深度學習
Deep Learning

Backpropagation

Instructor: 林英嘉 (Ying-Jia Lin)
2025/09/17

Course GitHub

Slido # DL0917

Outline

- Gradient Descent Recap [20 min]
- Backpropagation [60 min]
- NumPy [35 min]
- 作業說明 [15 min]
- Quiz [20 min]

深度學習課程地圖

深度學習基礎

Week 2	多層感知機
Week 2	梯度下降
Week 3	反向傳播法
Week 4	最佳化方法 (Optimizers)

模型訓練技巧

圖神經網路

電腦視覺

- 卷積神經網路
- Vision Transformers

自然語言處理

- 循環神經網路
- Transformers
- Large Language Models

(深度) 強化學習

- Policy Gradients
- Actor-Critic
- PPO

[Recap] Minimize a Regression Model

- 假設我們今天要用 linear regression 來訓練一層的 MLP，模型輸出是 $\hat{y} = wx + b$
- 以均方誤差 (Mean Squared Error) 為例：

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \longleftarrow \text{模型輸出跟正確答案的平均差距}$$

把 $(wx_i + b)$ 代入 $\hat{y}_i$ $\longrightarrow$

$$\boxed{= \frac{1}{n}\sum_{i=1}^{n}\left(y_i - (wx_i + b)\right)^2}$$

- 其中:

訓練目標是讓這個公式在n筆訓練資料的平均差距越小越好

  - $\mathcal{L}$ 代表 Loss function；n 代表有 n 筆訓練資料
  - y_i 爲任一筆 ground-truth、$\hat{y}_i$ 爲任一筆 prediction (model output)

[Recap] Minimize a Regression Model

$$\frac{\partial \mathcal{L}}{\partial w} = \frac{1}{n}\sum_{i=1}^{n} 2\big(y_{\mathrm{i}} - (wx_{\mathrm{i}} + b)\big) \cdot (-x_{\mathrm{i}})$$

$$\frac{\partial \mathcal{L}}{\partial b} = \frac{1}{n}\sum_{i=1}^{n} 2\big(y_i - (wx_{\mathrm{i}} + b)\big) \cdot (-1)$$

更新w：

$$w_t = w_{t-1} - \eta \frac{\partial \mathcal{L}}{\partial w_{t-1}}$$

現在這個時間點的權重值

上一次的時間點的權重值

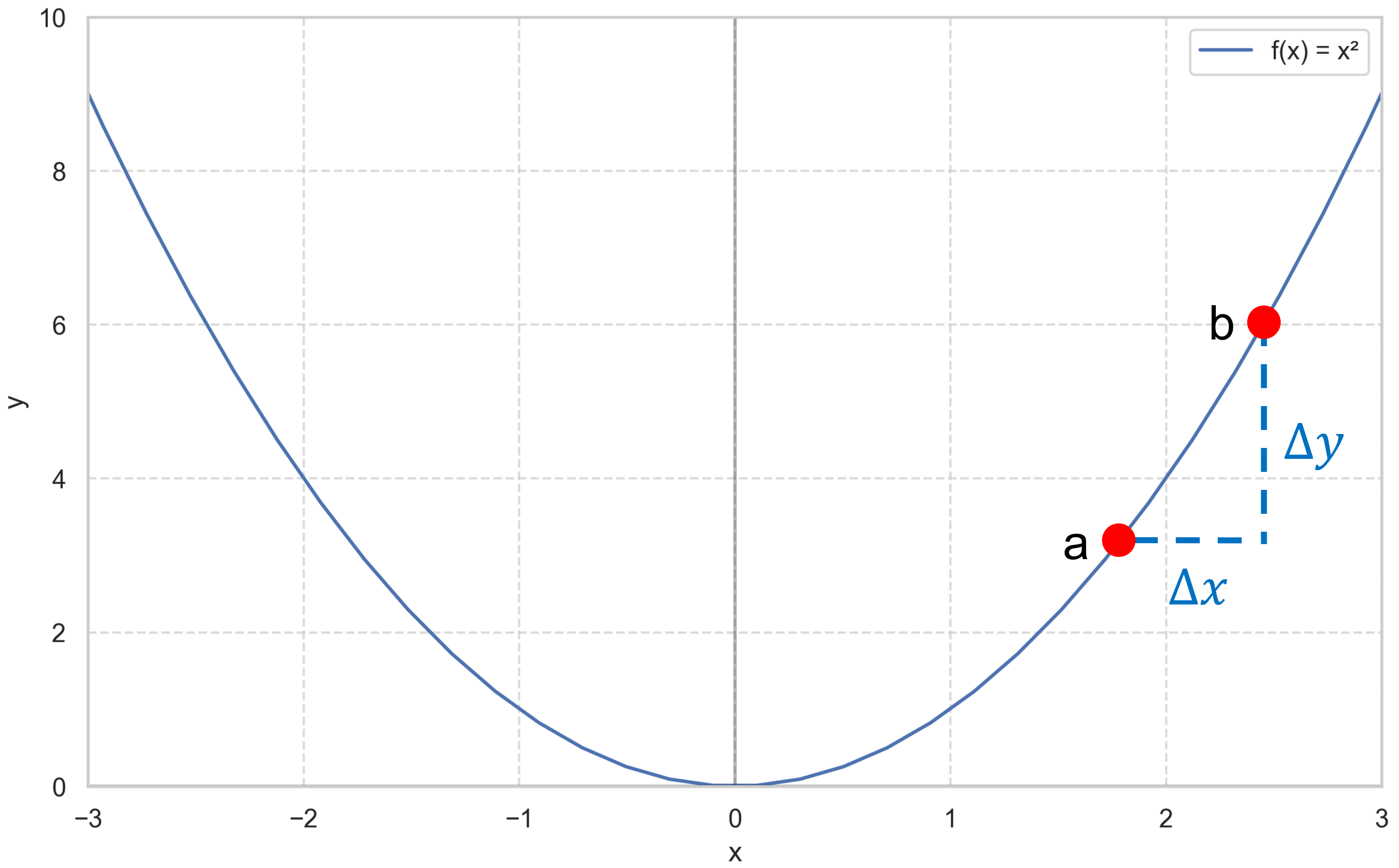
更新b：

$$b_t = b_{t-1} - \eta \frac{\partial \mathcal{L}}{\partial b_{t-1}}$$

現在這個時間點的偏置項

上一次的時間點的偏置項

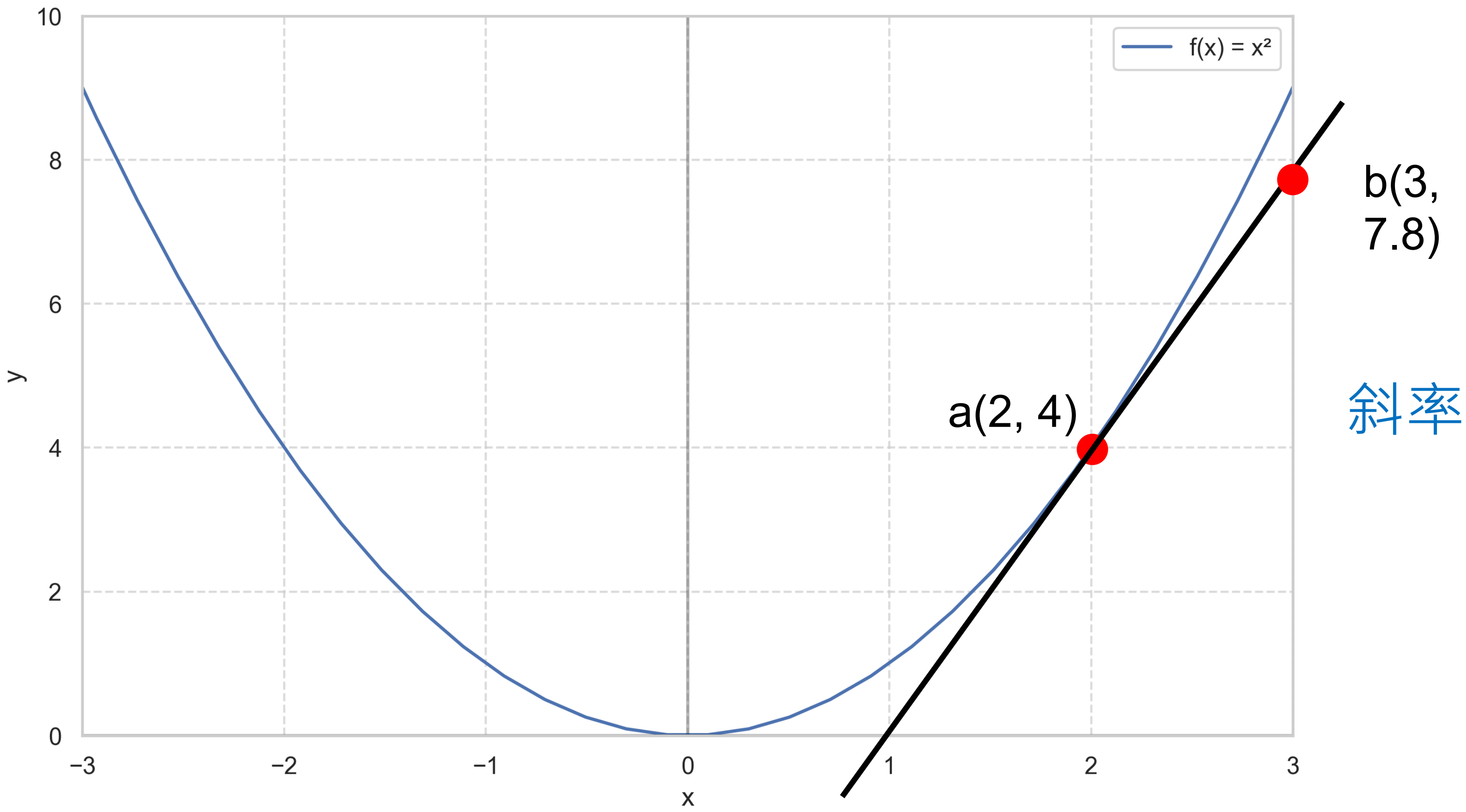
[Recap] 斜率 (slope)



範例曲線： $f(x) = x^2$

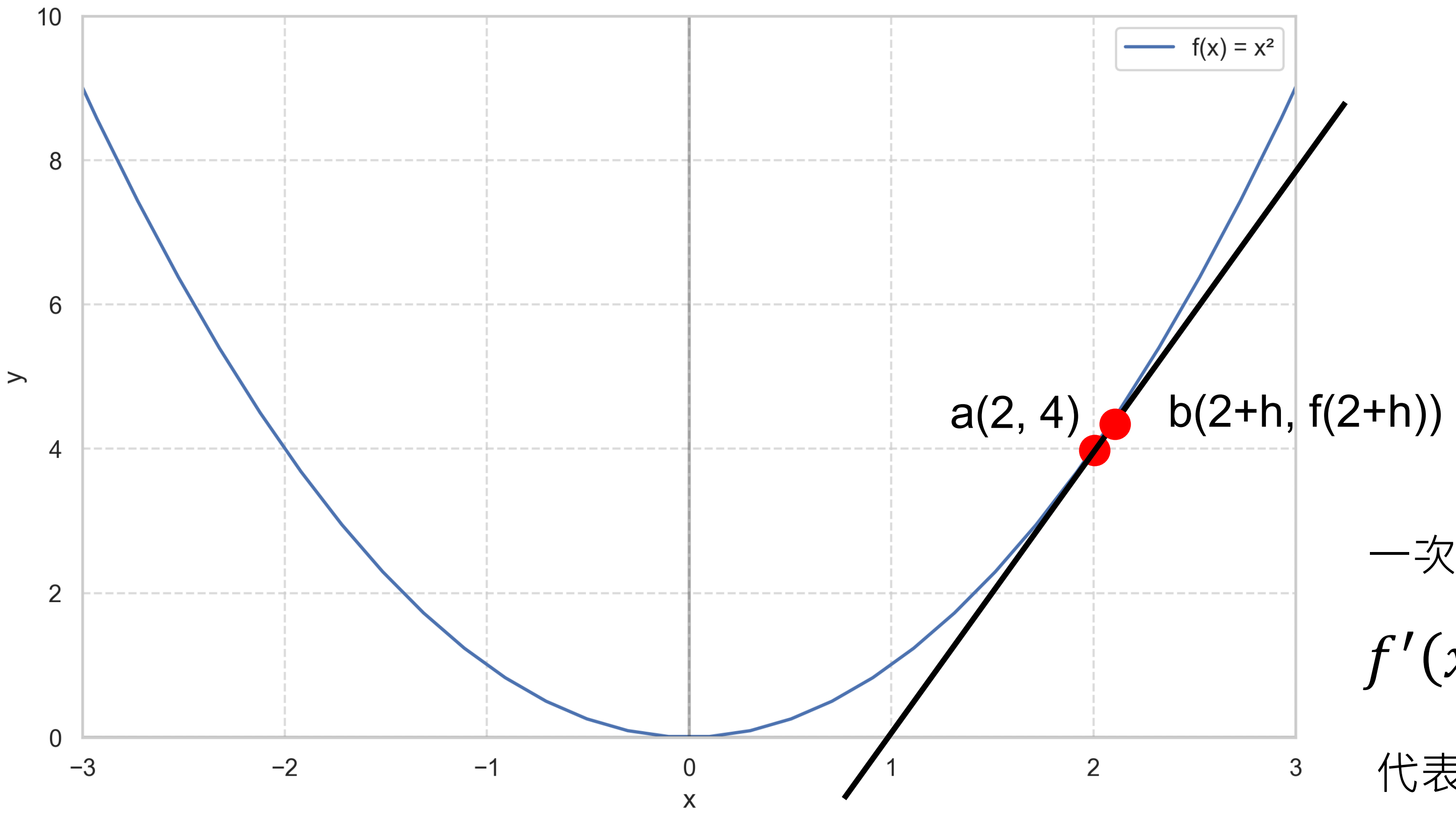
斜率$= \frac{\Delta y}{\Delta x}$

[Recap] 斜率 (slope)



$$斜率 = \frac{\Delta y}{\Delta x} = \frac{7.8-4}{3-2}$$

[Recap] 切線斜率 (slope)



$$斜率 = \frac{\Delta y}{\Delta x} = \frac{f(2+h)-4}{(2+h-2)\cong 0}$$

一次導函數：

$$f'(x) = \lim_{h\to 0} \frac{f(x+h)-f(x)}{h}$$

代表瞬時變化率 (h→0)

[Recap] 梯度 (gradients)

- 在單維度情況 (只有一個變數 x, Univariate)下，梯度就等同於一次導數：

$$\nabla_x f = \frac{df(x)}{dx} = f'(x)$$

切線斜率

- 在多維度情況 (有多個變數 x, Multivariate)下，梯度是所有偏導數的向量：

$$\nabla_{\mathrm{x}} f = \left[\frac{\partial f}{\partial x_1}, \quad \dots, \quad \frac{\partial f}{\partial x_n}\right]$$

切線斜率是梯度的分量

[Recap] Gradient Descent (梯度下降)

Assume x is a trainable parameter (weight), f is a differentiable function:

Gradient descent: $$x' = x - \eta \nabla_x f(x)$$

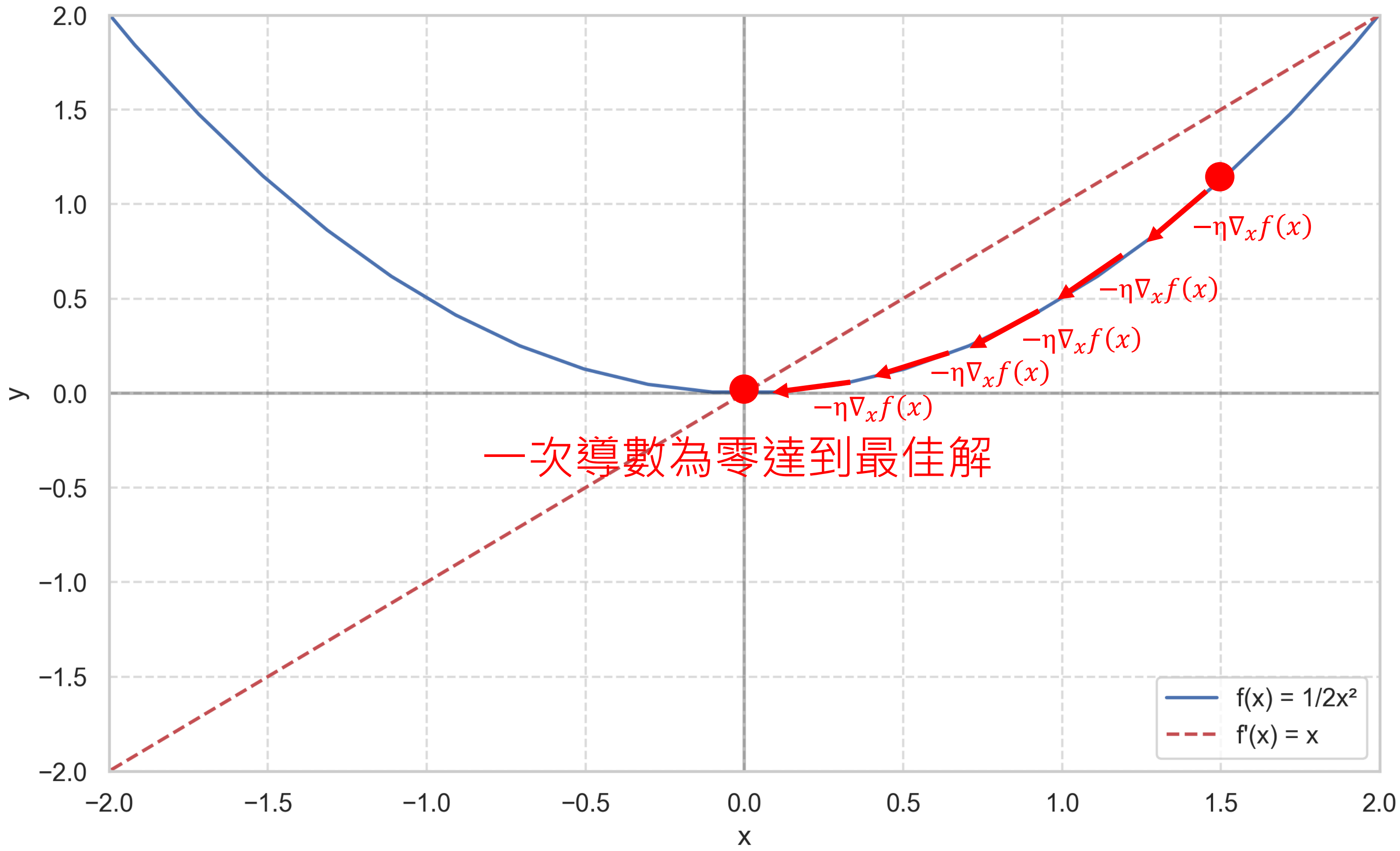
η is the learning rate (伊塔/欸塔) used for gradient descent.

- 調整每次更新參數時的**幅度**

[Recap] 梯度下降簡易範例

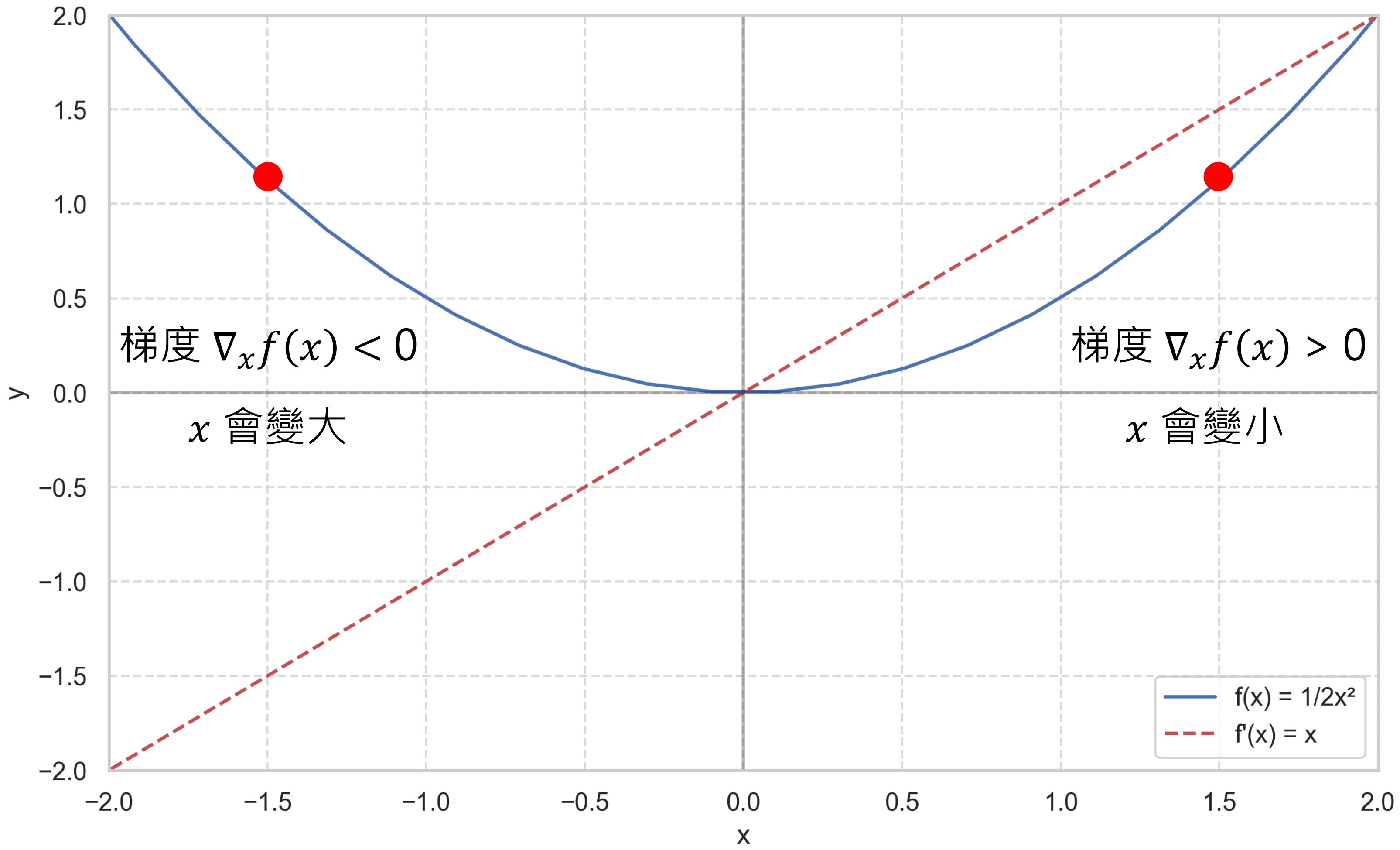
梯度下降公式：$x' = x - \eta\nabla_x f(x)$

- 使用 f(x) = 1/2x² 作為範例 (x同樣代表模型參數，為了方便理解，暫時不考慮輸入資料點)



以兩個點來觀察 Gradient Descent 的特性

$$x' = x - \eta\nabla_x f(x)$$



梯度特性

- 在單維度中：
  - 梯度 (即導數) 表示函數在該點的變化率（切線斜率）
  - 因為只有一個變數，變化方向只有正向或負向
  - 梯度下降核心概念：我們以梯度的反向來更新模型，以找到最小值
- 在多維度中：
  - 梯度是一個向量
    - (梯度的大小，也就是梯度的範數 $|\nabla_x f|$，表示函數在該點的變化率)
  - 梯度下降核心概念：我們以梯度的反向來更新模型，以找到最小值

Convergence and gradients

- 在經過很多個 steps (t 很大) 的更新之後，gradients 等於零的情況：

$$\nabla_{\theta_t}\mathcal{L}(\theta_t) = 0$$

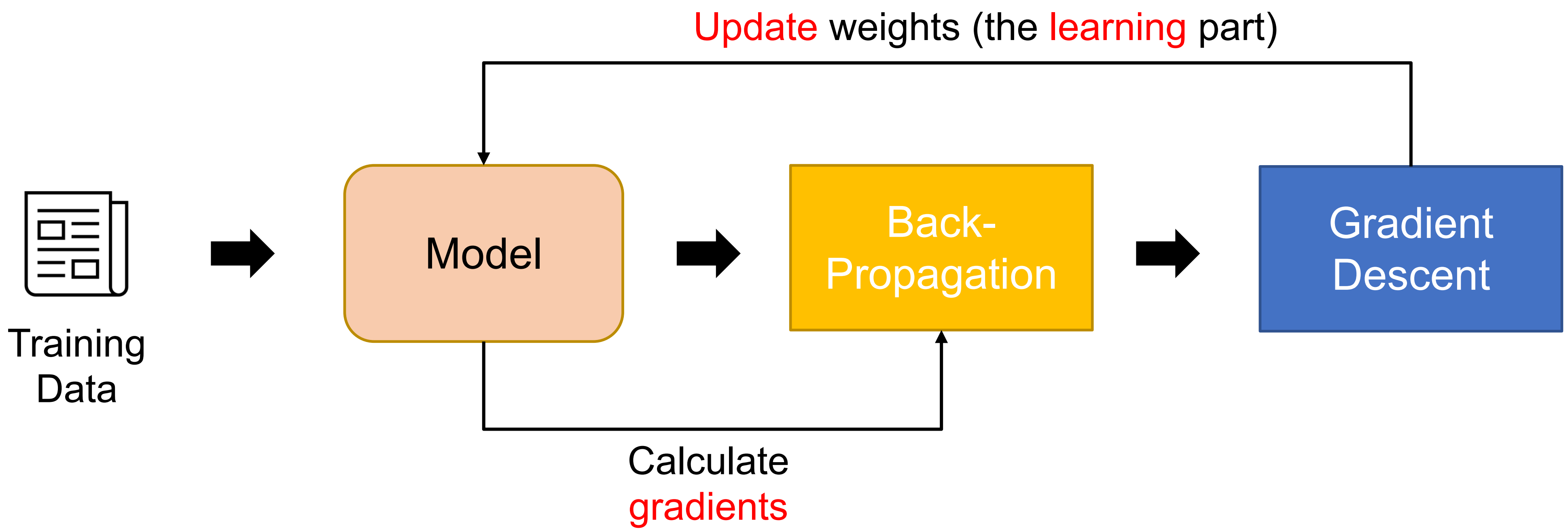
- $\mathcal{L}$: loss function
- θ_t: 在 t 個時間點 (step) 的的參數組合 (包含各層的w跟b)
- $\nabla_{\theta_t}\mathcal{L}(\theta_t)$: 梯度

此時代表模型可能已經達到目標函數的最佳解

Backpropagation

Training Process of a Deep Learning Model

- 深度學習模型被訓練的流程



What is Backpropagation? (BP)

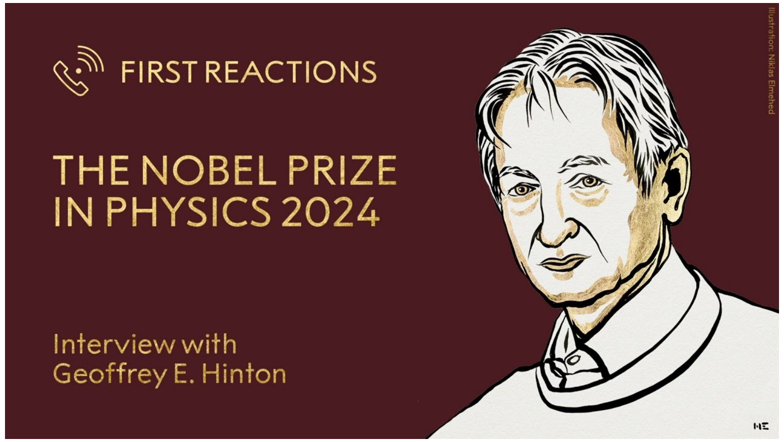
- Backpropagation is the application of the **chain rule** from calculus for computing the gradient.

Learning representations by back-propagating errors

David E. Rumelhart*, Geoffrey E. Hinton† & Ronald J. Williams*

* Institute for Cognitive Science, C-015, University of California, San Diego, La Jolla, California 92093, USA
† Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Philadelphia 15213, USA

We describe a new learning procedure, back-propagation, for networks of neurone-like units. The procedure repeatedly adjusts the weights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the net and the desired output vector. As a result of the weight adjustments, internal 'hidden' units which are not part of the input or output come to represent important features of the task domain, and the regularities in the task are captured by the interactions of these units. The ability to create useful new features distinguishes back-propagation from earlier, simpler methods such as the perceptron-convergence procedure[1].

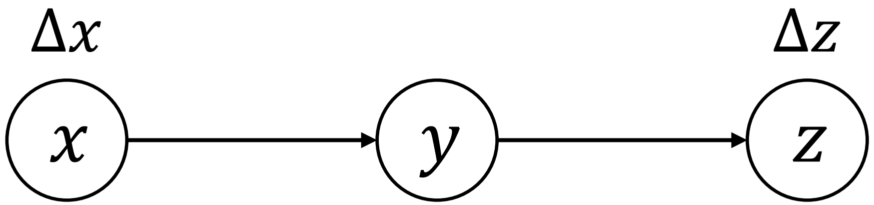


https://www.youtube.com/watch?v=-icD_KmvnnM

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors." nature 323.6088 (1986): 533-536.

Backpropagation 推導

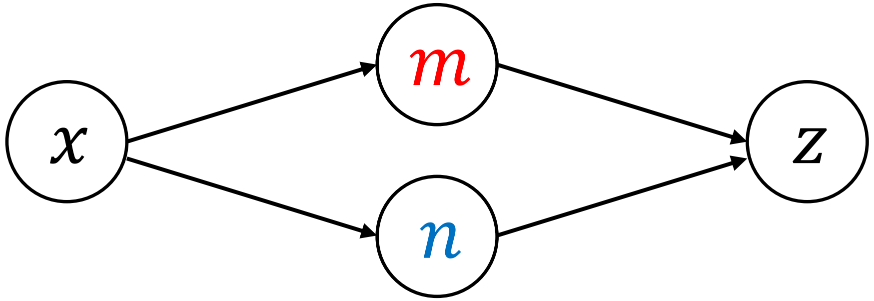
(Calculus) Chain Rule - 1



$$\frac{dz}{dx} = \underline{\frac{dz}{dy}}\,\underline{\frac{dy}{dx}}$$

- $\frac{dz}{dx}$ 代表 x 對 z 的影響，但 <u>x 的變化會影響到 y</u> ，接著 <u>y 影響到 z</u>

(Calculus) Chain Rule - 2

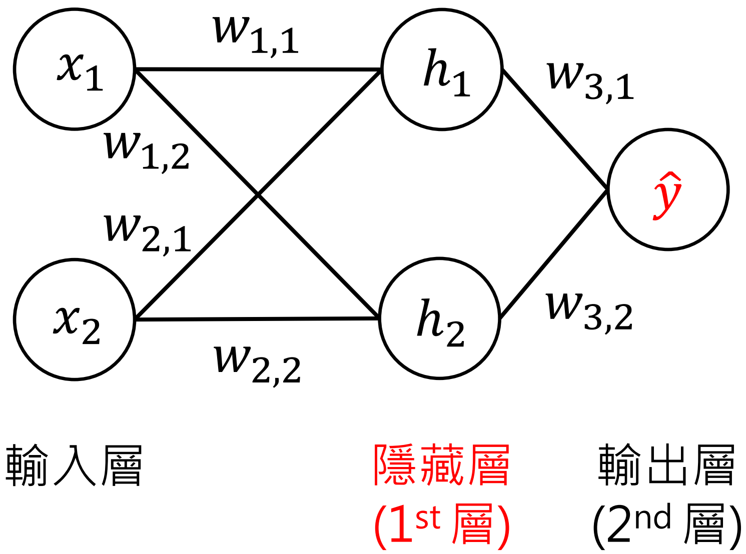


$$\frac{dz}{dx} = \frac{dz}{dm}\frac{dm}{dx} + \frac{dz}{dn}\frac{dn}{dx}$$

- $\frac{dz}{dx}$ 代表 x 對 z 的影響
    - 但 z 的變化由 m 和 n 共同影響

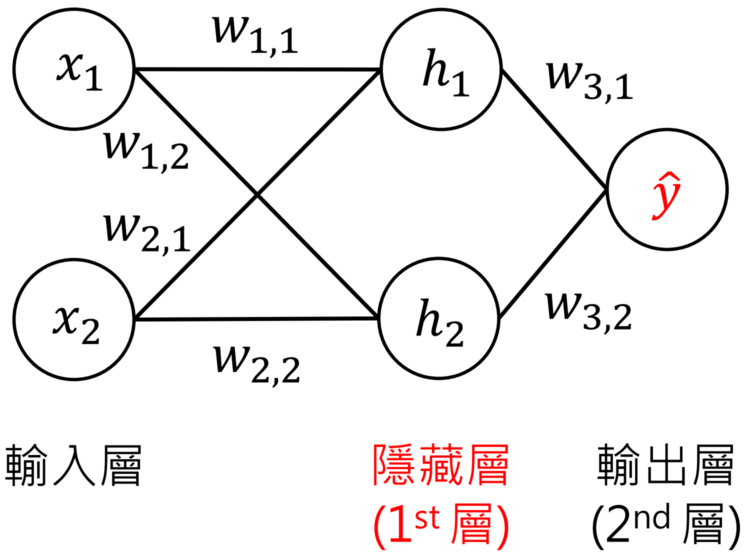
MLP 多層感知機 (今天的範例模型)

⬇️ A two-layer MLP (multi-layer perceptron)



MLP 多層感知機 (今天的範例模型)

⬇️ A two-layer MLP (multi-layer perceptron)



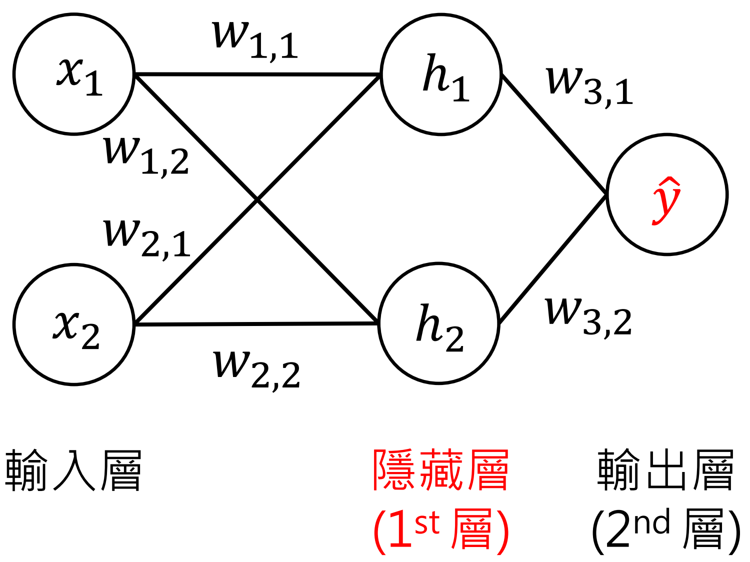
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

(教學用) 簡化 ⟶ $\mathcal{L} = \frac{1}{2}(y - \hat{y})^2$

- 現在只探討單一樣本，所以沒有1/n跟總和的部分
- 為了簡潔，這邊先乘上 1/2

Forward Pass (由輸入層到輸出層進行)

⬇️ A two-layer MLP (multi-layer perceptron)



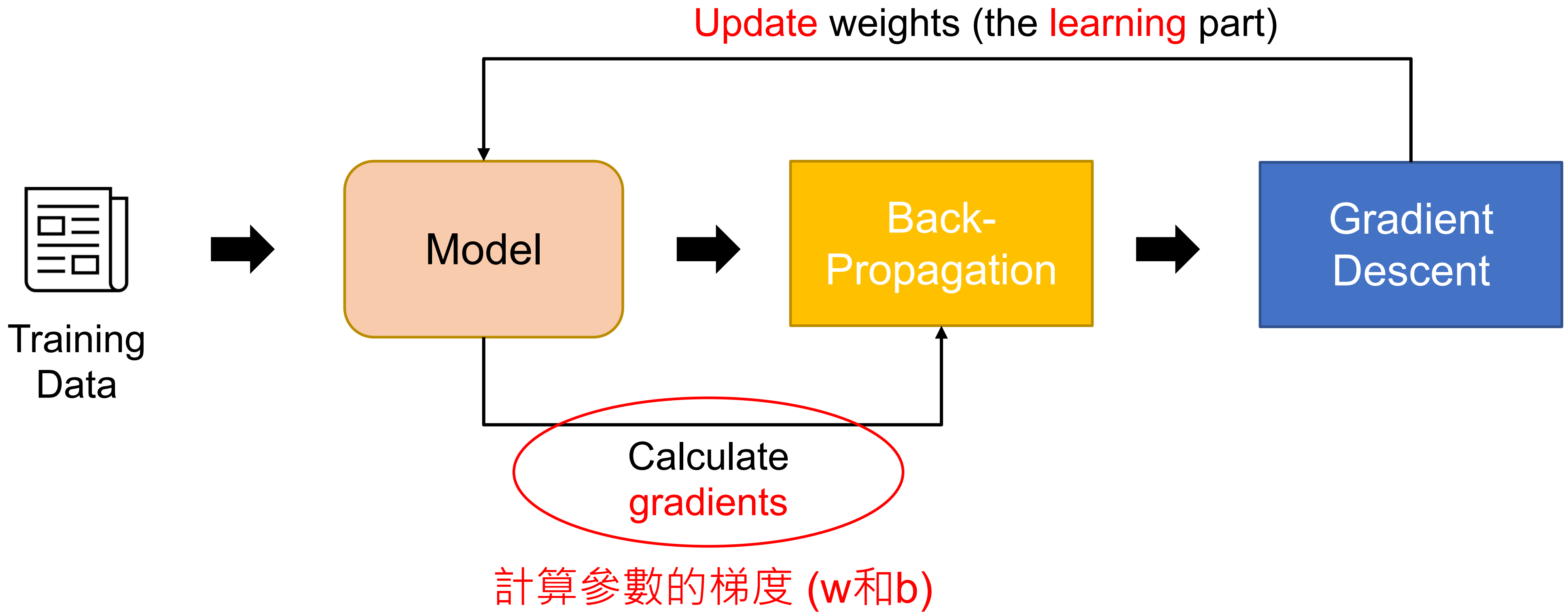
$$h_1 = w_{1,1}x_1 + w_{2,1}x_2 + b_1$$

$$h_2 = w_{1,2}x_1 + w_{2,2}x_2 + b_2$$

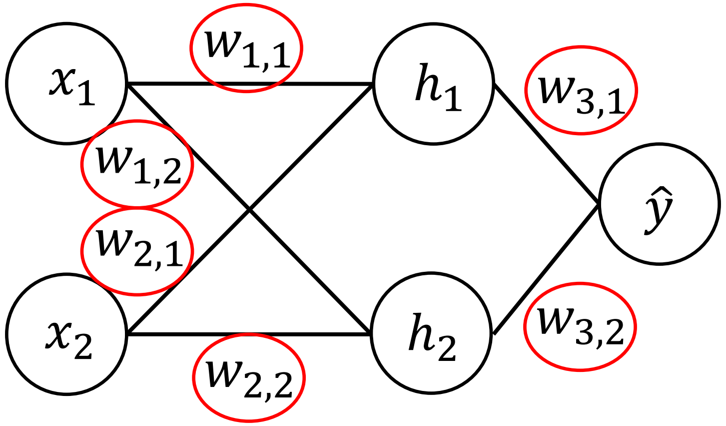
$$\hat{y} = w_{3,1}h_1 + w_{3,2}h_2 + b_3$$

Training Process of a Deep Learning Model

- 深度學習模型被訓練的流程



計算參數的梯度 (w和b)



$$h_1 = w_{1,1}x_1 + w_{2,1}x_2 + b_1$$

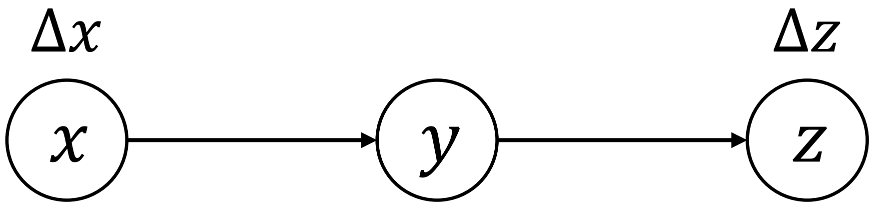
$$h_2 = w_{1,2}x_1 + w_{2,2}x_2 + b_2$$

$$\hat{y} = w_{3,1}h_1 + w_{3,2}h_2 + b_3$$

$$\mathcal{L}'(\mathbf{w},\mathbf{b}) = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial w_{1,1}} \\ \frac{\partial \mathcal{L}}{\partial w_{1,2}} \\ \vdots \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial b_3} \end{bmatrix} = \nabla_{\mathbf{w},\mathbf{b}}\mathcal{L}$$

為了簡化計算流程，後面的推導只會涵蓋 w 的部分

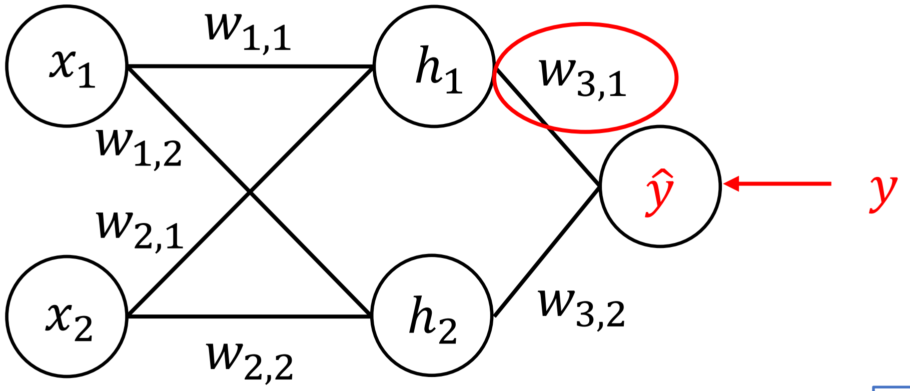
(Calculus) Chain Rule - 1



$$\frac{dz}{dx} = \underline{\frac{dz}{dy}}\,\underline{\frac{dy}{dx}}$$

- $\frac{dz}{dx}$ 代表 x 對 z 的影響，但 <u>x 的變化會影響到 y</u>，接著 <u>y 影響到 z</u>

Backpropagation (Layer: 2, $w_{3,1}$)



$$\frac{\partial \mathcal{L}}{\partial w_{3,1}} = \frac{\partial \hat{y}}{\partial w_{3,1}} \cdot \boxed{\frac{\partial \mathcal{L}}{\partial \hat{y}}}$$

grad_table

微分項	值
$\frac{\partial \mathcal{L}}{\partial \hat{y}}$	$-(y-\hat{y})$

$$\mathcal{L} = \frac{1}{2}(y-\hat{y})^2$$

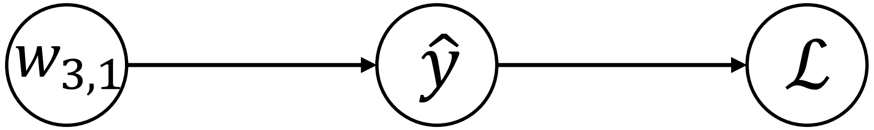
$\mathcal{L}$ 帶入

$$\frac{\partial \mathcal{L}}{\partial \hat{y}} = \frac{\partial[\frac{1}{2}(y-\hat{y})^2]}{\partial \hat{y}}$$

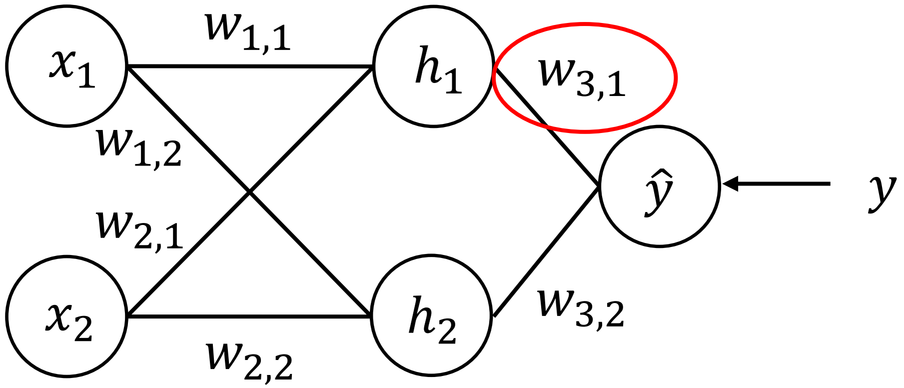
The generalized power rule
(廣義冪次規則)

$$= \frac{1}{2} \cdot 2(y-\hat{y}) \cdot (-1)$$

$$= -(y-\hat{y})$$



Backpropagation (Layer: 2, $w_{3,1}$)



$$\frac{\partial \mathcal{L}}{\partial w_{3,1}} = \boxed{\frac{\partial \hat{y}}{\partial w_{3,1}}} \cdot \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

grad_table

微分項	值
$\frac{\partial \mathcal{L}}{\partial \hat{y}}$	$-(y-\hat{y})$
$\frac{\partial \hat{y}}{\partial w_{3,1}}$	h_1

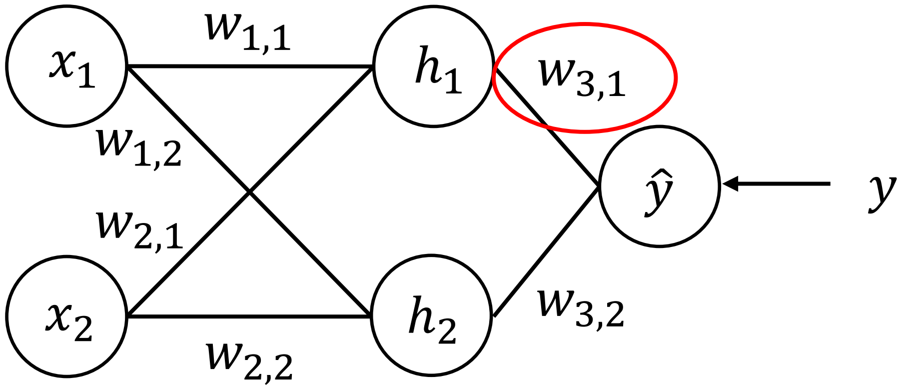
$\hat{y}$ 帶入

$$\hat{y} = w_{3,1}h_1 + w_{3,2}h_2 + b_3$$

$$\frac{\partial \hat{y}}{\partial w_{3,1}} = \frac{\partial [w_{3,1}h_1 + w_{3,2}h_2 + b_3]}{\partial w_{3,1}}$$

$$= h_1$$

Backpropagation (Layer: 2, $w_{3,1}$)



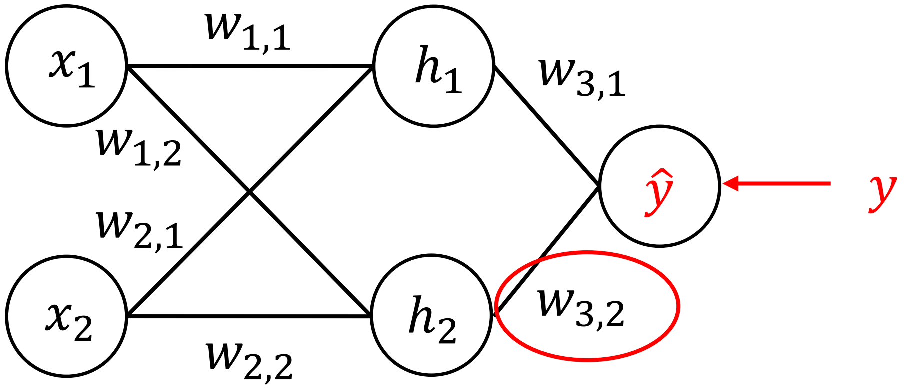
$$\frac{\partial \mathcal{L}}{\partial w_{3,1}} = \frac{\partial \hat{y}}{\partial w_{3,1}} \cdot \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

$$= h_1 \cdot -(y - \hat{y})$$

grad_table

微分項	值
$\frac{\partial \mathcal{L}}{\partial \hat{y}}$	$-(y - \hat{y})$
$\frac{\partial \hat{y}}{\partial w_{3,1}}$	h_1

Backpropagation (Layer: 2, $w_{3,2}$)



$$\frac{\partial \mathcal{L}}{\partial w_{3,2}} = \frac{\partial \hat{y}}{\partial w_{3,2}} \cdot \boxed{\frac{\partial \mathcal{L}}{\partial \hat{y}}}$$

grad_table

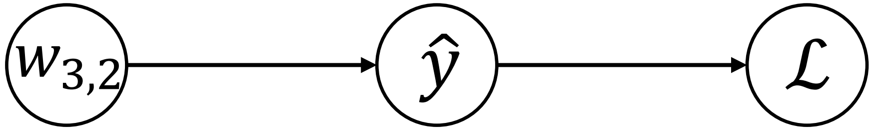
微分項	值
$\boxed{\frac{\partial \mathcal{L}}{\partial \hat{y}}}$	$-(y-\hat{y})$
$\frac{\partial \hat{y}}{\partial w_{3,1}}$	h_1

$$\mathcal{L} = \frac{1}{2}(y-\hat{y})^2$$

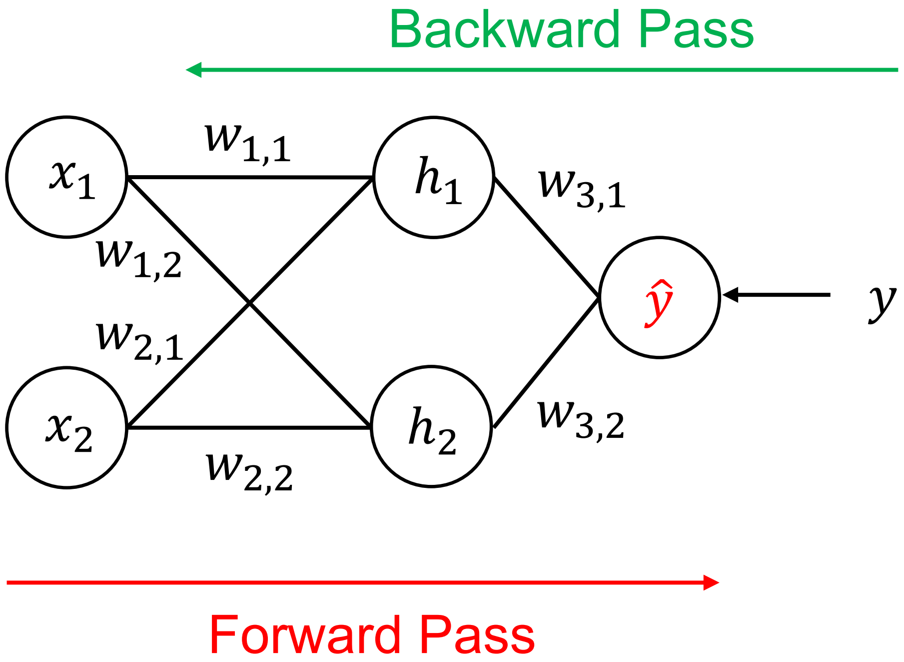
因為 $\hat{y} = w_{3,1}h_1 + w_{3,2}h_2 + b_3$

所以 $\frac{\partial \hat{y}}{\partial w_{3,2}} = h_2$

因此 $\frac{\partial \mathcal{L}}{\partial w_{3,2}} = h_2 \cdot -(y-\hat{y})$



反向傳播法觀察 (1)



$$\hat{y} = w_{3,1}h_1 + w_{3,2}h_2 + b_3$$

$$\frac{\partial \mathcal{L}}{\partial w_{3,1}} = \boxed{\frac{\partial \hat{y}}{\partial w_{3,1}}} \cdot \boxed{\frac{\partial \mathcal{L}}{\partial \hat{y}}}$$

$$\frac{\partial \mathcal{L}}{\partial w_{3,2}} = \boxed{\frac{\partial \hat{y}}{\partial w_{3,2}}} \cdot \boxed{\frac{\partial \mathcal{L}}{\partial \hat{y}}}$$

$$\frac{\partial \mathcal{L}}{\partial \hat{y}} = \frac{\partial [\frac{1}{2}(y - \hat{y})^2]}{\partial \hat{y}}$$

grad_table

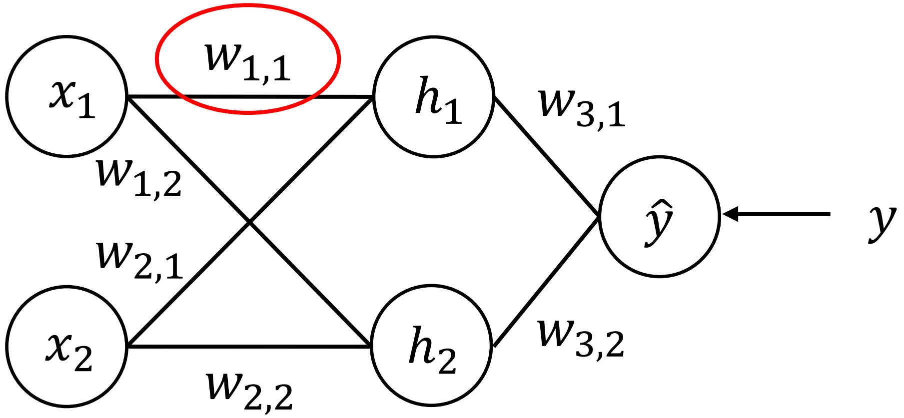
微分項	值
$\frac{\partial \mathcal{L}}{\partial \hat{y}}$	$-(y - \hat{y})$
$\frac{\partial \hat{y}}{\partial w_{3,1}}$	h_1

💡反向傳播法是在算loss對參數的梯度
= Forward Pass 結果的偏微分 x Backward Pass 結果的偏微分

傳遞過程定義

- Forward Pass:
  - [定義] 將輸入資料經由神經網路的層層運算，產生最終輸出
  - [重要功能] 記錄所有中間計算結果，供反向傳播計算梯度使用
- Backward Pass:
  - [定義] 使用 Chain rule，將 loss 對參數的梯度從輸出層由後往前計算

Backpropagation (Layer: 1, $w_{1,1}$)



$$\frac{\partial \mathcal{L}}{\partial w_{1,1}} = \frac{\partial h_1}{\partial w_{1,1}} \cdot \frac{\partial \mathcal{L}}{\partial h_1}$$

$$= \frac{\partial h_1}{\partial w_{1,1}} \cdot \boxed{\frac{\partial \hat{y}}{\partial h_1}} \cdot \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

grad_table

微分項	值
$\frac{\partial \mathcal{L}}{\partial \hat{y}}$	$-(y - \hat{y})$
$\frac{\partial \hat{y}}{\partial w_{3,1}}$	h_1
$\frac{\partial \hat{y}}{\partial h_1}$	$w_{3,1}$

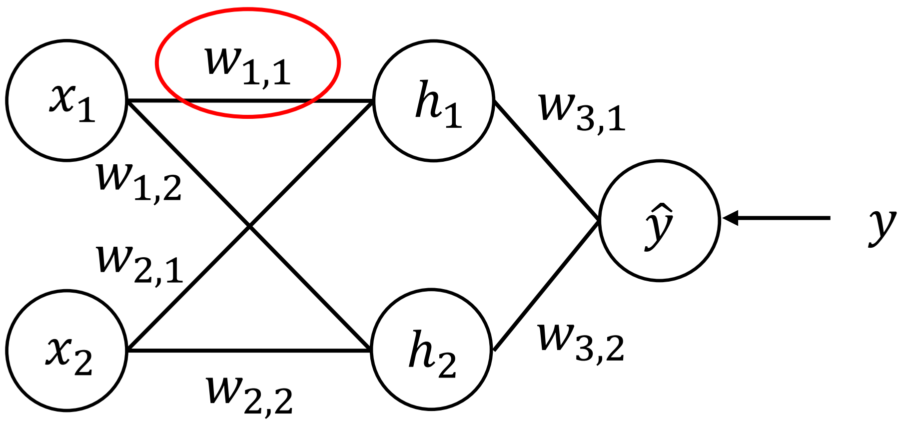
$$\hat{y} = w_{3,1}h_1 + w_{3,2}h_2 + b_3$$

$$\frac{\partial \hat{y}}{\partial h_1} = \frac{\partial [w_{3,1}h_1 + w_{3,2}h_2 + b_3]}{\partial h_1}$$

$$= w_{3,1}$$

$$w_{1,1} \rightarrow h_1 \rightarrow \hat{y} \rightarrow \mathcal{L}$$

Backpropagation (Layer: 1, $w_{1,1}$)



$$\frac{\partial \mathcal{L}}{\partial w_{1,1}} = \frac{\partial h_1}{\partial w_{1,1}} \cdot \frac{\partial \mathcal{L}}{\partial z_1}$$

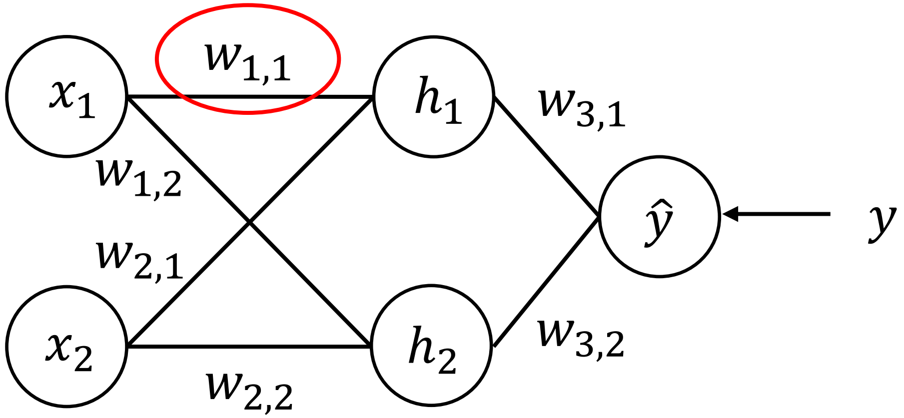
$$= \frac{\partial h_1}{\partial w_{1,1}} \cdot \frac{\partial \hat{y}}{\partial h_1} \cdot \boxed{\frac{\partial \mathcal{L}}{\partial \hat{y}}}$$

`grad_table`

微分項	值
$\boxed{\frac{\partial \mathcal{L}}{\partial \hat{y}}}$	$-(y-\hat{y})$
$\frac{\partial \hat{y}}{\partial w_{3,1}}$	h_1
$\frac{\partial \hat{y}}{\partial h_1}$	$w_{3,1}$

已經算過了，直接取記錄過的值就可以了！

Backpropagation (Layer: 1, $w_{1,1}$)



$$\frac{\partial \mathcal{L}}{\partial w_{1,1}} = \frac{\partial h_1}{\partial w_{1,1}} \cdot \frac{\partial \mathcal{L}}{\partial h_1}$$

$$= \boxed{\frac{\partial h_1}{\partial w_{1,1}}} \cdot \frac{\partial \hat{y}}{\partial h_1} \cdot \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

grad_table

微分項	值
$\frac{\partial \mathcal{L}}{\partial \hat{y}}$	$-(y-\hat{y})$
$\frac{\partial \hat{y}}{\partial w_{3,1}}$	h_1
$\frac{\partial \hat{y}}{\partial h_1}$	$w_{3,1}$
$\frac{\partial h_1}{\partial w_{1,1}}$	x_1

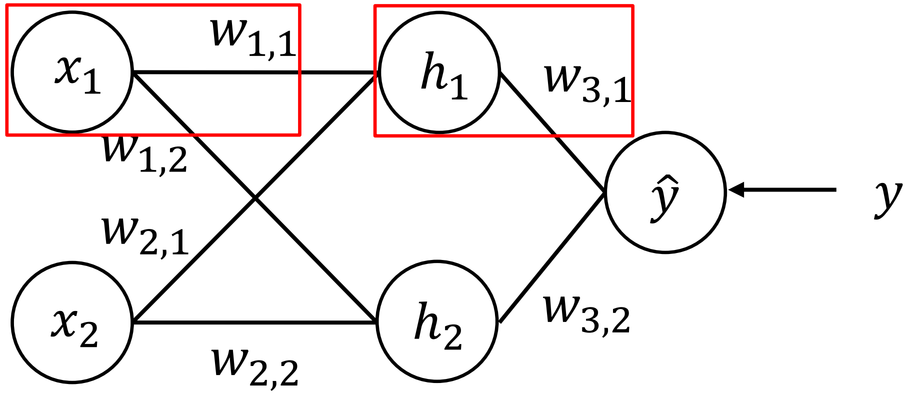
$$h_1 = w_{1,1}x_1 + w_{2,1}x_2 + b_1$$

$$\frac{\partial h_1}{\partial w_{1,1}} = \frac{\partial (w_{1,1}x_1 + w_{2,1}x_2 + b_1)}{\partial w_{1,1}} = x_1$$

因此

$$\frac{\partial \mathcal{L}}{\partial w_{1,1}} = x_1 \cdot -(y-\hat{y}) \cdot w_{3,1}$$

反向傳播法觀察 (2): Forward Pass



微分項	值
$\frac{\partial \mathcal{L}}{\partial \hat{y}}$	$-(y-\hat{y})$
$\frac{\partial \hat{y}}{\partial w_{3,1}}$	h_1
$\frac{\partial \hat{y}}{\partial h_1}$	$w_{3,1}$
$\frac{\partial h_1}{\partial w_{1,1}}$	x_1

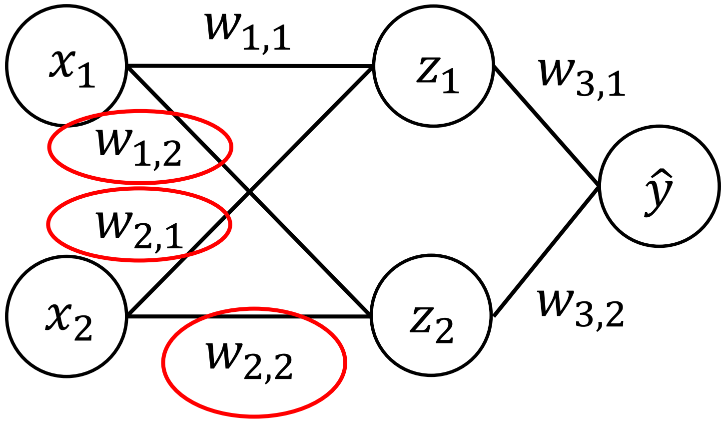
$$\hat{y} = w_{3,1}h_1 + w_{3,2}h_2 + b_3$$

$$h_1 = w_{1,1}x_1 + w_{2,1}x_2 + b_1$$

$$h_2 = w_{1,2}x_1 + w_{2,2}x_2 + b_2$$

💡Forward Pass 結果的偏微分等於輸入的值

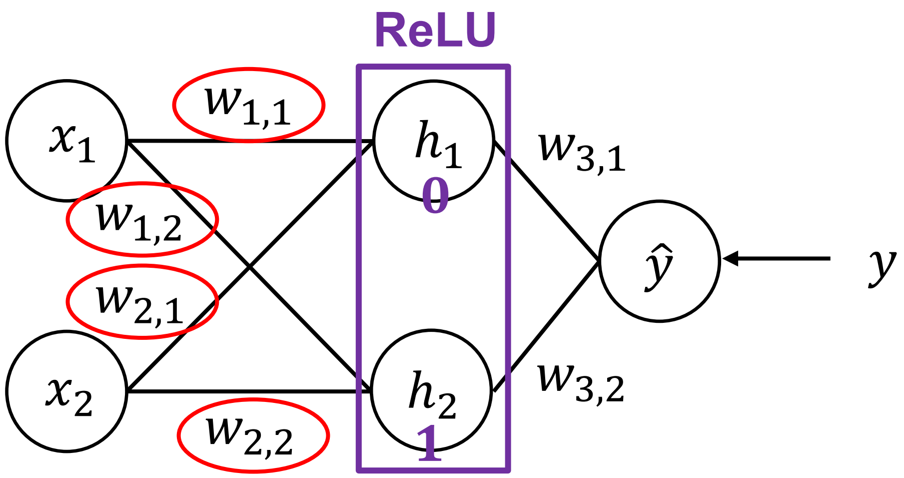
還有哪些參數還沒算到 Gradients



$$\mathcal{L}'(\mathbf{w},\mathbf{b}) = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial w_{1,1}} \\ \frac{\partial \mathcal{L}}{\partial w_{1,2}} \\ \vdots \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial b_3} \end{bmatrix} = \nabla_{\mathbf{w},\mathbf{b}}\mathcal{L}$$

剩下的 w 和 b 的算法其實大同小異

如果有 ReLU...



$$\frac{\partial \mathcal{L}}{\partial w_{1,1}} = \overset{0}{\frac{\partial h_1}{\partial w_{1,1}}} \cdot \frac{\partial \mathcal{L}}{\partial h_1} = \frac{\partial h_1}{\partial w_{1,1}} \cdot \frac{\partial \hat{y}}{\partial h_1} \cdot \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

$$\frac{\partial \mathcal{L}}{\partial w_{1,2}} = \frac{\partial h_2}{\partial w_{1,2}} \cdot \frac{\partial \mathcal{L}}{\partial h_2} = \frac{\partial h_2}{\partial w_{1,2}} \cdot \frac{\partial \hat{y}}{\partial h_2} \cdot \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

$$\frac{\partial \mathcal{L}}{\partial w_{2,1}} = \overset{0}{\frac{\partial h_1}{\partial w_{2,1}}} \cdot \frac{\partial \mathcal{L}}{\partial h_1} = \frac{\partial h_1}{\partial w_{2,1}} \cdot \frac{\partial \hat{y}}{\partial h_1} \cdot \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

$$\frac{\partial \mathcal{L}}{\partial w_{2,2}} = \frac{\partial h_2}{\partial w_{2,2}} \cdot \frac{\partial \mathcal{L}}{\partial h_2} = \frac{\partial h_2}{\partial w_{2,2}} \cdot \frac{\partial \hat{y}}{\partial h_2} \cdot \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

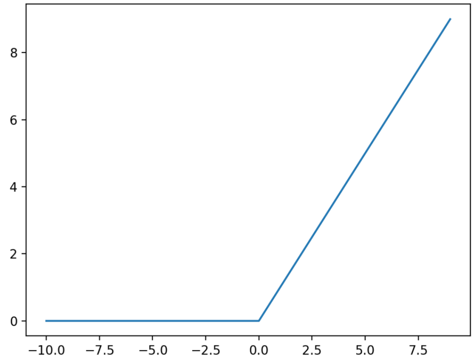


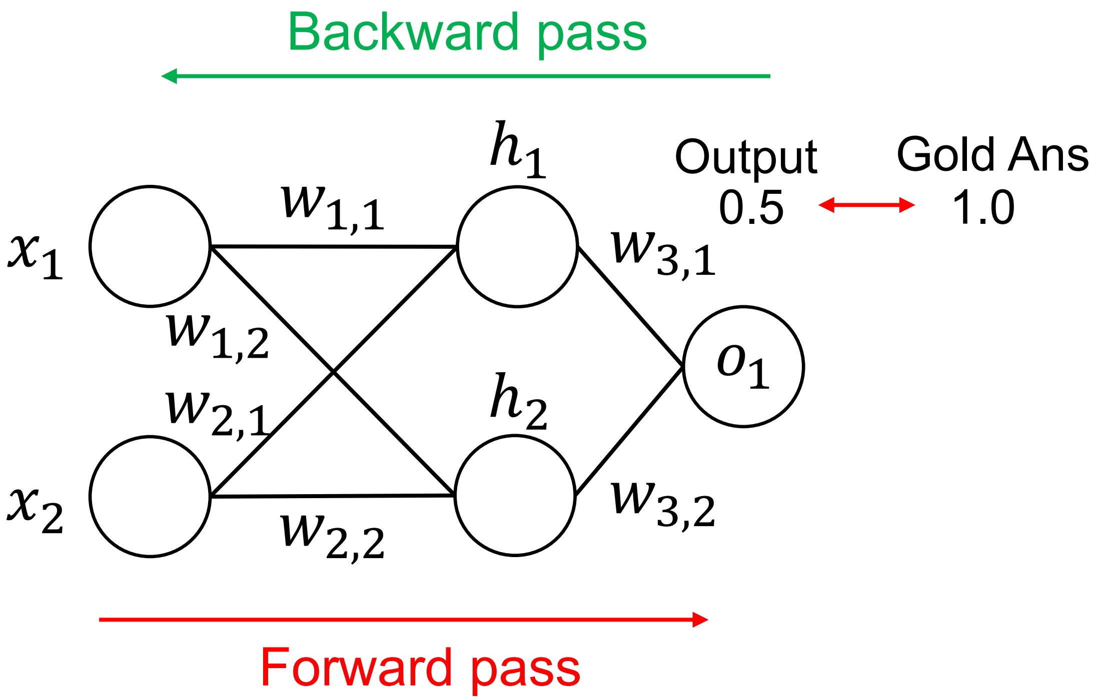
Figure source: https://paperswithcode.com/method/relu

Gradient Descent vs. Back-propagation

	Gradient Descent	Back-propagation
功能	根據梯度更新權重值 (weights)	計算神經網路中的梯度，以供梯度下降使用
最終目標	透過梯度找出 function 最佳解 (最小化目標函數)	計算梯度，以便使用梯度下降找到最佳解

Summary for Backpropagation (BP)

- BP 利用Chain rule，加上數值紀錄機制，有效率地為所有的模型參數計算梯度
- Forward Pass: 算出每個節點的values
- Backward Pass: error signal
- 核心精神是 Cache 機制



Can we train deep NN without BP?

- Calculating partial derivatives for the parameters by hands is time-consuming.
- Alternatives of BP:
  - https://stackoverflow.com/questions/55287004/are-there-alternatives-to-backpropagation

Thank you!

Instructor: 林英嘉

✉️ yjlin@cgu.edu.tw