

# Computing Gradient

Hung-yi Lee

李宏毅

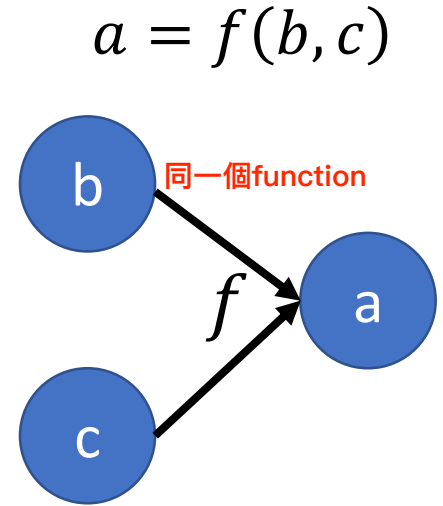
# Introduction

- Backpropagation: an efficient way to compute the gradient
- Prerequisite
  - Backpropagation for feedforward net:
    - [http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS\\_2015\\_2/Lecture/DNN%20backprop.ecm.mp4/index.html](http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/Lecture/DNN%20backprop.ecm.mp4/index.html)
    - Simple version: <https://www.youtube.com/watch?v=ibJpTrp5mcE>
  - Backpropagation through time for RNN:  
[http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS\\_2015\\_2/Lecture/RNN%20training%20\(v6\).ecm.mp4/index.html](http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/Lecture/RNN%20training%20(v6).ecm.mp4/index.html)
- Understanding backpropagation by computational graph
  - Tensorflow, Theano, CNTK, etc.

# Computational Graph

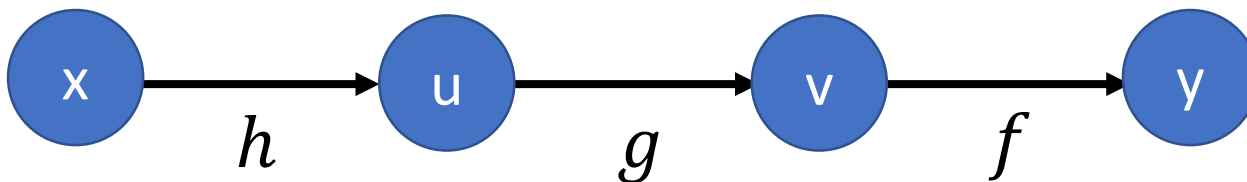
# Computational Graph

- A “language” describing a function
  - **Node**: variable (scalar, vector, tensor <sup>matrix</sup> .....
  - **Edge**: operation (simple function)



Example  $y = f(g(h(x)))$

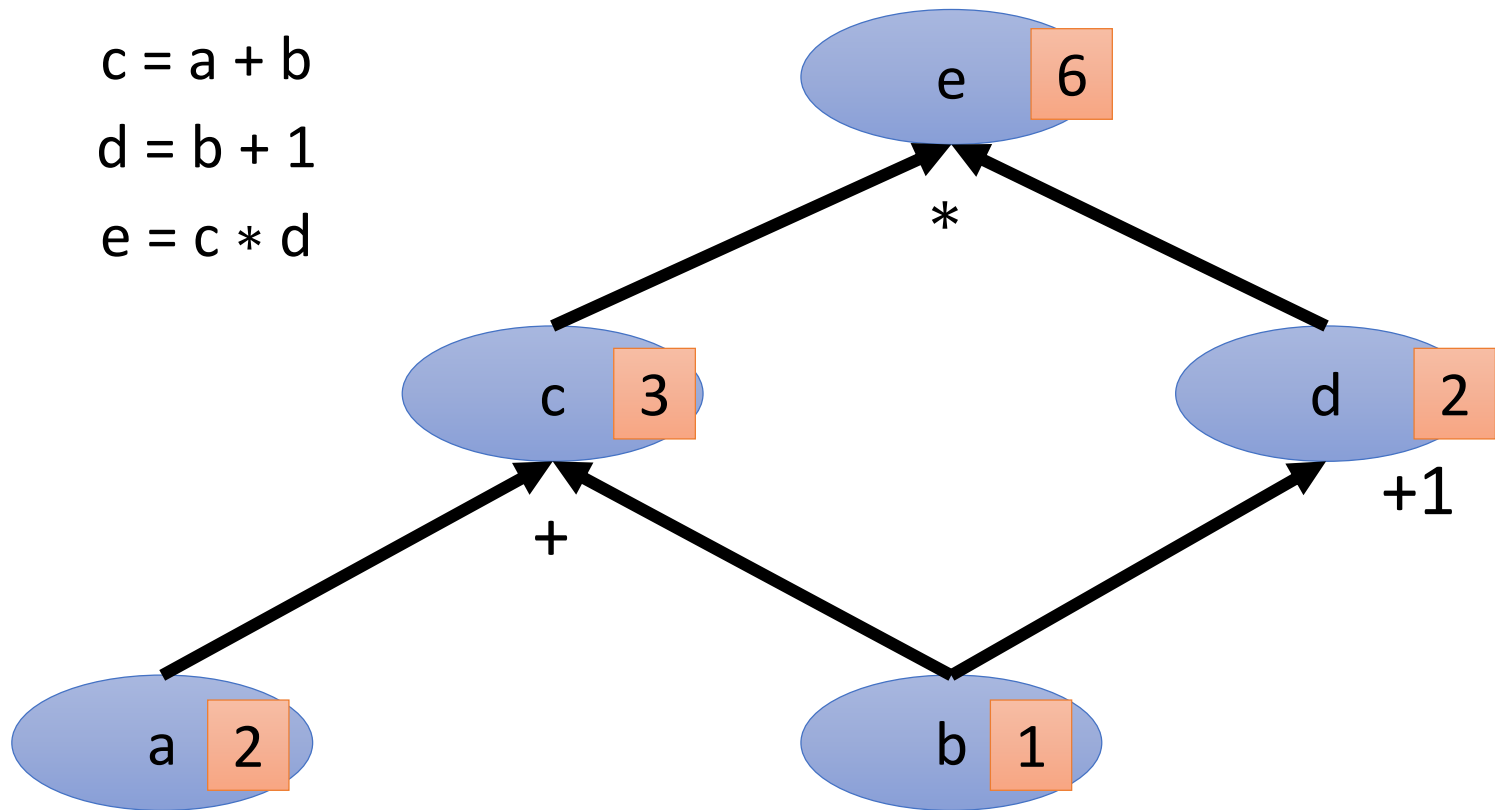
$$u = h(x) \quad v = g(u) \quad y = f(v)$$



# Computational Graph

好處是求每個node gradient是很方便的

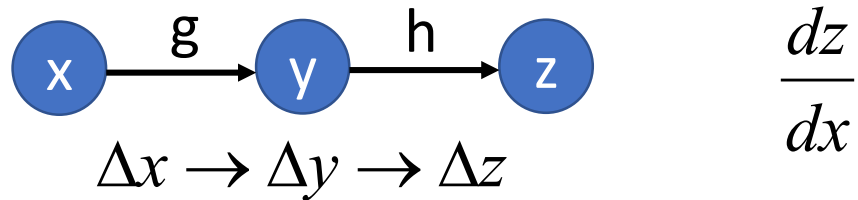
- Example:  $e = (a+b) * (b+1)$



# Review: Chain Rule

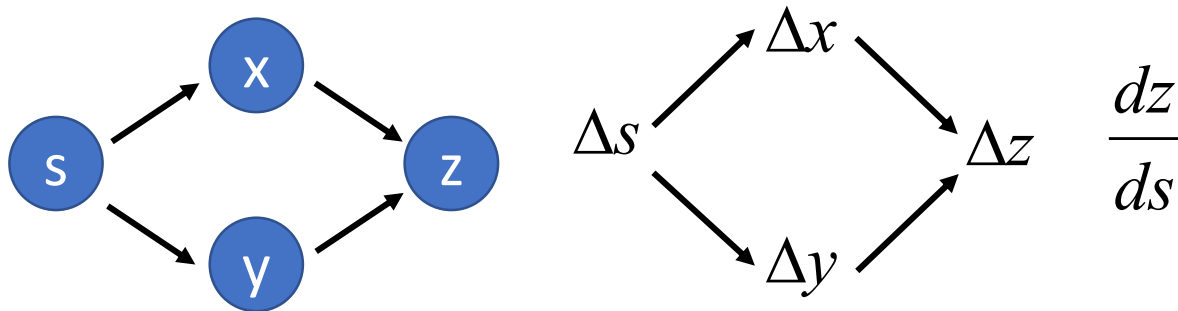
## Case 1

$$z = f(x) \quad \longrightarrow \quad y = g(x) \quad z = h(y)$$



## Case 2

$$z = f(s) \quad \longrightarrow \quad x = g(s) \quad y = h(s) \quad z = k(x, y)$$



# Computational Graph

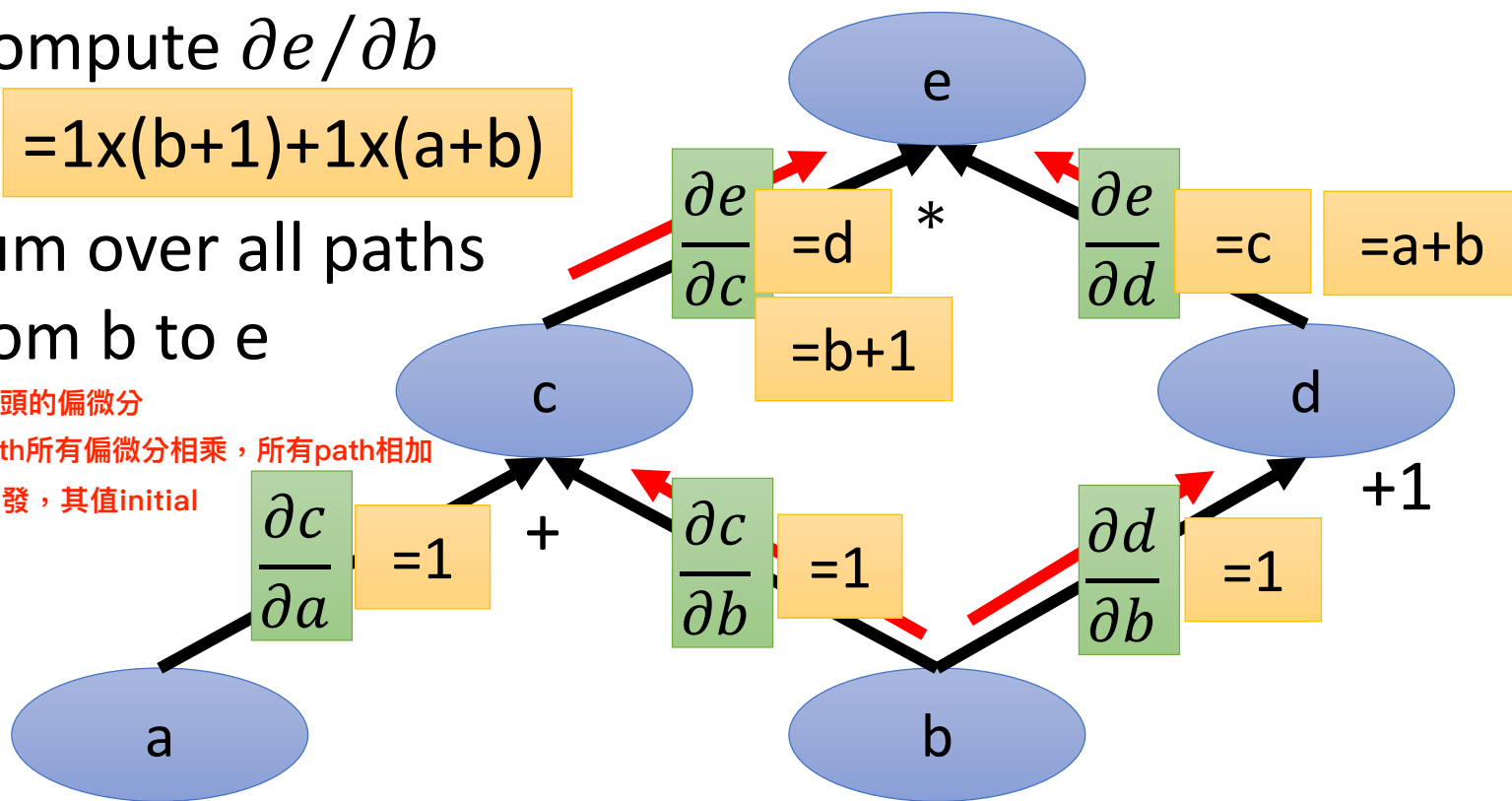
- Example:  $e = (a+b) * (b+1)$

$$\frac{\partial e}{\partial b} = \frac{\partial e}{\partial c} \frac{\partial c}{\partial b} + \frac{\partial e}{\partial d} \frac{\partial d}{\partial b}$$

Compute  $\partial e / \partial b$

$$= 1 \times (b+1) + 1 \times (a+b)$$

Sum over all paths  
from b to e



第一步：算每個箭頭的偏微分

第二步：同一條path所有偏微分相乘，所有path相加

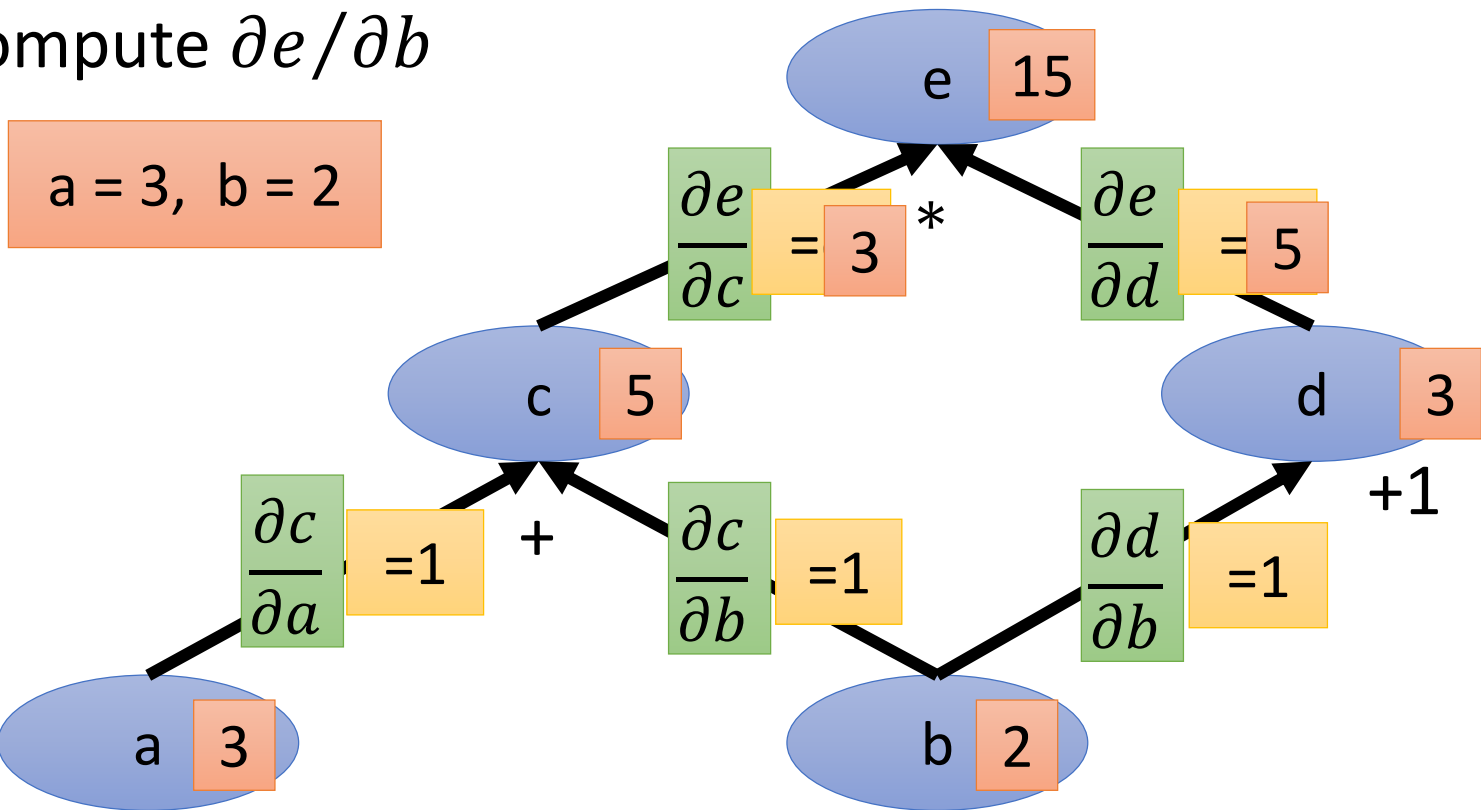
第三步：從哪裡出發，其值initial  
為1代入

# Computational Graph

- Example:  $e = (a+b) * (b+1)$

$$\frac{\partial e}{\partial b} = \frac{\partial e}{\partial c} \frac{\partial c}{\partial b} + \frac{\partial e}{\partial d} \frac{\partial d}{\partial b}$$

Compute  $\partial e / \partial b$



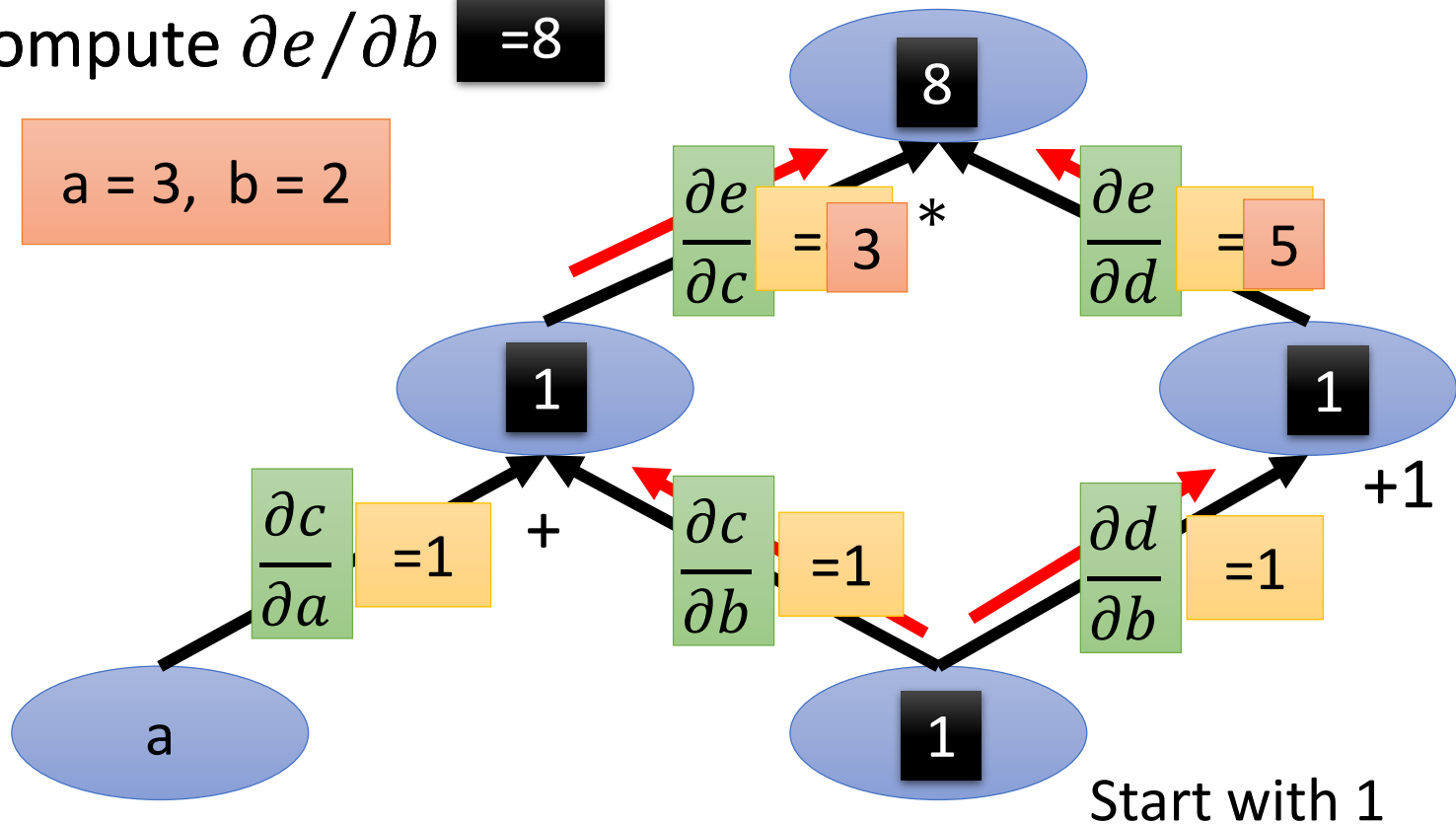


# Computational Graph

- Example:  $e = (a+b) * (b+1)$

Compute  $\partial e / \partial b$  **=8**

$a = 3, b = 2$

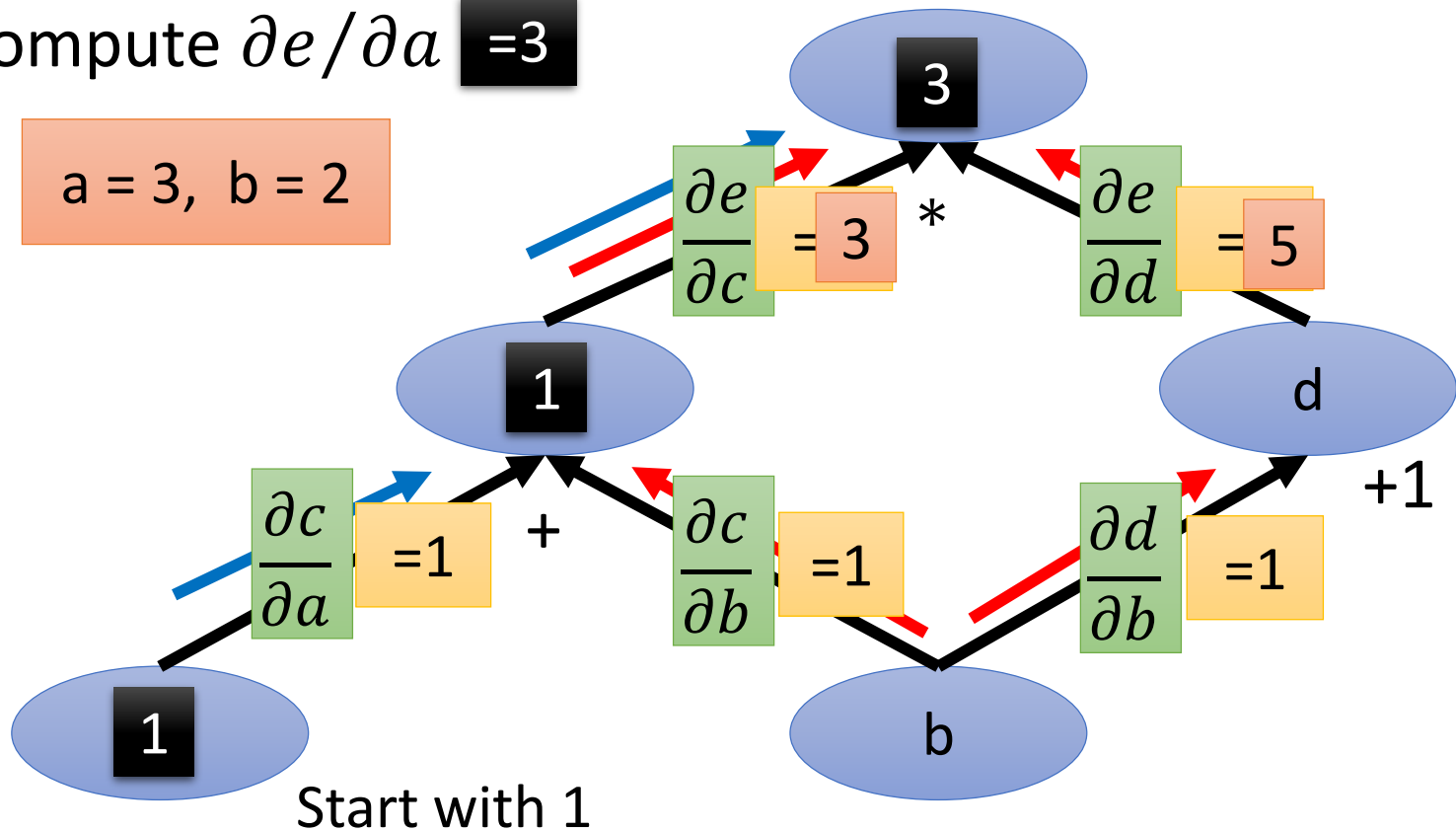


# Computational Graph

- Example:  $e = (a+b) * (b+1)$

Compute  $\partial e / \partial a$  **=3**

$a = 3, b = 2$



# Computational Graph

**back propagation：從root開始走比較有效率**

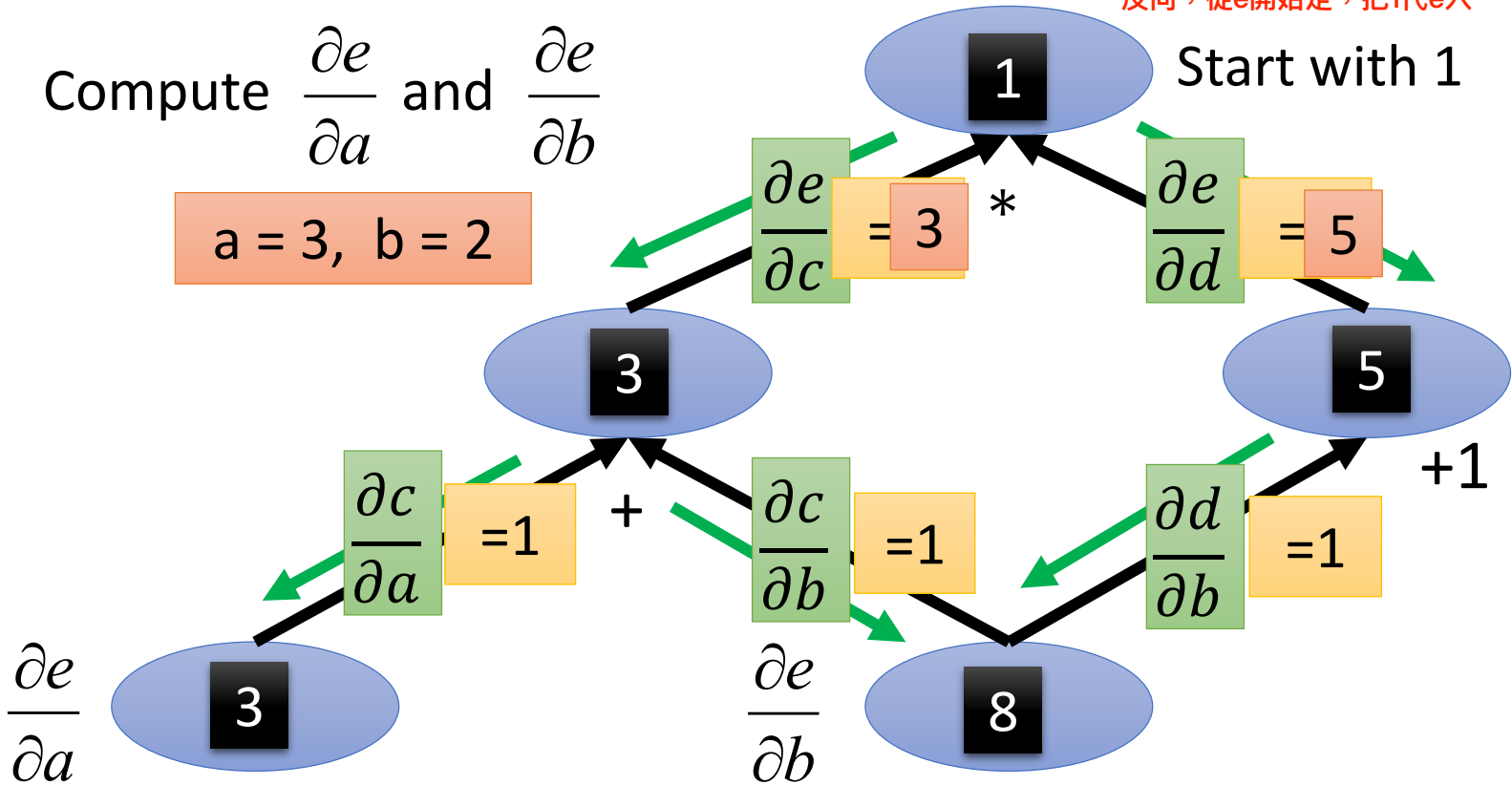
## Reverse mode

## What is the benefit?

- Example:  $e = (a+b) * (b+1)$

Compute  $\frac{\partial e}{\partial a}$  and  $\frac{\partial e}{\partial b}$

a = 3, b = 2



反向，從e開始走，把1代e入

## Start with 1

# Computational Graph

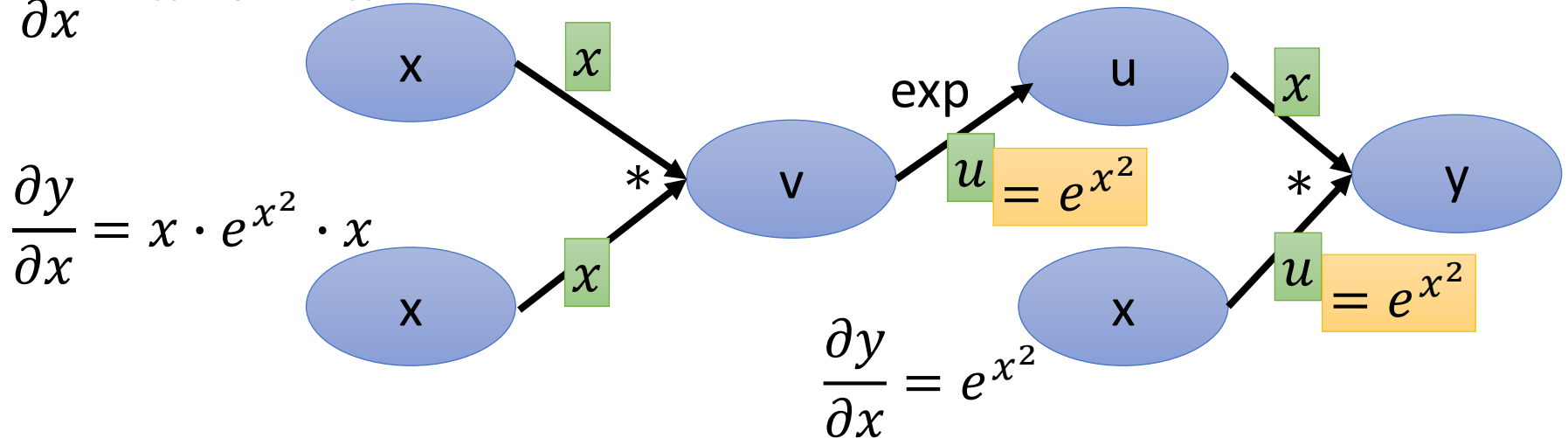
- **Parameter sharing:** the same parameters appearing in different nodes

一開始先把所有共用的parameter當作不一樣的參數來看，算完後再直接加起來

$$y = xe^{x^2} \quad \frac{\partial y}{\partial x} = ? \quad e^{x^2} + x \cdot e^{x^2} \cdot 2x$$

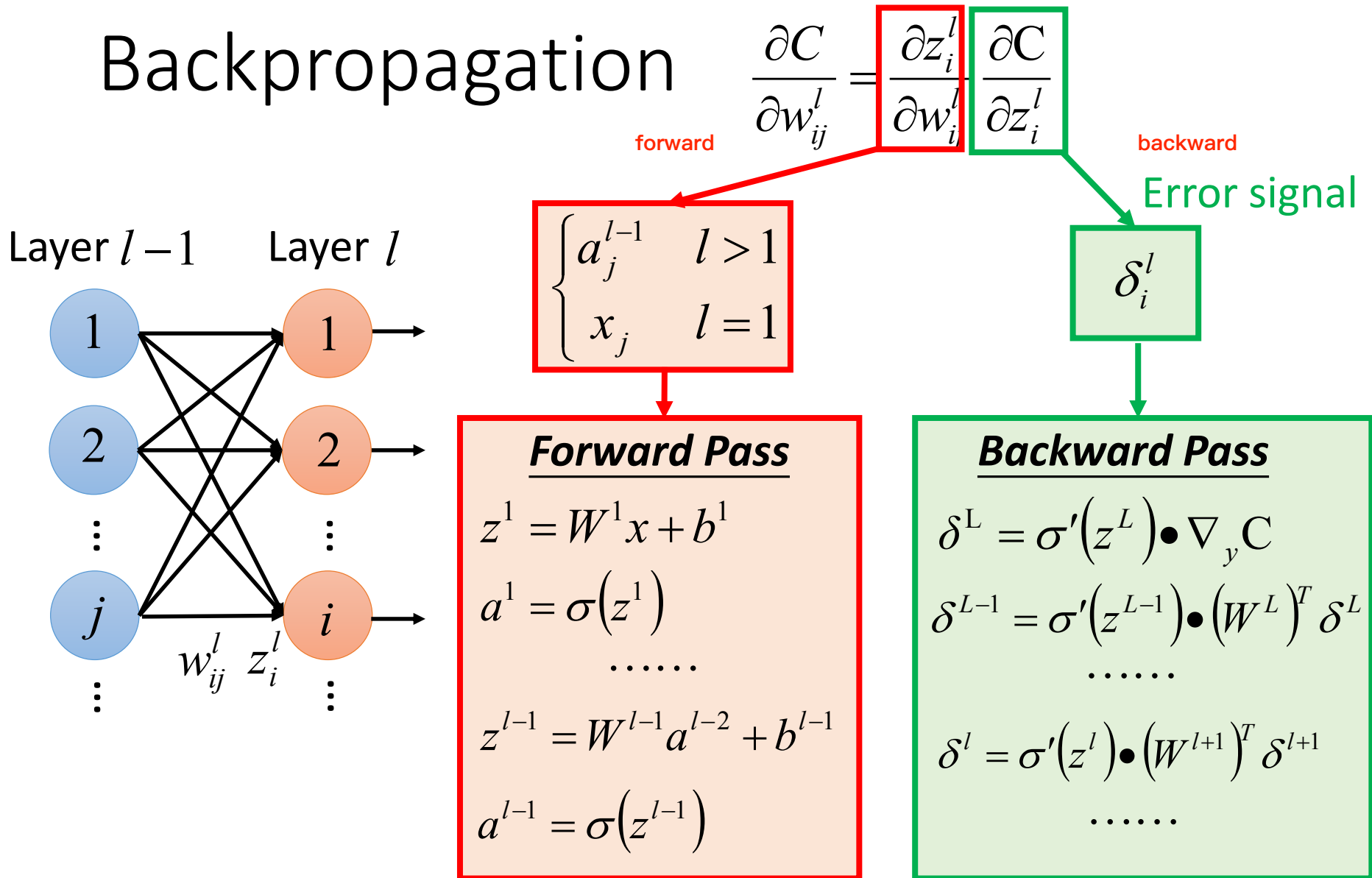
$$\frac{\partial y}{\partial x} = x \cdot e^{x^2} \cdot x$$

$$\frac{\partial y}{\partial x} = x \cdot e^{x^2} \cdot x$$



# Computational Graph for Feedforward Net

# Review: Backpropagation

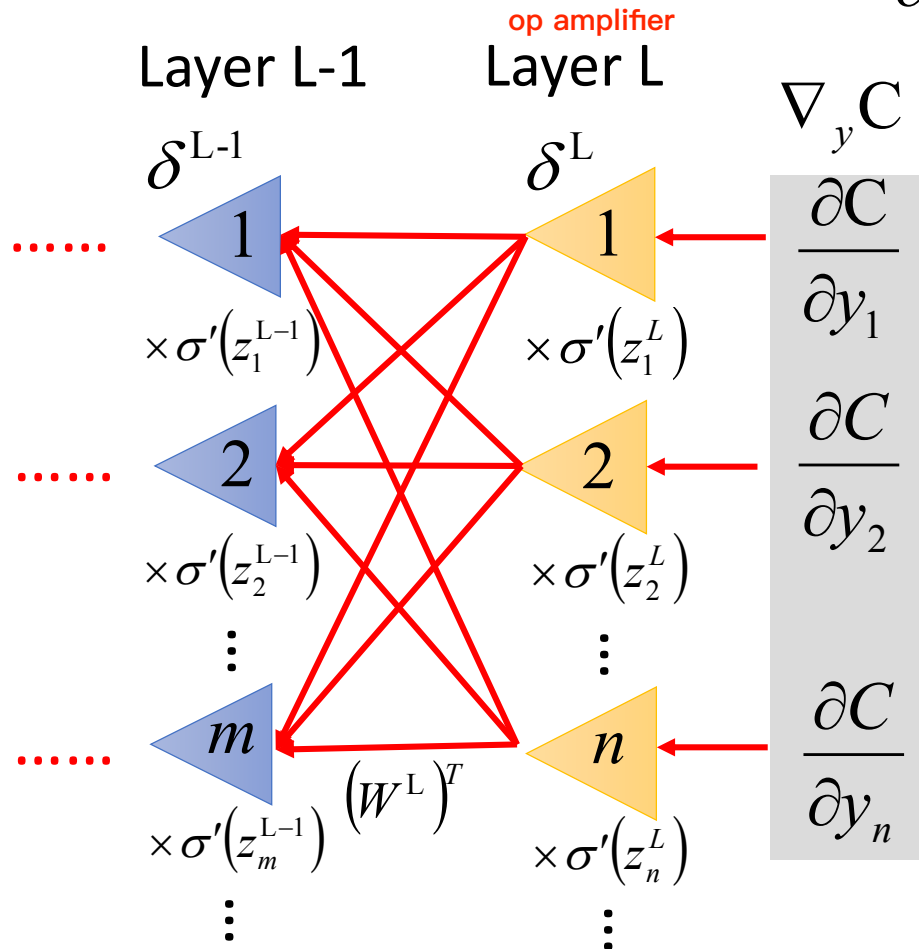


# Review: Backpropagation

把network整個逆轉

$$\frac{\partial C}{\partial w_{ij}^l} = \frac{\partial z_i^l}{\partial w_{ij}^l} \frac{\partial C}{\partial z_i^l}$$

Error signal



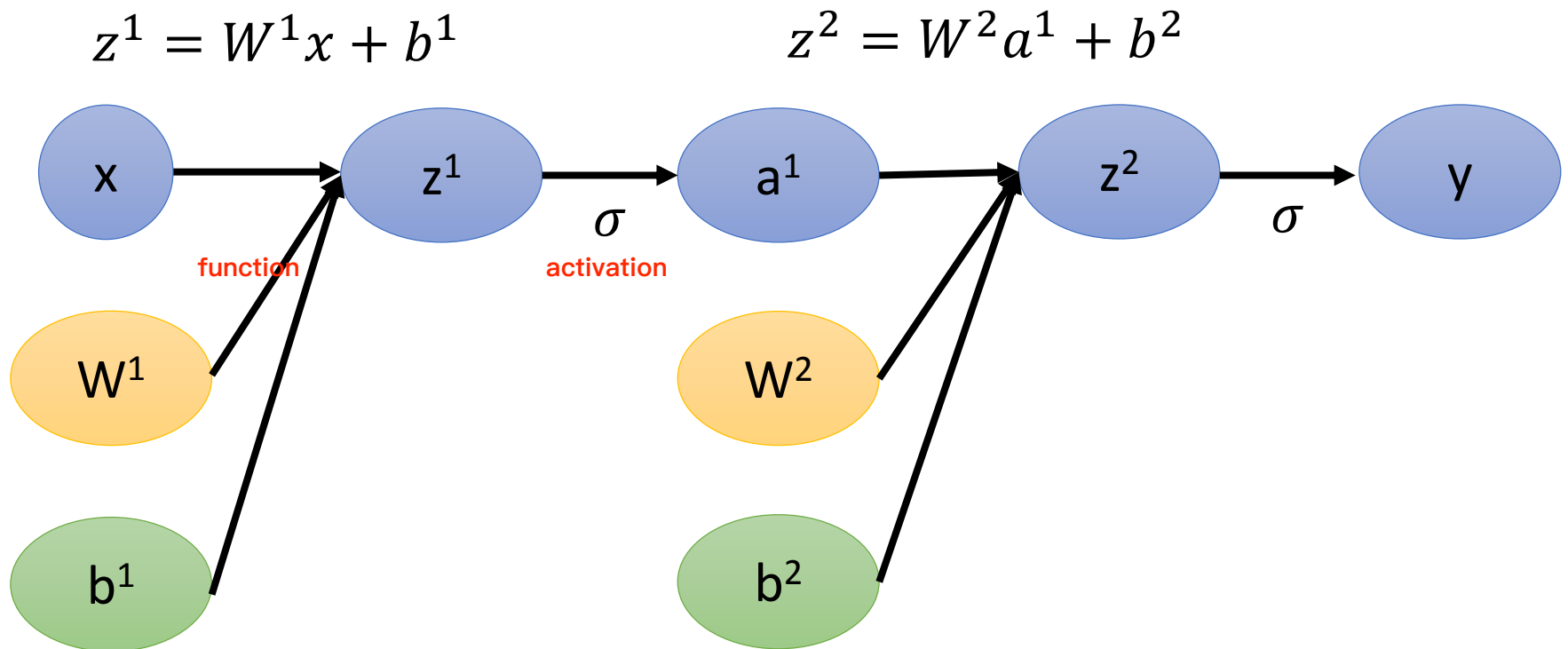
(we do not use softmax here)

## Backward Pass

$$\begin{aligned} \delta^L &= \sigma'(z^L) \bullet \nabla_y C \\ \delta^{L-1} &= \sigma'(z^{L-1}) \bullet (W^L)^T \delta^L \\ &\dots\dots \\ \delta^l &= \sigma'(z^l) \bullet (W^{l+1})^T \delta^{l+1} \\ &\dots\dots \end{aligned}$$

# Feedforward Network

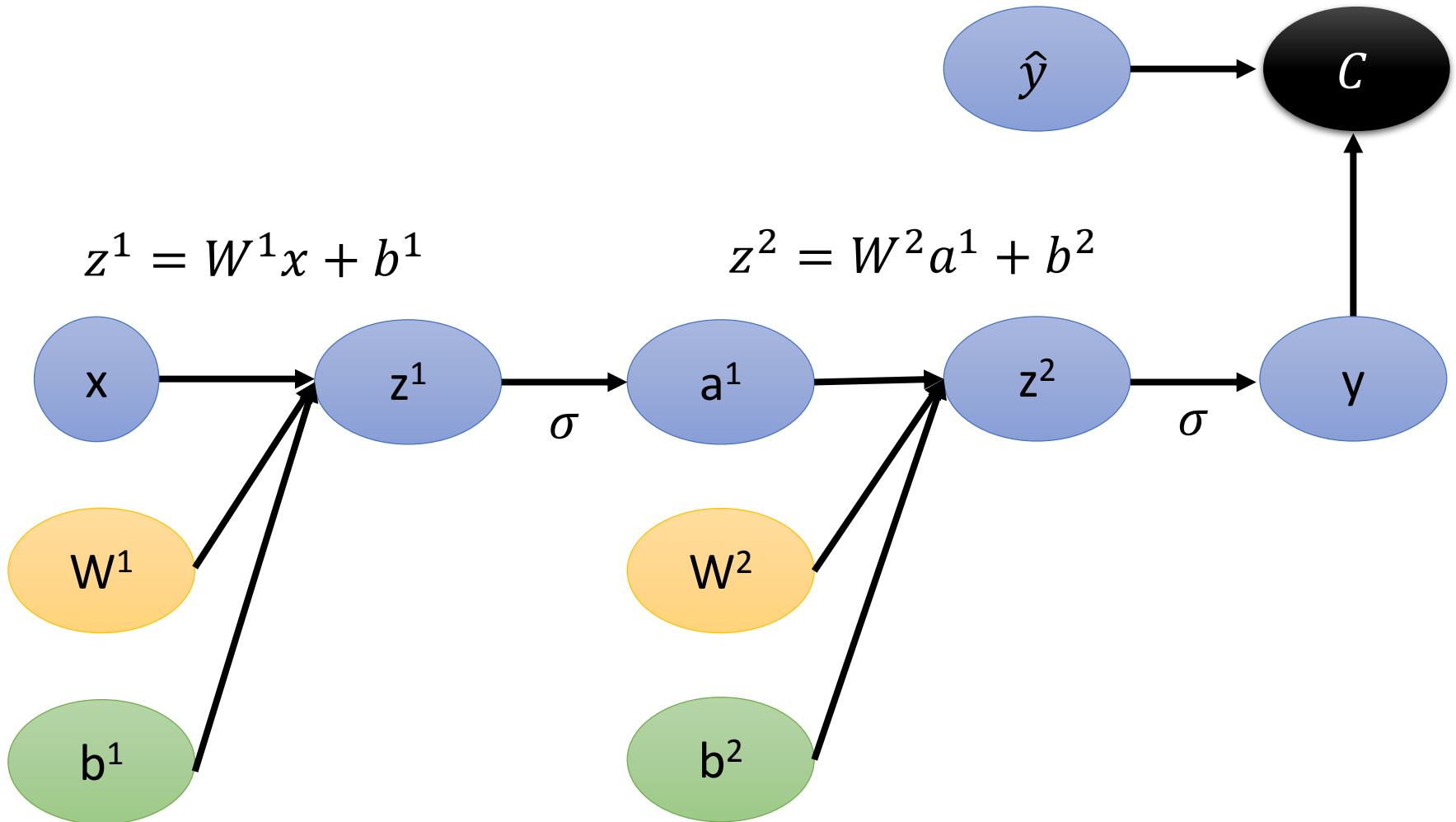
$$y = \sigma(W^L \dots \sigma(W^2 \sigma(W^1 x + b^1) + b^2) \dots + b^L)$$





# Loss Function of Feedforward Network

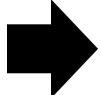
cost function  
 $C = L(y, \hat{y})$



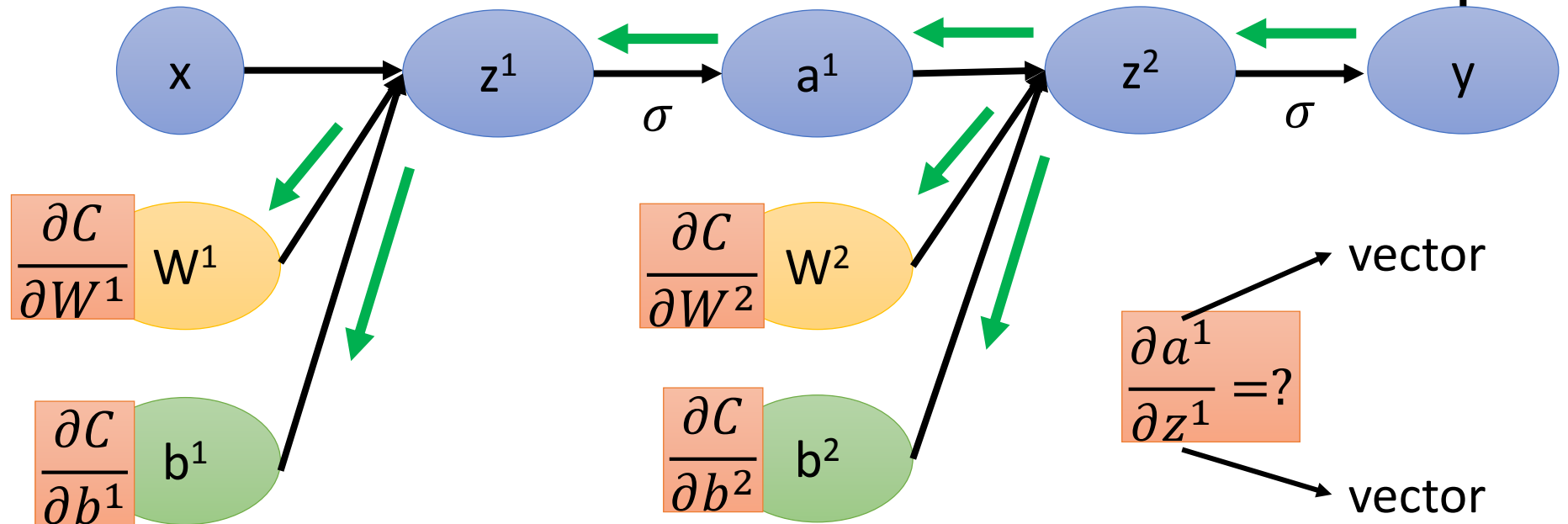
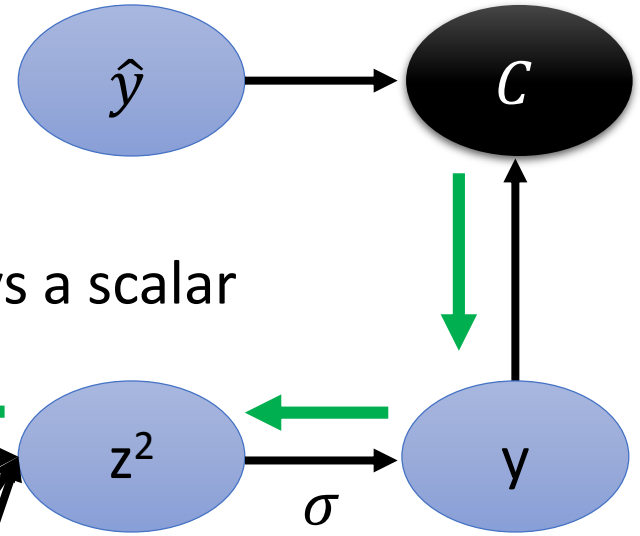
# Gradient of Cost Function

To compute the gradient ...

第一步 Computing the partial derivative on the edge

第二步 Using reverse mode  Output is always a scalar

$$C = L(y, \hat{y})$$



# Jacobian Matrix

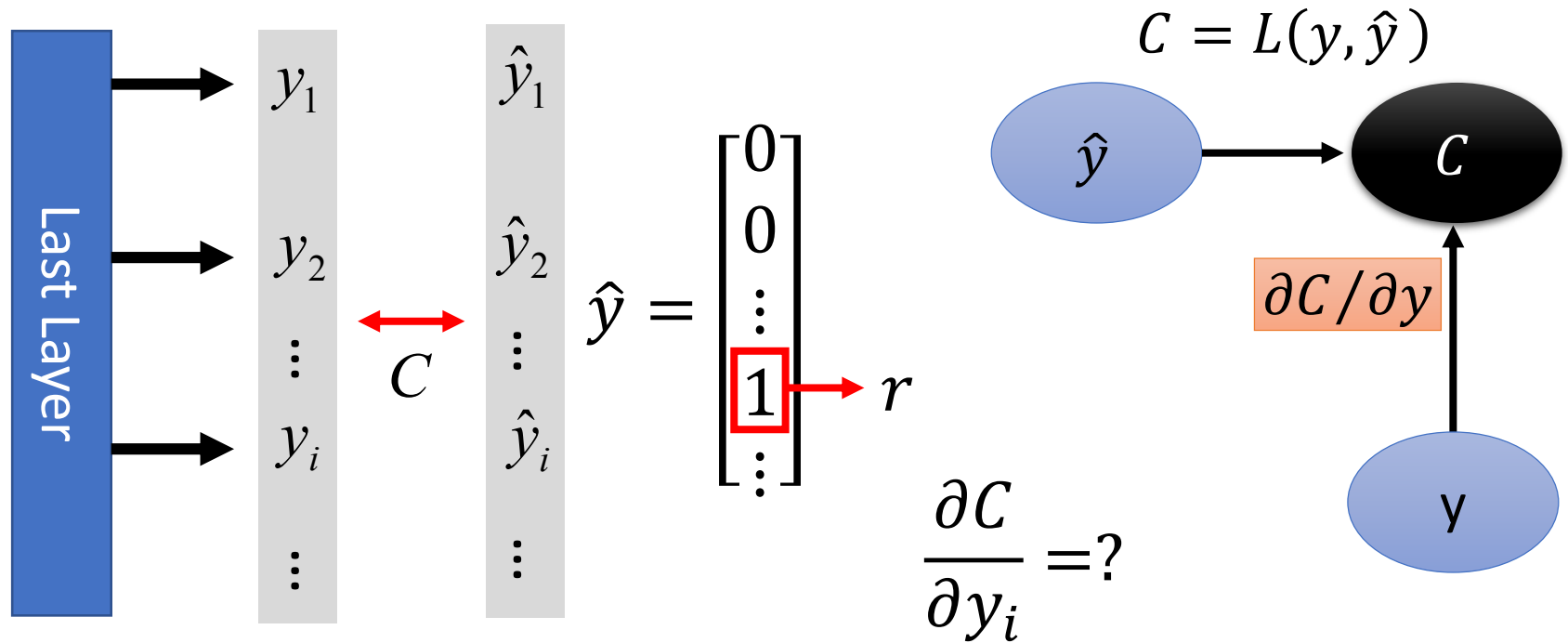
$$y = f(x) \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

$$\frac{\partial y}{\partial x} = \begin{array}{ccc} \text{dy1/dx1} & \text{dy1/dx2} & \text{dy1/dx3} \\ \text{dy2/dx1} & \text{dy2/dx2} & \text{dy2/dx3} \end{array} \left. \vphantom{\begin{array}{ccc} \text{dy1/dx1} & \text{dy1/dx2} & \text{dy1/dx3} \\ \text{dy2/dx1} & \text{dy2/dx2} & \text{dy2/dx3} \end{array}} \right\} \begin{array}{l} \text{size of y} \\ \text{size of x} \end{array}$$

## Example

$$\begin{bmatrix} x_1 + x_2 x_3 \\ 2x_3 \end{bmatrix} = f \left( \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \right) \quad \frac{\partial y}{\partial x} = \begin{bmatrix} 1 & x_3 & x_2 \\ 0 & 0 & 2 \end{bmatrix}$$

# Gradient of Cost Function



Cross Entropy:  $C = -\log y_r$

$$\frac{\partial C}{\partial y} = [ \quad ]$$

$i = r:$

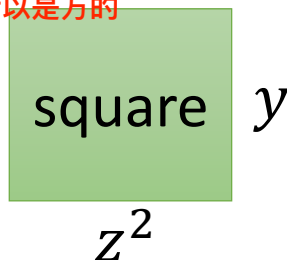
$$\partial C / \partial y_r = -1/y_r$$

$$i \neq r: \partial C / \partial y_i = 0$$

# Gradient of Cost Function

只通過一個activation function並不會改變維度，所以是方的

$\frac{\partial y}{\partial z^2}$  is a Jacobian matrix



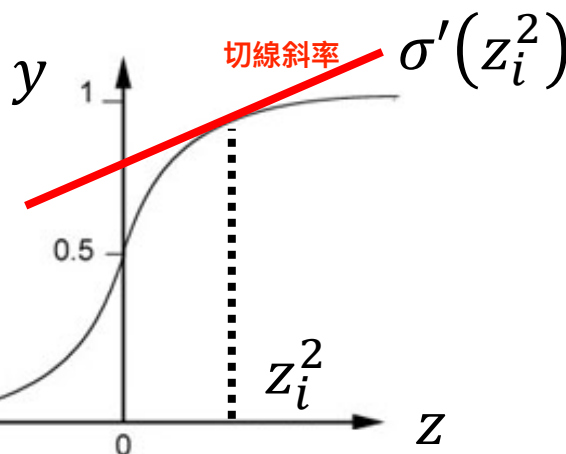
i-th row, j-th column:  $\partial y_i / \partial z_j^2$

$$i \neq j: \partial y_i / \partial z_j^2 = 0$$

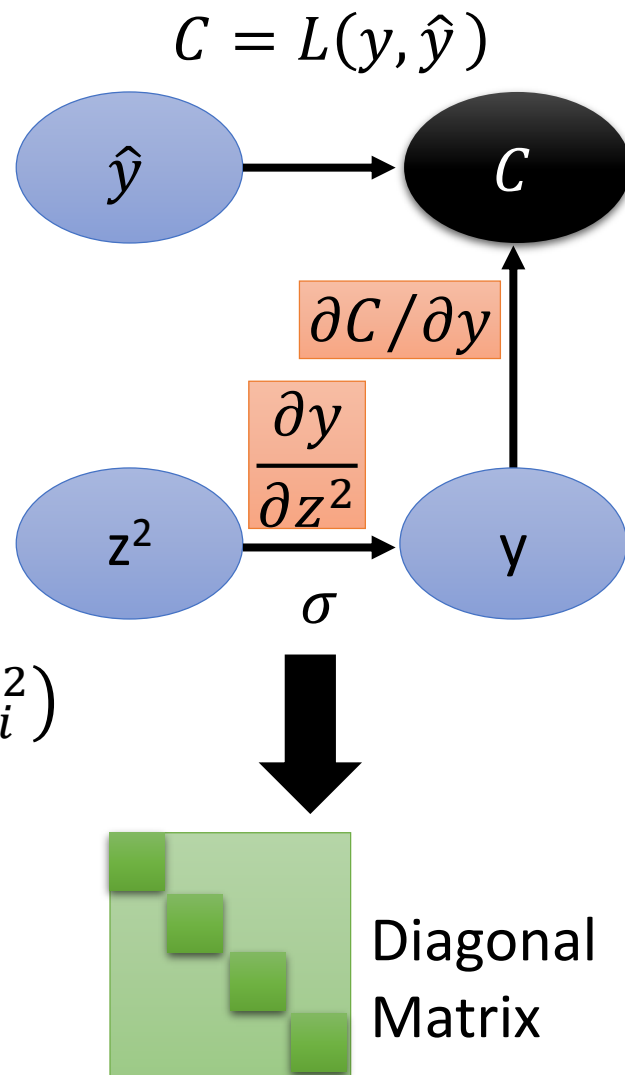
$$i = j: \partial y_i / \partial z_i^2 = \sigma'(z_i^2)$$

$$y_i = \sigma(z_i^2)$$

How about softmax? 😊



不會是diagonal，因為每個維度都會影響結果(weight sum)



$\frac{\partial z^2}{\partial a^1}$  is a Jacobian matrix

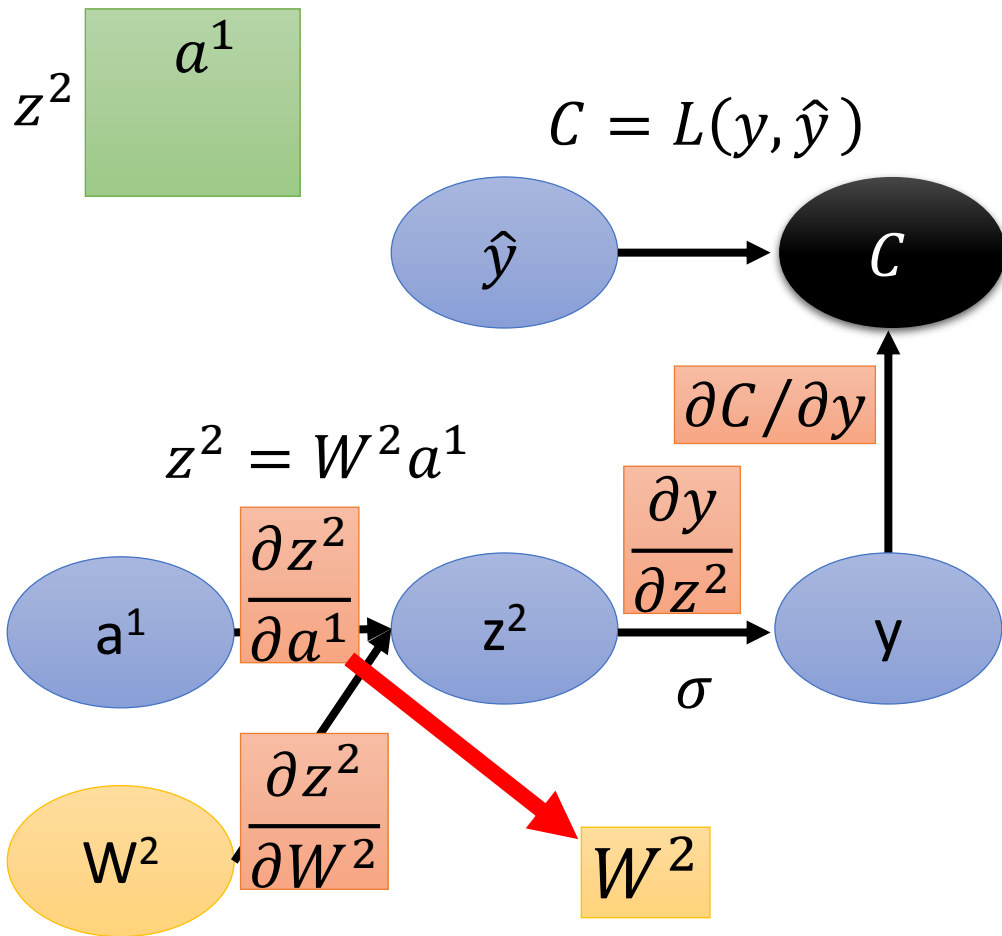
正好就是W2 !

i-th row, j-th column:

$$\frac{\partial z_i^2}{\partial a_j^1} = w_{ij}^2$$

$$z_i^2 = w_{i1}^2 a_1^1 + w_{i2}^2 a_2^1 + \dots + w_{in}^2 a_n^1$$

$$\begin{bmatrix} z_1^l \\ z_2^l \\ \vdots \\ z_i^l \\ \vdots \end{bmatrix} = \begin{bmatrix} w_{11}^l & w_{12}^l & \dots \\ w_{21}^l & w_{22}^l & \dots \\ \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} a_1^{l-1} \\ a_2^{l-1} \\ \vdots \\ a_i^{l-1} \\ \vdots \end{bmatrix} + \begin{bmatrix} b_1^l \\ b_2^l \\ \vdots \\ b_i^l \\ \vdots \end{bmatrix}$$



$$\frac{\partial z^2}{\partial W^2} = m$$

mxn

因為被拉直了

(j-1)xn+k

i

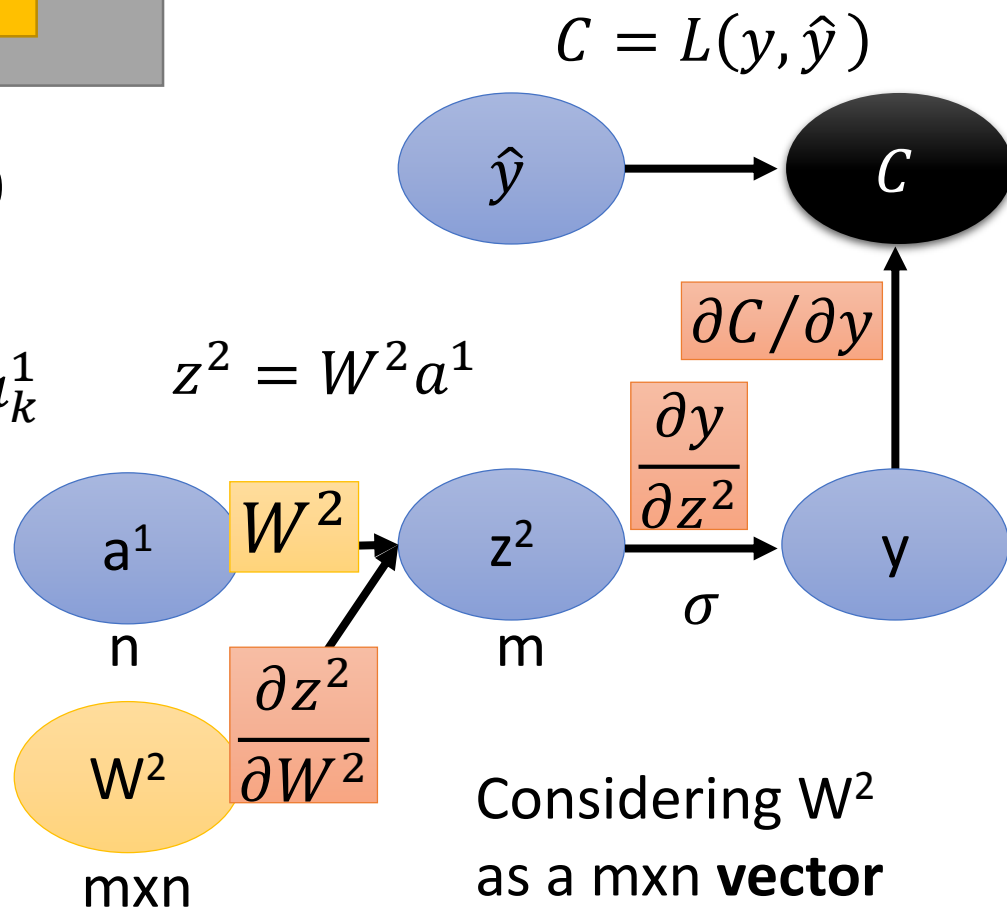
$\frac{\partial z_i^2}{\partial W_{jk}^2}$

$$\frac{\partial z_i^2}{\partial W_{jk}^2} = ? \quad i \neq j: \frac{\partial z_i^2}{\partial W_{jk}^2} = 0$$

$$i = j: \frac{\partial z_i^2}{\partial W_{ik}^2} = a_k^1$$

$$z_i^2 = w_{i1}^2 a_1^1 + w_{i2}^2 a_2^1 + \dots + w_{in}^2 a_n^1$$

$$\begin{bmatrix} z_1^l \\ z_2^l \\ \vdots \\ z_i^l \\ \vdots \end{bmatrix} = \begin{bmatrix} w_{11}^l & w_{12}^l & \dots \\ w_{21}^l & w_{22}^l & \dots \\ \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} a_1^{l-1} \\ a_2^{l-1} \\ \vdots \\ a_i^{l-1} \\ \vdots \end{bmatrix} + \begin{bmatrix} b_1^l \\ b_2^l \\ \vdots \\ b_i^l \\ \vdots \end{bmatrix}$$



照理來說是3維的tensor，但這邊把二維拉成一維的vector

(considering  $\partial z^2 / \partial W^2$  as a tensor makes thing easier)

$$\frac{\partial z^2}{\partial W^2} = m$$

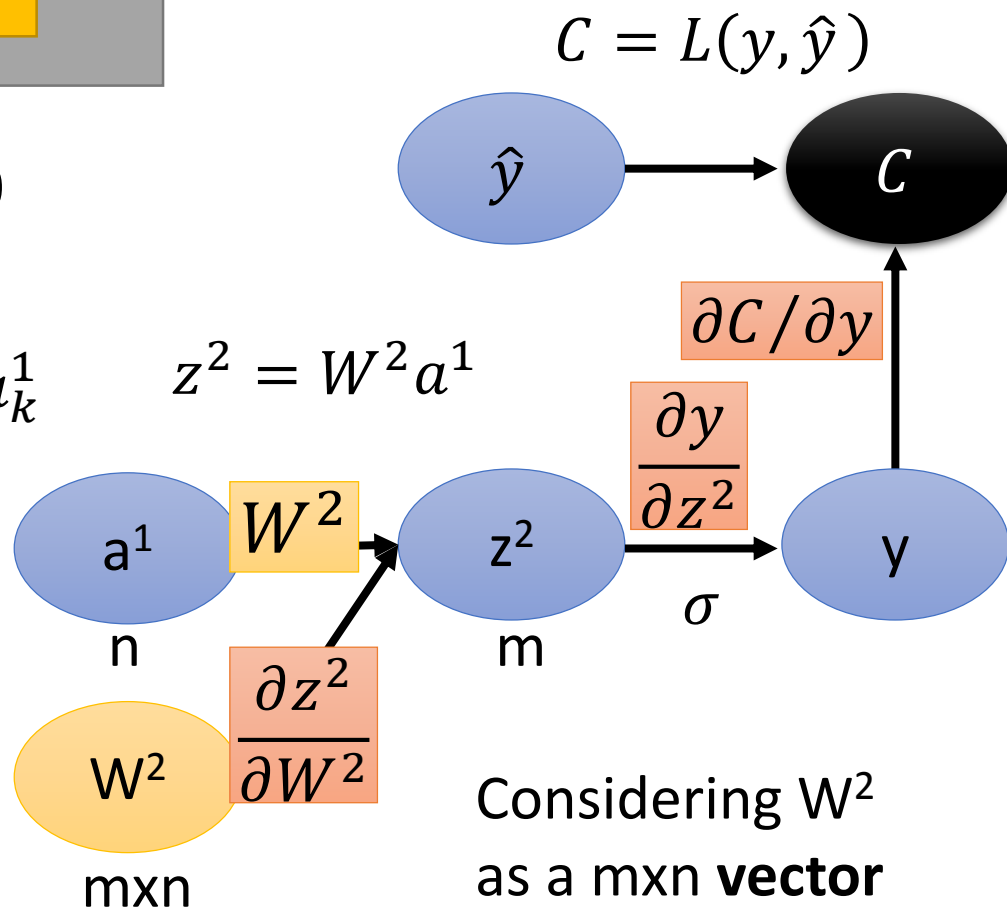
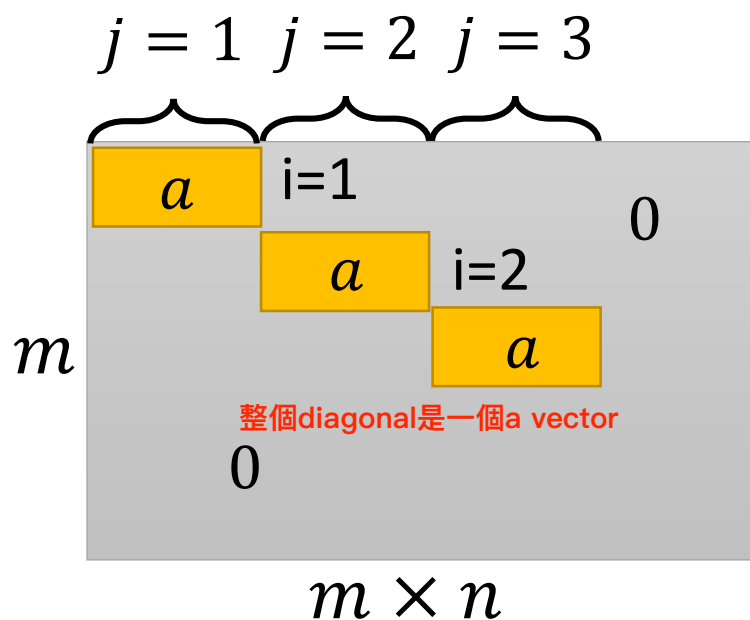
mxn

(j-1)xn+k
i

$$\frac{\partial z_i^2}{\partial W_{jk}^2}$$

$$\frac{\partial z_i^2}{\partial W_{jk}^2} = ? \quad i \neq j: \frac{\partial z_i^2}{\partial W_{jk}^2} = 0$$

$$i = j: \frac{\partial z_i^2}{\partial W_{ik}^2} = a_k^1$$

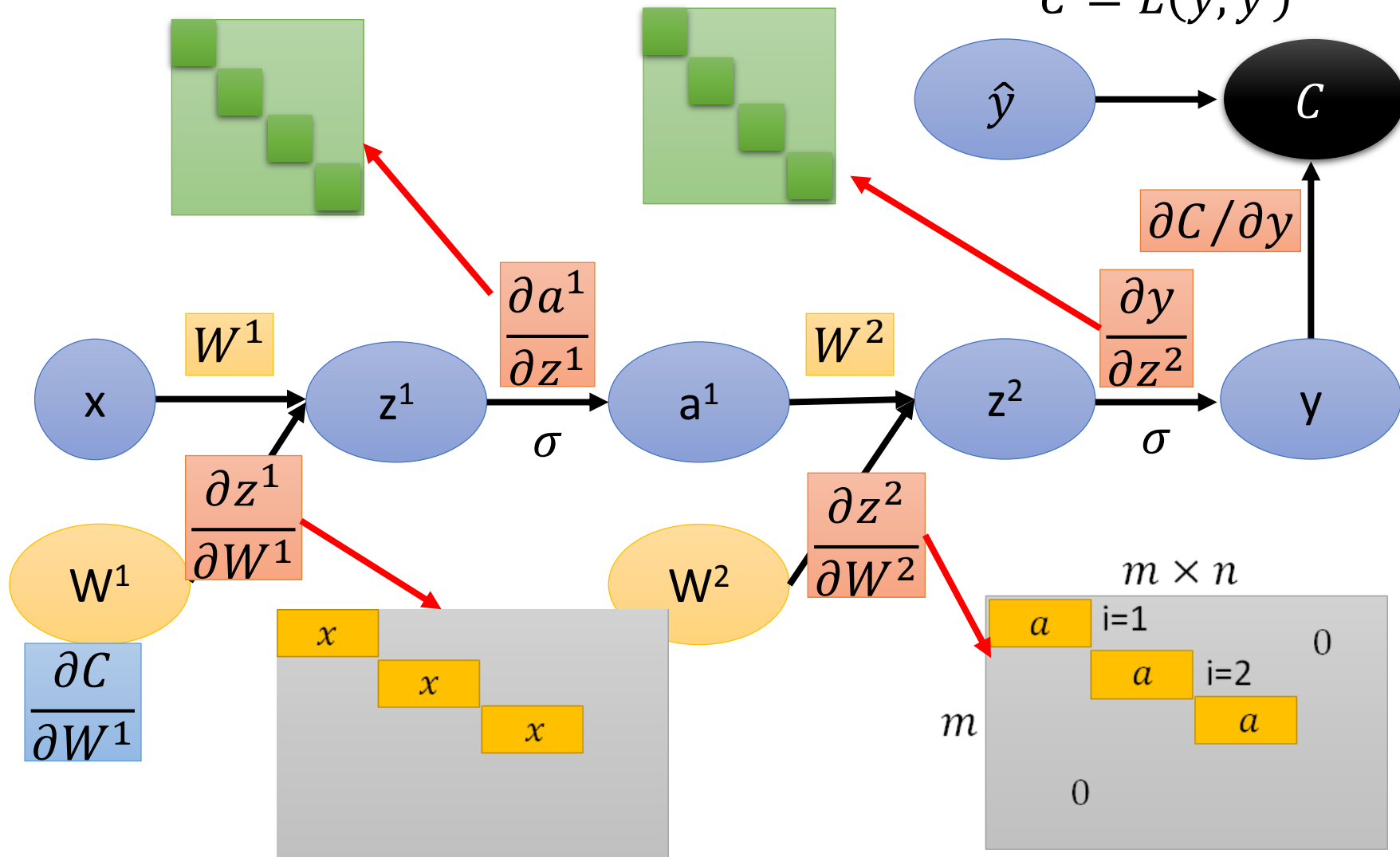




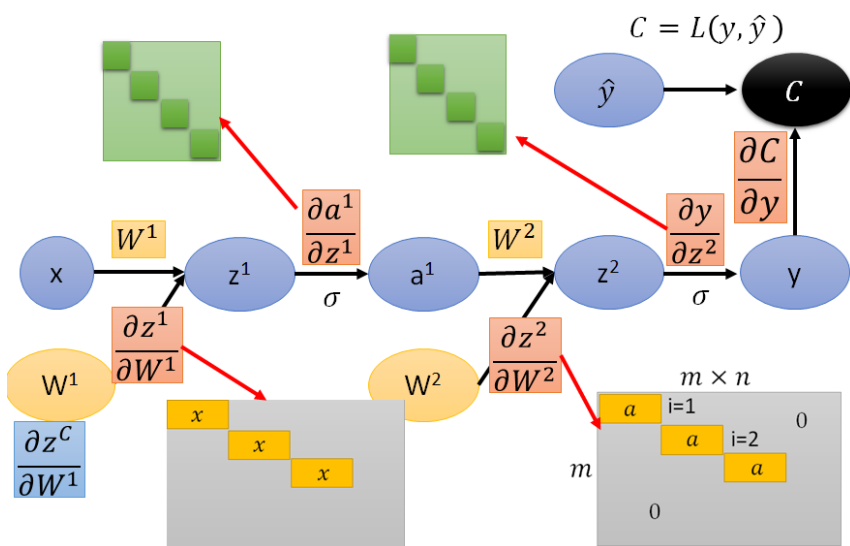
$$\frac{\partial \mathcal{C}}{\partial W^1} = \frac{\partial \mathcal{C}}{\partial y} \frac{\partial y}{\partial z^2} W^2 \frac{\partial a^1}{\partial z^1} \frac{\partial z^1}{\partial W^1} = [\cdots \frac{\partial \mathcal{C}}{\partial W_{ij}^1} \cdots]$$

rows=1

$$\mathcal{C} = L(y, \hat{y})$$



# Question



$$\frac{\partial C}{\partial w_{ij}^l} = \frac{\partial z_i^l}{\partial w_{ij}^l} \frac{\partial C}{\partial z_i^l}$$

Error signal

## Forward Pass

$$\begin{aligned} z^1 &= W^1 x + b^1 \\ a^1 &= \sigma(z^1) \\ &\dots\dots \\ z^{l-1} &= W^{l-1} a^{l-2} + b^{l-1} \\ a^{l-1} &= \sigma(z^{l-1}) \end{aligned}$$

## Backward Pass

$$\begin{aligned} \delta^L &= \sigma'(z^L) \bullet \nabla_y C \\ \delta^{L-1} &= \sigma'(z^{L-1}) \bullet (W^L)^T \delta^L \\ &\dots\dots \\ \delta^l &= \sigma'(z^l) \bullet (W^{l+1})^T \delta^{l+1} \\ &\dots\dots \end{aligned}$$

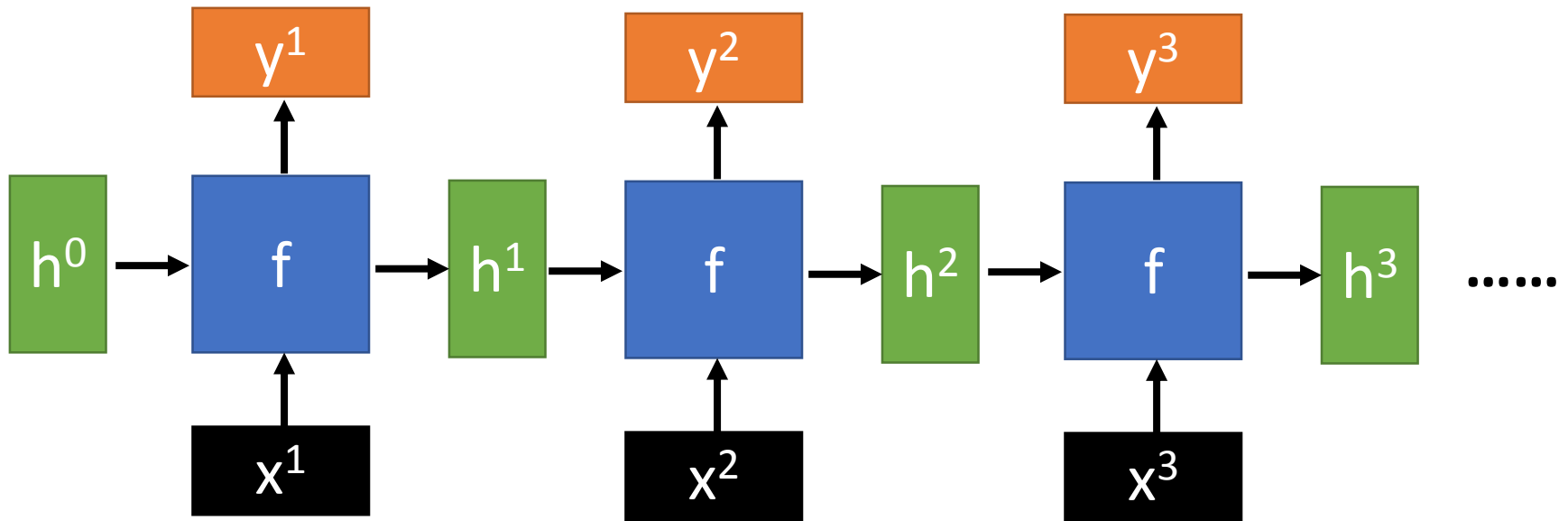
Q: Only backward pass for computational graph?

computational graph還是需要forward pass，雖然上述方法只有提到backward pass，但是a,x等值還是需要先forward算出來

Q: Do we get the same results from the two different approaches?

# Computational Graph for Recurrent Network

# Recurrent Network



$$y^t, h^t = f(x^t, h^{t-1}; W^i, W^h, W^o)$$

$$h^t = \sigma(W^i x^t + W^h h^{t-1})$$

$$y^t = \text{softmax}(W^o h^t)$$

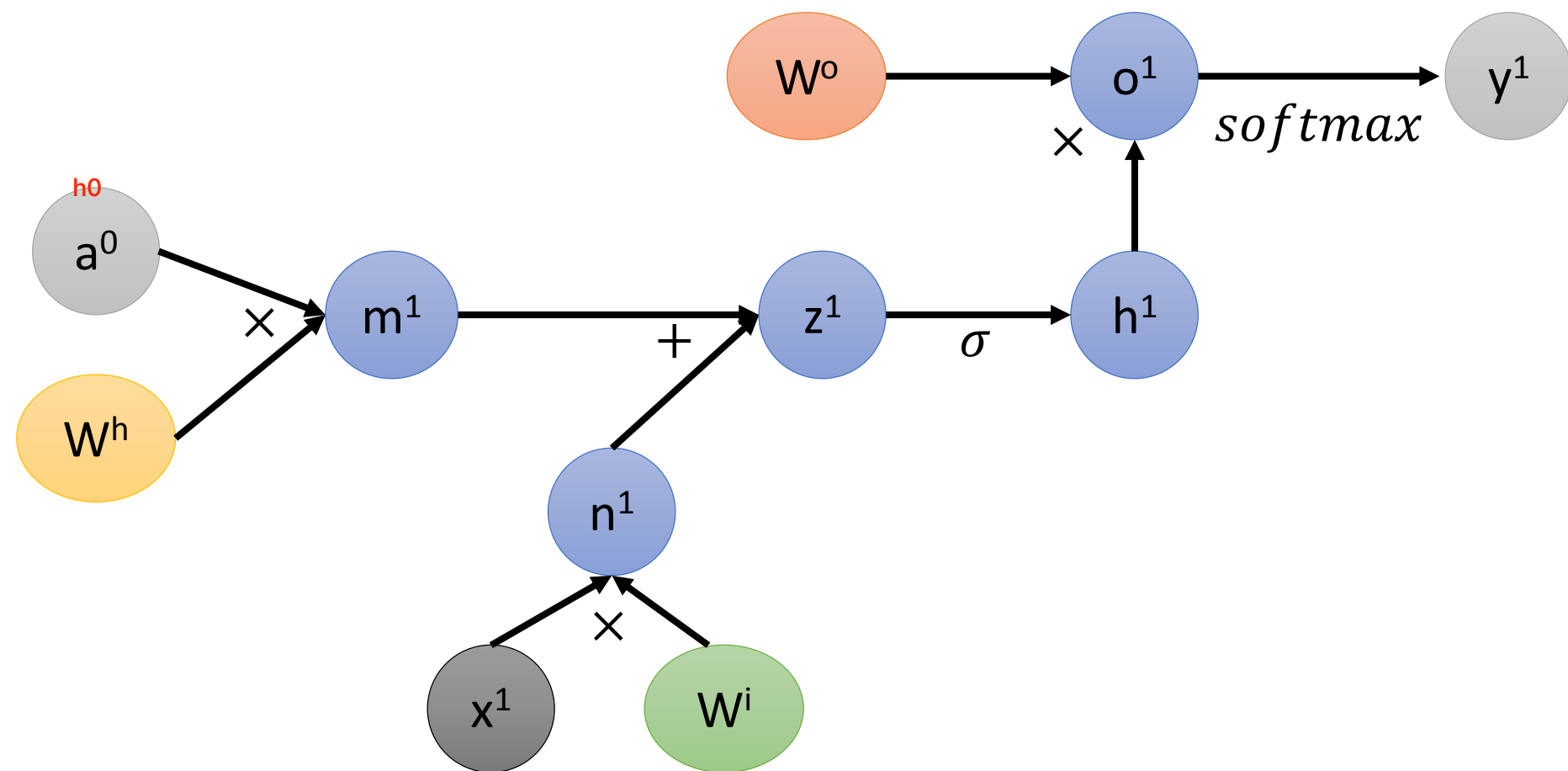
(biases are ignored here)

# Recurrent Network

$$y^t, h^t = f(x^t, h^{t-1}; W^i, W^h, W^o)$$

$$a^t = \sigma(W^i x^t + W^h h^{t-1})$$

$$y^t = \text{softmax}(W^o h^t)$$

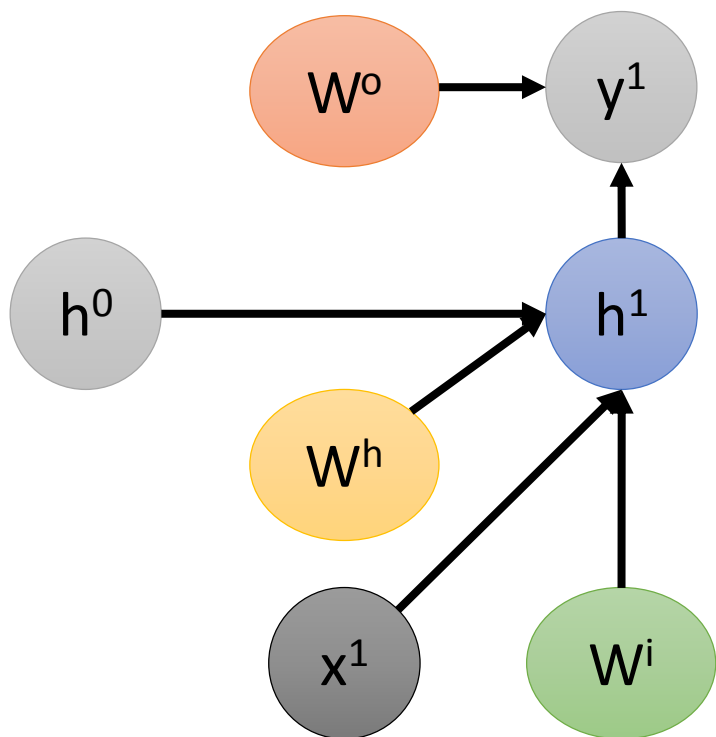


# Recurrent Network

$$y^t, h^t = f(x^t, h^{t-1}; W^i, W^h, W^o)$$

$$a^t = \sigma(W^i x^t + W^h h^{t-1})$$

$$y^t = \text{softmax}(W^o h^t)$$



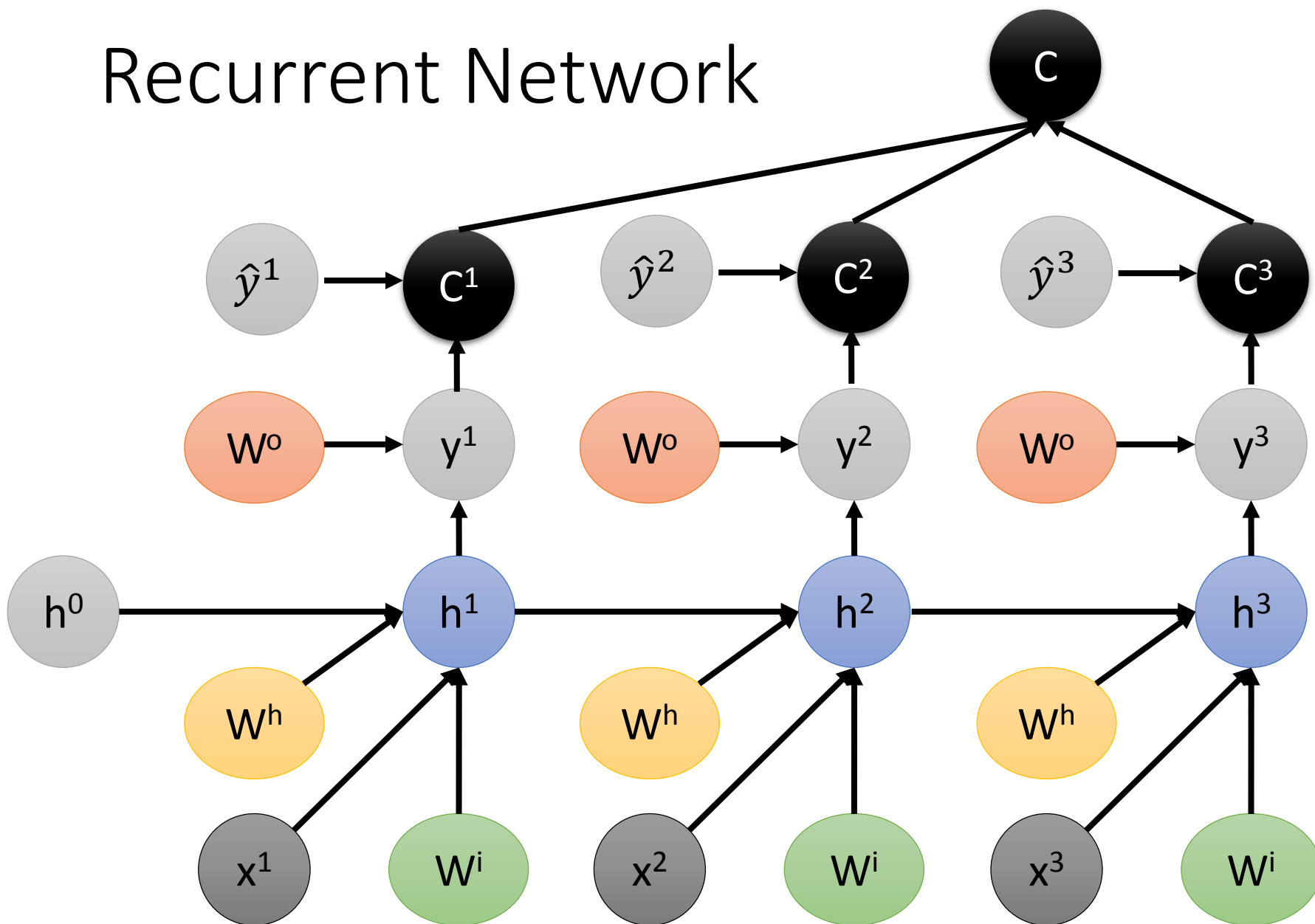
$$y^t = \text{softmax}(W^o h^t)$$

簡化RNN cell 表示法

$$h^t = \sigma(W^i x^t + W^h h^{t-1})$$

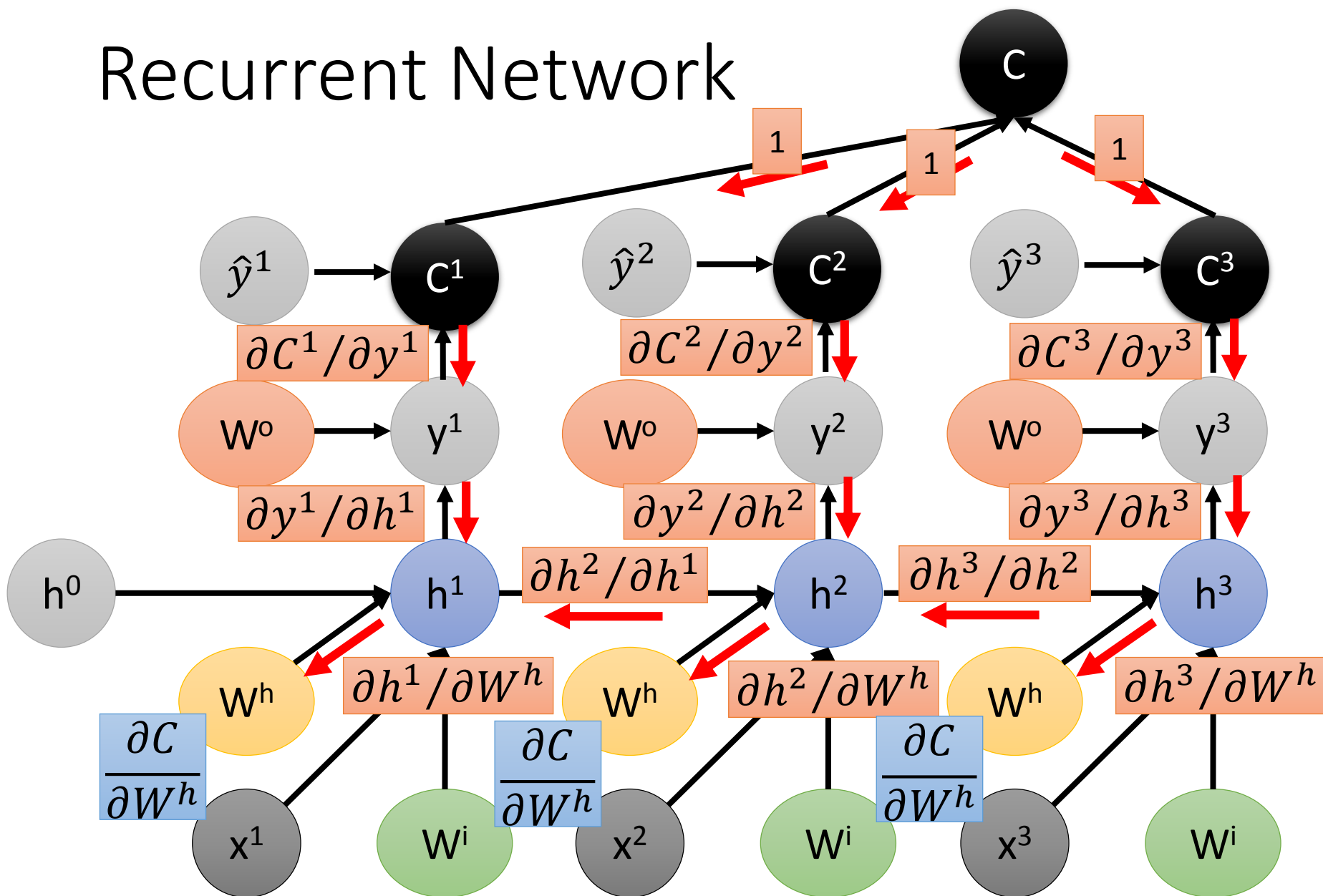
# Recurrent Network

$$C = C^1 + C^2 + C^3$$



# Recurrent Network

$$C = C^1 + C^2 + C^3$$

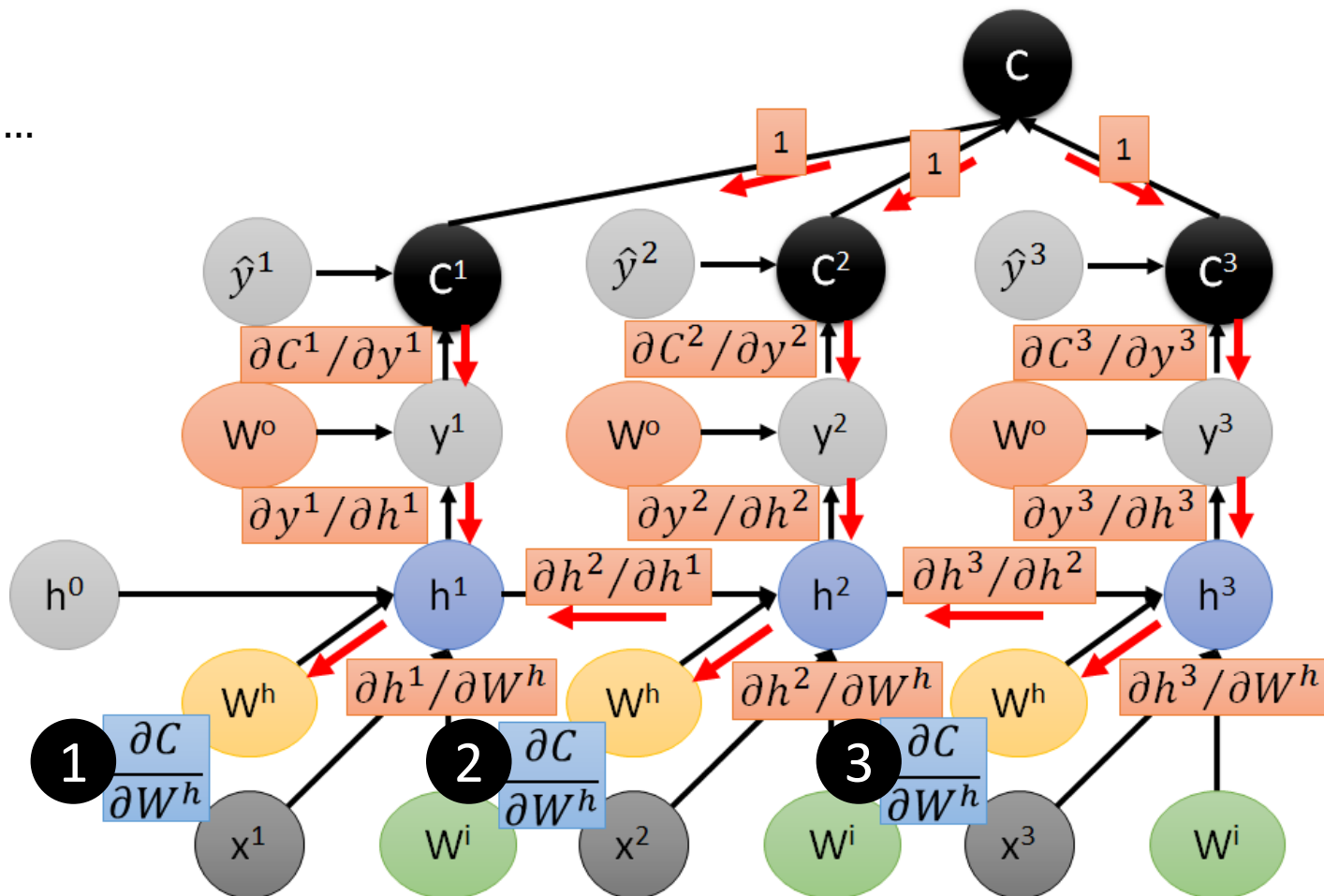




$$\textcircled{1} \frac{\partial C}{\partial W^h} = \left[ \overset{\text{第一條path}}{\frac{\partial C^1}{\partial y^1} \frac{\partial y^1}{\partial h^1}} + \overset{\text{第二條path}}{\frac{\partial C^2}{\partial y^2} \frac{\partial y^2}{\partial h^2} \frac{\partial h^2}{\partial h^1}} + \overset{\text{第三條path}}{\frac{\partial C^3}{\partial y^3} \frac{\partial y^3}{\partial h^3} \frac{\partial h^3}{\partial h^2} \frac{\partial h^2}{\partial h^1}} \right] \frac{\partial h^1}{\partial W^h}$$

$$\textcircled{2} \frac{\partial C}{\partial W^h} = \dots \dots \quad \frac{\partial C}{\partial W^h} = \textcircled{1} \frac{\partial C}{\partial W^h} + \textcircled{2} \frac{\partial C}{\partial W^h} + \textcircled{3} \frac{\partial C}{\partial W^h}$$

$$\textcircled{3} \frac{\partial C}{\partial W^h} = \dots \dots$$



# Reference

- Textbook: Deep Learning
  - Chapter 6.5
- Calculus on Computational Graphs: Backpropagation
  - <https://colah.github.io/posts/2015-08-Backprop/>
- On chain rule, computational graphs, and backpropagation
  - <http://outlace.com/Computational-Graph/>

# Acknowledgement

- 感謝 翁丞世 同學找到投影片上的錯誤