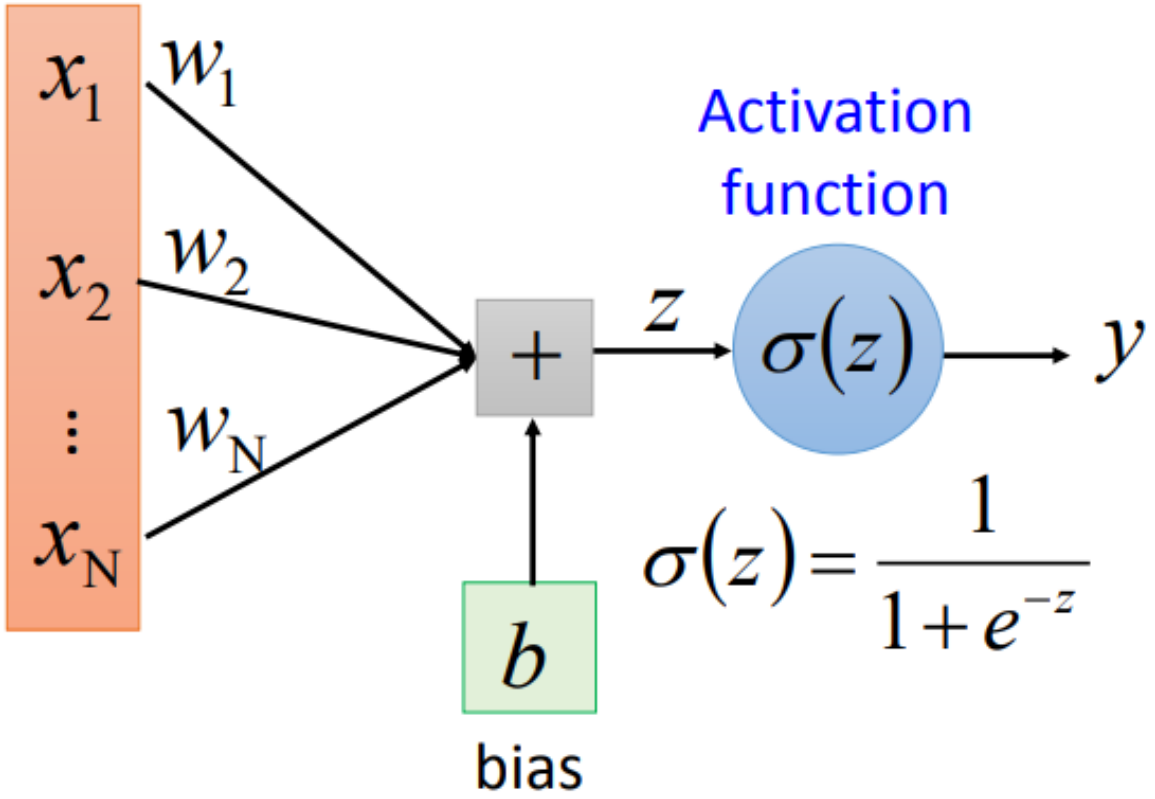


- **Artificial Neurons** (人工神經元)
  - 輸入(inputs)
  - 權重(weights)
  - 加權總和＋偏置(bias)
  - 活化函數(activation function)

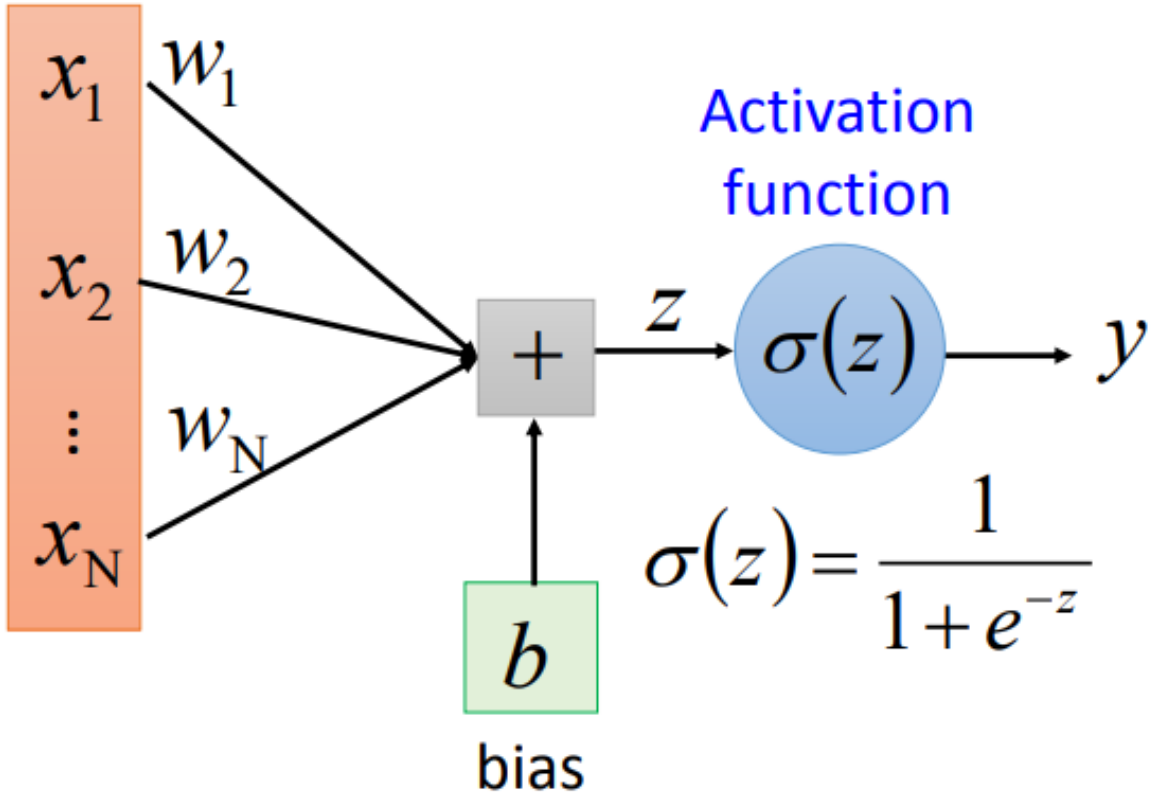


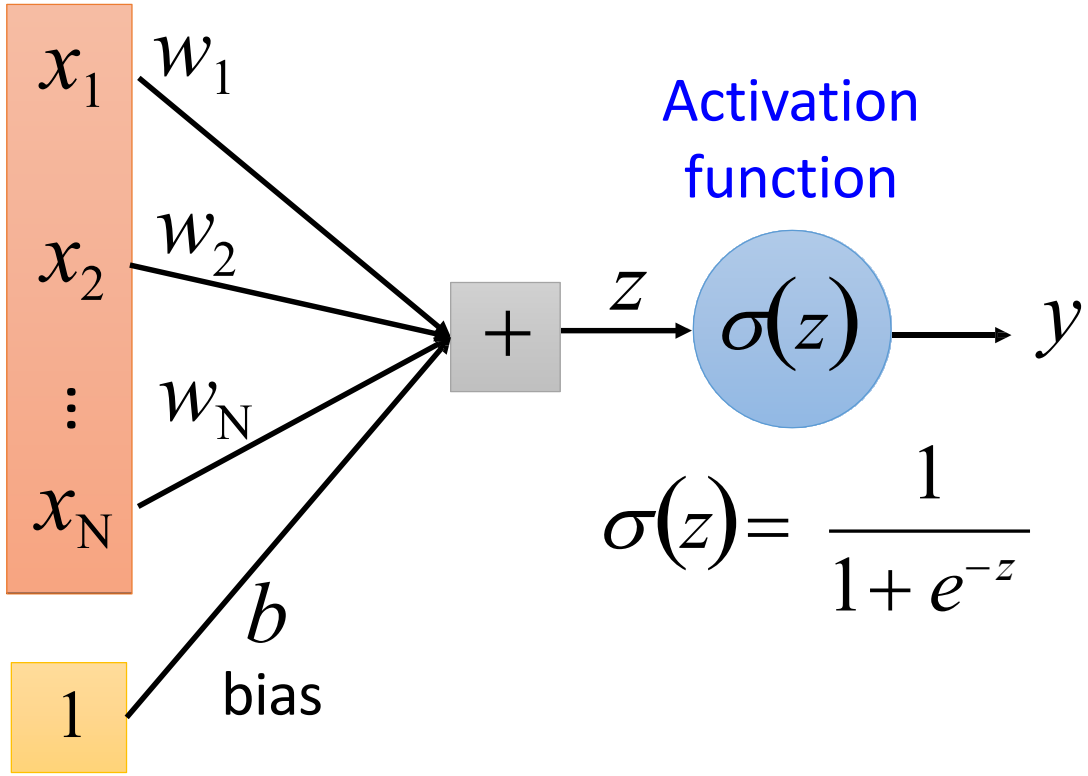
Mathematical Model of Artificial Neurons

- $z = \sum_{i=1}^{n} w_i x_i + b\text{ias}$
- $\text{y} = f(z)$
  - x_i：輸入特徵
  - w_i：權重
  - b：偏置
  - $f(\cdot)$：活化函數



Mathematical Model of Artificial Neurons





Mathematical Model of Artificial Neurons

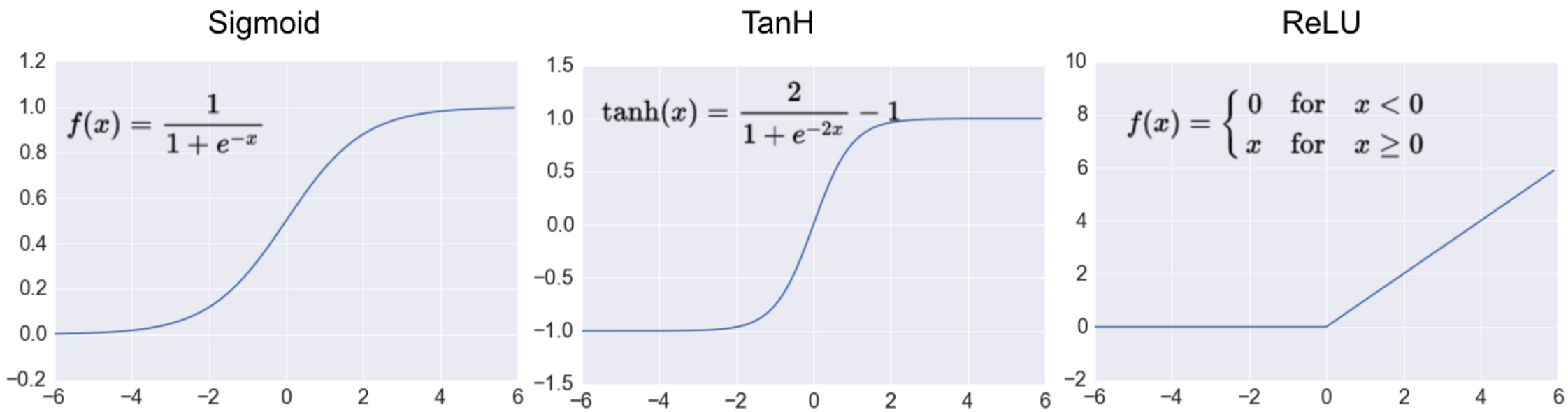
- **神經元公式：**

$$y = f(\sum_{i=1}^{n} w_i x_i + b)$$

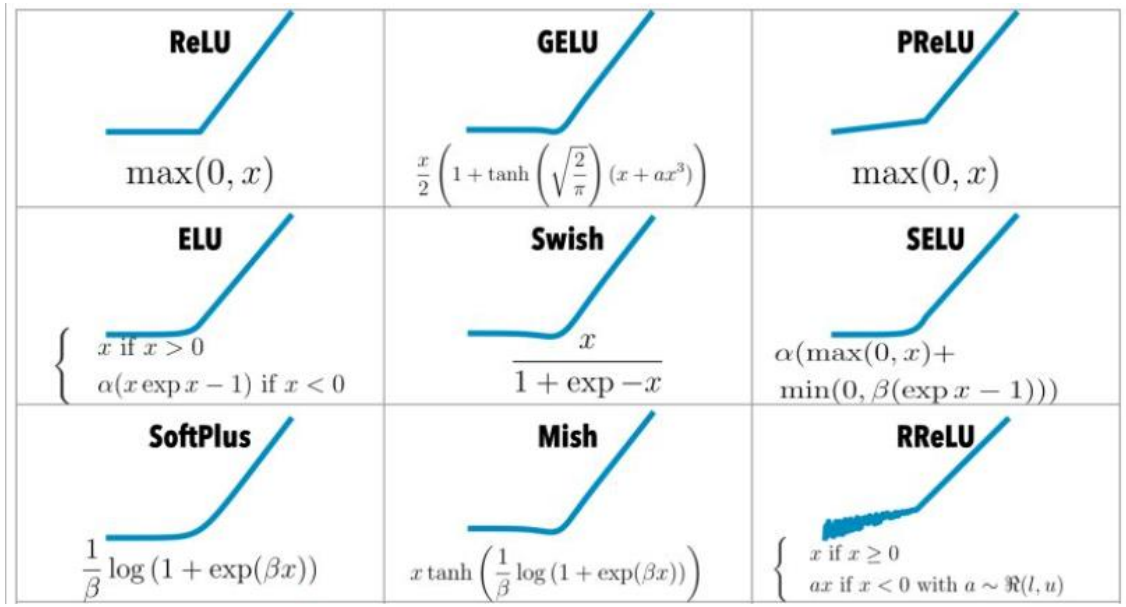
- x_i (Inputs)：來自感測器或前一層的特徵數據。
- w_i (Weights)：學習到的「經驗值」，代表特徵的重要性。
- b (Bias)：調整門檻值，確保模型在輸入為 0 時仍有靈活性。
- f (Activation Function)：引入「非線性」，讓網路能處理複雜問題。

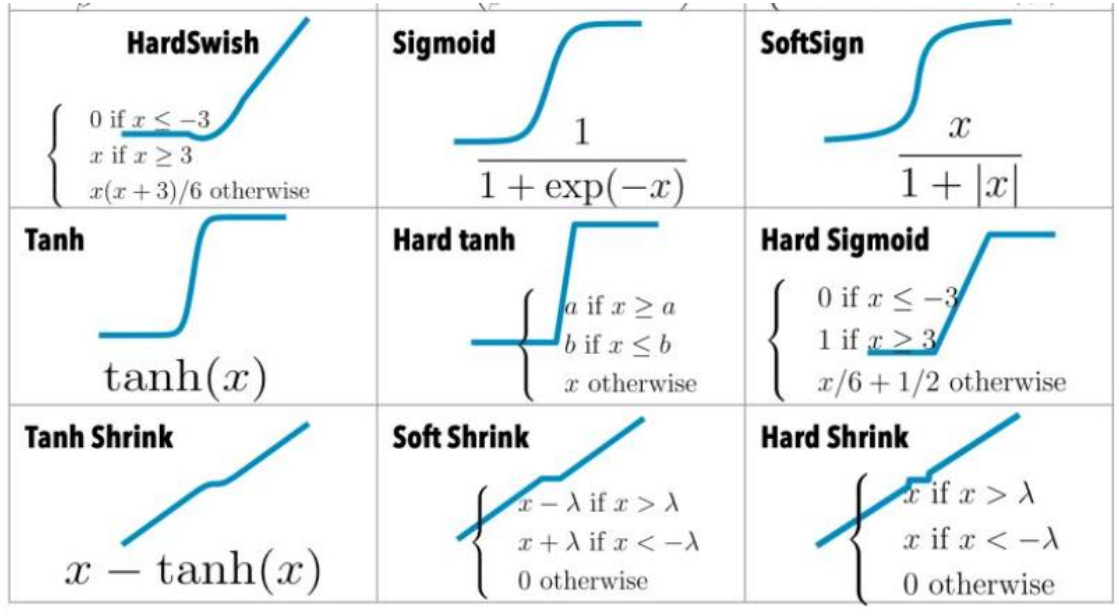
Common Activation Functions

- **Activation Function**：引入「非線性」處理，讓網路能處理複雜問題



Different Types of Activation Functions





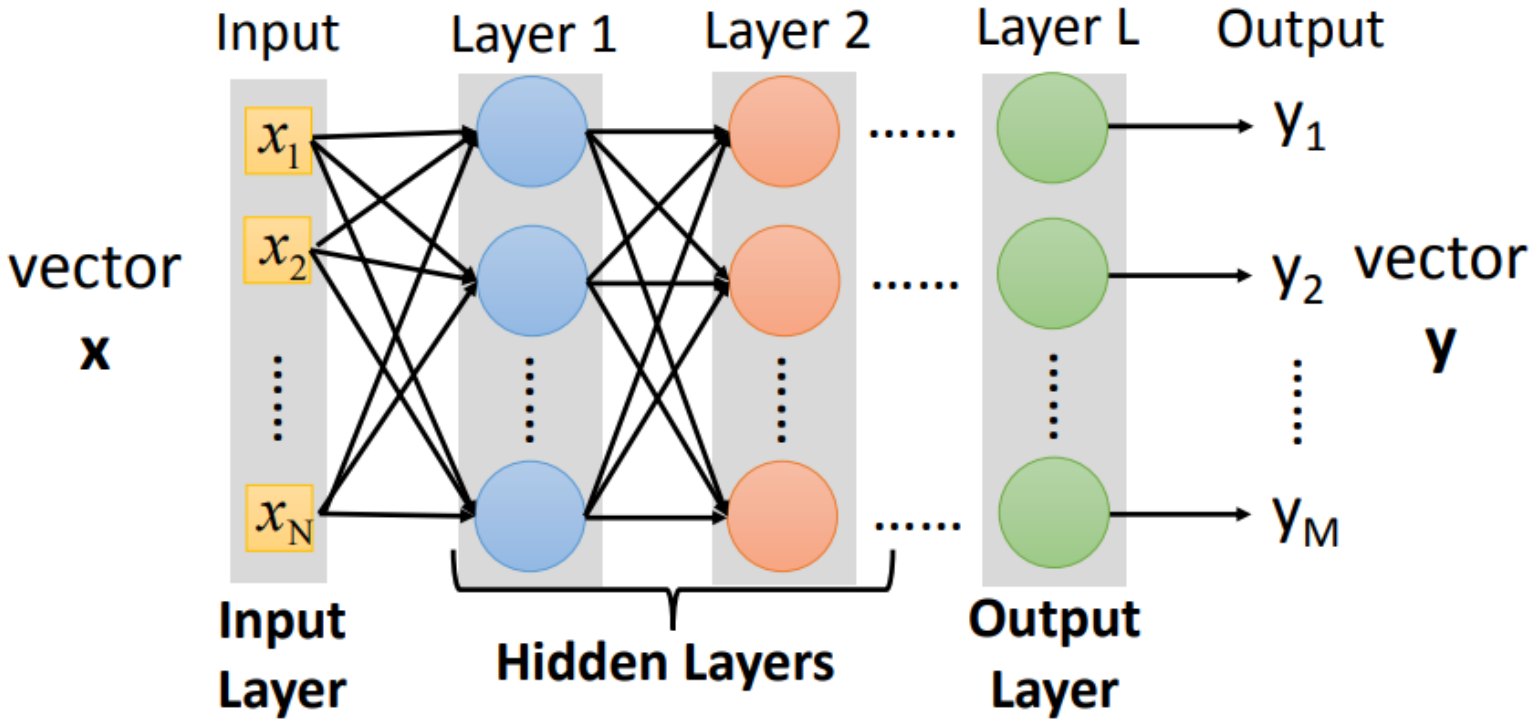
Why is an activation function needed?

- **直觀解釋**
  - 如果沒有 *activation function*，不論疊加多少層，結果永遠是線性組合(一條直線)。

- **常用函數**
  - Sigmoid：將輸出壓縮在 0 ~1 之間(適合機率)。
  - ReLU (Rectified Linear Unit)：
    - 目前的主流，解決梯度消失問題，
    - $f(x) = \max(0, x)$。
  - Tanh：輸出範圍 -1 ~ 1。

From a Single Neuron to a Network (The Architecture)

$$f: R^N \rightarrow R^M$$



- Fully connected feedforward network
- Deep Neural Network: many hidden layers

Basic Architecture of Neural Networks

- **Three Types of Layers**
  - Input Layer (輸入層)
  - Hidden Layer (隱藏層)
  - Output Layer (輸出層)

- **Classified by Depth**
  - Perceptron (單層感知器)
  - Multi-Layer Perceptron (多層感知器)
  - Deep Neural Network (深度神經網路)

Feedforward Propagation (前向傳播)

- **概念**
  - 資料從輸入層 → 隱藏層 → 輸出層
  - 逐層計算加權總和與活化函數

- **目的**
  - 根據目前權重，產生預測結果 $\hat{y}$

Loss Function (損失函數)

- **Loss Function 用途：**
  - 衡量模型預測與真實答案的差距

- **常見損失函數：**
  - 均方誤差(MSE)：回歸問題
  - Cross-Entropy：分類問題

Different Types of Loss Functions

Classification Loss Functions (Binary + Multi-class)

Binary Cross Entropy (BCE)	Loss function for binary classification tasks.	$\mathcal{L}_{BCE} = \frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(x_i)) + (1-y_i) \cdot log(1-p(x_i))$
Hinge Loss	Penalizes wrong and right (but less confident) predictions. Commonly used in SVMs.	$\mathcal{L}_{\text{Hinge}} = max(0, 1-(f(x)\cdot y))$
Cross Entropy Loss	Extension of BCE loss to multi-class classification.	$\mathcal{L}_{CE} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}.log(f(x_{ij}))$ $N : samples; M : classes$
KL Divergence	Minimizes the divergence between predicted and true probability distribution	$\mathcal{L}_{KL} = \sum_{i=1}^{N} y_i \cdot log(\frac{y_i}{f(x_i)})$

Regression Loss Functions

Mean Bias Error	Captures average bias in prediction. But is rarely used for training.	$\mathcal{L}_{MBE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i))$
Mean Absolute Error	Measures absolute average bias in prediction. Also called L1 Loss.	$\mathcal{L}_{MAE} = \frac{1}{N}\sum_{i=1}^{N}\lvert y_i - f(x_i)\rvert$
Mean Squared Error	Average squared distance between actual and predicted. Also called L2 Loss.	$\mathcal{L}_{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i))^2$
Root Mean Squared Error	Square root of MSE. Loss and dependent variable have same units.	$\mathcal{L}_{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i))^2}$
Huber Loss	A combination of MSE and MAE. It is parametric loss function.	$\mathcal{L}_{\text{Huberloss}} = \begin{cases} \frac{1}{2}(y_i - f(x_i))^2 & : \lvert y_i - f(x_i)\rvert \le \delta \\ \delta(\lvert y_i - f(x_i)\rvert - \frac{1}{2}\delta) & : otherwise \end{cases}$
Log Cosh Loss	Similar to Huber Loss + non-parametric. But computationally expensive.	$\mathcal{L}_{\text{LogCosh}} = \frac{1}{N}\sum_{i=1}^{N} log(cosh(f(x_i) - y_i))$

Binary Cross Entropy (BCE)

- **使用時機：二元分類問題**
  - 例如：會買 / 不會買、有病 / 沒病、垃圾信 / 非垃圾信

- 公式：

$$L = -\left[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})\right]$$

- $y \in \{0, 1\}$：真實標籤
- $\hat{y} \in (0, 1)$：模型預測為「類別 1」的機率

Binary Cross Entropy (二分類交叉熵)

$$L = -\left[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})\right]$$

使用者	真實標籤 y	預測機率 $\hat{y}$
A	1（會買）	0.9
B	1（會買）	0.2
C	0（不會買）	0.1
D	0（不會買）	0.8

A：真的會買，模型也很有信心

$$L = -\log(0.9) \approx 0.105$$

B：真的會買，但模型很不確定

$$L = -\log(0.2) \approx 1.609$$

C：真的不會買，模型也預測不會

$$L = -\log(1 - 0.1) = -\log(0.9) \approx 0.105$$

D：真的不會買，但模型預測會買

$$L = -\log(1 - 0.8) = -\log(0.2) \approx 1.609$$

Cross Entropy Loss (交叉熵損失)

- **使用時機：多類別分類（單一正確答案）**
  - 例如：手寫數字 0–9、鳶尾花分類

- 公式：

$$L = -\sum_{i=1}^{C} y_i \log(\hat{y}_i)$$

$$\text{Loss} = -(0 \cdot \log(0.1) + 1 \cdot \log(0.7) + 0 \cdot \log(0.2)) = -\log(0.7) \approx 0.357$$

- C：類別數
- y_i：One-hot 真實標籤
- $\hat{y}_i$：Softmax 輸出的機率

Cross Entropy Loss (交叉熵損失)

例子：手寫數字辨識（0、1、2）

真實答案：1

One-hot 標籤

$$y = [0, 1, 0]$$

模型預測 1（表現好）

$$\hat{y} = [0.1,\ 0.8,\ 0.1]$$

$$L = -\log(0.8) \approx 0.223$$

模型預測 2（表現差）

$$\hat{y} = [0.7,\ 0.2,\ 0.1]$$

$$L = -\log(0.2) \approx 1.609$$

Backpropagation (反向傳播法)

- **核心概念**
  - 利用鏈鎖法則(Chain Rule)
  - 將誤差由輸出層往回傳
  - 計算每個權重的梯度

- **目標**
  - 更新權重，使損失函數最小化

Update Weights by Gradient Descent (梯度下降法)

- **權重更新公式**

$$w := w - \eta \frac{\partial L}{\partial w}$$

  - η：學習率（Learning Rate）

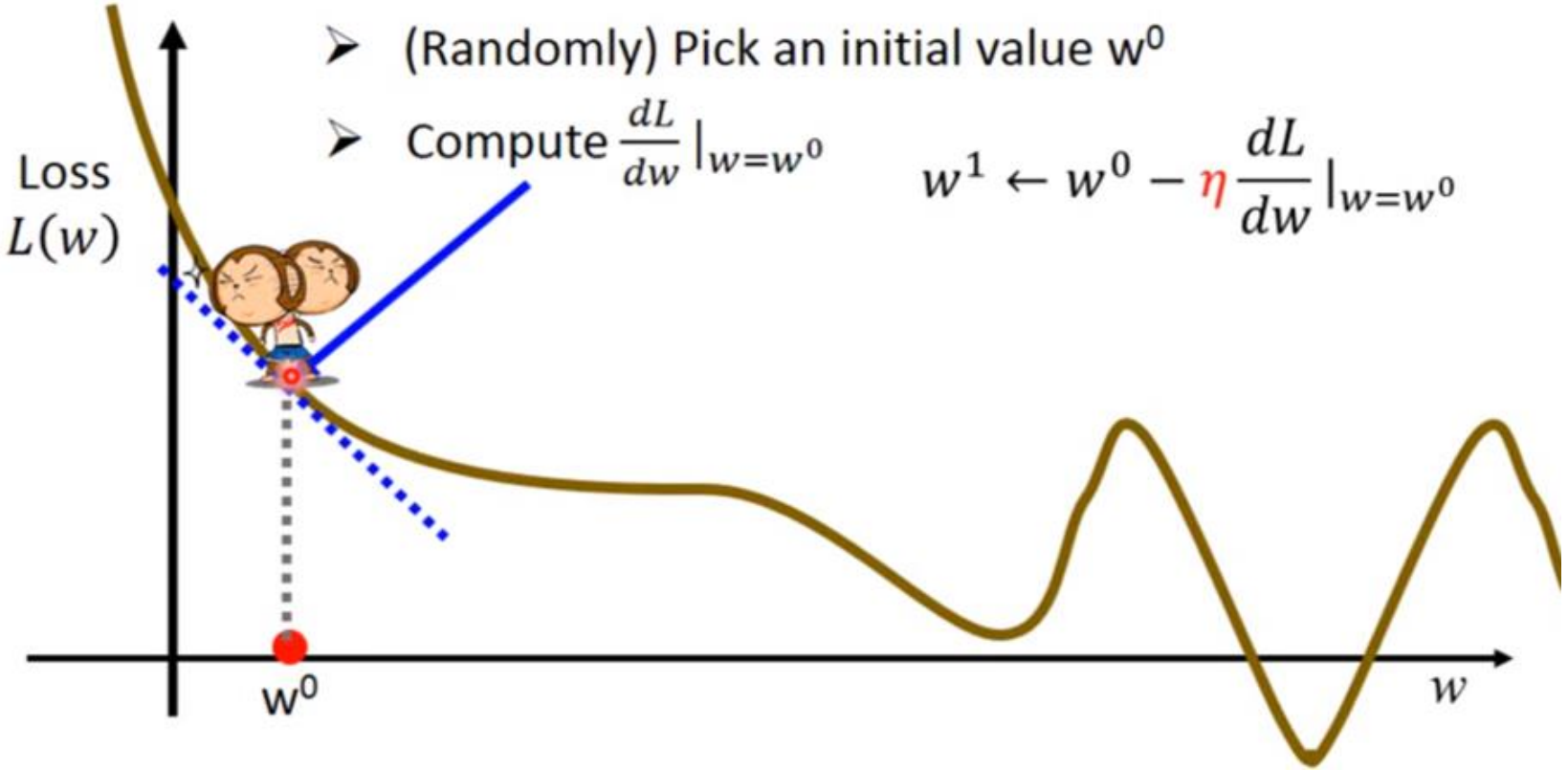
- **變形**
  - Adam（常用）
  - Batch Gradient Descent
  - Stochastic Gradient Descent (SGD)

Gradient Descent (梯度下降法)
Blind Man Goes Down the Mountain

$$w^* = arg \min_w L(w)$$

- Consider loss function $L(w)$ with one parameter w:
  - (Randomly) Pick an initial value w^0
  - Compute $\frac{dL}{dw}|_{w=w^0}$

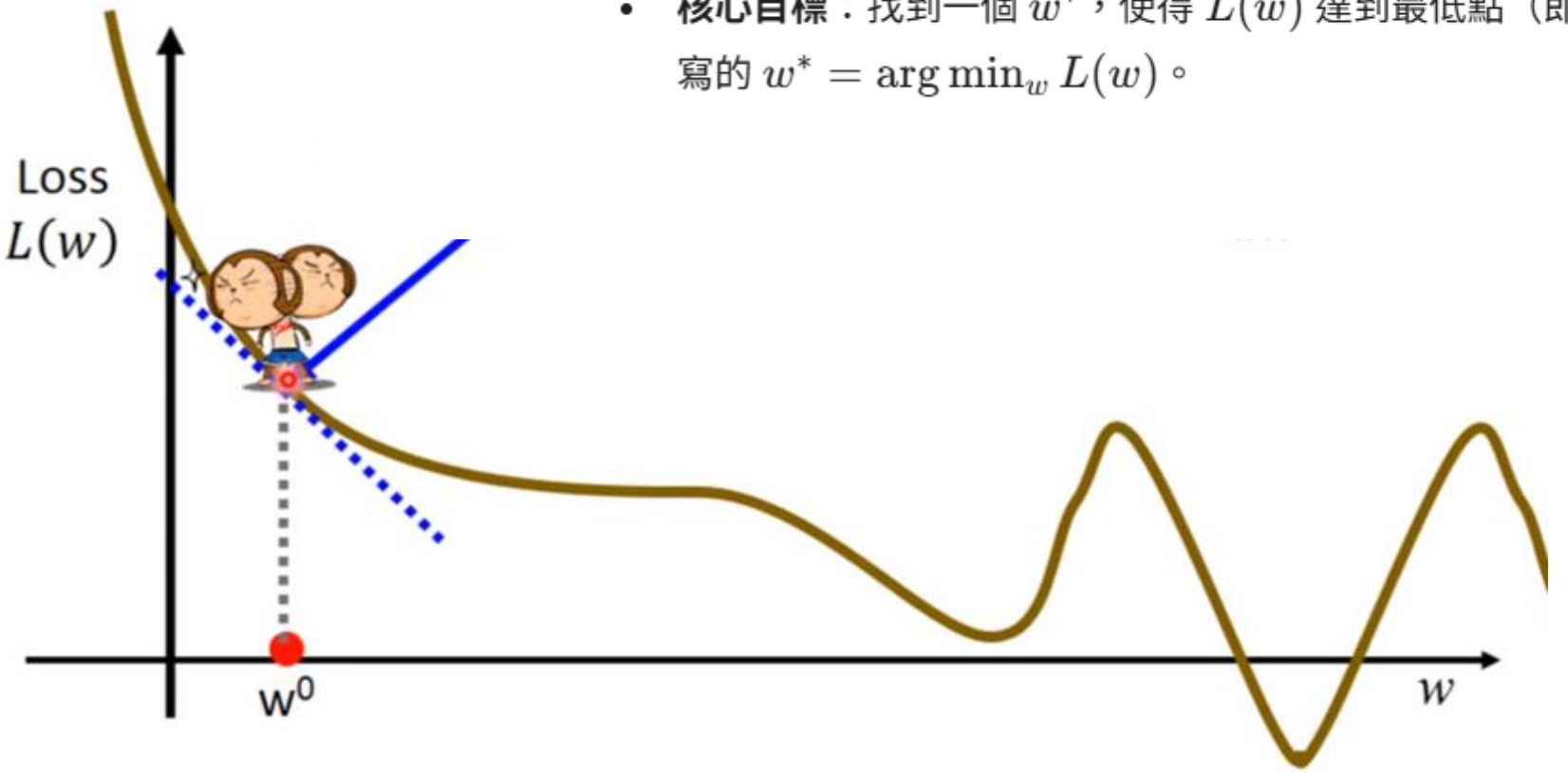
$$w^1 \leftarrow w^0 - \eta \frac{dL}{dw}|_{w=w^0}$$



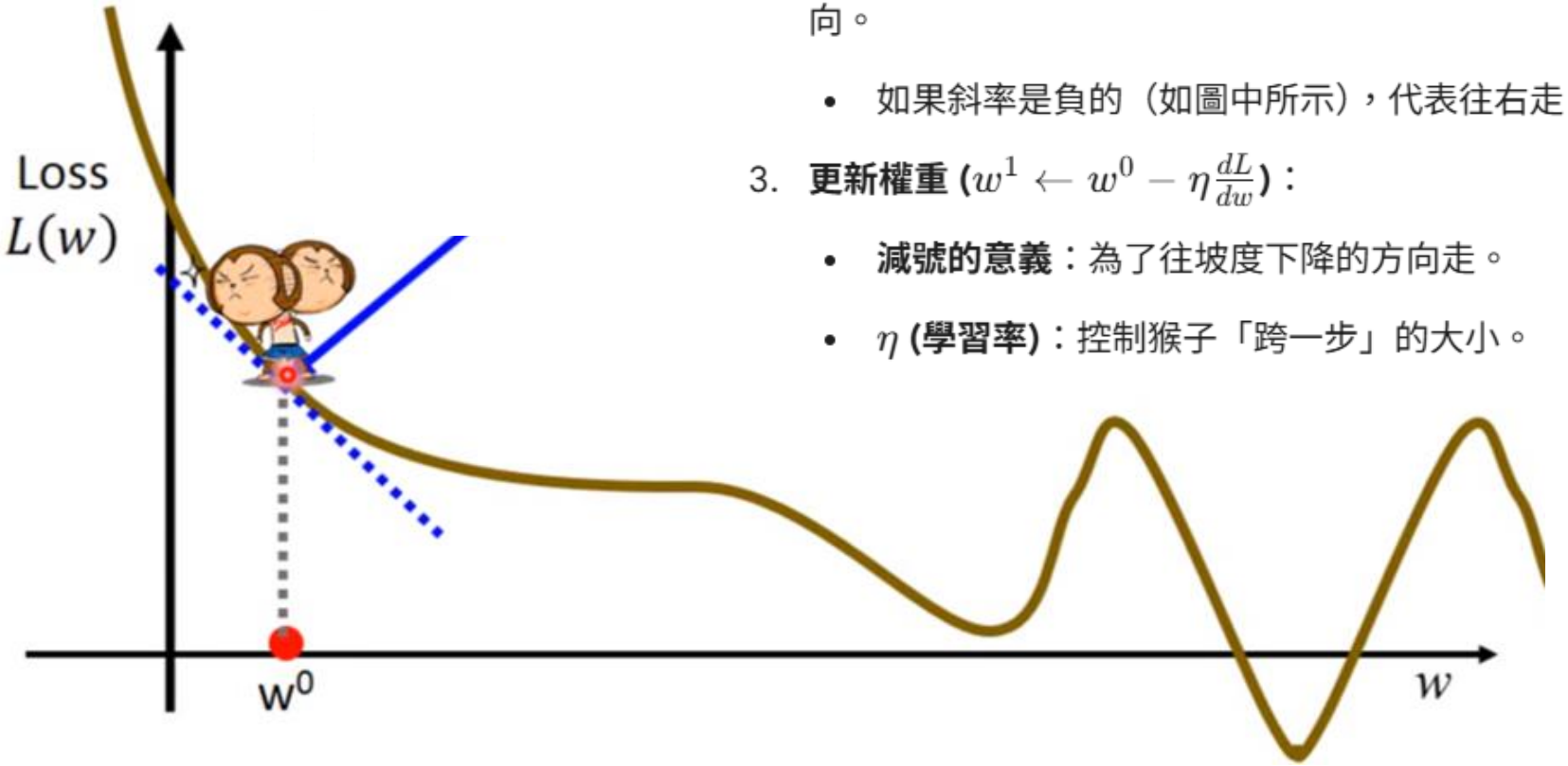
Gradient Descent (梯度下降法)
Blind Man Goes Down the Mountain

1. 場景設定：什麼是學習？

- **橫軸 (w)**：代表神經元的**權重**。這就是電腦可以調整的「旋鈕」。
- **縱軸 ($L(w)$)**：代表**損失函數（Loss）**，也就是錯誤率。
- **核心目標**：找到一個 w^*，使得 $L(w)$ 達到最低點（即圖中右側最深的谷底）。這就是公式所寫的 $w^* = \arg\min_w L(w)$。



Gradient Descent (梯度下降法)
Blind Man Goes Down the Mountain



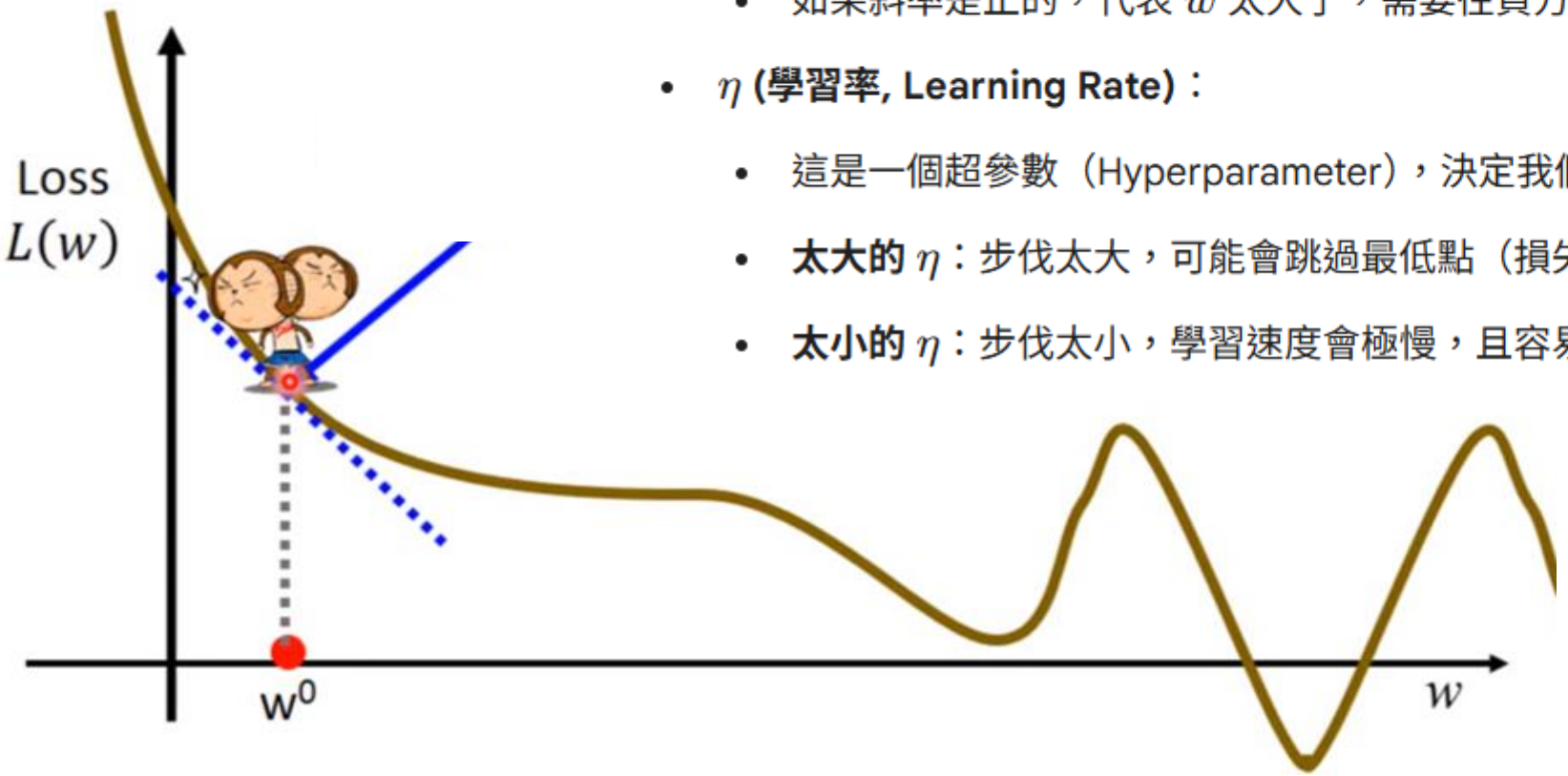
2. 步驟拆解：如何移動？

圖中的猴子代表「目前的模型狀態」，牠正試圖走到山谷：

1. **隨機初始化 (w^0)**： 一開始電腦什麼都不懂，所以隨機選一個權重起點（紅點 w^0）。
2. **計算梯度 ($\frac{dL}{dw}$)**： 在 w^0 這個位置畫一條**切線**（藍色虛線）。這條線的**斜率**告訴我們坡度的方向。
   - 如果斜率是負的（如圖中所示），代表往右走（增加 w）可以降低 Loss。
3. **更新權重 ($w^1 \leftarrow w^0 - \eta\frac{dL}{dw}$)**：
   - **減號的意義**：為了往坡度下降的方向走。
   - η **(學習率)**：控制猴子「跨一步」的大小。

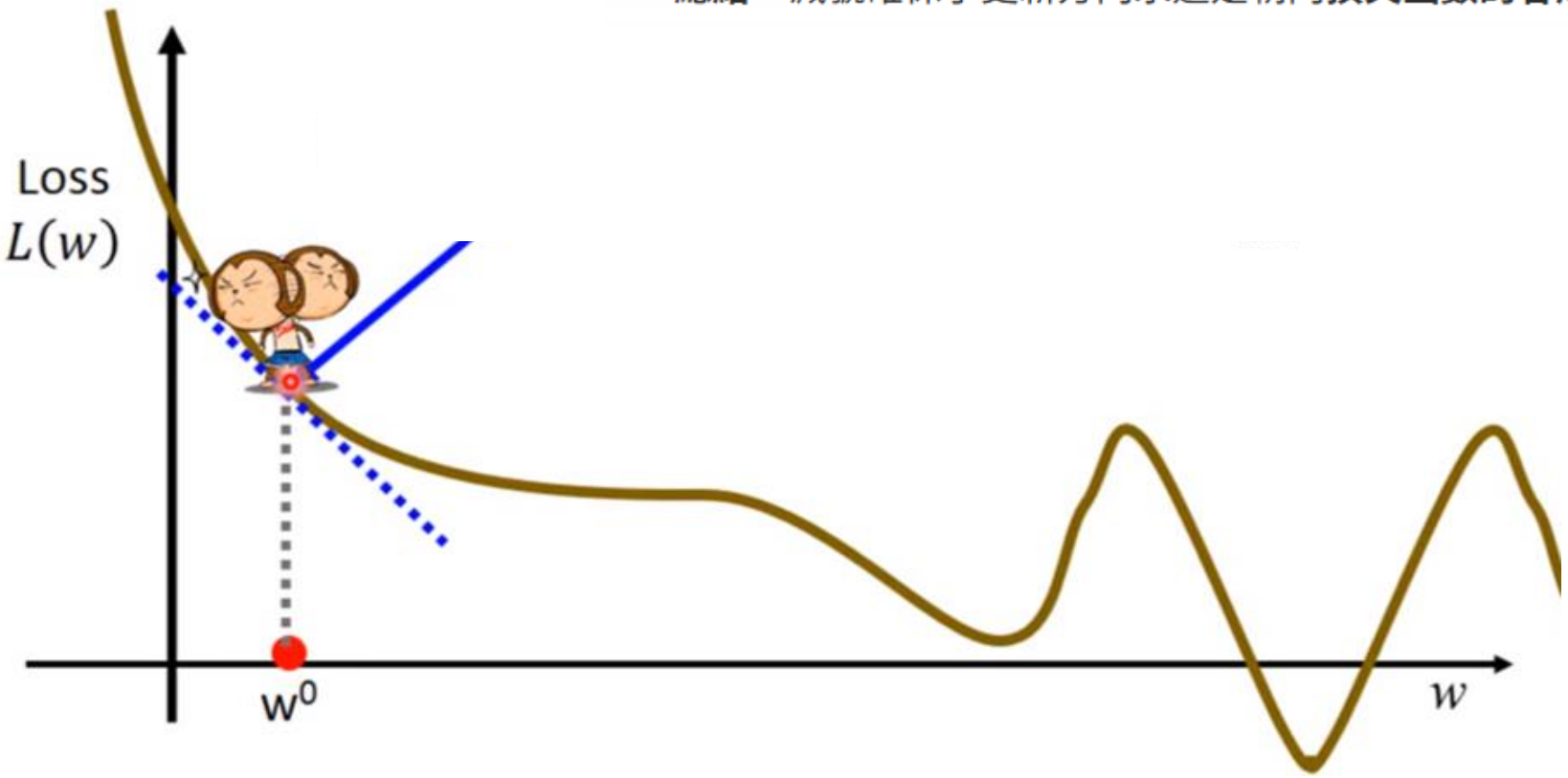
Gradient Descent (梯度下降法)
Blind Man Goes Down the Mountain

- $\frac{\partial L}{\partial w}$ **(梯度/斜率)**：
  - 這是在微積分中對權重求偏微分。
  - **直觀理解**：它告訴我們「如果權重 w 增加一點點，損失 L 會往哪個方向變化？」。
  - 如果斜率是正的，代表 w 太大了，需要往負方向調整。
- η **(學習率, Learning Rate)**：
  - 這是一個超參數（Hyperparameter），決定我們「跨步」的大小。
  - **太大的** η：步伐太大，可能會跳過最低點（損失函數的谷底），導致模型無法收斂。
  - **太小的** η：步伐太小，學習速度會極慢，且容易陷入局部最小值（Local Minimum）。



Gradient Descent (梯度下降法)
Blind Man Goes Down the Mountain

- 為何用減號?
- **原理**：如果斜率 $\frac{\partial L}{\partial w}$ 是正的（往右上斜），為了降低 L，我們必須讓 w 向左走（減小）；如果斜率是負的（往右下斜），則減去一個負數會變成加，讓 w 向右走。
- **總結**：減號確保了更新方向永遠是朝向**損失函數的谷底**前進。



陷入局部最佳解

- 陷入局部佳解
  - 圖中右側有好幾個坑洞。
  - 如果猴子的步伐太小，牠可能會掉進左邊比較淺的坑（局部最小值）
  - 沒辦法到達最右邊最深的地方（全域最小值 Global Minimum）。

- 學習率的重要性：
  - 學習率太小：猴子走得太慢，要下山得花一輩子。
  - 學習率太大：猴子可能會用力過猛，直接跳過山谷，翻到對面的山坡上，導致模型永遠無法收斂。

Introduction to Differential Calculus (微分簡介)

- 微分是在問：當 x 改變一點點時，y 會改變多少？
- 也就是「變化率」或「斜率」。

數學定義

$$\frac{dy}{dx} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

例 1：直線

$$y = 3x \quad \Rightarrow \quad \frac{dy}{dx} = 3$$

(x 每增加 1，y 就增加 3)

Basic Concepts of Differentiation (微分基礎概念)

- Second-degree Function (二次函式)

$$y = x^2$$

$$\frac{dy}{dx} = 2x$$

- First-degree Function (一次函式)

$$f(x) = x \qquad y = 3x$$

$$\frac{df}{dx} = 1 \qquad \frac{dy}{dx} = 3$$

- Constant Function (常數函式)

$$f(x) = 5$$

$$\frac{df}{dx} = 0$$

- Polynomial Function (多項式函式)

$$f(x) = 3x^2 + 2x + 7$$

$$\frac{df}{dx} = 6x + 2$$

Basic Concepts of Differentiation (微分基礎概念)

- Exponential Function (指數函數)

$$\frac{d}{dx}\,a^x = a^x \ln a \qquad (a > 0,\ a \neq 1)$$

- Logarithmic Function (對數函數)

$$\frac{d}{dx}\,\log_a(x) = \frac{1}{x \ln a} \qquad (a > 0,\ a \neq 1,\ x > 0)$$

- Natural Exponential Function
(自然指數函數)

$$f(x) = e^x$$

$$\frac{df}{dx} = e^x$$

- Natural Logarithmic Function
(自然對數函式)

$$f(x) = \ln x \qquad \ln(x) \text{ 是「以 } e \text{ 為底的對數」}$$

$$\frac{df}{dx} = \frac{1}{x} \qquad \ln(x) = \log_e(x)$$

$$e \approx 2.71828$$

Differentiation of Logarithm (對數的微分)

對於底數為 a ($a > 0$ 且 $a \neq 1$) 的對數函數 $\log_a(x)$，其微分公式為：

$$\frac{d}{dx}\log_a(x) = \frac{1}{x \ln a}$$

推導過程： 利用**換底公式**，可以將 $\log_a(x)$ 轉換為 $\frac{\ln x}{\ln a}$。因為 $\frac{1}{\ln a}$ 是常數，所以微分時只需對 $\ln x$ 微分：

$$\frac{d}{dx}\left(\frac{\ln x}{\ln a}\right) = \frac{1}{\ln a} \cdot \frac{d}{dx}(\ln x) = \frac{1}{\ln a} \cdot \frac{1}{x} = \frac{1}{x \ln a}$$

如果對數裡面的內容不是單純的 x，而是一個函數 $u(x)$，則需要使用連鎖律 (Chain Rule)：

- **自然對數：** $\frac{d}{dx}\ln(u) = \frac{1}{u} \cdot \frac{du}{dx}$
- **一般對數：** $\frac{d}{dx}\log_a(u) = \frac{1}{u \ln a} \cdot \frac{du}{dx}$

Introduction to Partial Differentiation (偏微分簡介)

- 式子內有多個變數，一次只看一個變數的影響，假設其他變數不會動

數學記號

若：

$$f(x,y) = x^2 + 3xy + y^2$$

- 對 x 偏微分：

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

- 對 y 偏微分：

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

對 x 偏微分（把 y 當常數）

$$\frac{\partial f}{\partial x} = 2x + 3y$$

對 y 偏微分（把 x 當常數）

$$\frac{\partial f}{\partial y} = 3x + 2y$$

Differentiation vs Partial Differentiation (微分 vs 偏微分)

- Differentiation (微分)
  - 變數數量：1個
  - 代表整體變化率

- Partial Differentiation (偏微分)
  - 變數數量：多個
  - 代表單一方向變化率

Chain Rule (連鎖律)

- $z = 2x$
- $y = z^2$

- x → z → y

- **Chain Rule**

$$\frac{dy}{dx} = \frac{dy}{dz} \cdot \frac{dz}{dx}$$

- $z = 2x$ → $\frac{dz}{dx} = 2$
- $y = z^2$ → $\frac{dy}{dz} = 2z$

- x → z → y

- **Chain Rule**

$$\frac{dy}{dx} = (2z) \cdot (2)$$

將 $z = 2x$ 代回：$\frac{dy}{dx} = 2 \cdot (2x) \cdot 2$

最終結果： $\frac{dy}{dx} = 8x$

Apply Chain Rule to Update Weights and Bias
(應用連鎖律於權重更新及偏移量更新)

1. 定義函數關係

- **線性單元 (Linear Unit)：** $z = wx + b \implies \frac{dz}{dx} = w$
- **激活函數 (Activation Function)：** 若 $f(z)$ 為 Sigmoid 函數，則其導數性質為：$\frac{d\hat{y}}{dz} = f(z)(1 - f(z))$ 因為 $\hat{y} = f(z)$，故可簡化為：$\hat{y}(1 - \hat{y})$

---

2. 變數流向圖

依賴路徑： $x \xrightarrow{\text{linear}} z \xrightarrow{\text{sigmoid}} \hat{y}$

---

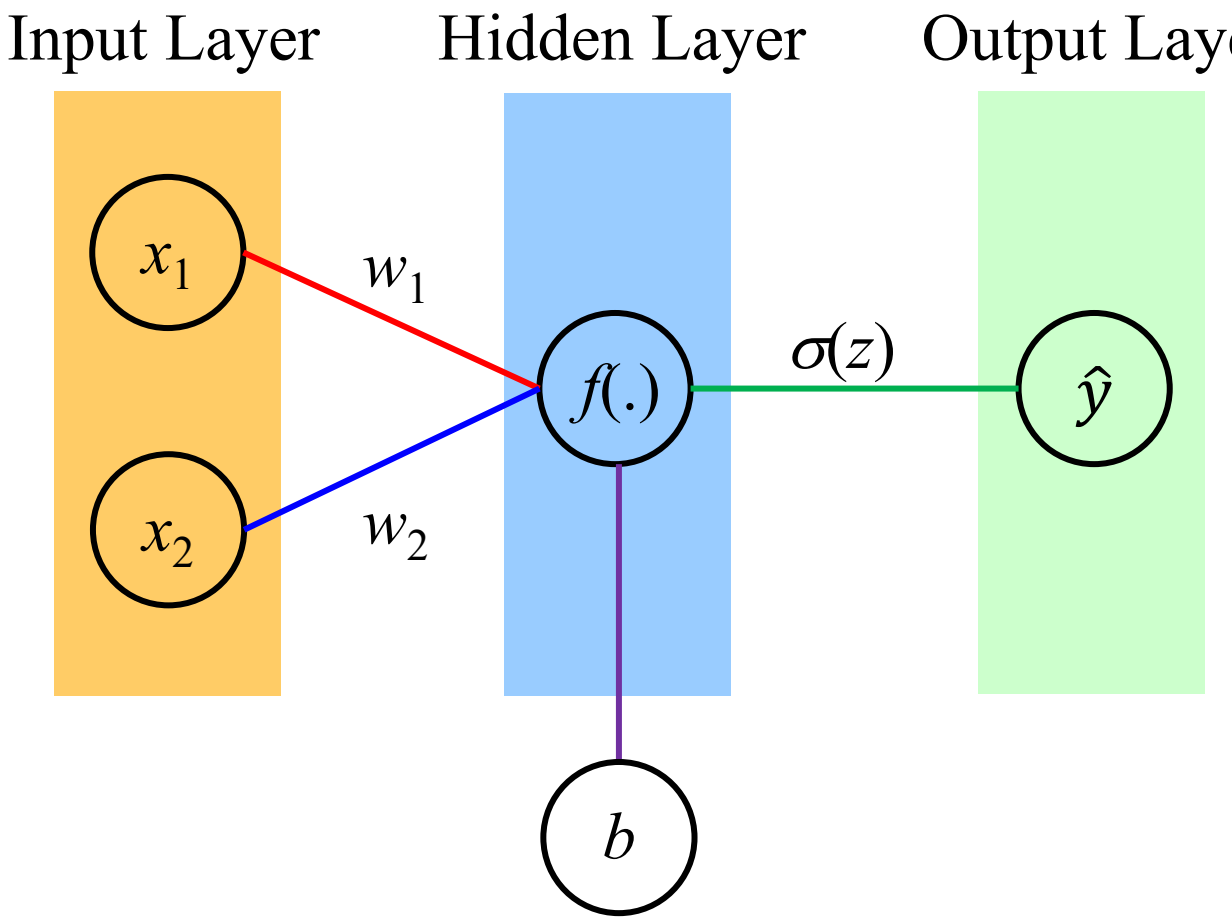
3. 應用連鎖律求解 $\frac{d\hat{y}}{dx}$

為了求出輸入值 x 對輸出值 $\hat{y}$ 的影響，我們將各段導數相乘：

$$\frac{d\hat{y}}{dx} = \frac{d\hat{y}}{dz} \cdot \frac{dz}{dx}$$

將 $\frac{d\hat{y}}{dz} = \hat{y}(1 - \hat{y})$ 與 $\frac{dz}{dx} = w$ 帶入：$\frac{d\hat{y}}{dx} = \hat{y}(1 - \hat{y}) \cdot w$

A Simple Example for Backpropagation



Update Weights w_1, w_2 and Bias b

Forward（前向傳播）

$$z = w_1 x_1 + w_2 x_2 + b$$

$$\hat{y} = \sigma(z) = \frac{1}{1 + e^{-z}}$$

Loss（Binary Cross Entropy）

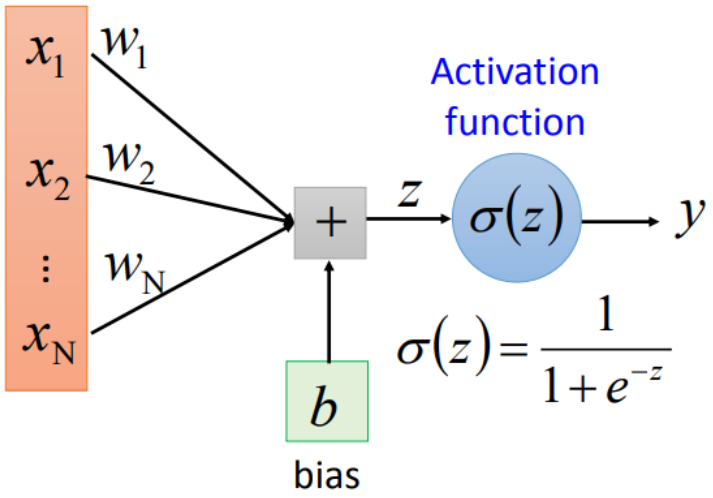
$$L = -\left[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})\right]$$

我們要更新：

- w_1, w_2, b

公式：

$$\theta := \theta - \eta \frac{\partial L}{\partial \theta}$$



連鎖律應用：Cross Entropy 與 Sigmoid 的梯度推導

1. 核心連鎖律公式

欲求出損失函數 L 對線性輸出 z 的變化率，公式如下：

$$\frac{\partial L}{\partial z} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z}$$

---

2. 各組成部分的導數

- **Sigmoid 激活函數的導數 (藍色部分):** 若 $\hat{y} = \sigma(z)$，則：

$$\frac{\partial \hat{y}}{\partial z} = \hat{y}(1 - \hat{y})$$

- **Cross Entropy 損失函數的導數 (紅色部分):** 對於二元交叉熵 $L = -(y \ln \hat{y} + (1 - y) \ln(1 - \hat{y}))$，其導數為：

$$\frac{\partial L}{\partial \hat{y}} = \frac{\hat{y} - y}{\hat{y}(1 - \hat{y})}$$

3. 最終梯度化簡 (綠色結果)

將上述兩者相乘，分母與分子項互相抵消：

$$\frac{\partial L}{\partial z} = \left(\frac{\hat{y} - y}{\hat{y}(1 - \hat{y})} \right) \cdot (\hat{y}(1 - \hat{y}))$$

結論： $\frac{\partial L}{\partial z} = \hat{y} - y$

Cross Entropy 與 Sigmoid 的梯度推導

- **BCE 損失函數 (L)**：針對單一樣本，其公式為：$L = -[y \ln(\hat{y}) + (1-y)\ln(1-\hat{y})]$。

我們要算的是 z（進入神經元的值）對 L（最後的錯誤）的影響，公式如下：

$$\frac{\partial L}{\partial z} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z}$$

第一步：對 L 求 $\hat{y}$ 的偏微分

$$\frac{\partial L}{\partial \hat{y}} = -\left(\frac{y}{\hat{y}} - \frac{1-y}{1-\hat{y}}\right) = \frac{\hat{y}-y}{\hat{y}(1-\hat{y})}$$

對權重 w_1

$$\frac{\partial L}{\partial w_1} = (\hat{y} - y)\, x_1$$

對權重 w_2

$$\frac{\partial L}{\partial w_2} = (\hat{y} - y)\, x_2$$

對偏置 b

$$\frac{\partial L}{\partial b} = (\hat{y} - y)$$

權重更新

$$w_1 := w_1 - \eta(\hat{y} - y)x_1$$

$$w_2 := w_2 - \eta(\hat{y} - y)x_2$$

偏置更新

$$b := b - \eta(\hat{y} - y)$$

1. 基本關係式回顧

首先，我們知道神經網路中的線性輸出 z 定義為：

$$z = w_1x_1 + w_2x_2 + b$$

2. 拆解連鎖律

若要求損失函數 L 對權重 w_1 的偏導數（梯度），連鎖律公式如下：

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial z} \cdot \frac{\partial z}{\partial w_1}$$

- **第一部分 $\frac{\partial L}{\partial z}$：** 根據您前一頁的推導，在 Cross Entropy + Sigmoid 的組合下，我們已經得出：

$$\frac{\partial L}{\partial z} = (\hat{y} - y)$$

- **第二部分 $\frac{\partial z}{\partial w}$：** 針對 $z = w_1x_1 + w_2x_2 + b$ 對各個參數求導：
  - 對 w_1 求導：$\frac{\partial z}{\partial w_1} = x_1$
  - 對 w_2 求導：$\frac{\partial z}{\partial w_2} = x_2$
  - 對 b 求導：$\frac{\partial z}{\partial b} = 1$

假設：

- $x_1 = 1,\ x_2 = 2$
- $w_1 = 0.1,\ w_2 = 0.2,\ b = 0$
- $y = 1,\ \eta = 0.1$

Forward

$$z = 0.1(1) + 0.2(2) = 0.5$$

$$\hat{y} = \sigma(0.5) \approx 0.62$$

梯度

$$\hat{y} - y = -0.38$$

更新

$$w_1 = 0.1 - 0.1(-0.38)(1) = 0.138$$

$$w_2 = 0.2 - 0.1(-0.38)(2) = 0.276$$

$$b = 0 - 0.1(-0.38) = 0.038$$

何謂梯度

- **梯度（Gradient）** 可以理解為「斜率」在高維空間中的推廣。

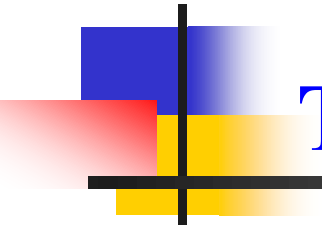
- **梯度的兩大特性：**
  - 方向：指向函數值增加最快的方向。
  - 大小（模）：代表在該方向上的變化率（坡度有多陡）。

梯度消失（Vanishing Gradient）

- 神經網路的訓練是基於「反向傳播」（Backpropagation）。當我們要更新網路前幾層（靠近輸入層）的參數時，必須從輸出層開始，一層一層地向前計算導數。根據微積分的連鎖律，這個過程本質上是**一系列導數的連乘**：
- 如果每一層的導數都小於 1，隨著網路層數（n）增加，這些小數連續相乘後的結果會以指數等級的速度萎縮。當網路達到數十層甚至上百層時，傳到前幾層的梯度就會變得微乎其微。

梯度消失（Vanishing Gradient）

- 早期的神經網路習慣使用 Sigmoid 或 Tanh 作為激活函數，它們是造成梯度消失的頭號戰犯：
  - Sigmoid 函數： 其導數最大值僅為 0.25（發生在輸入為 0 時）。這意味著每經過一層，梯度至少會衰減為原來的 1/4。
  - 飽和問題： 當神經元的輸入非常大或非常小時，Sigmoid 的曲線會變得非常平坦，其導數趨近於 0。一旦神經元進入這個「飽和區」，梯度就會消失，參數更新幾乎停滯。



Thank you!!