

Computing Gradient

Hung-yi Lee

李宏毅

Introduction

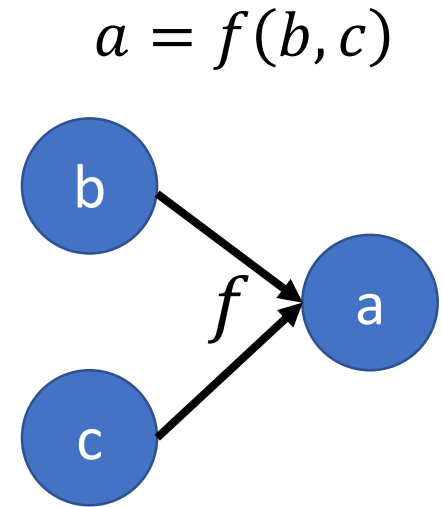
- Backpropagation: an efficient way to compute the gradient
- Prerequisite
 - Backpropagation for feedforward net:
 - http://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/Lecture/DNN%20backprop.ecm.mp4/index.html
 - Simple version: <https://www.youtube.com/watch?v=ibJpTrp5mcE>
 - Backpropagation through 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 可以看作是一种描述 Model 的语言，一个节点 (node) 表示一个变量、数值、向量、矩阵，一个边表示一个操作、简单的函数。

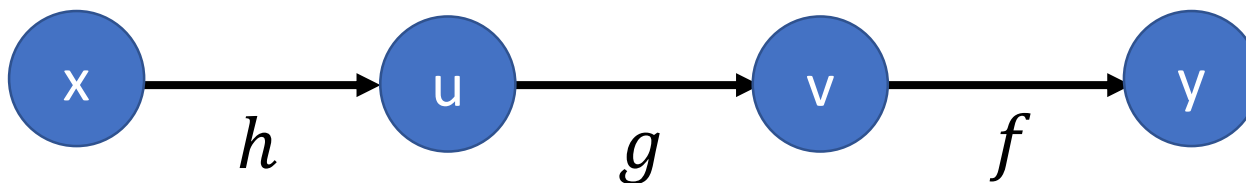
Computational Graph

- A “language” describing a function
 - **Node**: variable (scalar, vector, tensor)
 - **Edge**: operation (simple function)



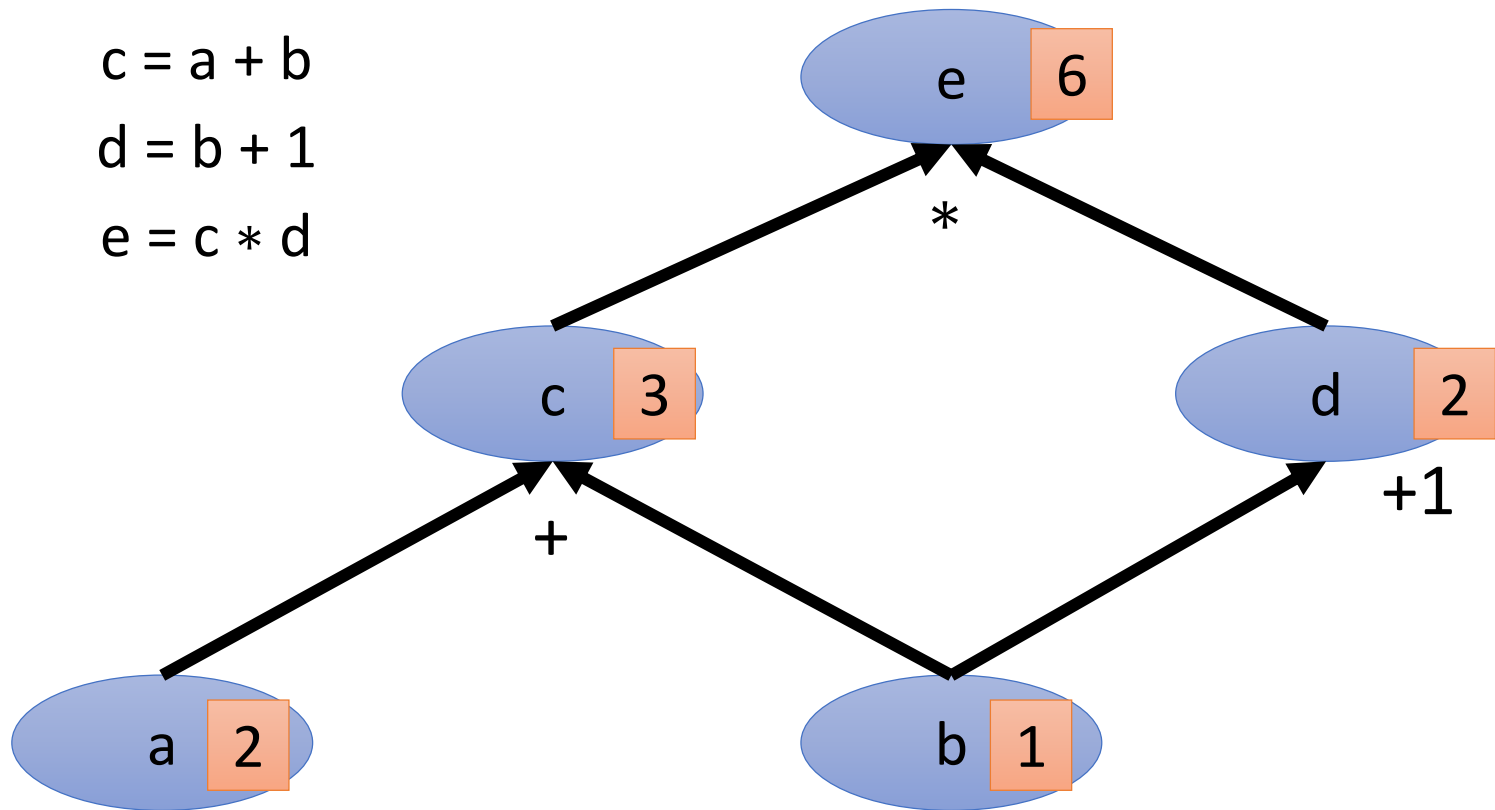
Example $y = f(g(h(x)))$

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



Computational Graph

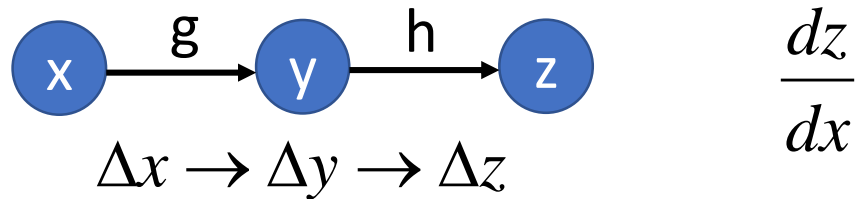
- Example: $e = (a+b) * (b+1)$



Review: Chain Rule

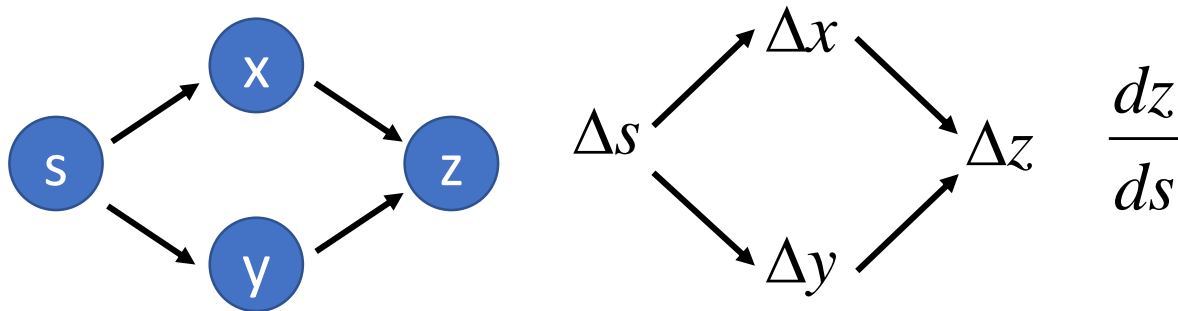
Case 1

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



Case 2

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



Computational Graph

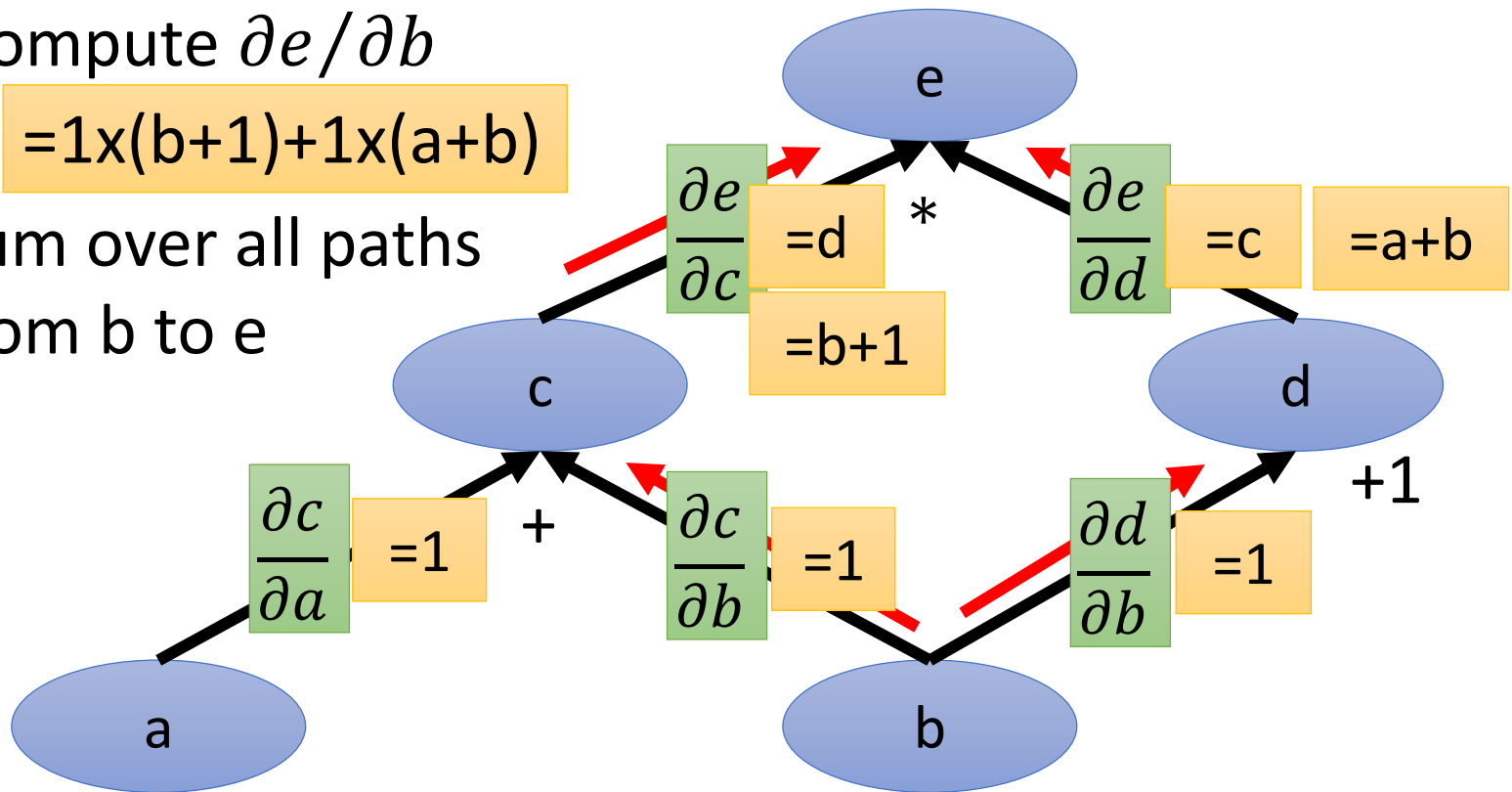
- Example: $e = (a+b) * (b+1)$

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

Compute $\partial e / \partial b$

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

Sum over all paths
from b to e

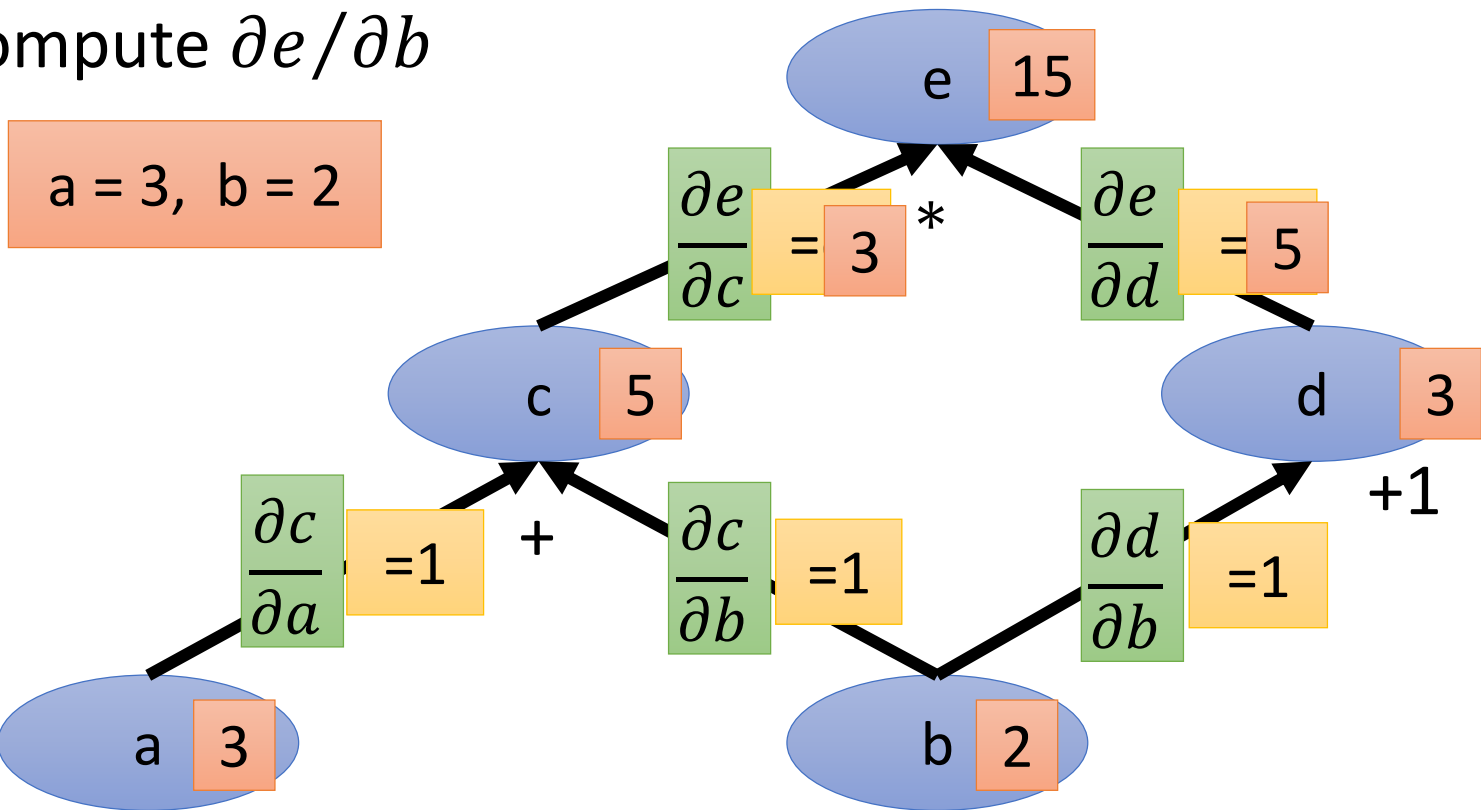


Computational Graph

- 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 $\partial e / \partial b$

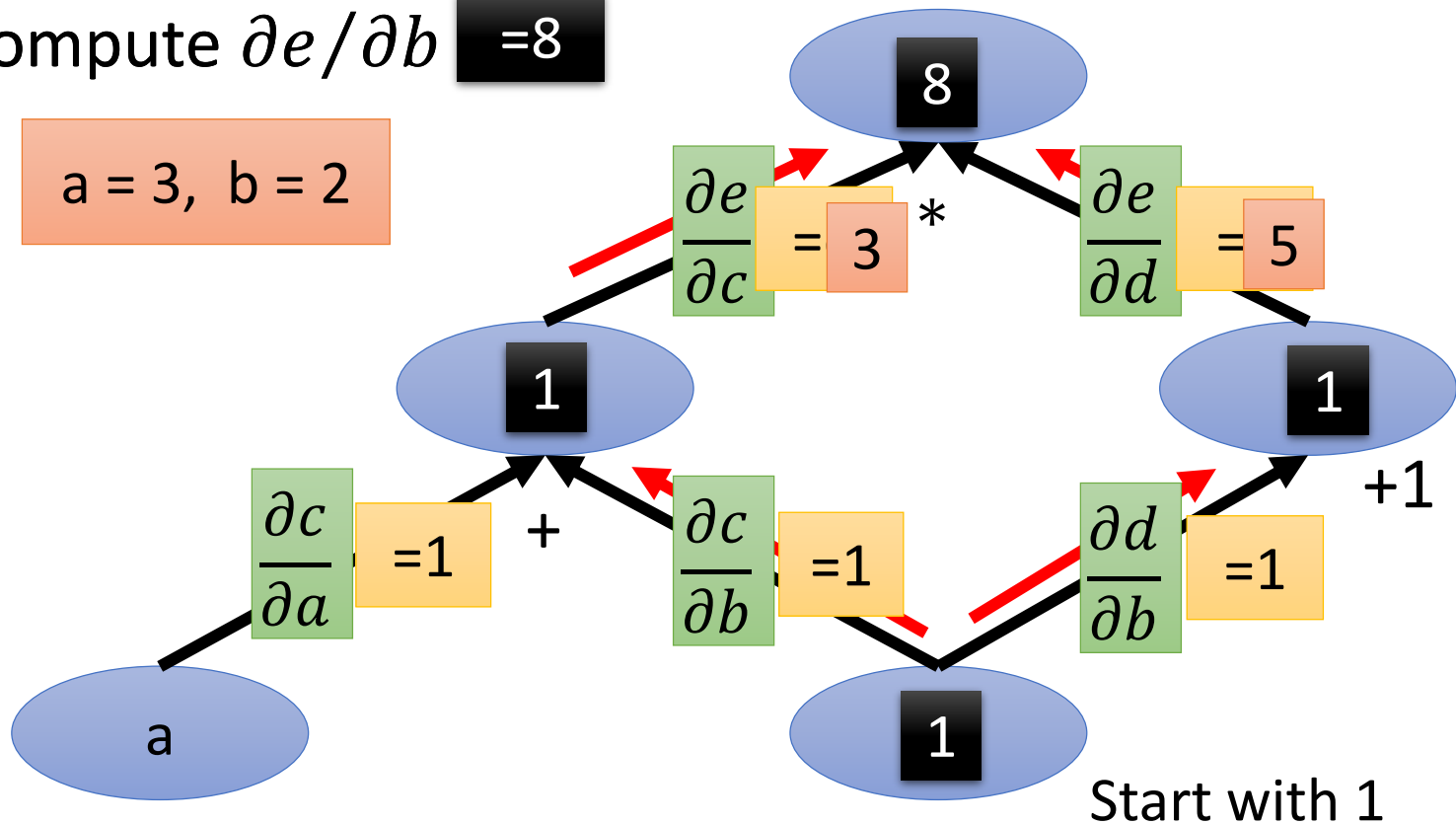


Computational Graph

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

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

$a = 3, b = 2$

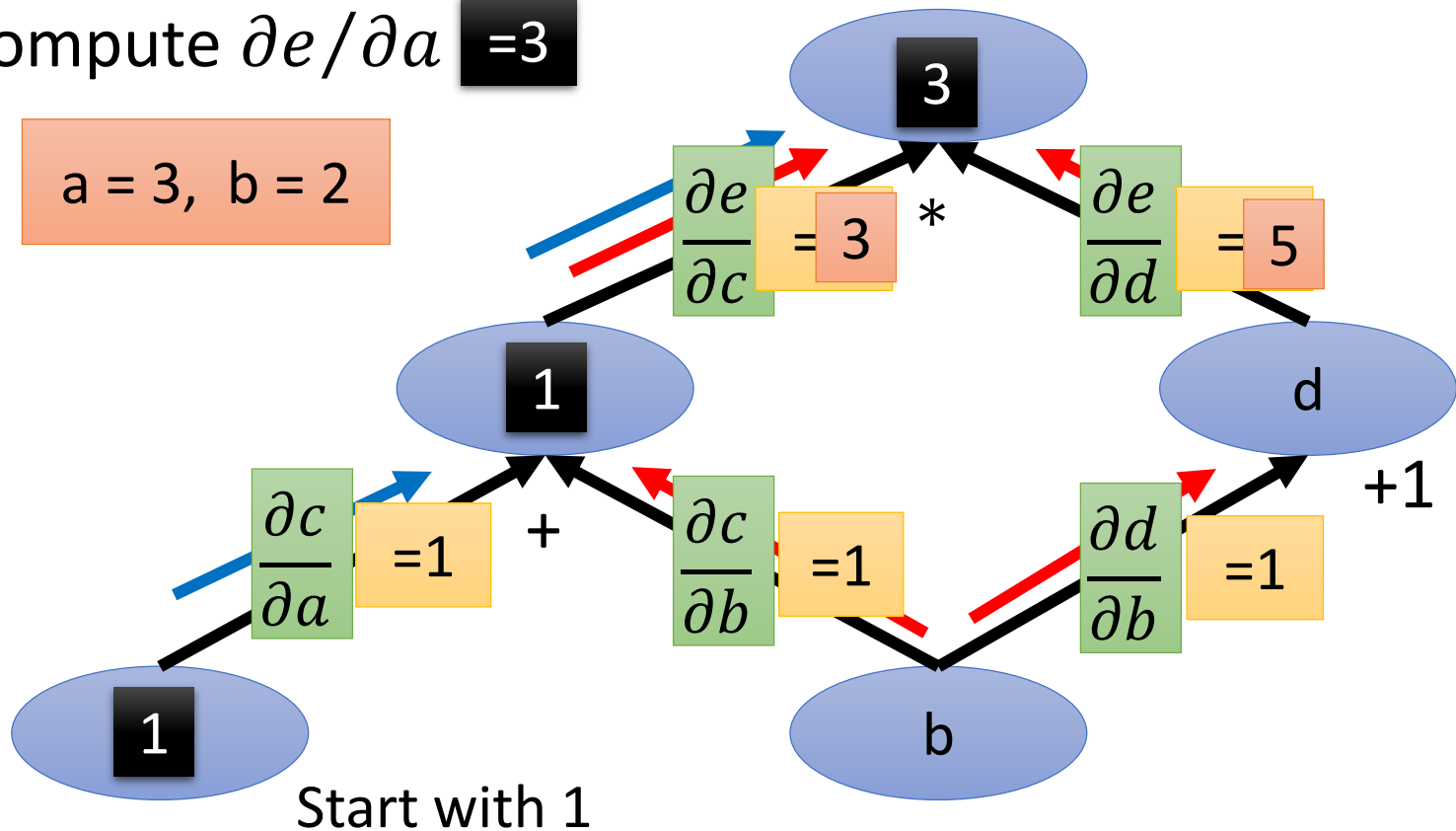


Computational Graph

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

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

$a = 3, b = 2$



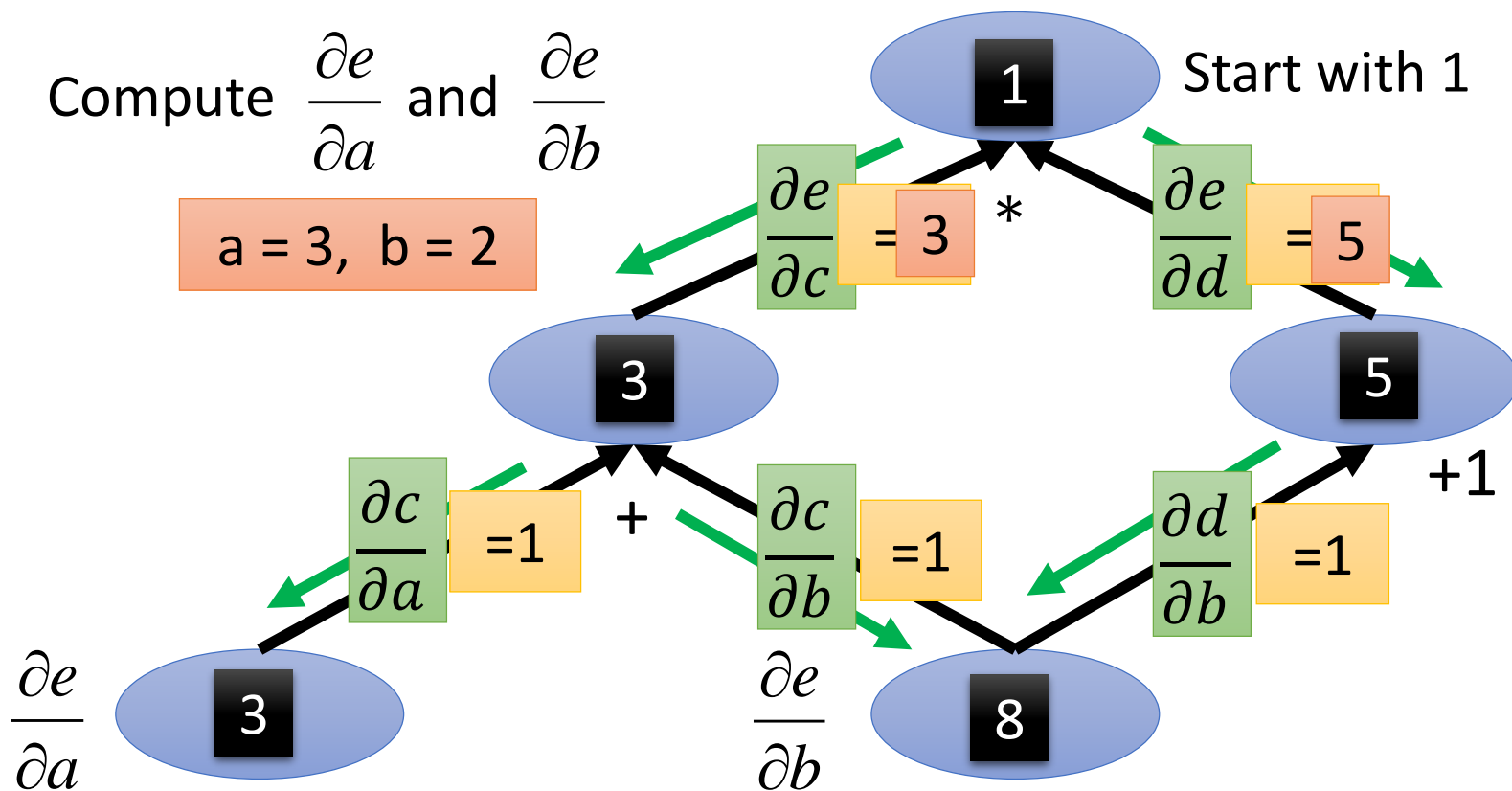
求解偏微分值，通过反向模式更好些。
要求出每一个节点变量对下一个节点变量的微分值，然后一次将每条线上的微分值乘起来。注意要把所有的路线全部加起来。

Computational Graph

Reverse mode

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

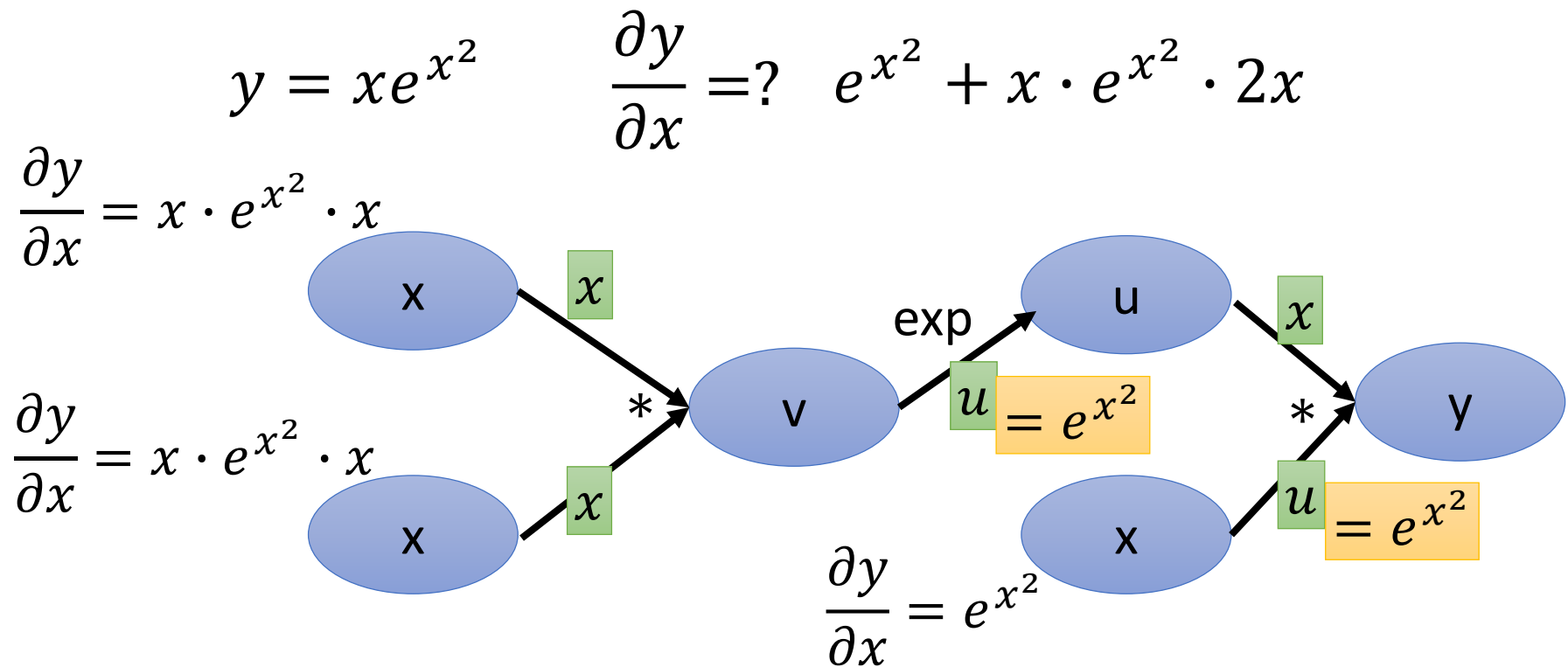
What is the benefit?



对于相同的参数，在微观上要将其分为不一样的变量，如下面的三个x当作x1，x2，x3. 宏观上又是相同的，最后将对x的求导全加起来。

Computational Graph

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

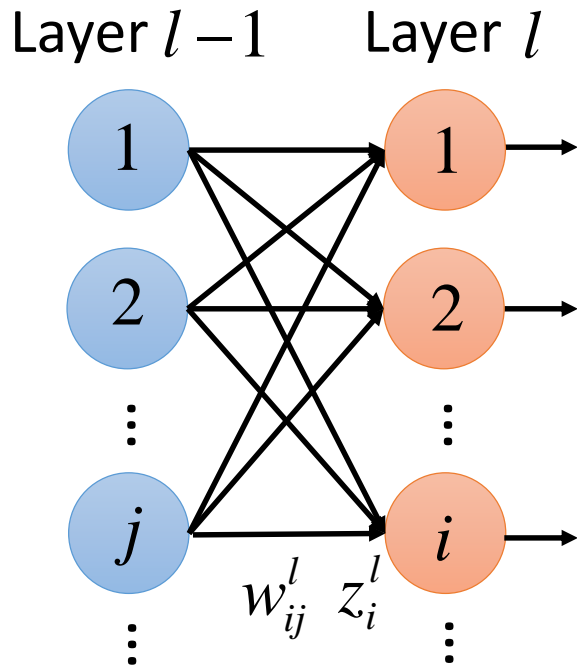


Computational Graph for Feedforward Net

Review: Backpropagation

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

Error signal



$$\begin{cases} a_j^{l-1} & l > 1 \\ x_j & l = 1 \end{cases}$$

Forward Pass

$$z^1 = W^1 x + b^1$$

$$a^1 = \sigma(z^1)$$

.....

$$z^{l-1} = W^{l-1} a^{l-2} + b^{l-1}$$

$$a^{l-1} = \sigma(z^{l-1})$$

Backward Pass

$$\delta^L = \sigma'(z^L) \bullet \nabla_y C$$

$$\delta^{L-1} = \sigma'(z^{L-1}) \bullet (W^L)^T \delta^L$$

.....

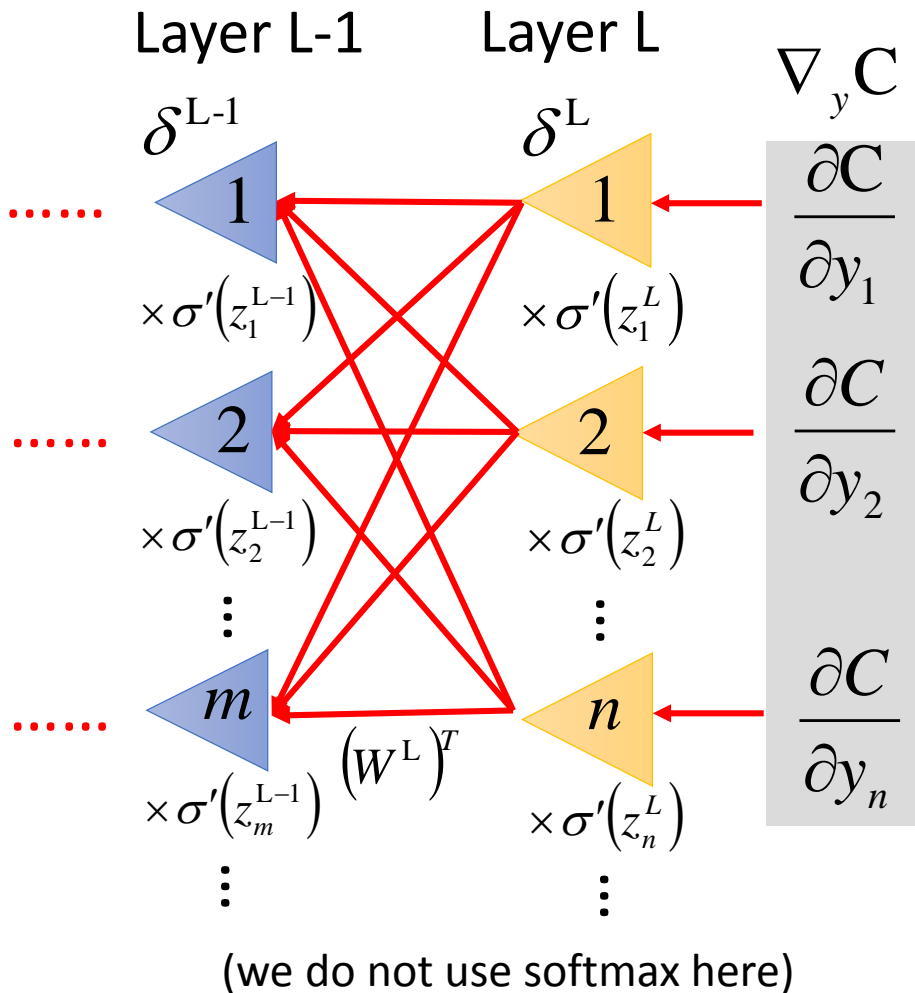
$$\delta^l = \sigma'(z^l) \bullet (W^{l+1})^T \delta^{l+1}$$

.....

Review: Backpropagation

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

Error signal



Backward Pass

$$\delta^L = \sigma'(z^L) \bullet \nabla_y C$$

$$\delta^{L-1} = \sigma'(z^{L-1}) \bullet (W^L)^T \delta^L$$

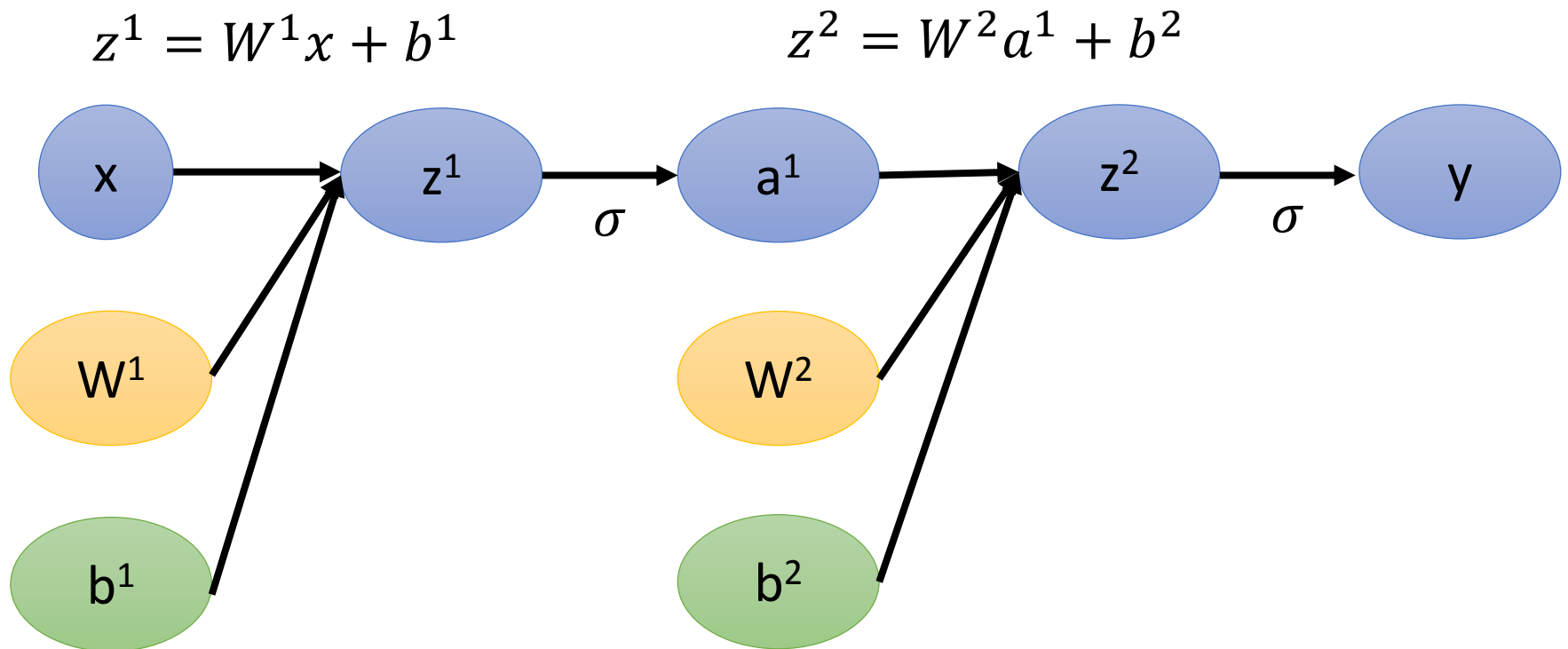
.....

$$\delta^l = \sigma'(z^l) \bullet (W^{l+1})^T \delta^{l+1}$$

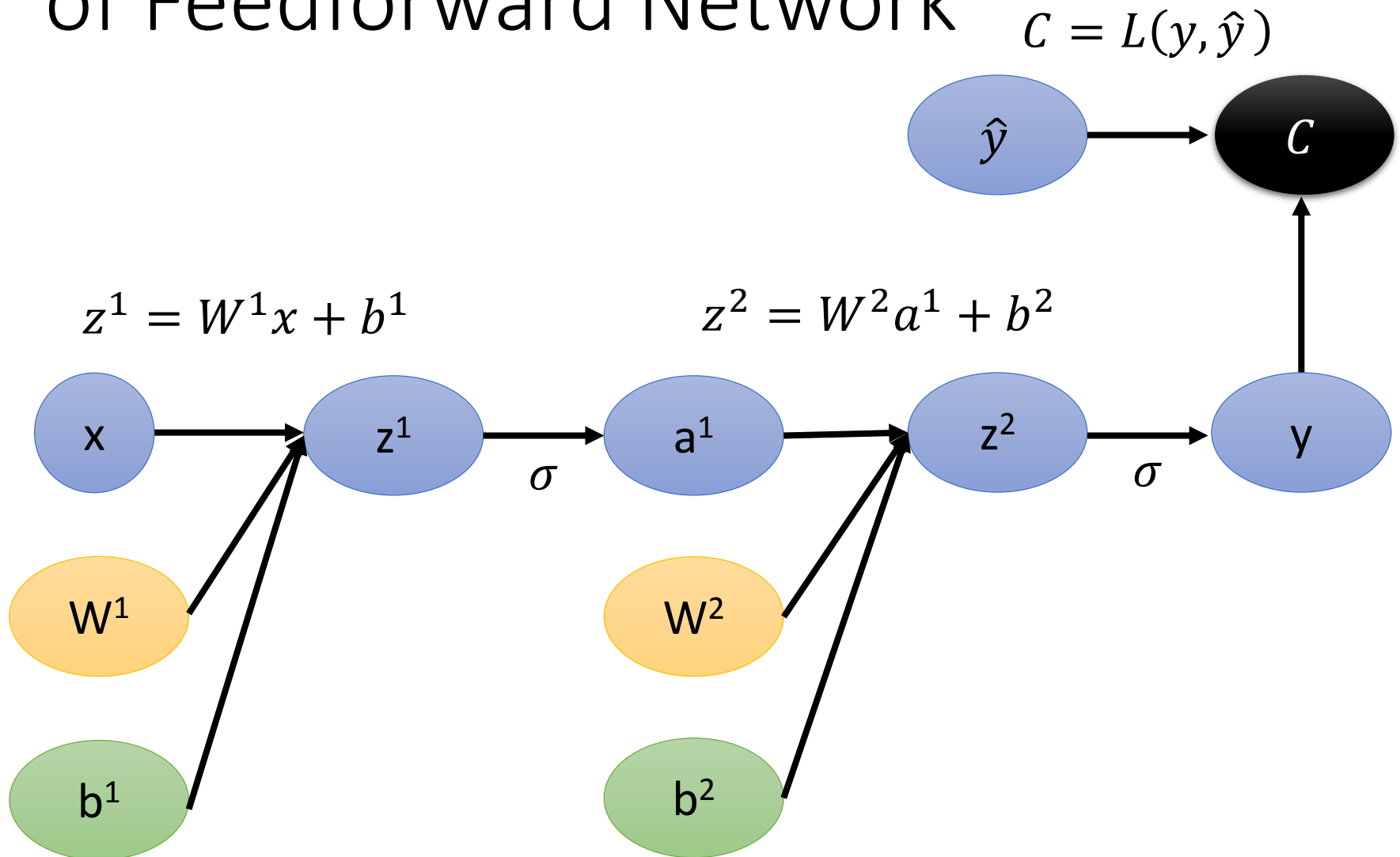
.....

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



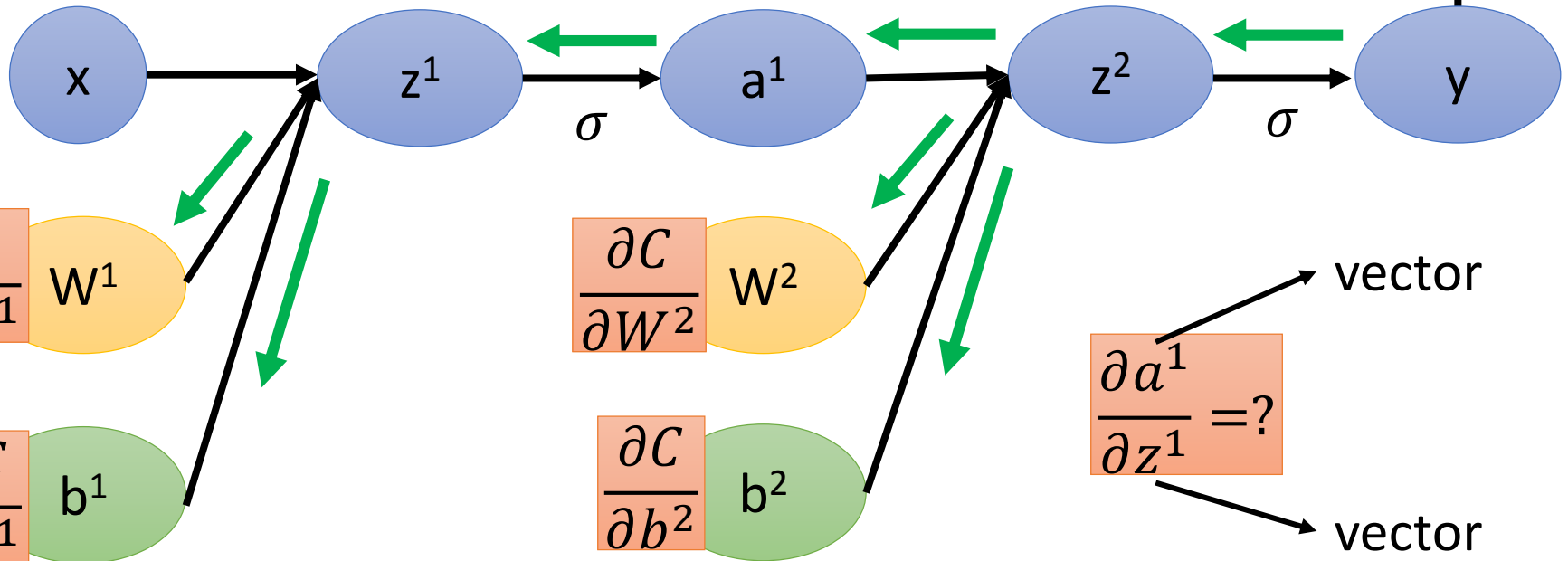
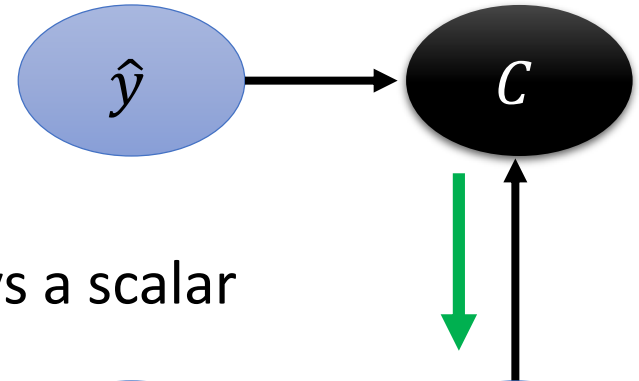
Gradient of Cost Function

To compute the gradient ...

Computing the partial derivative
on the edge

Using reverse mode  Output is always a scalar

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



Jacobian Matrix

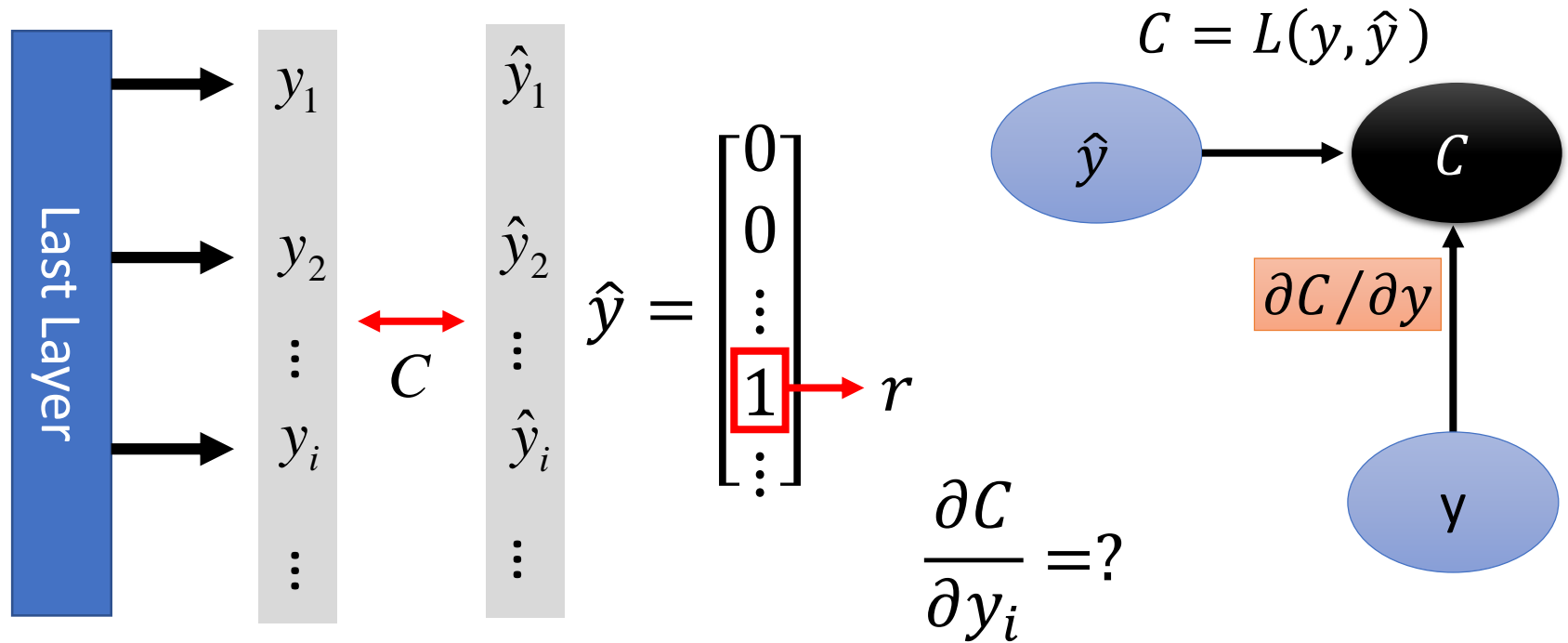
$$y = f(x) \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

$$\frac{\partial y}{\partial x} = \underbrace{\hspace{15em}}_{\text{size of } x} \underbrace{\hspace{1em}}_{\text{size of } y}$$

Example

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

Gradient of Cost Function



Cross Entropy: $C = -\log y_r$

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

$i = r:$

$$\frac{\partial C}{\partial y_r} = -1/y_r$$

$$i \neq r: \frac{\partial C}{\partial y_i} = 0$$

Gradient of Cost Function

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

square y
 z^2

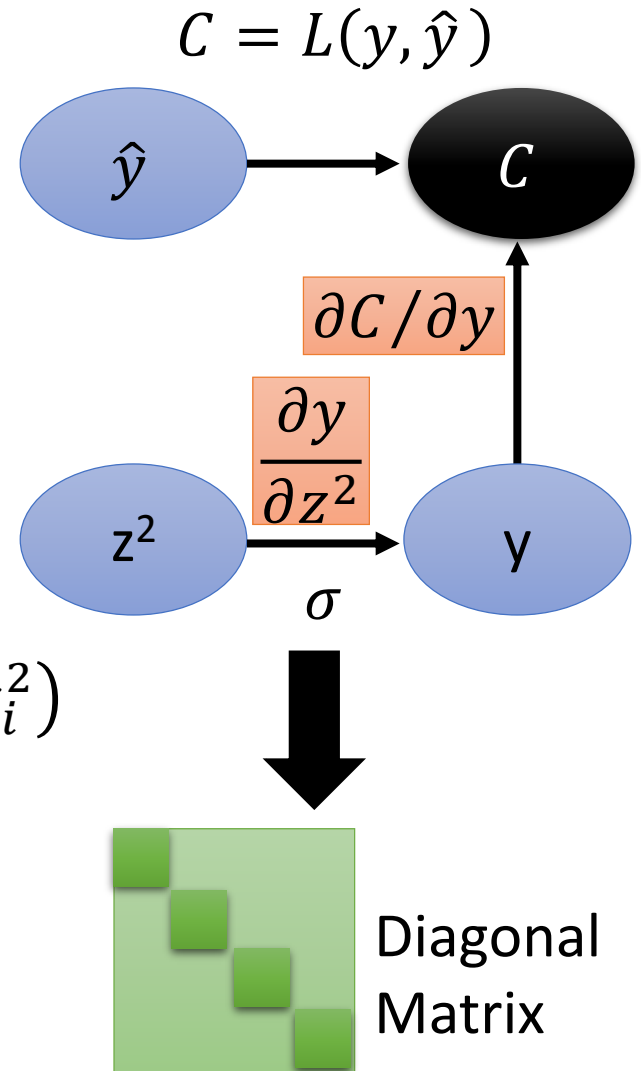
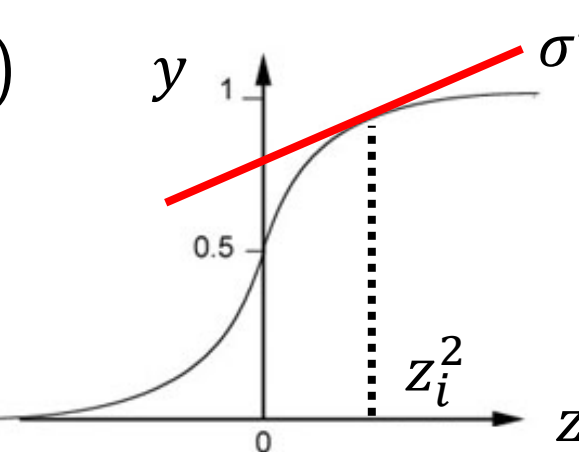
i-th row, j-th column: $\partial y_i / \partial z_j^2$

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

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

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

How about
softmax? 😊



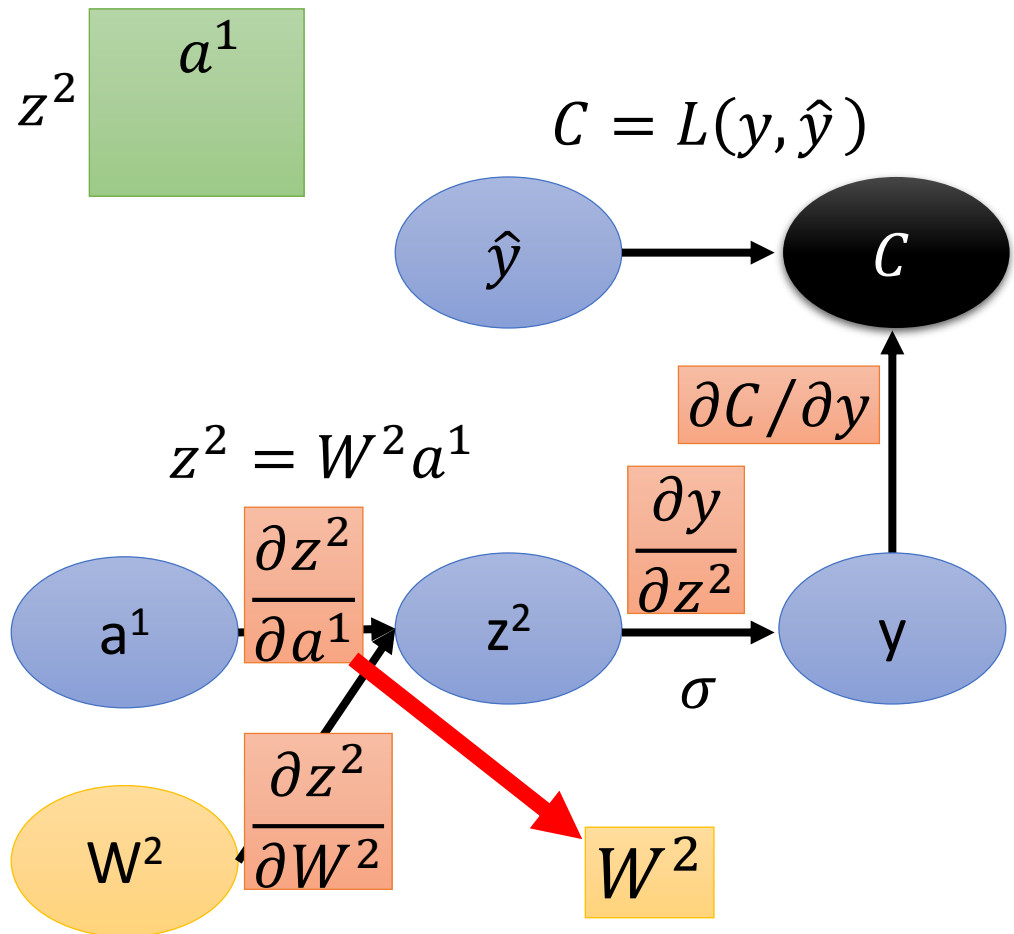
$\frac{\partial z^2}{\partial a^1}$ is a Jacobian matrix

i-th row, j-th column:

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

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

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



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

mxn

(j-1)xn+k

i

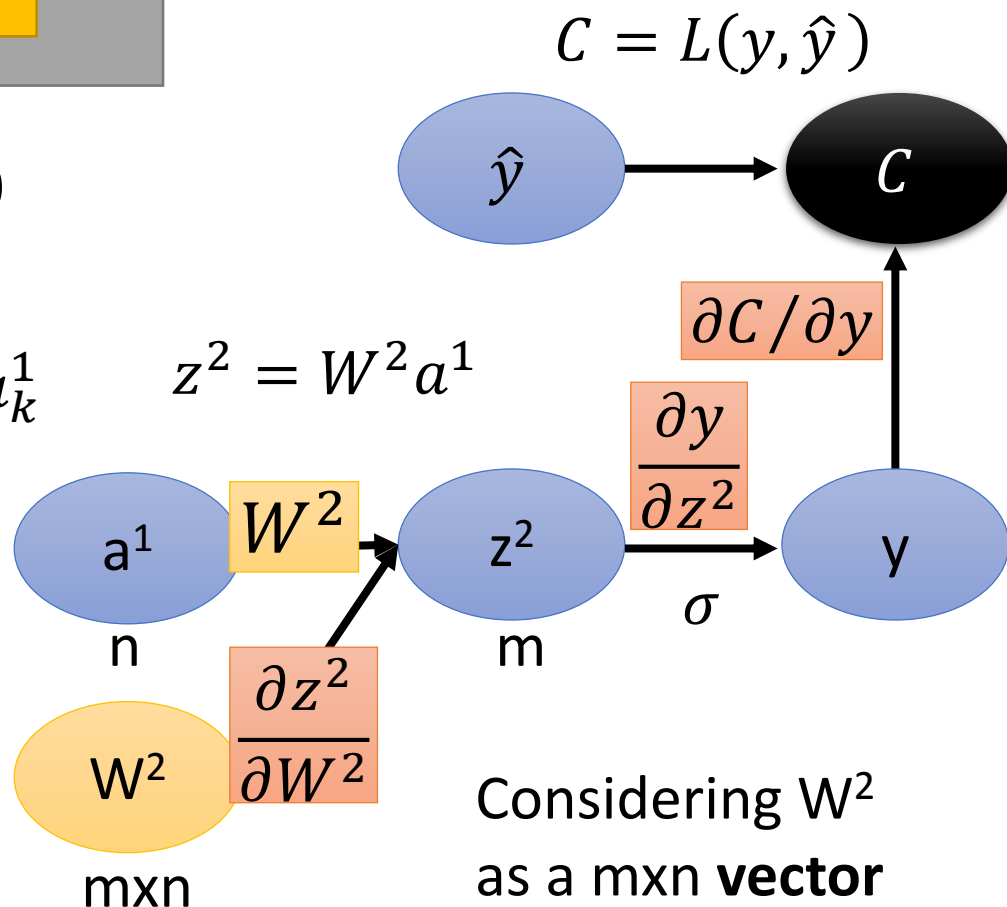
$\nearrow \frac{\partial z_i^2}{\partial W_{jk}^2}$

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

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

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

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



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

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

mxn

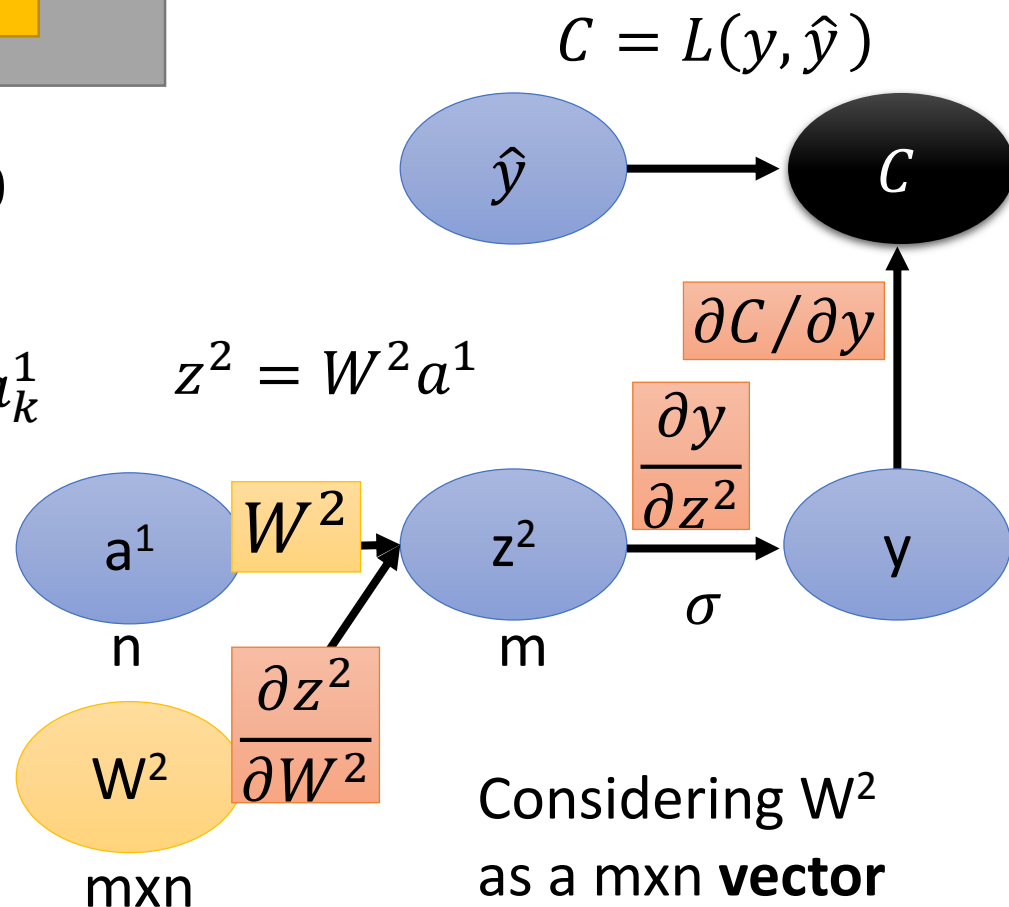
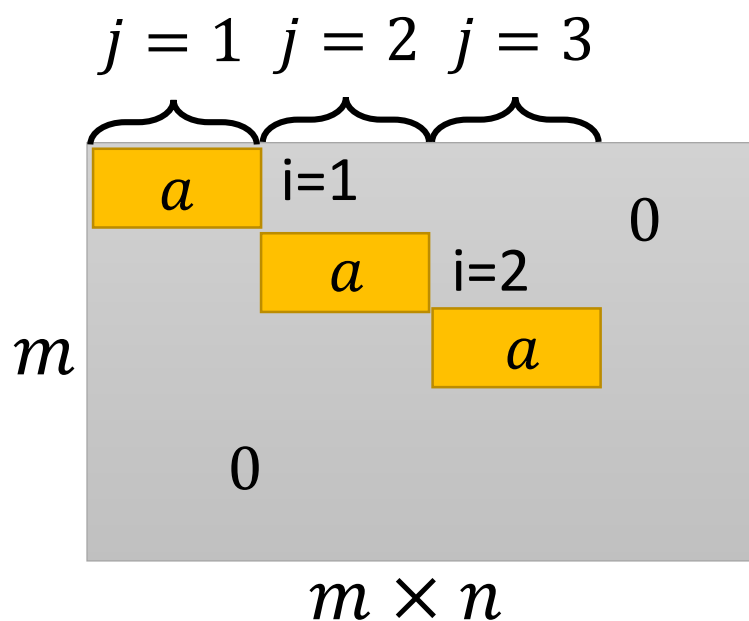
(j-1)xn+k

i

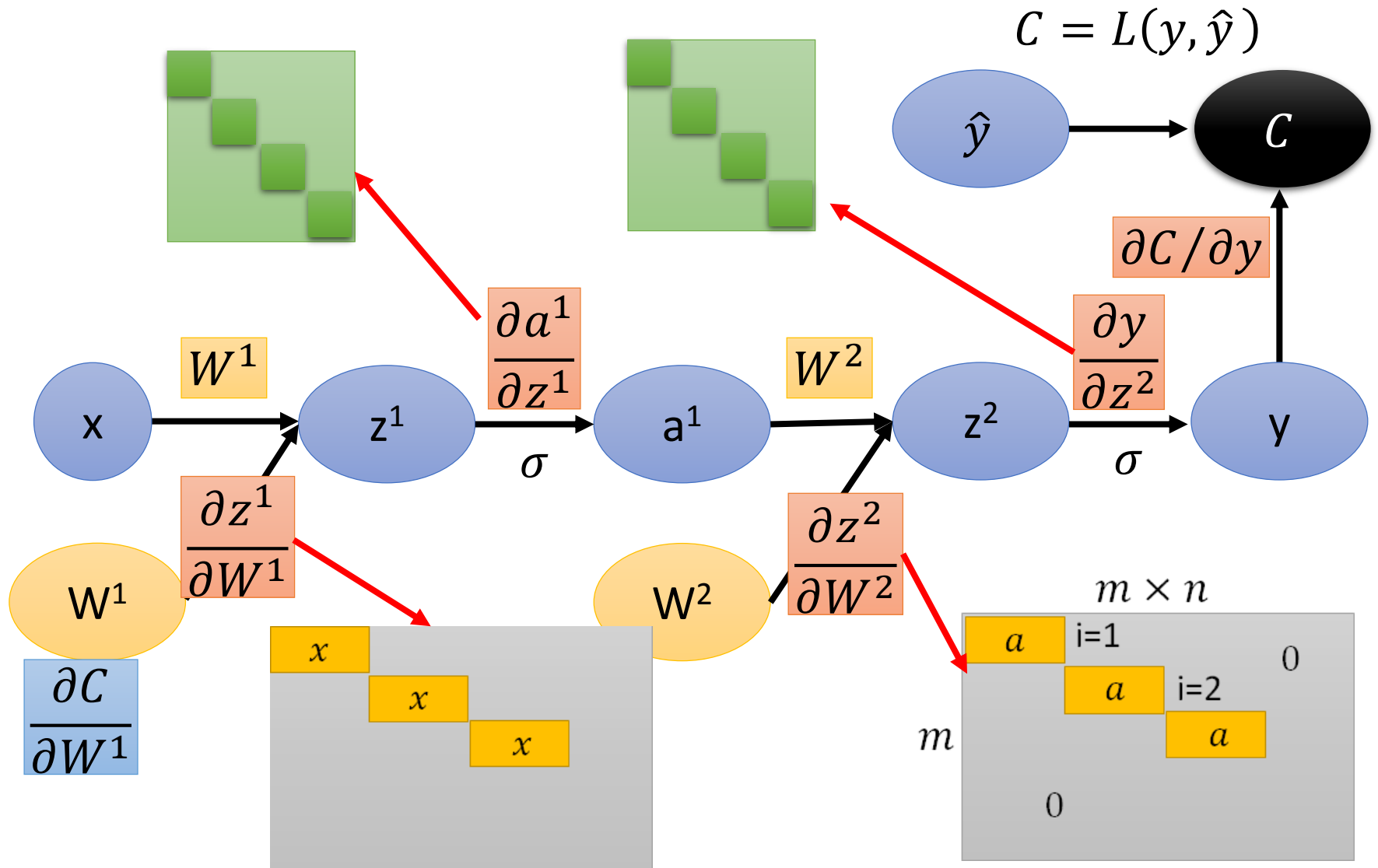
$$\frac{\partial z_i^2}{\partial W_{jk}^2}$$

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

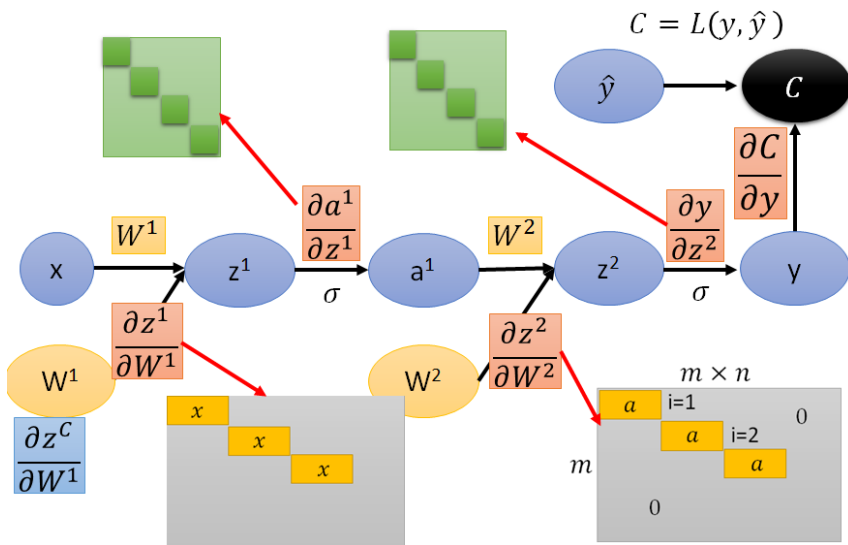
$$i = j: \frac{\partial z_i^2}{\partial W_{ik}^2} = a_k^1$$



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



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

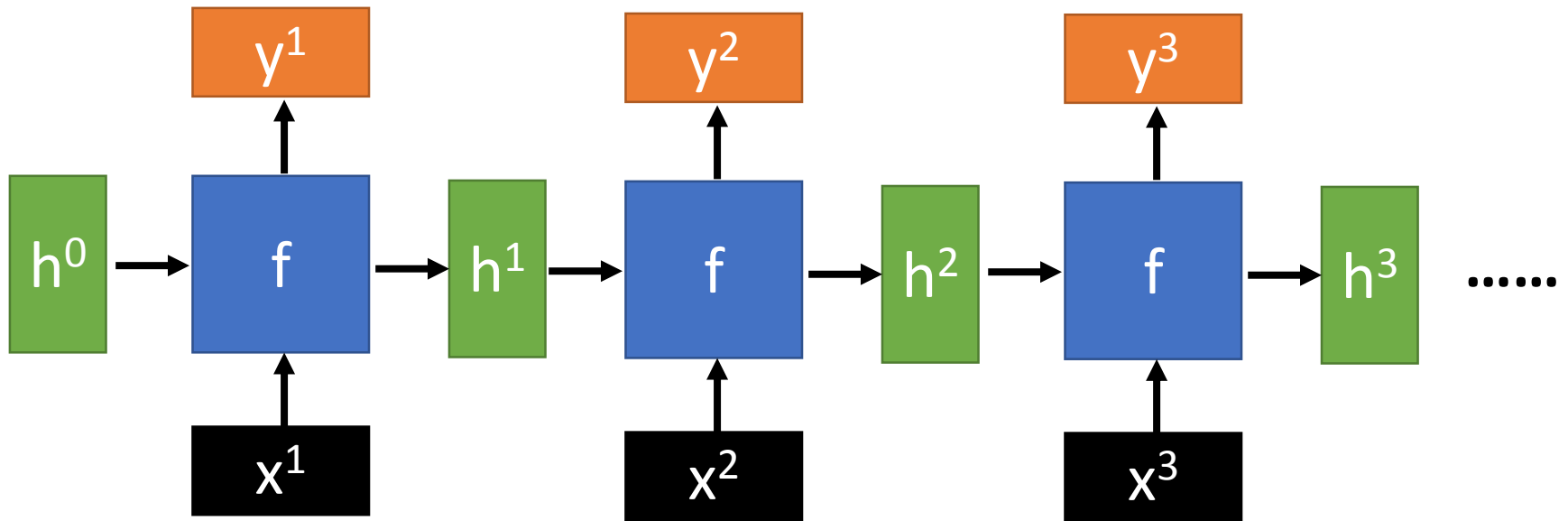
$$\begin{aligned} \delta^L &= \sigma'(z^L) \bullet \nabla_y C \\ \delta^{L-1} &= \sigma'(z^{L-1}) \bullet (W^L)^T \delta^L \\ &\dots\dots \\ \delta^l &= \sigma'(z^l) \bullet (W^{l+1})^T \delta^{l+1} \\ &\dots\dots \end{aligned}$$

Q: Only backward pass for computational graph?

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

Computational Graph for Recurrent Network

Recurrent Network



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

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

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

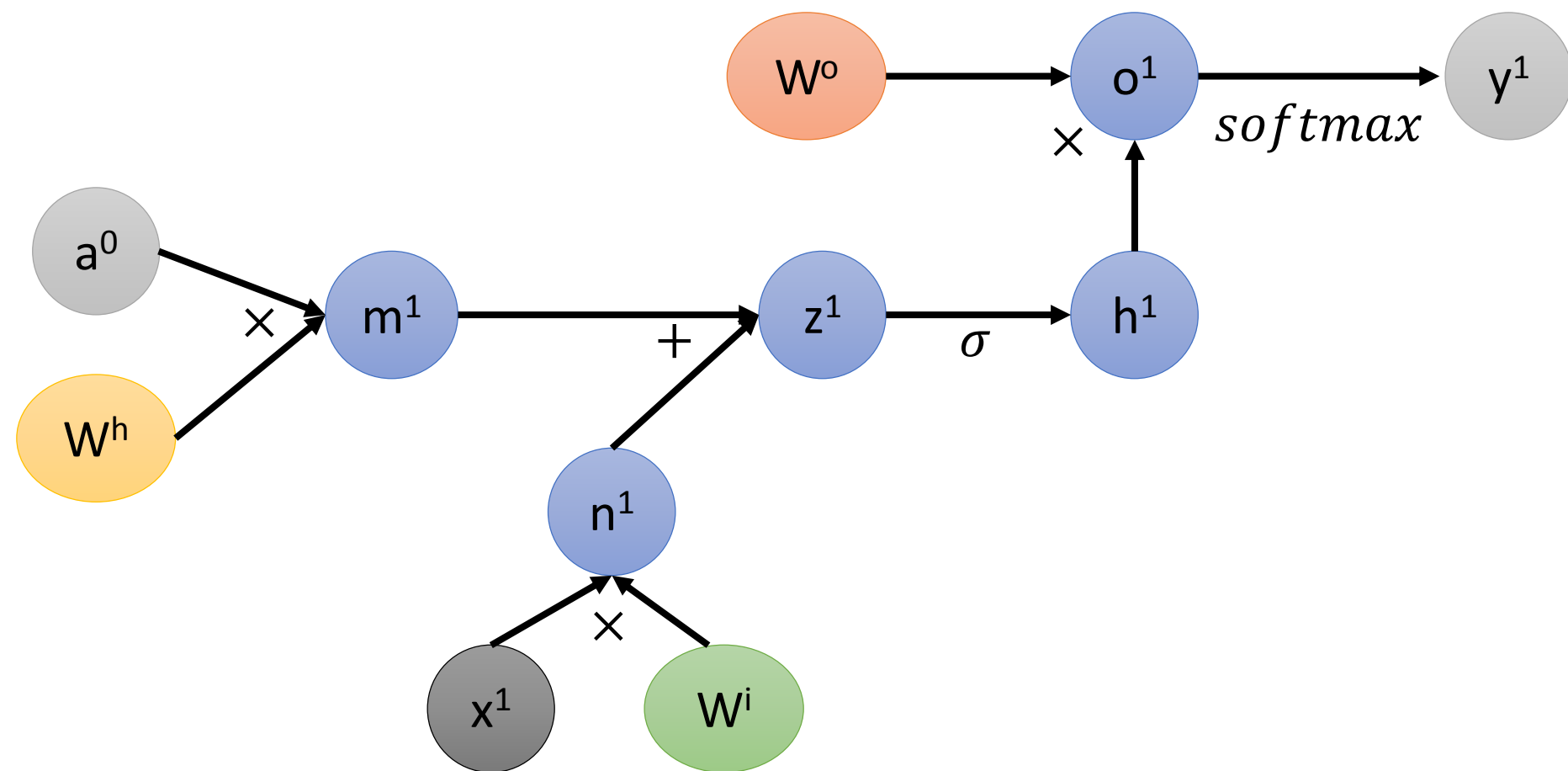
(biases are ignored here)

Recurrent Network

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

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

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

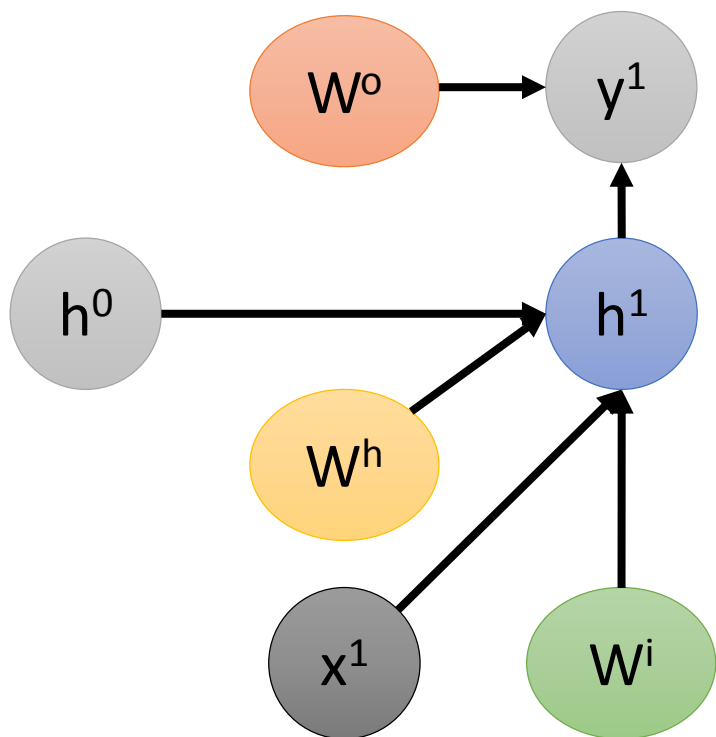


Recurrent Network

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

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

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

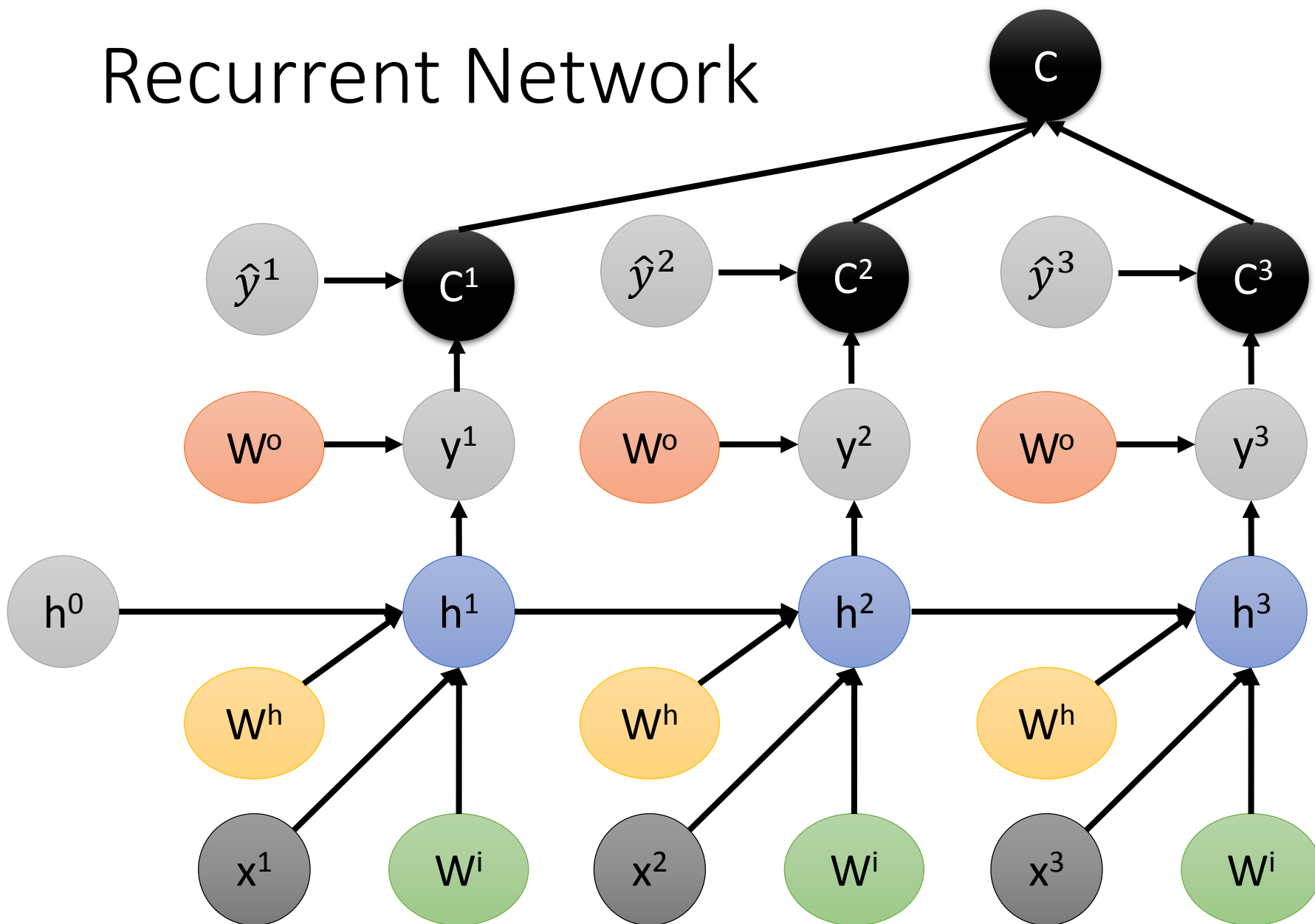


$$y^t = \text{softmax}(W^o h^t)$$

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

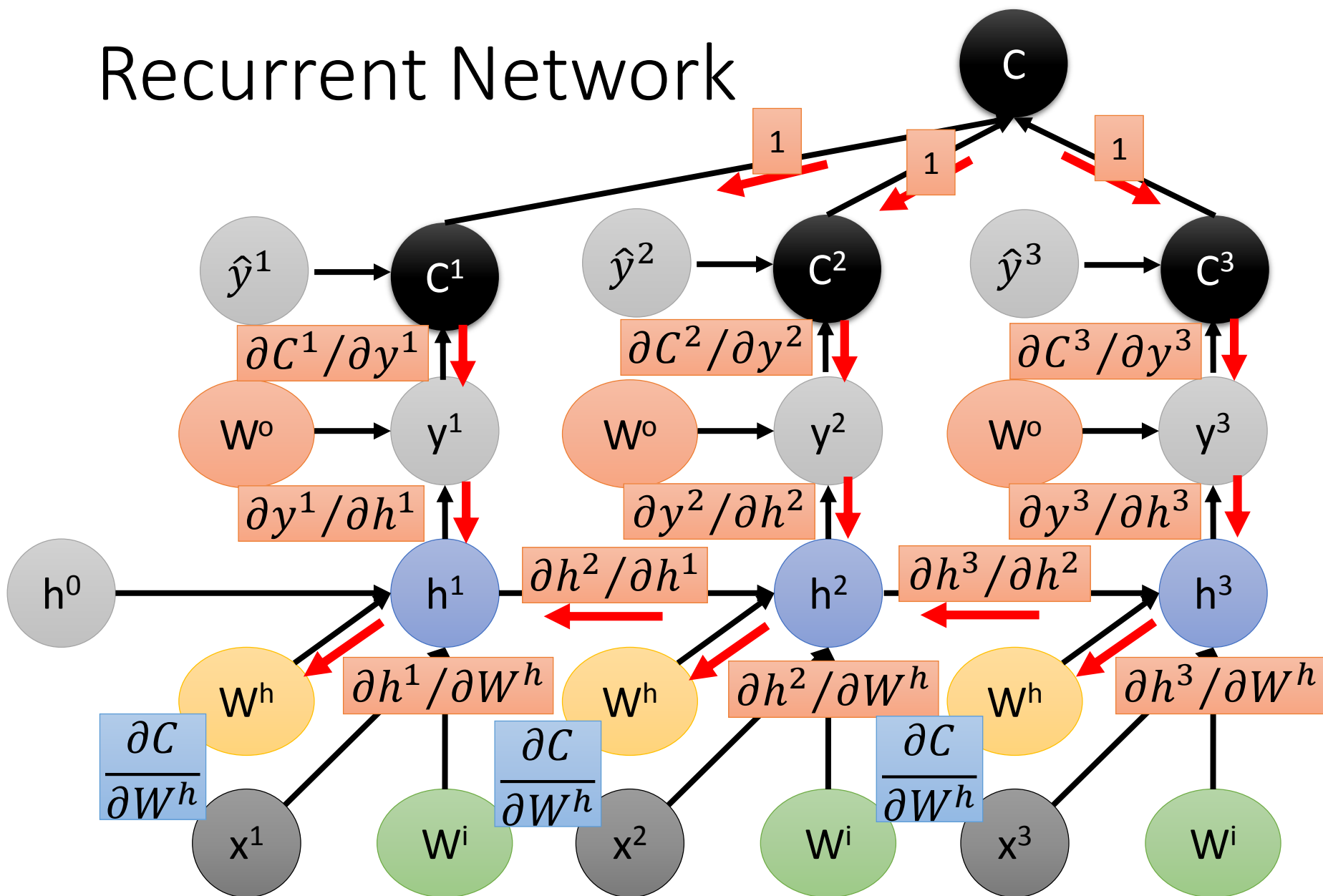
Recurrent Network

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



Recurrent Network

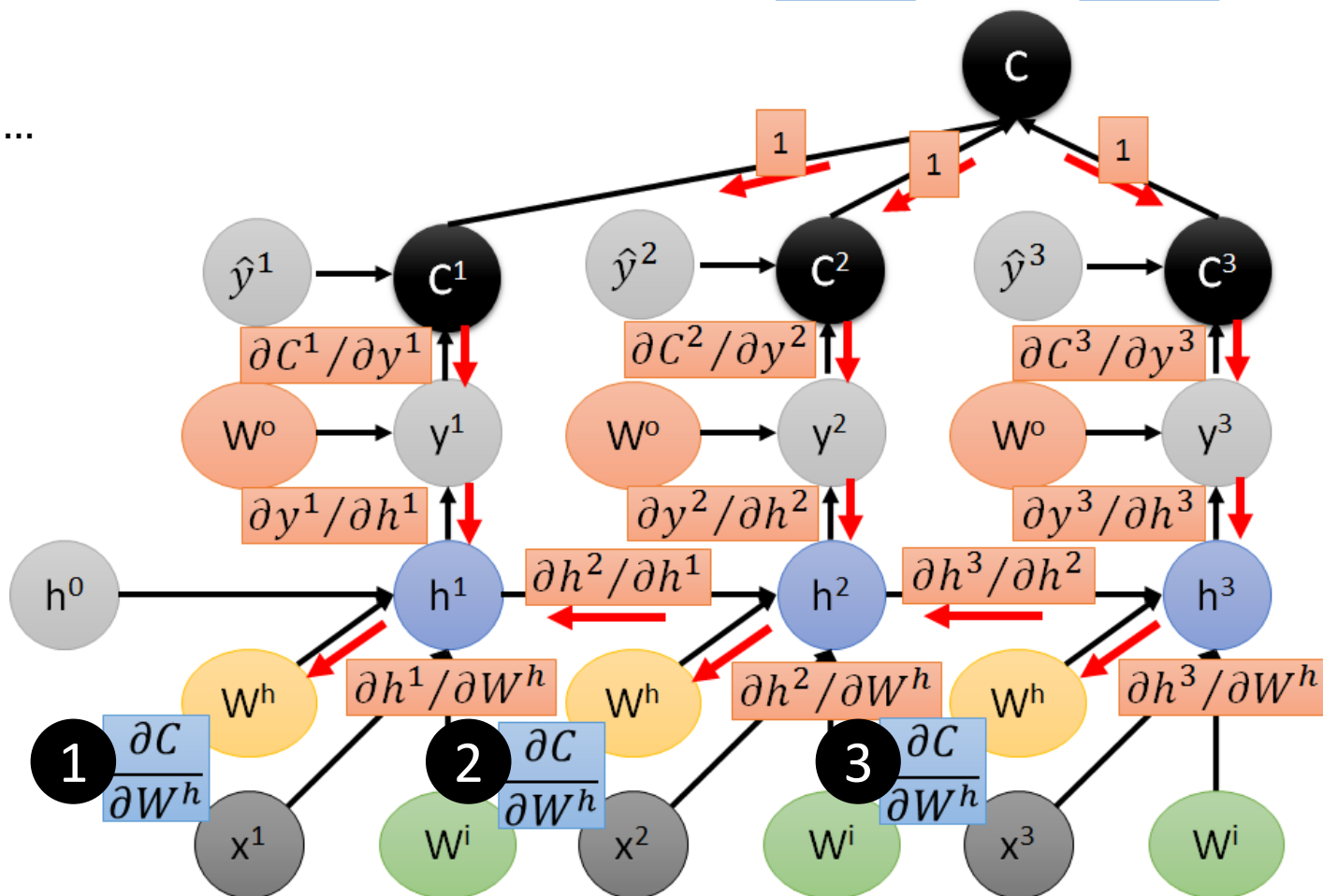
$$C = C^1 + C^2 + C^3$$



$$\textcircled{1} \frac{\partial C}{\partial W^h} = \left[\frac{\partial C^1}{\partial y^1} \frac{\partial y^1}{\partial h^1} + \frac{\partial C^2}{\partial y^2} \frac{\partial y^2}{\partial h^2} \frac{\partial h^2}{\partial h^1} + \frac{\partial C^3}{\partial y^3} \frac{\partial y^3}{\partial h^3} \frac{\partial h^3}{\partial h^2} \frac{\partial h^2}{\partial h^1} \right] \frac{\partial h^1}{\partial W^h}$$

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

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



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