# 深度學習
# Deep Learning

## Backpropagation

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

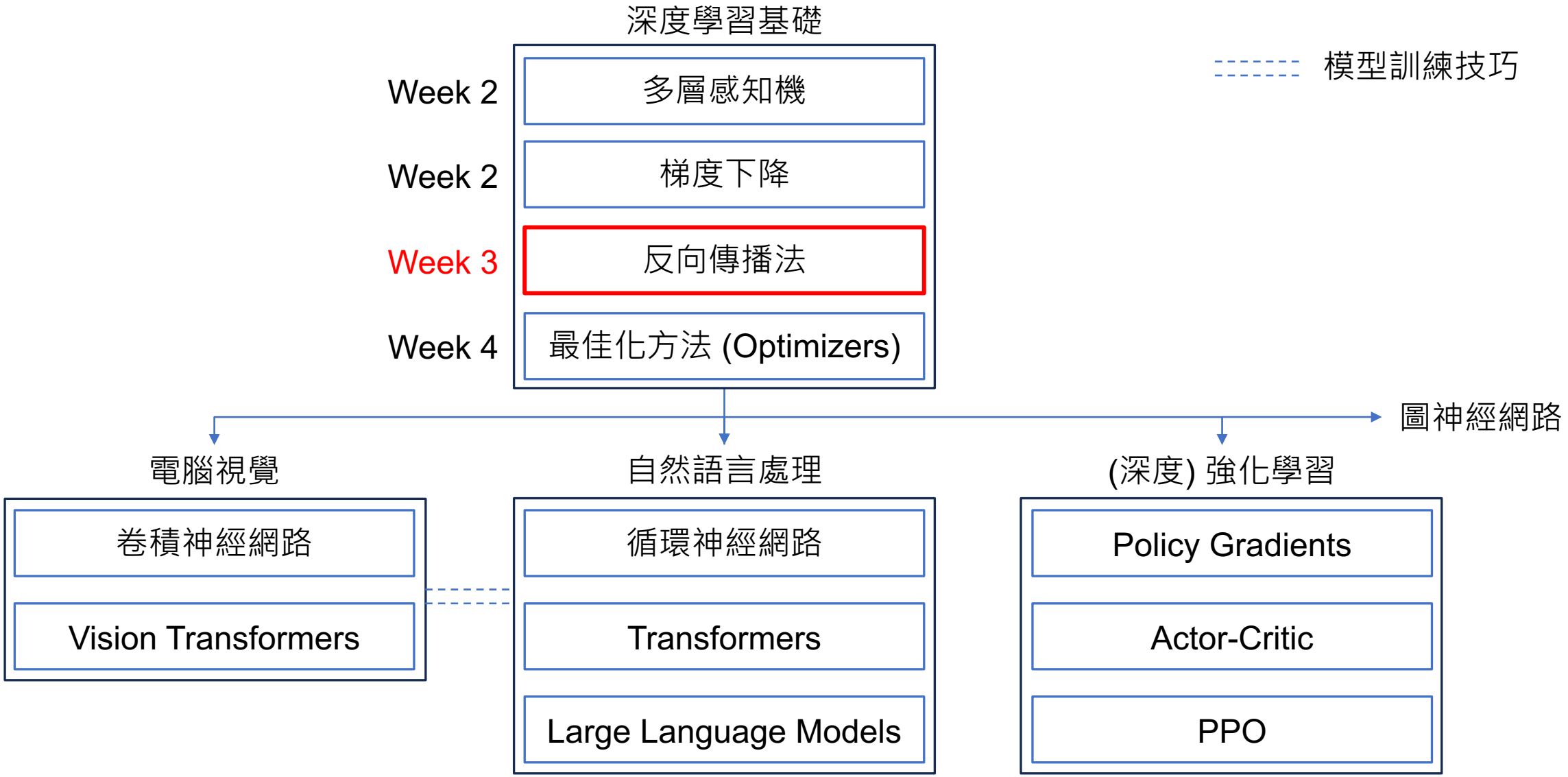
Course GitHub

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] Univariate gradients

- First-order derivative (s)
  - Univariate function: a scaler
  - Multivariate function: a vector
- Univariate Example:

Original function: $f(x) = x^2$

First-order derivative: $f'(x) = 2x = \nabla_x f$

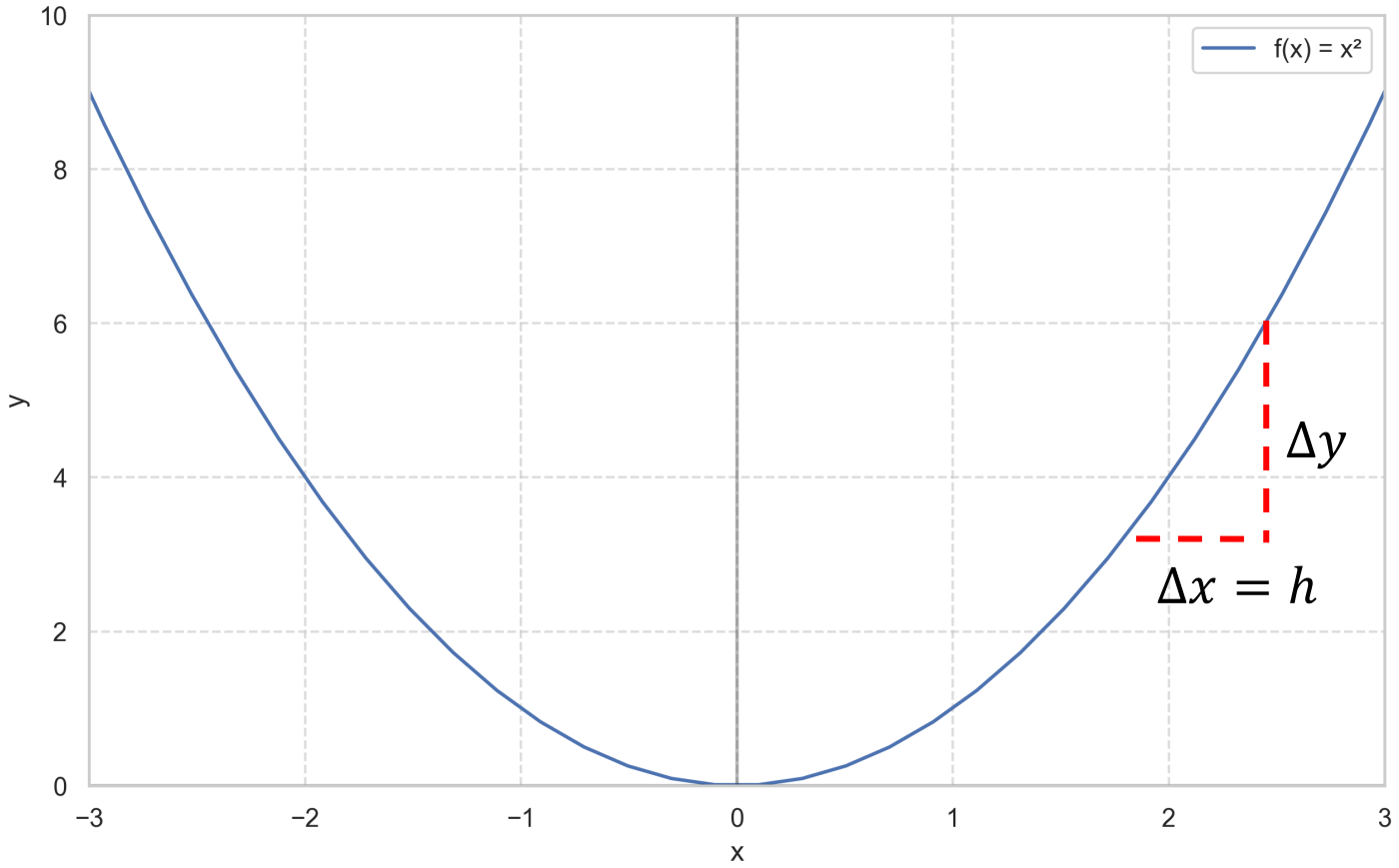
# [Recap] Multivariate gradients

- First-order derivative (s)
  - Univariate function: a scaler
  - Multivariate function: a vector
- Multivariate Example:

Original function: $f(x, y) = x^2 + y^2$

First-order derivatives: $f'(x, y) = \begin{bmatrix} 2x \\ 2y \end{bmatrix} = \nabla_{x,y} f$

# [Recap] What are gradients? (數學意義)




斜率$= \frac{\Delta y}{\Delta x}$　　$f(x) = x^2$

一次導函數：

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

$$= 2x$$

梯度等於一次導函數的值，
也就是某個點的斜率，
代表瞬時變化率 (h→0)

# [Recap] Gradient Descent (梯度下降)

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

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

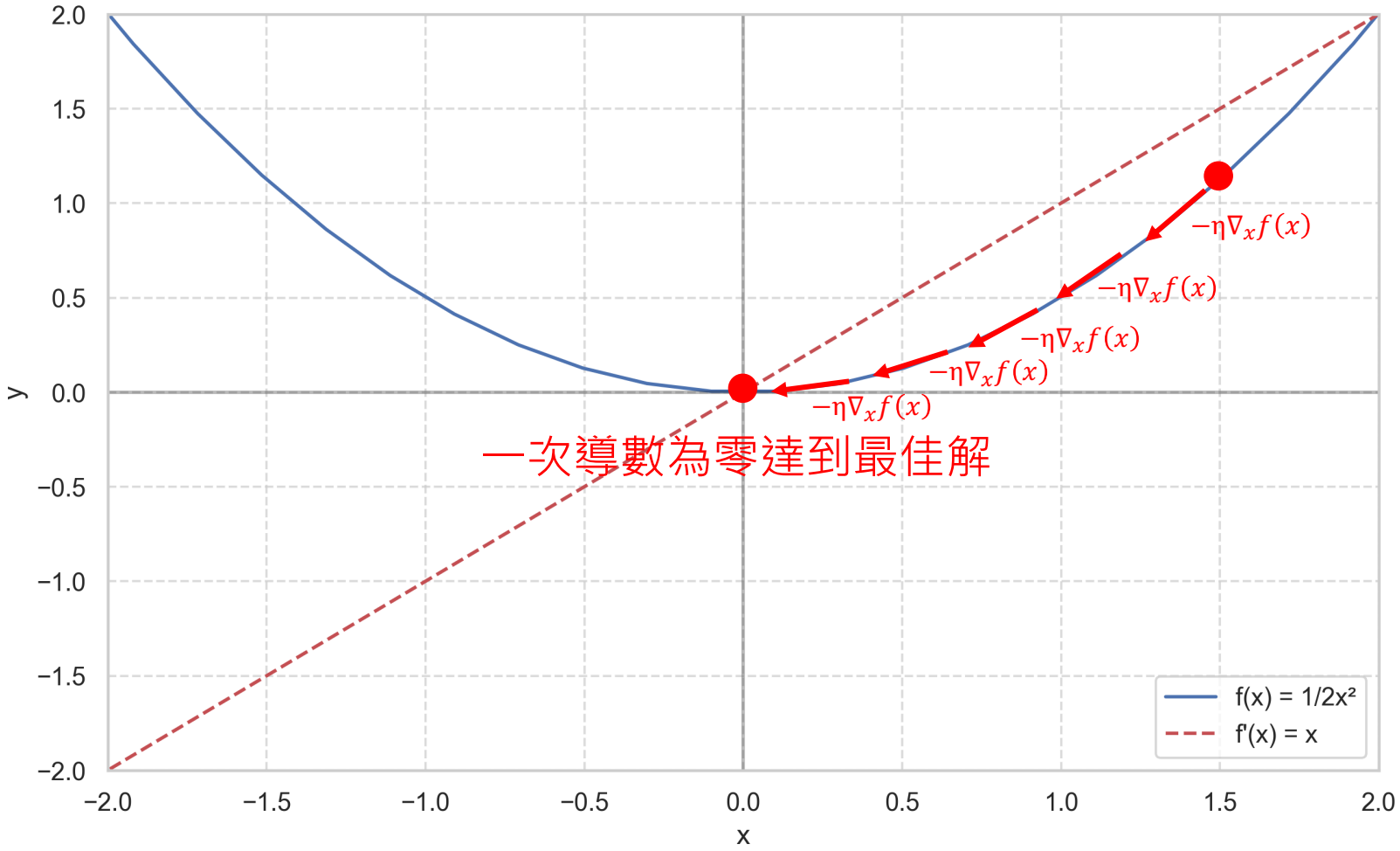
$\eta$ is the learning rate (伊塔/欸塔) used for gradient descent.

- 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)



# [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：

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

把 $(wx_i + b)$ 代入 $\hat{y}_i$ ⟶

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

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

- 其中:
  - $\mathcal{L}$ 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

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

對w進行偏微分

$$\frac{\partial \mathcal{L}}{\partial w} = \frac{1}{n}\sum_{i=1}^{n} 2\left(y_i - (wx_i + b)\right)\cdot(-x_i)$$

對b進行偏微分

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

# [Recap] Minimize a Regression Model

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

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

**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

# 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：

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

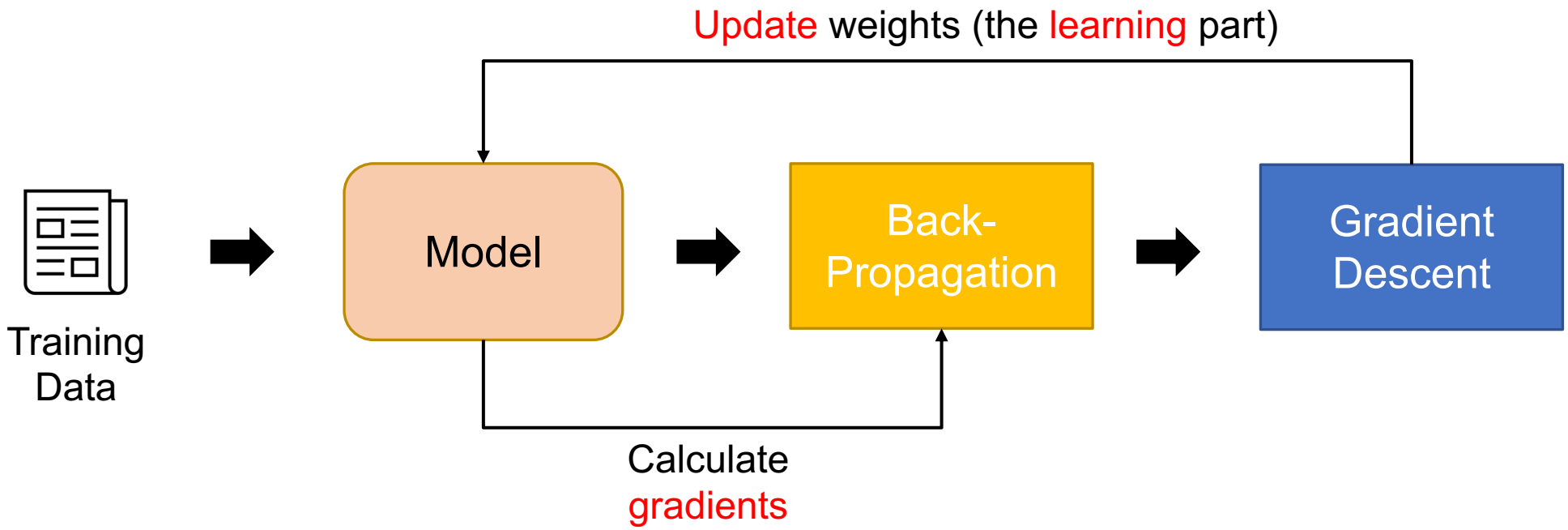
- $\mathcal{L}$: loss function
- $\theta_t$: 在 $t$ 個時間點 (step) 的的參數組合 (包含各層的w跟b)
- $\nabla_{\theta_t}\mathcal{L}(\theta_t)$: 梯度

**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)

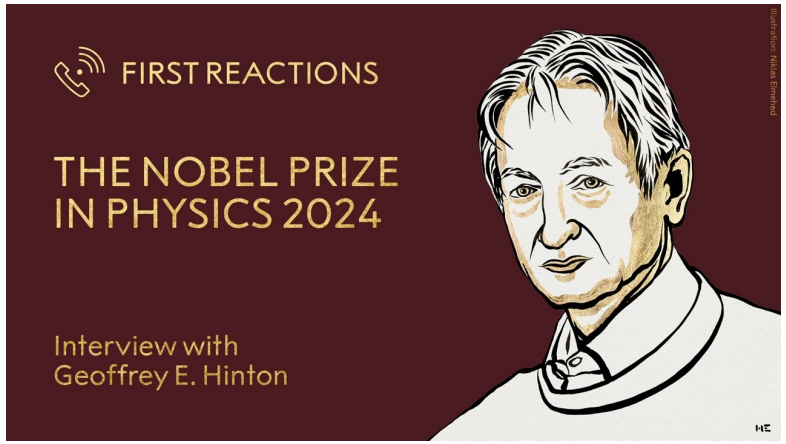
- Backpropagation is the application of the **chain rule** from calculus for computing the gradient.

**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].**

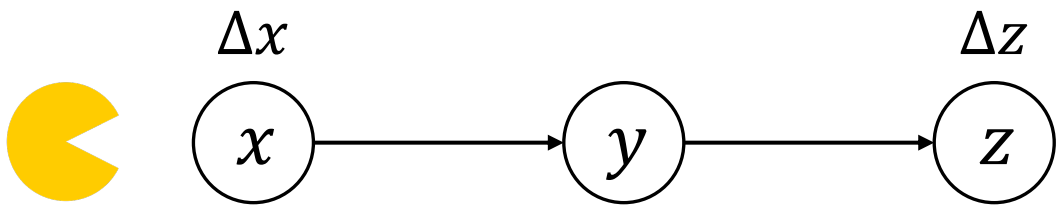



https://www.youtube.com/watch?v=-icD_KmvnnM

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

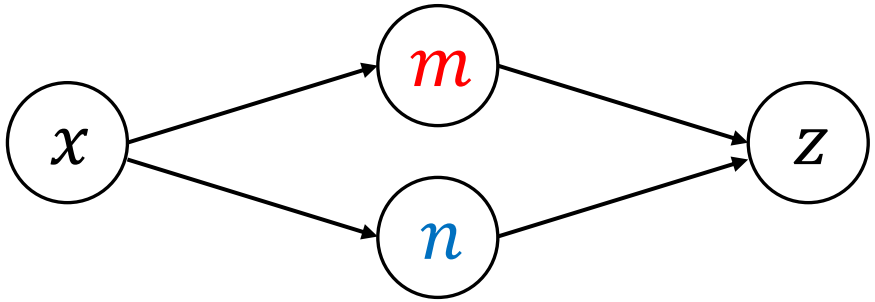
# (Calculus) Chain Rule - 1




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

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

# (Calculus) Chain Rule - 2

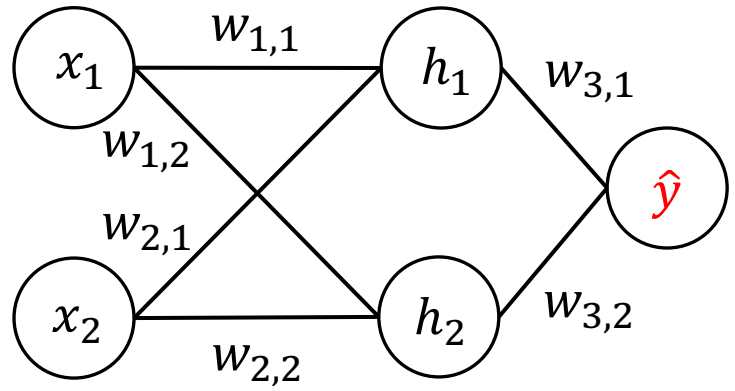



$$\frac{dz}{dx} = \frac{dz}{dm}\frac{dm}{dx} + \frac{dz}{dn}\frac{dn}{dx}$$

- $\frac{dz}{dx}$ 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)

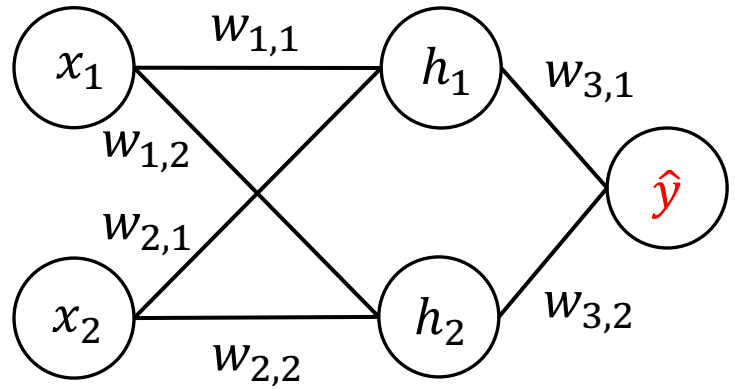
⬇️ A two-layer MLP (multi-layer perceptron)




輸入層　　隱藏層 (1st 層)　　輸出層 (2nd 層)

# MLP 多層感知機 (今天的範例模型)

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




輸入層　　隱藏層 (1st 層)　　輸出層 (2nd 層)

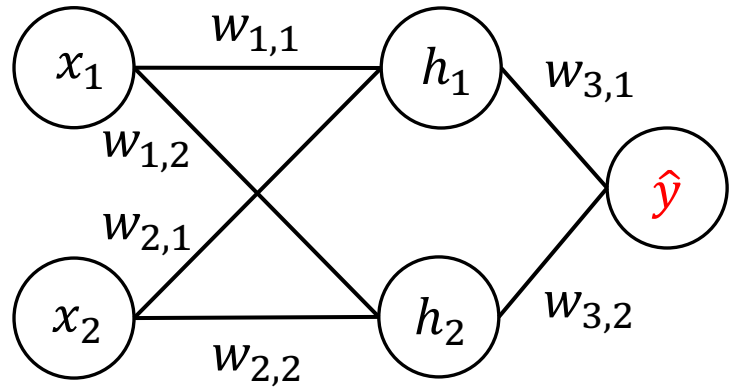
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

(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)




輸入層　　隱藏層 (1st 層)　　輸出層 (2nd 層)

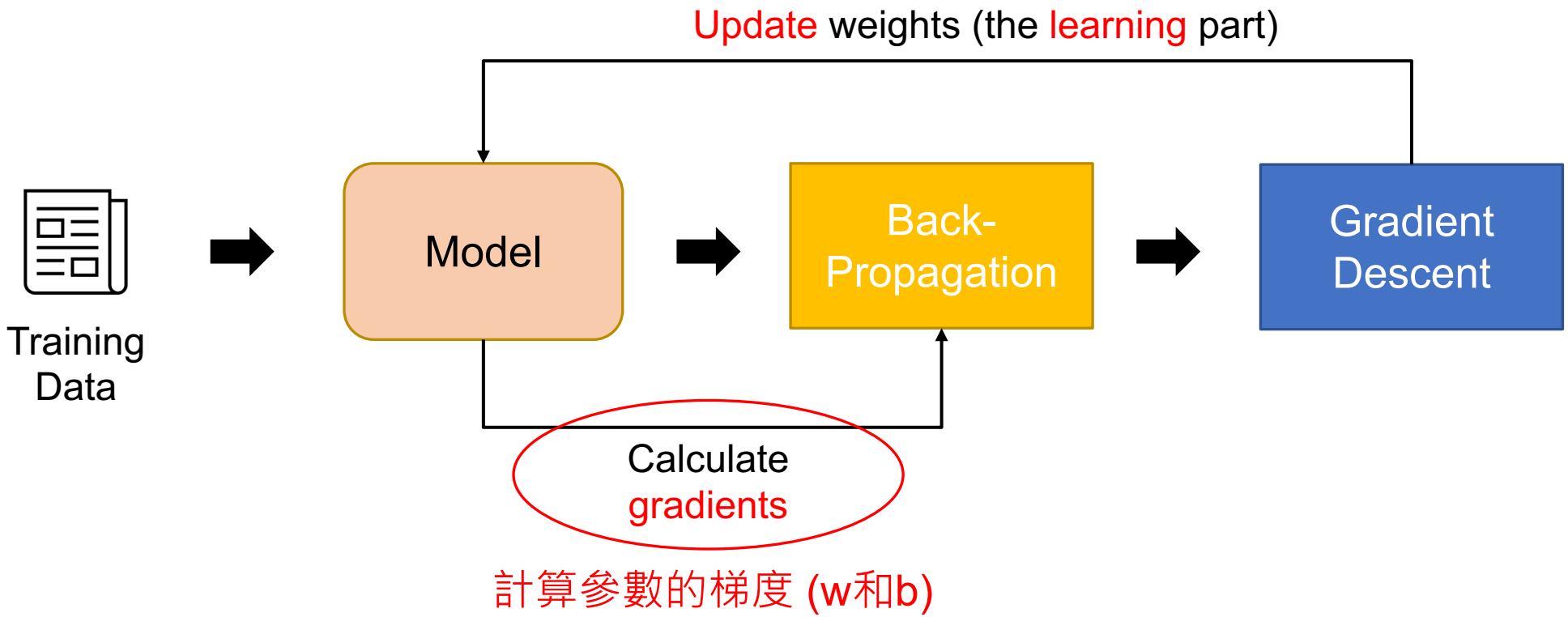
$$h_1 = w_{1,1}x_1 + w_{2,1}x_2 + b_1$$

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

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

# Training Process of a Deep Learning Model

- 深度學習模型被訓練的流程



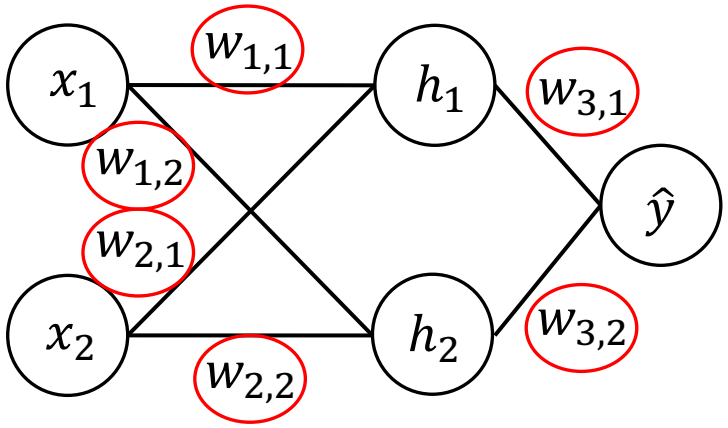
# [Recap] Univariate gradients

- First-order derivative (s)
  - Univariate function: a scaler
  - Multivariate function: a vector
- Multivariate Example:

Original function: $f(x, y) = x^2 + y^2$

First-order derivatives: $f'(x, y) = \begin{bmatrix} 2x \\ 2y \end{bmatrix} = \nabla_{x,y} f$

# 計算參數的梯度 (w和b)




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

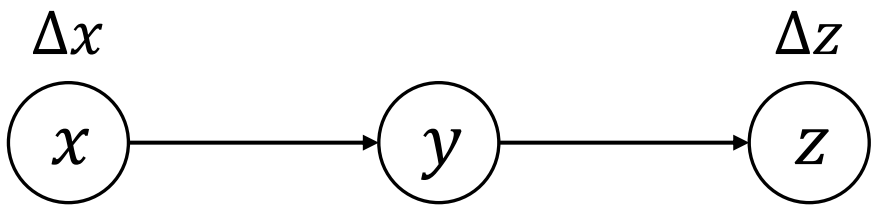
$$h_2 = w_{1,2}x_1 + w_{2,2}x_2 + b_2$$

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

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

為了簡化計算流程，後面的推導只會涵蓋 w 的部分

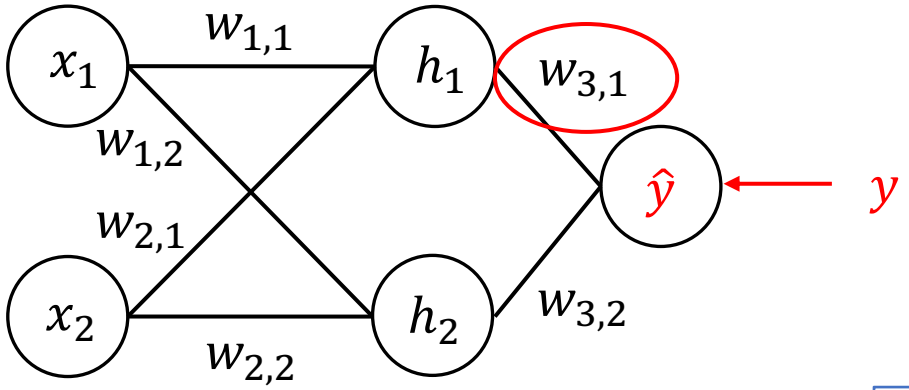
# (Calculus) Chain Rule - 1




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

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

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




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

**grad_table**

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

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

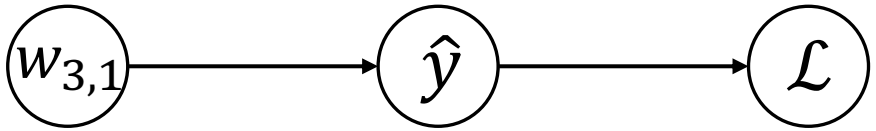
$\mathcal{L}$ 帶入

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

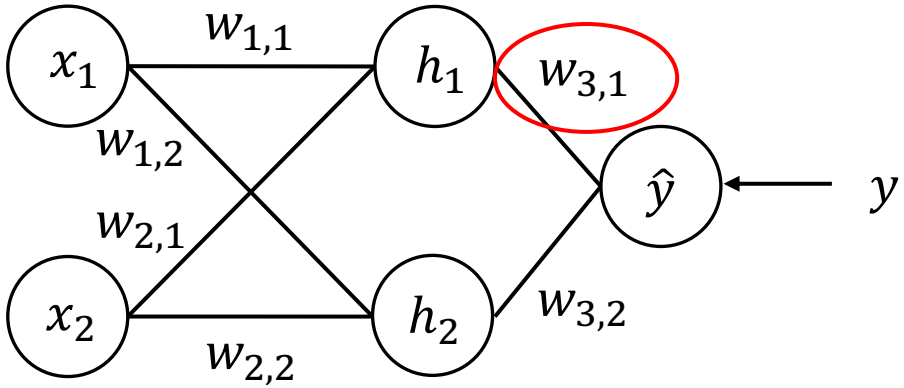
The generalized power rule
(廣義冪次規則)

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

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



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




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

**grad_table**

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

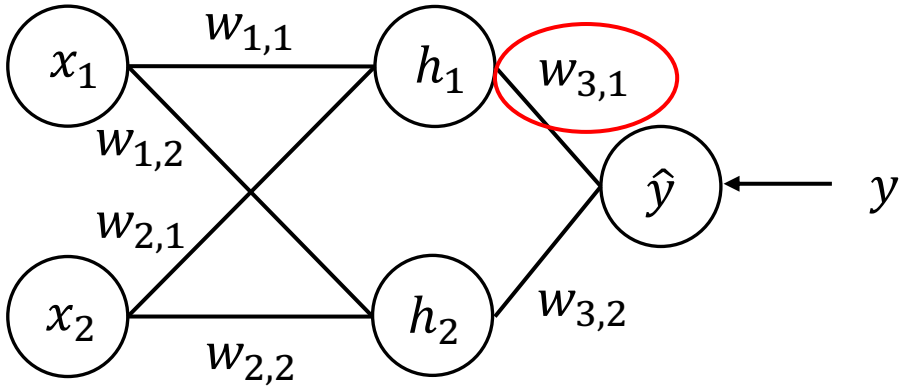
$\hat{y}$ 帯入

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

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

$$= h_1$$

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



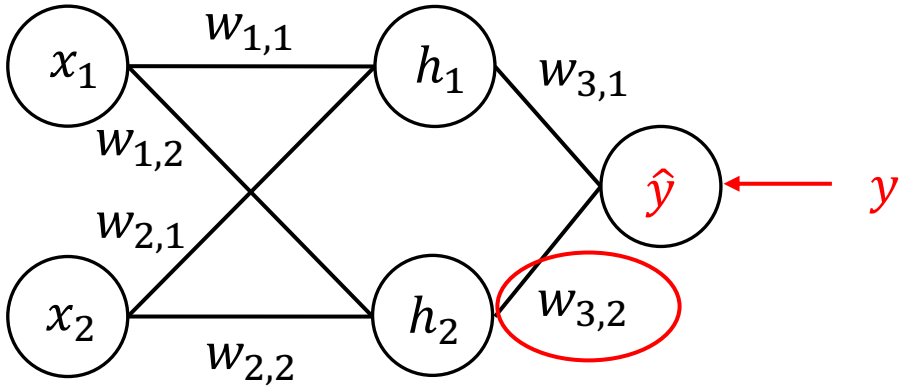

$$\frac{\partial \mathcal{L}}{\partial w_{3,1}} = \frac{\partial \hat{y}}{\partial w_{3,1}} \cdot \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

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

**grad_table**

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

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

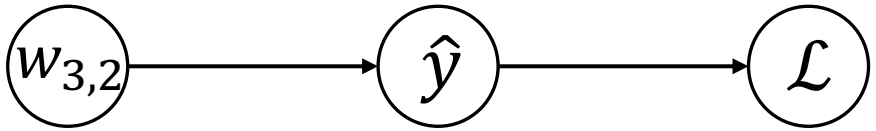



$$\frac{\partial \mathcal{L}}{\partial w_{3,2}} = \frac{\partial \hat{y}}{\partial w_{3,2}} \cdot \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

**grad_table**

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

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

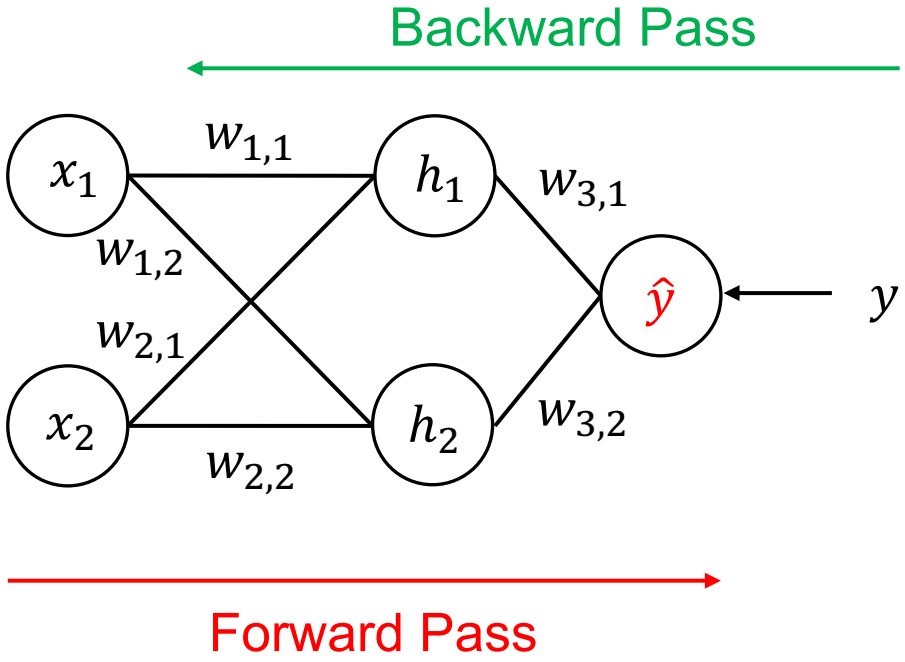



因為 $\hat{y} = w_{3,1}h_1 + w_{3,2}h_2 + b_3$

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)




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

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

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

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

`grad_table`

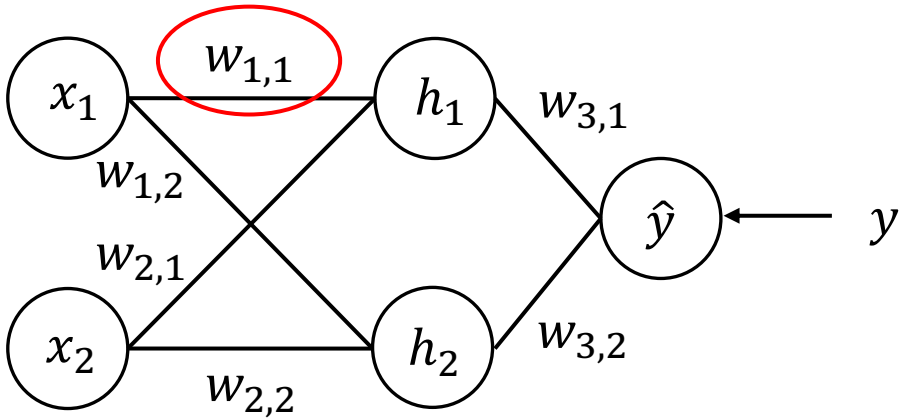
| 微分項 | 值 |
|---|---|
| $\frac{\partial \mathcal{L}}{\partial \hat{y}}$ | $-(y - \hat{y})$ |
| $\frac{\partial \hat{y}}{\partial w_{3,1}}$ | $h_1$ |

💡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, $w_{1,1}$)




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

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

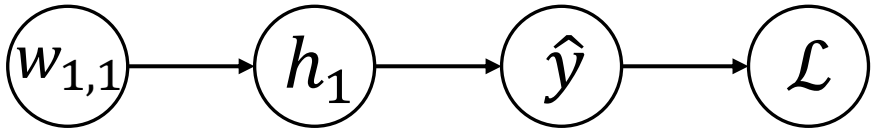
**grad_table**

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

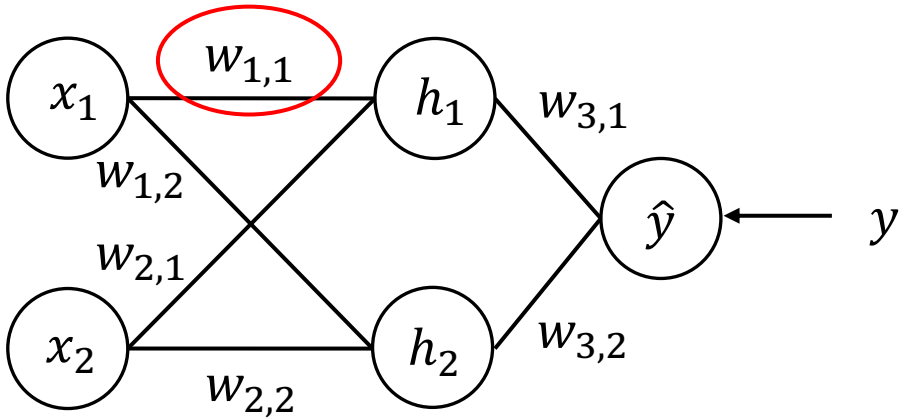
$$\hat{y} = w_{3,1}h_1 + w_{3,2}h_2 + b_3$$

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

$$= w_{3,1}$$



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




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

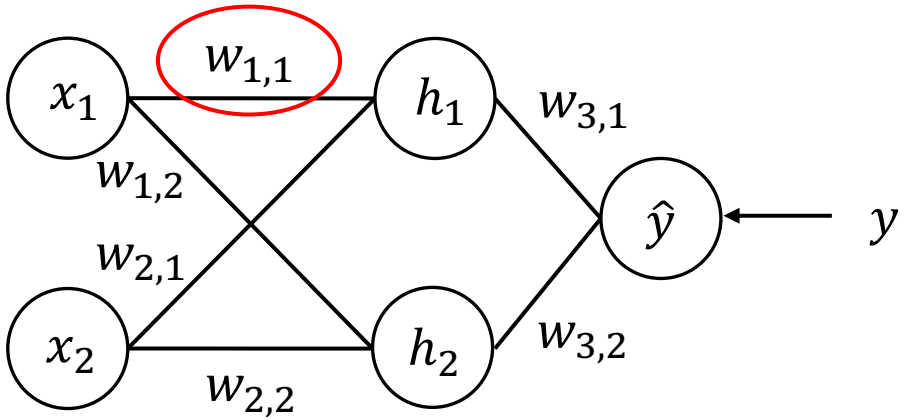
$$= \frac{\partial h_1}{\partial w_{1,1}} \cdot \frac{\partial \hat{y}}{\partial h_1} \cdot \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

**grad_table**

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

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 | 値 |
|---|---|
| $\frac{\partial \mathcal{L}}{\partial \hat{y}}$ | $-(y - \hat{y})$ |
| $\frac{\partial \hat{y}}{\partial w_{3,1}}$ | $h_1$ |
| $\frac{\partial \hat{y}}{\partial h_1}$ | $w_{3,1}$ |
| $\frac{\partial h_1}{\partial w_{1,1}}$ | $x_1$ |

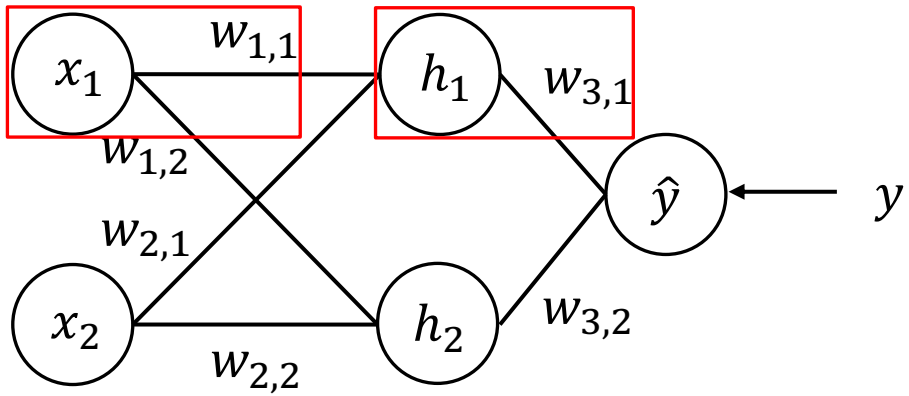
$$h_1 = w_{1,1}x_1 + w_{2,1}x_2 + b_1$$

因此

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

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

# 反向傳播法觀察 (2): Forward Pass




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

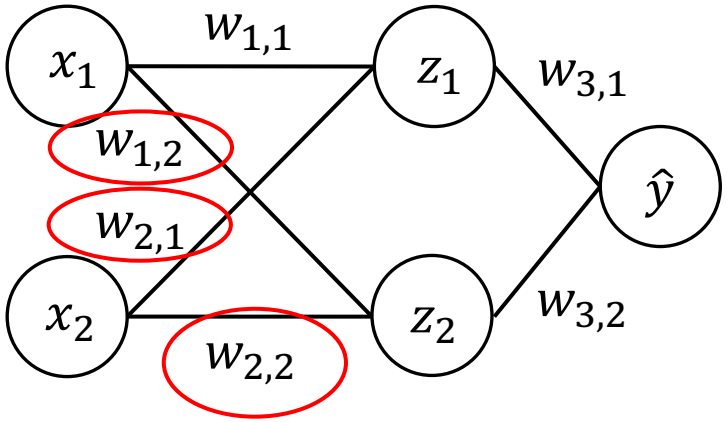
$$h_1 = w_{1,1}x_1 + w_{2,1}x_2 + b_1$$

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

💡Forward Pass 結果的偏微分等於輸入的值

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

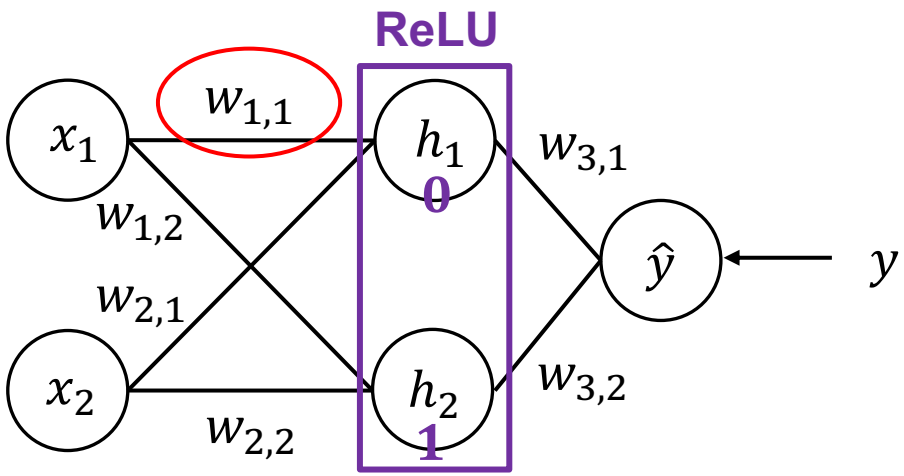
# 還有哪些參數還沒算到 Gradients




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

剩下的 w 和 b 的算法其實大同小異

# 如果有 ReLU...



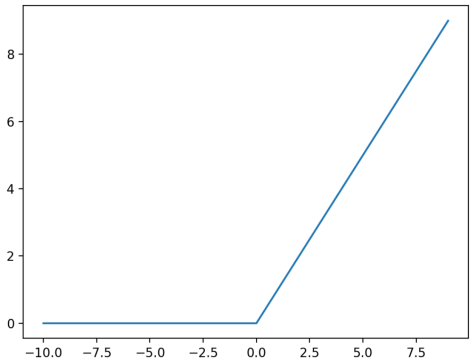




Figure source: https://paperswithcode.com/method/relu

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

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

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

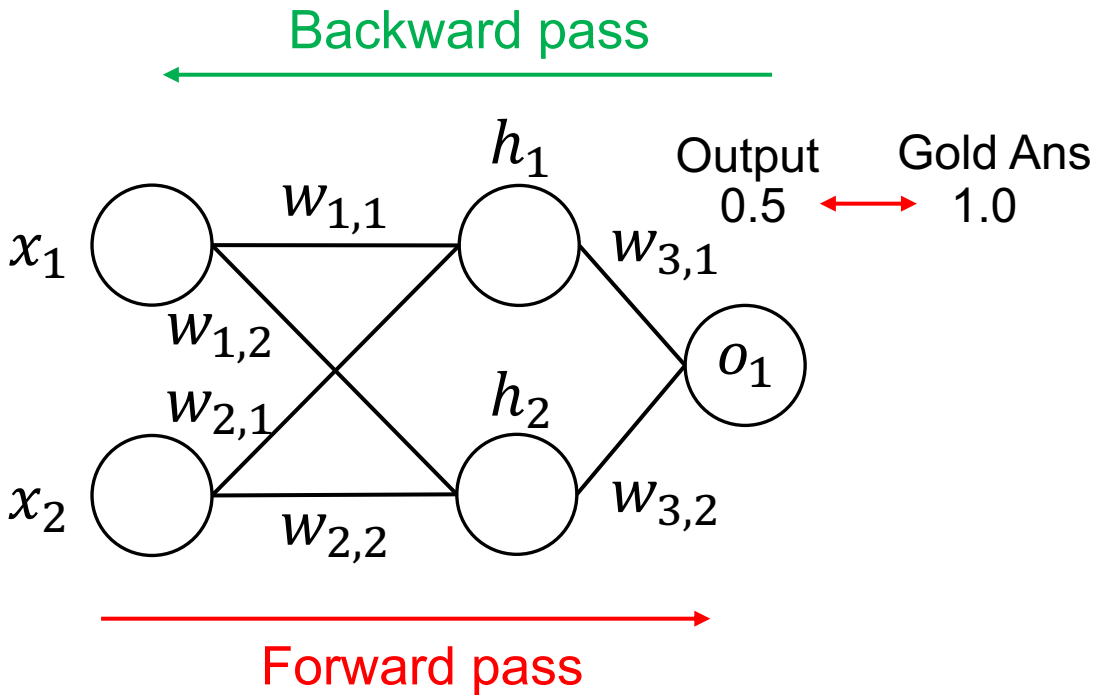
$$\frac{\partial \mathcal{L}}{\partial w_{2,2}} = \frac{\partial h_2}{\partial w_{2,2}} \cdot \frac{\partial \mathcal{L}}{\partial h_2} = \frac{\partial h_2}{\partial w_{2,2}} \cdot \frac{\partial \hat{y}}{\partial h_2} \cdot \frac{\partial \mathcal{L}}{\partial \hat{y}}$$

# 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