Computing Gradient

Hung-yi Lee
李宏毅

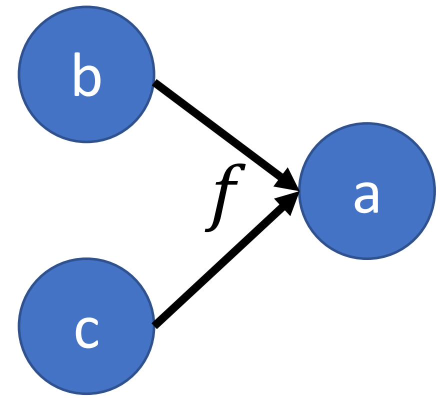
Introduction

- 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
- Understanding backpropagation by computational graph
  - Tensorflow, Theano, CNTK, etc.

Computational Graph

Computational Graph

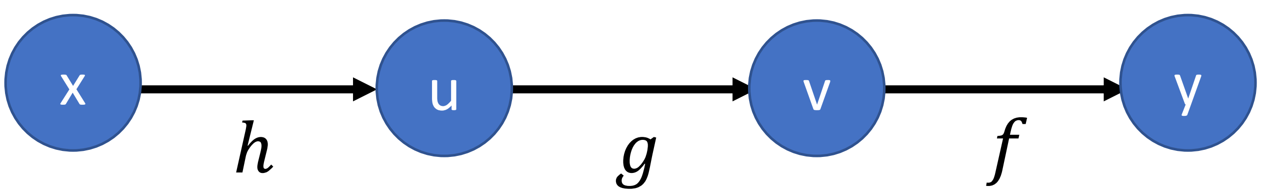
$$a = f(b, c)$$



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

<u>Example</u> $y = f\left(g\left(h(x)\right)\right)$

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



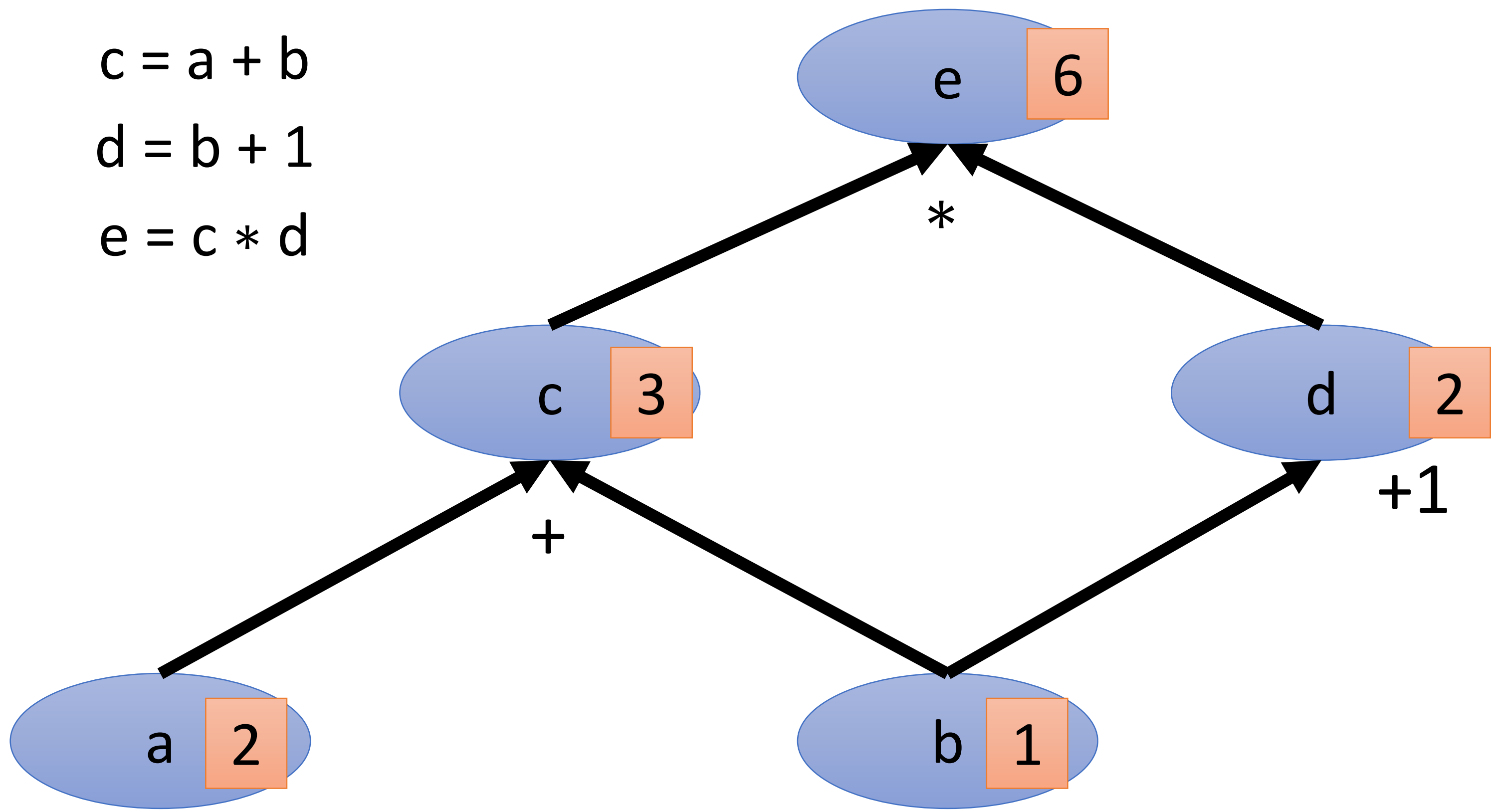
Computational Graph

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

$c = a + b$

$d = b + 1$

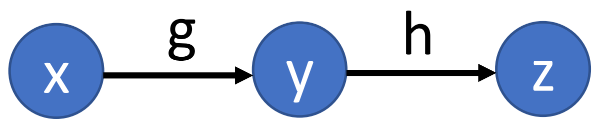
$e = c * d$



Review: Chain Rule

Case 1

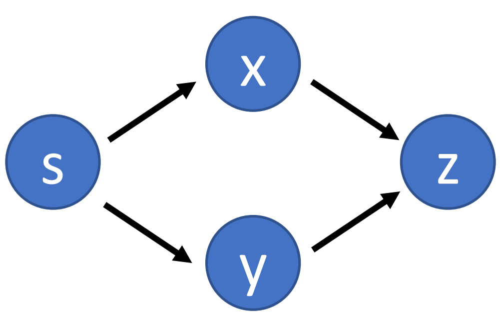
$$z = f(x) \Rightarrow y = g(x) \quad z = h(y)$$



$$\Delta x \to \Delta y \to \Delta z \qquad \frac{dz}{dx}$$

Case 2

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



$$\begin{tikzcd} & \Delta x \arrow[dr] & \\ \Delta s \arrow[ur] \arrow[dr] & & \Delta z \\ & \Delta y \arrow[ur] & \end{tikzcd} \qquad \frac{dz}{ds}$$

Computational Graph

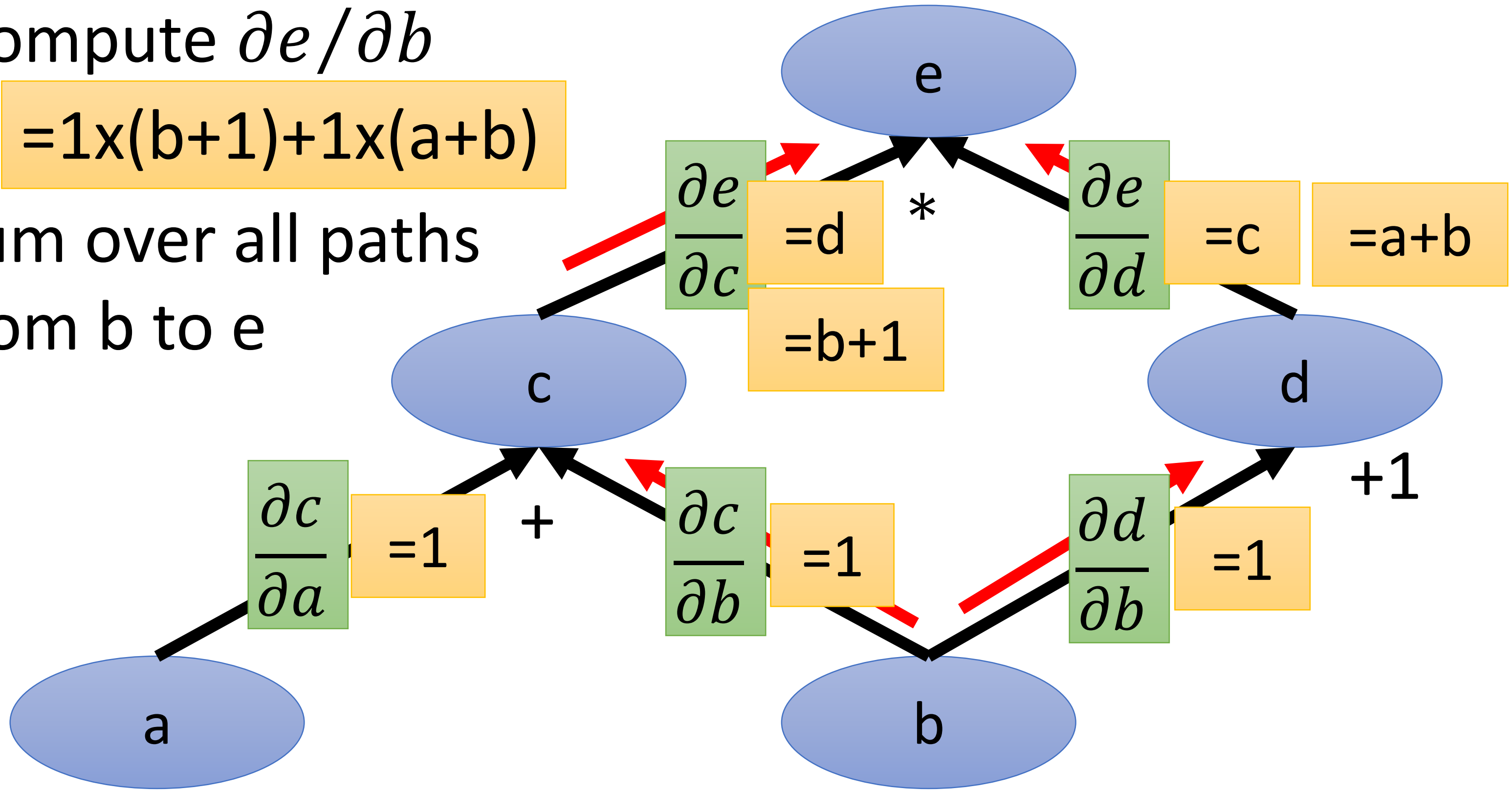
- Example: e = (a+b) ∗ (b+1)

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

Compute $\partial e / \partial b$

=1x(b+1)+1x(a+b)

Sum over all paths from b to e

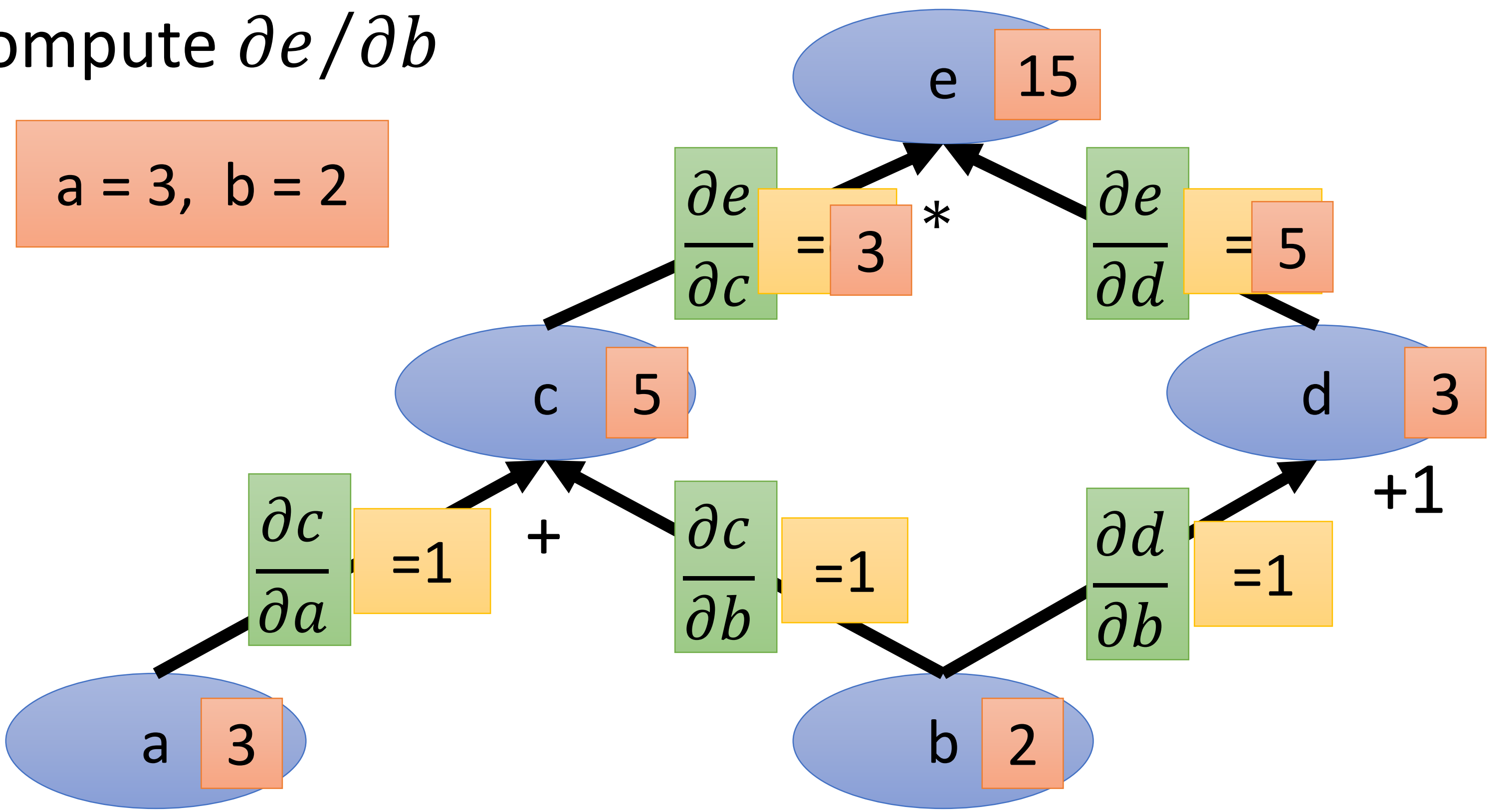


Computational Graph

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

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

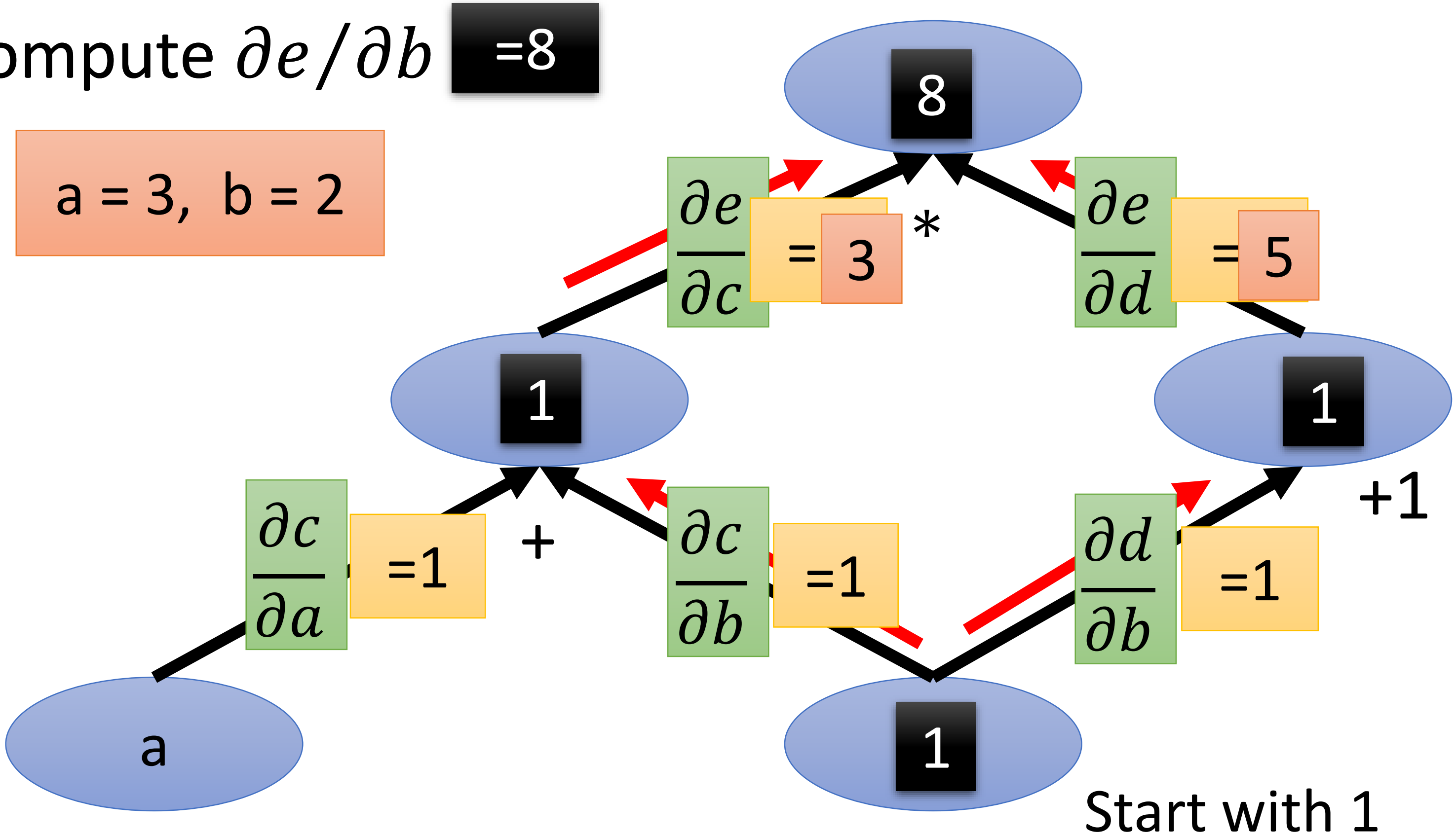
  Compute $\partial e / \partial b$



Computational Graph

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

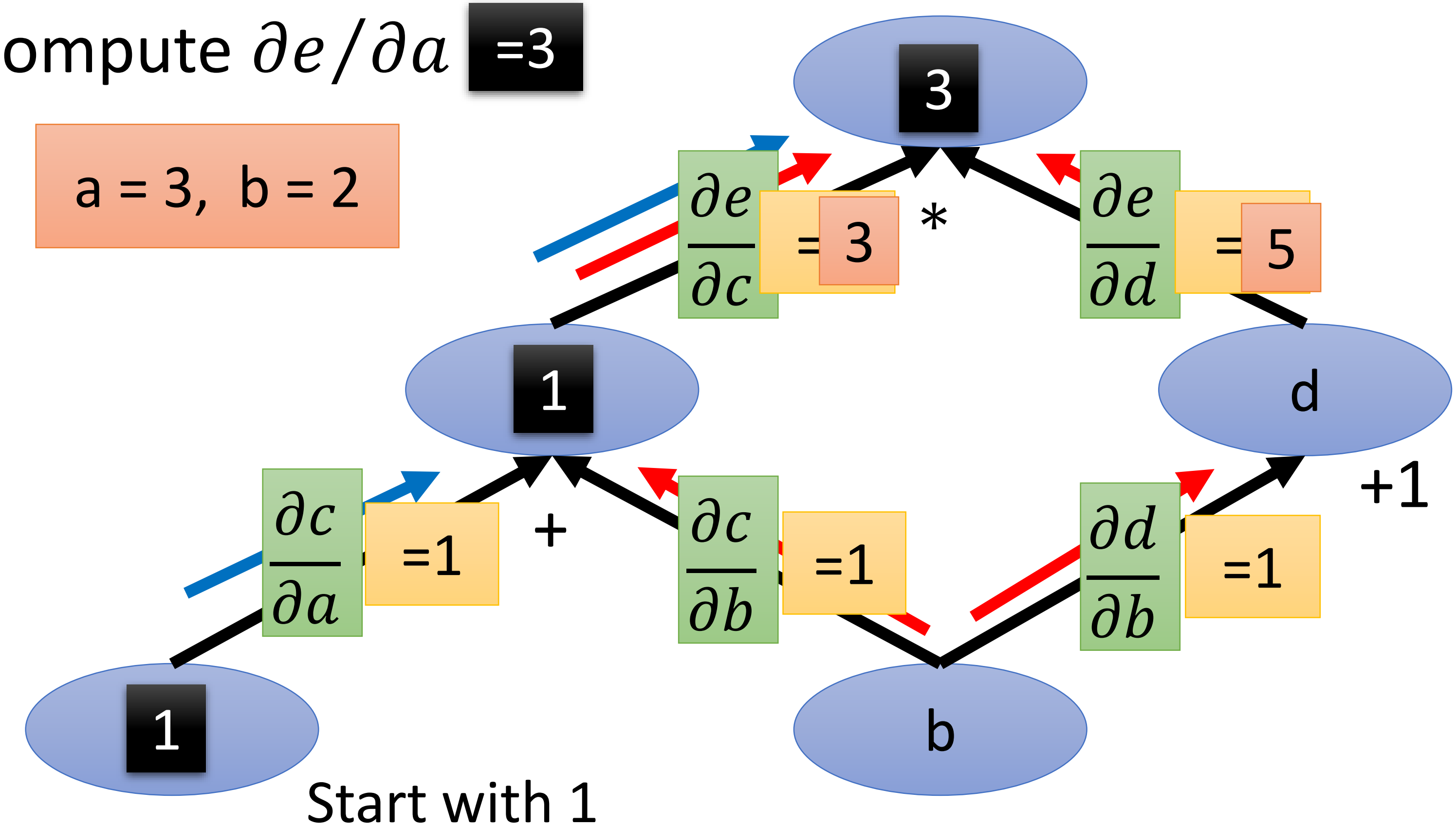
Compute $\partial e/\partial b$ =8



Computational Graph

- Example: e = (a+b) ∗ (b+1)

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



Computational Graph

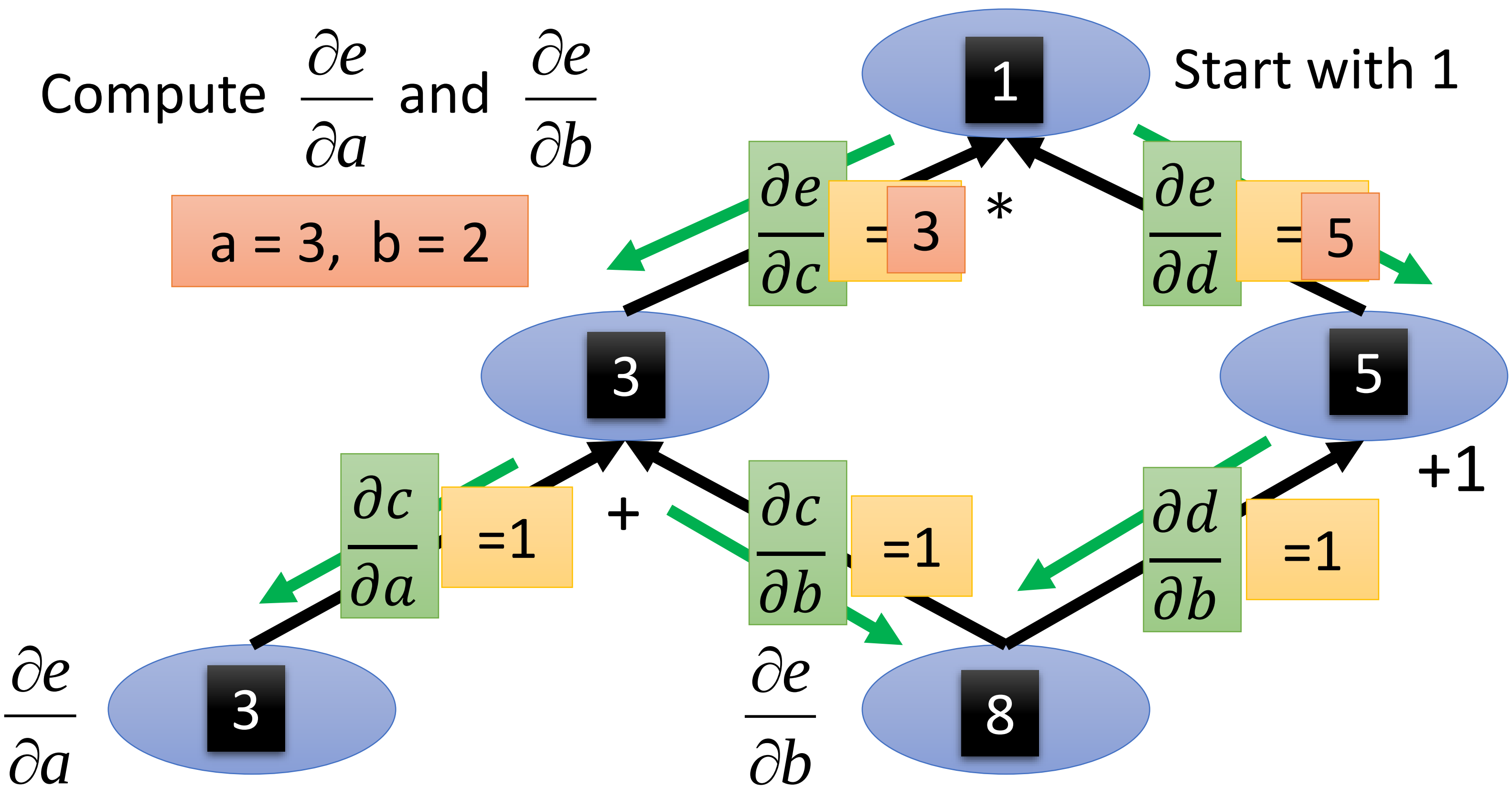
- Example: e = (a+b) ∗ (b+1)

Reverse mode

What is the benefit?

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

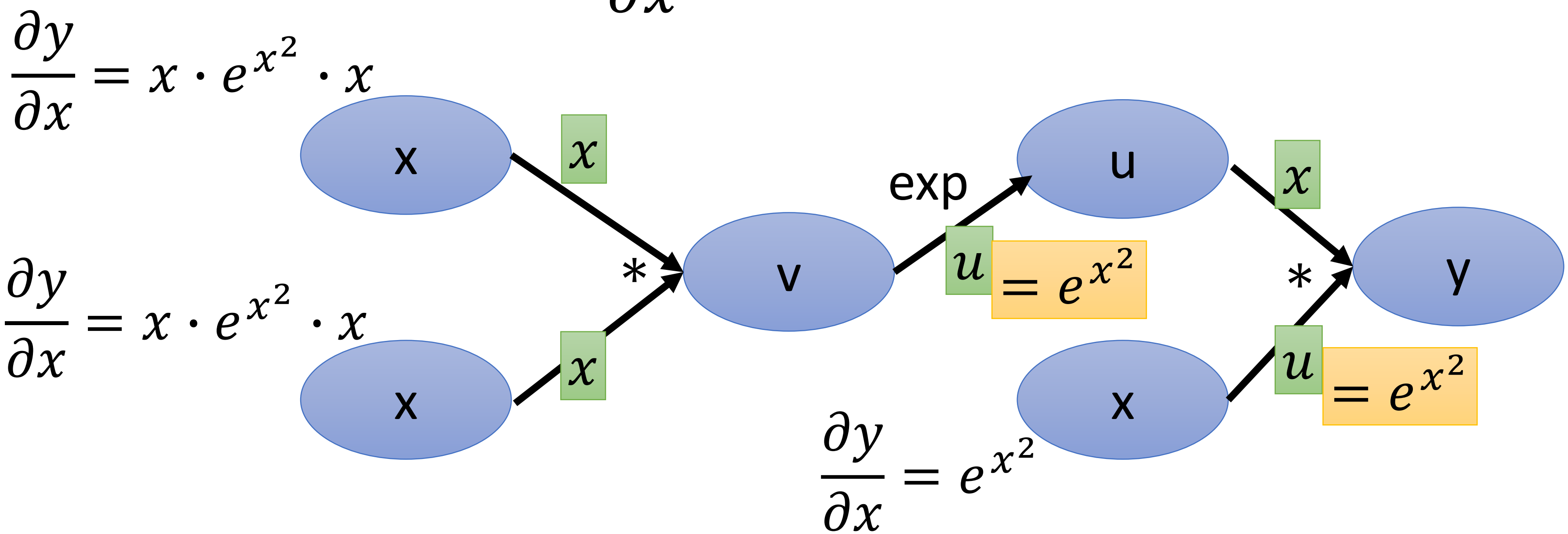
a = 3, b = 2



Computational Graph

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

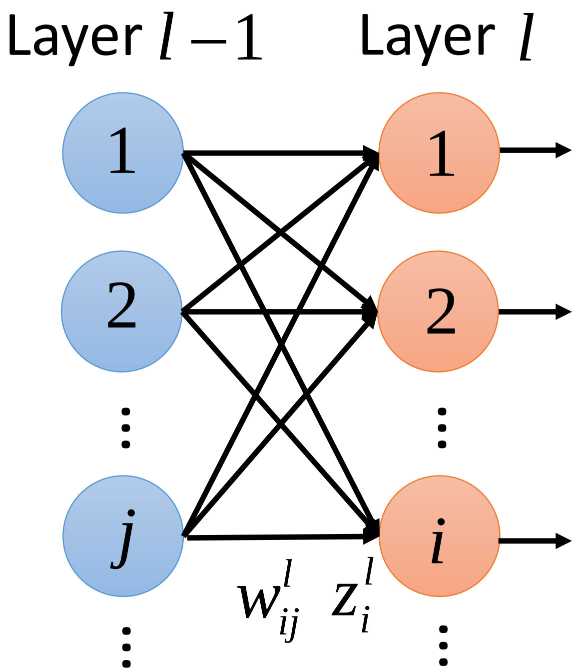
$$y = xe^{x^2} \qquad \frac{\partial y}{\partial x} =? \quad e^{x^2} + x \cdot e^{x^2} \cdot 2x$$



Computational Graph for Feedforward Net

Review: Backpropagation

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



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

Error signal

$$\delta_i^l$$

Forward Pass

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

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

$$\ldots\ldots$$

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

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

Backward Pass

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

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

$$\ldots\ldots$$

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

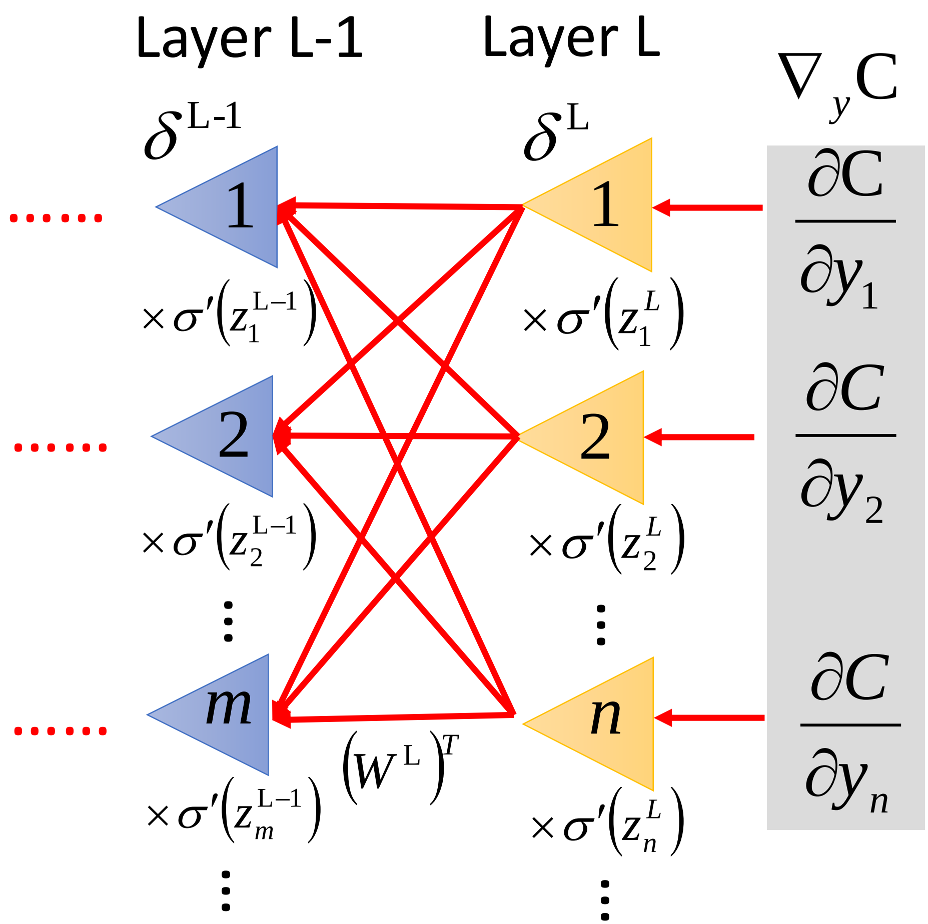
$$\ldots\ldots$$

Review: Backpropagation

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

Error signal

$$\boxed{\delta_i^l}$$



(we do not use softmax here)

Backward Pass

$$\delta^{\text{L}} = \sigma'\left(z^{L}\right) \bullet \nabla_y \text{C}$$

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

$$\cdots\cdots$$

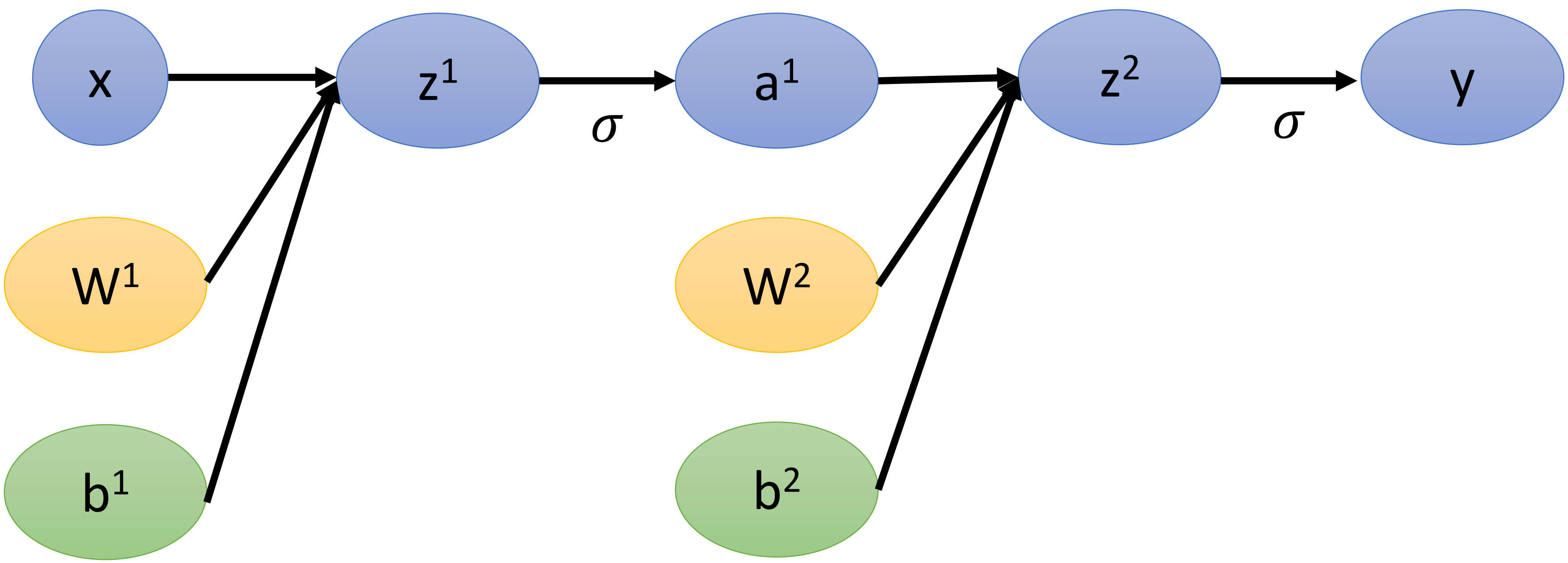
$$\delta^{l} = \sigma'\left(z^{l}\right) \bullet \left(W^{l+1}\right)^T \delta^{l+1}$$

$$\cdots\cdots$$

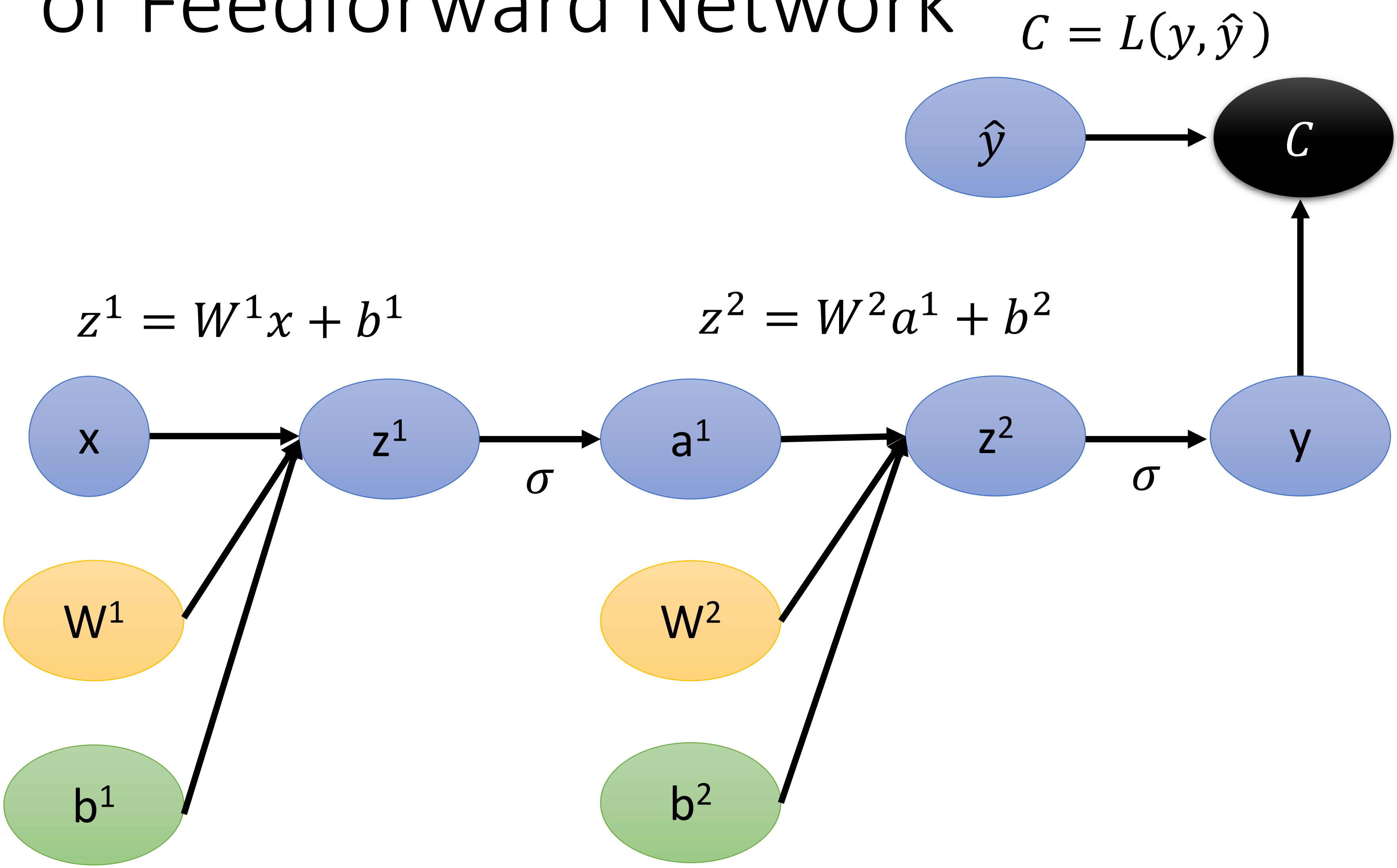
Feedforward Network

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

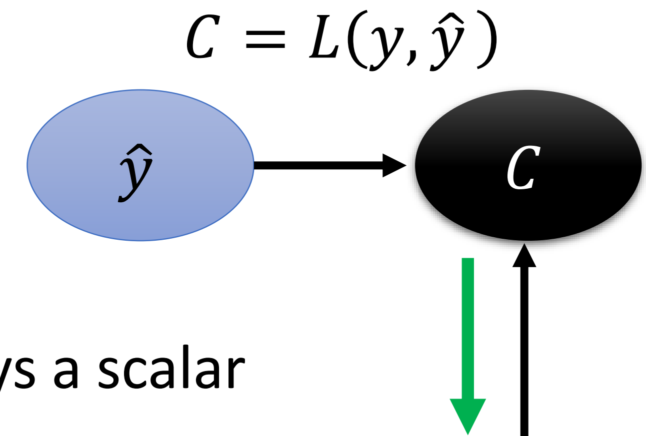
$$z^1 = W^1 x + b^1 \qquad z^2 = W^2 a^1 + b^2$$



Loss Function of Feedforward Network



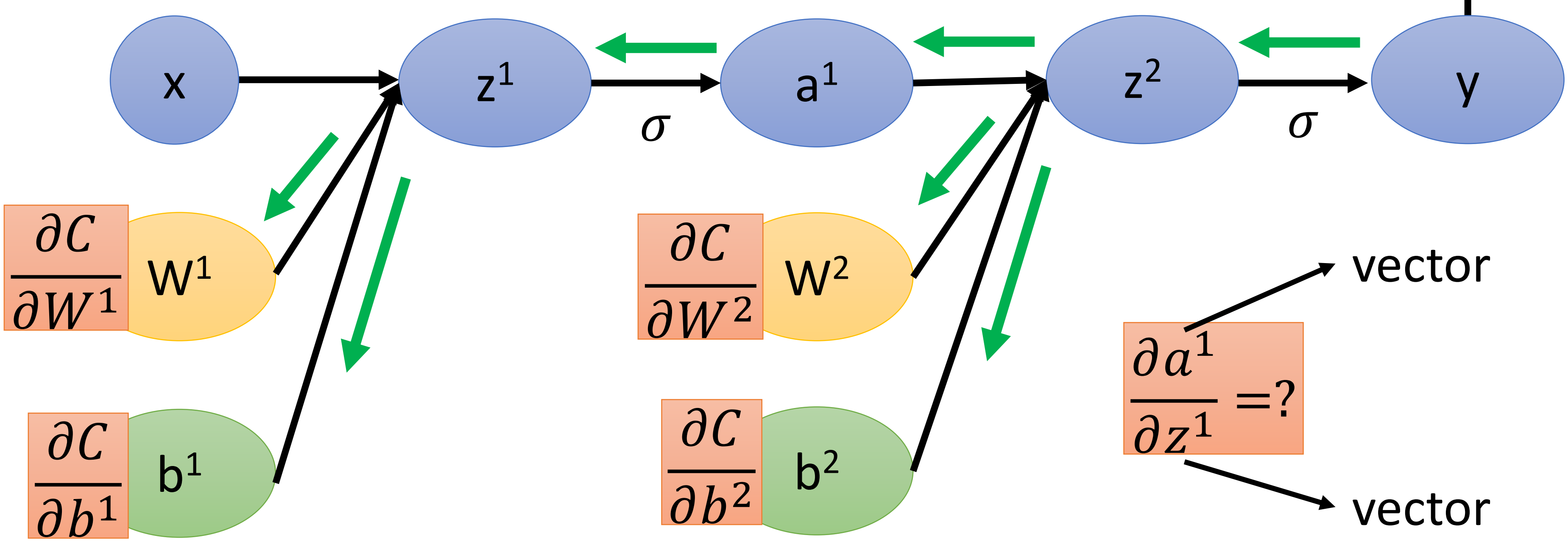
Gradient of Cost Function



To compute the gradient …

Computing the partial derivative on the edge

Using reverse mode ➡ Output is always a scalar



Jacobian Matrix

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

$$\frac{\partial y}{\partial x} =$$

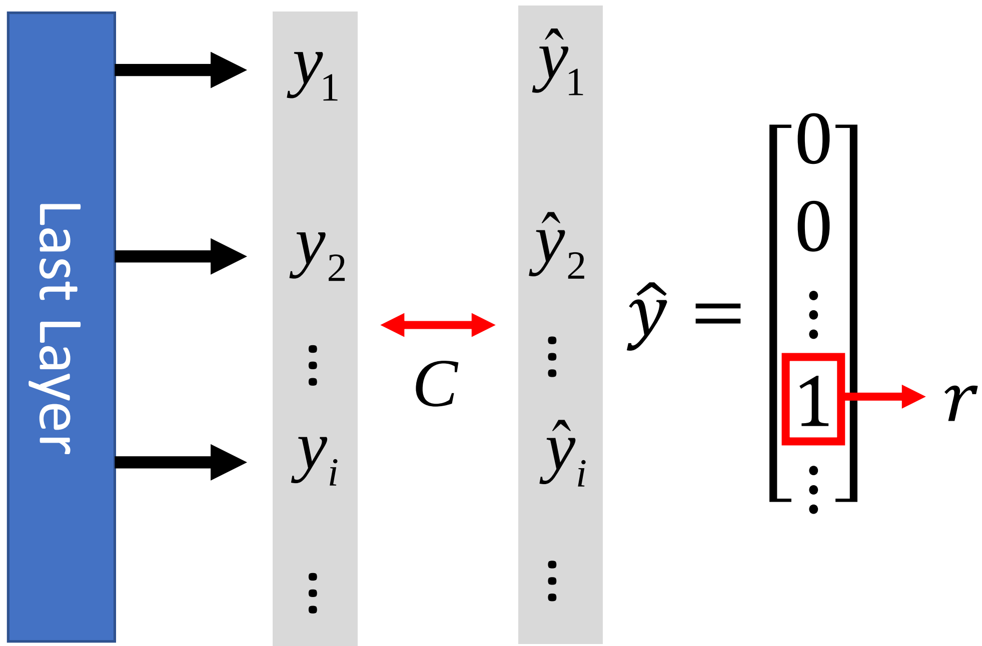
size of y

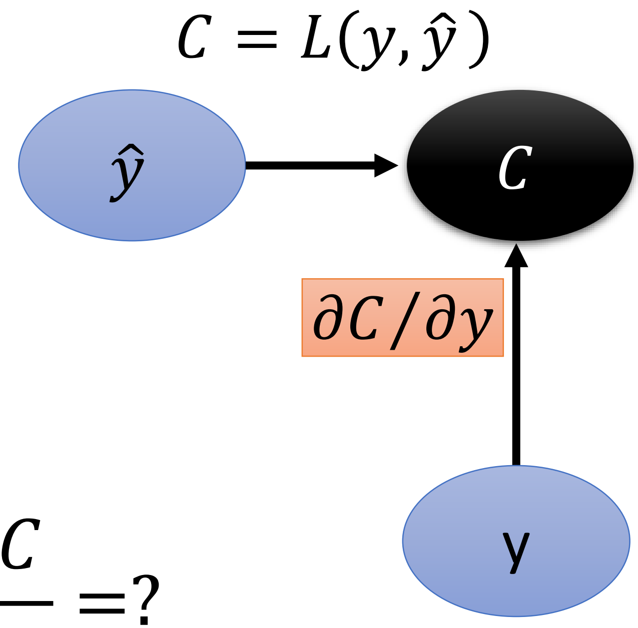
size of x

Example

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

Gradient of Cost Function





$$\frac{\partial C}{\partial y_i} = ?$$

Cross Entropy: $C = -log y_r$

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

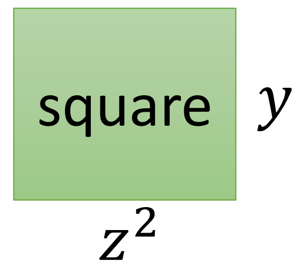
$i = r$:

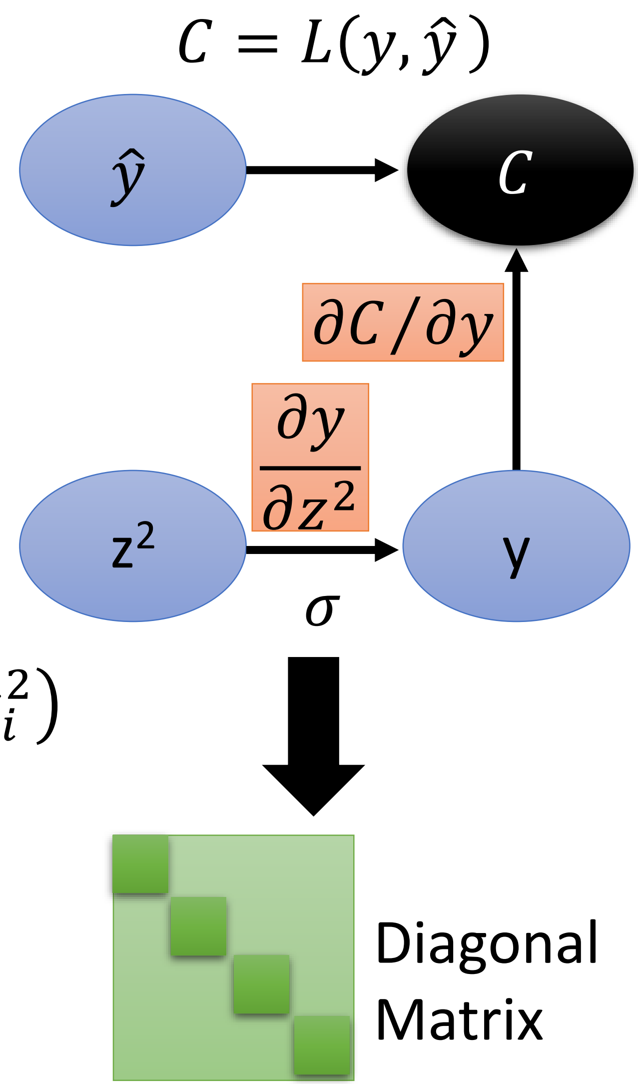
$$\partial C / \partial y_r = -1 / y_r$$

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

Gradient of Cost Function

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



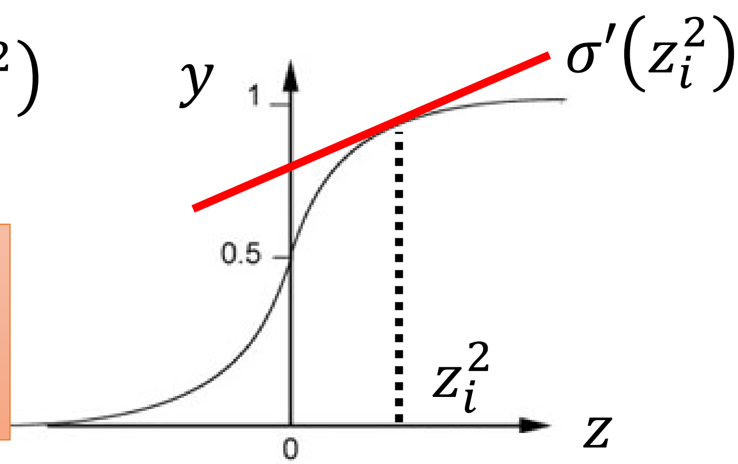


i-th row, j-th column: $\partial y_i / \partial z_j^2$

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

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

$y_i = \sigma(z_i^2)$



How about softmax? ☺

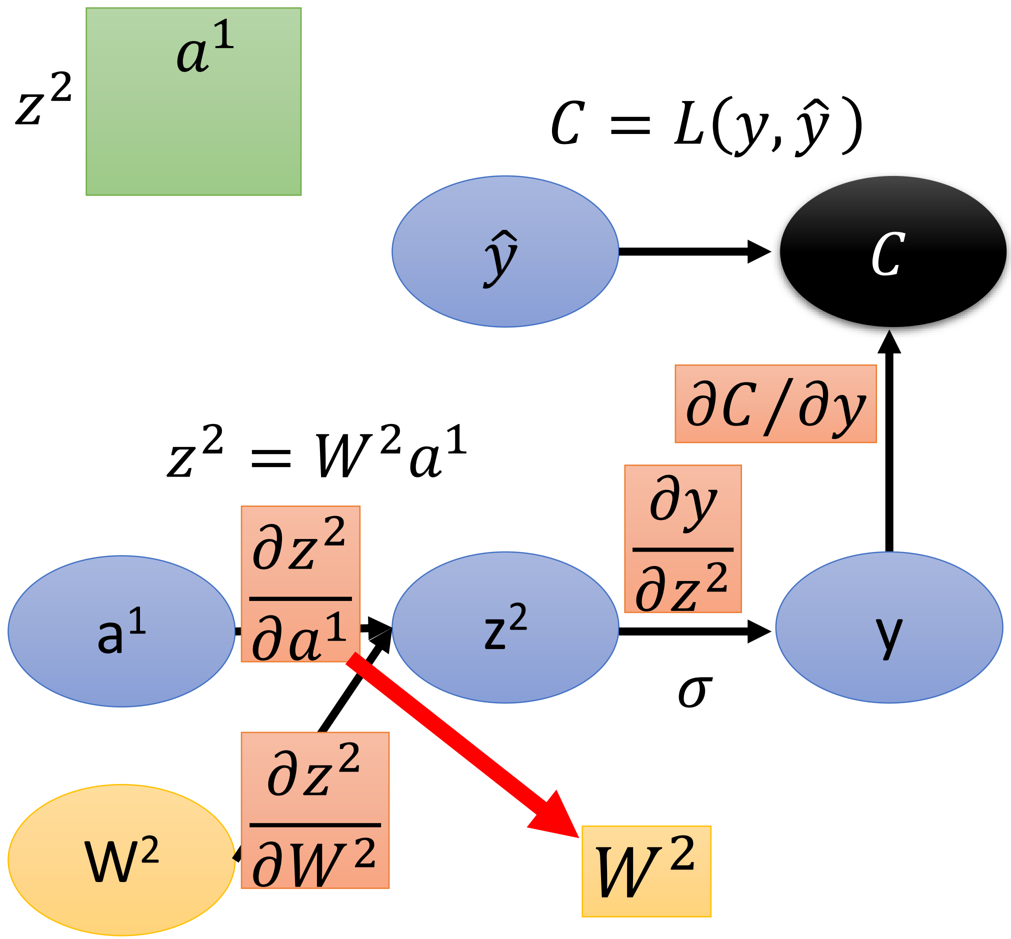
$\frac{\partial z^2}{\partial a^1}$ is a Jacobian matrix

i-th row, j-th column:

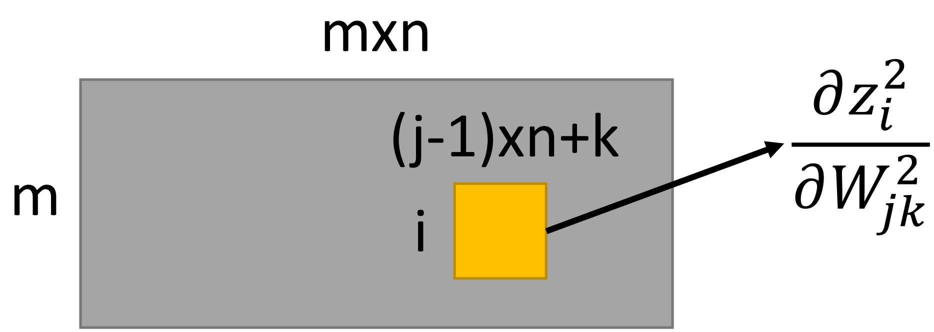
$$\frac{\partial z_i^2}{\partial a_j^1} =$$

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

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



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



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

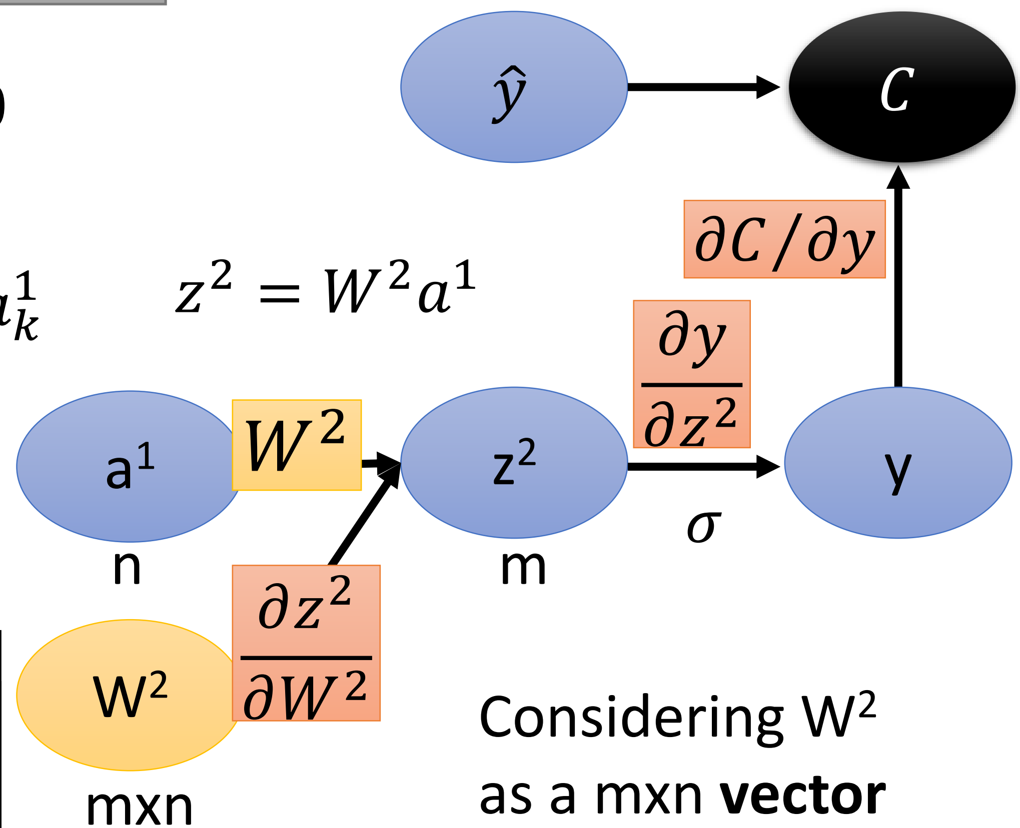
$$i = j: \frac{\partial z_i^2}{\partial W_{ik}^2} = a_k^1$$

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

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

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

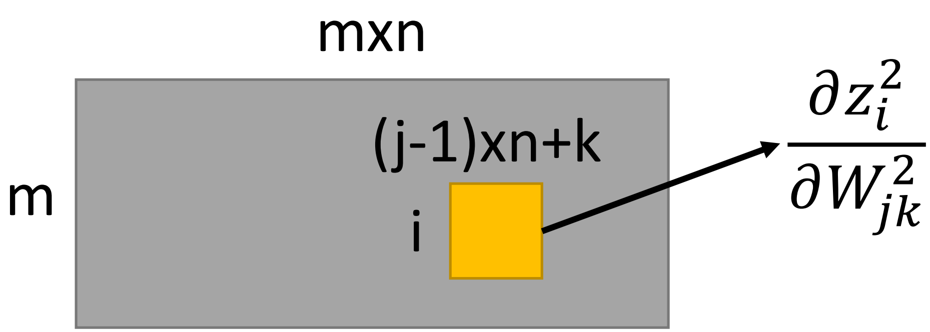
$$z^2 = W^2 a^1$$



Considering W^2 as a mxn **vector**

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

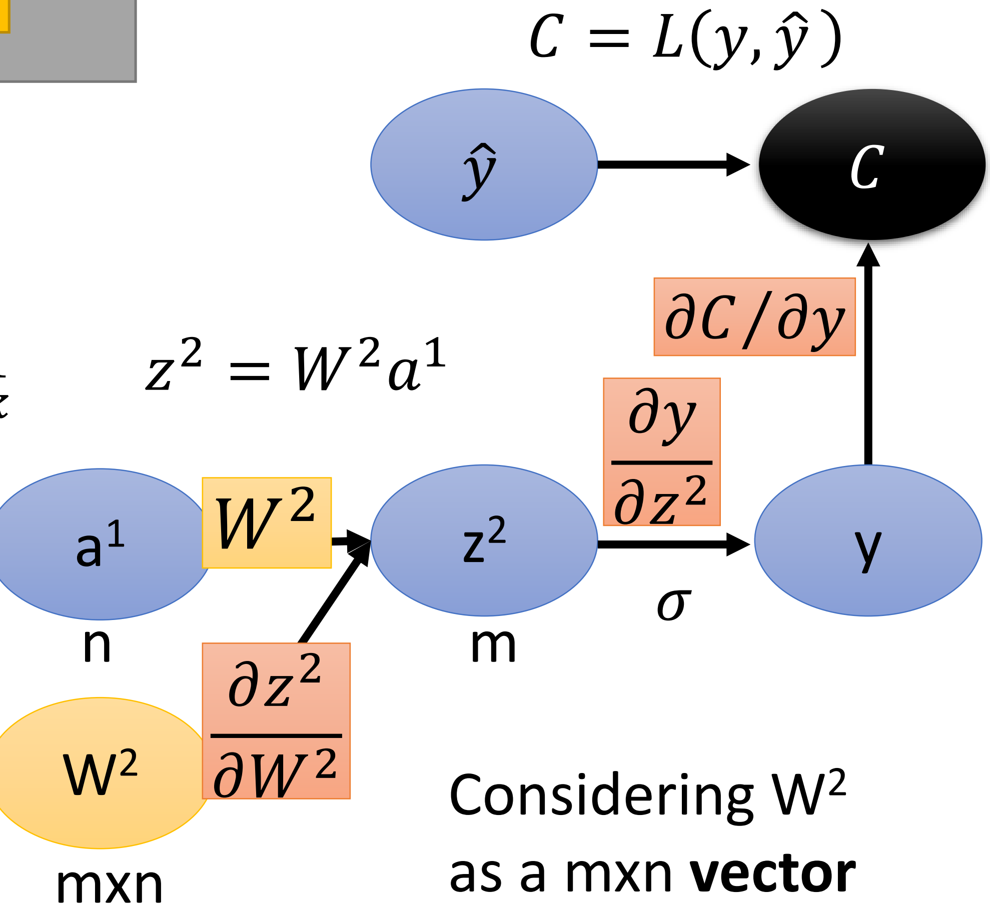
$$\frac{\partial z^2}{\partial W^2} =$$



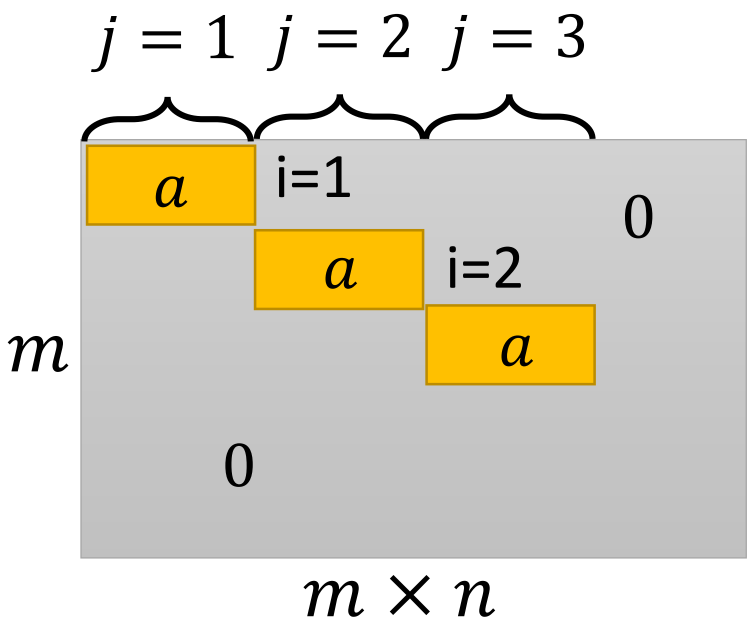
$$\frac{\partial z_i^2}{\partial W_{jk}^2} =?$$

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

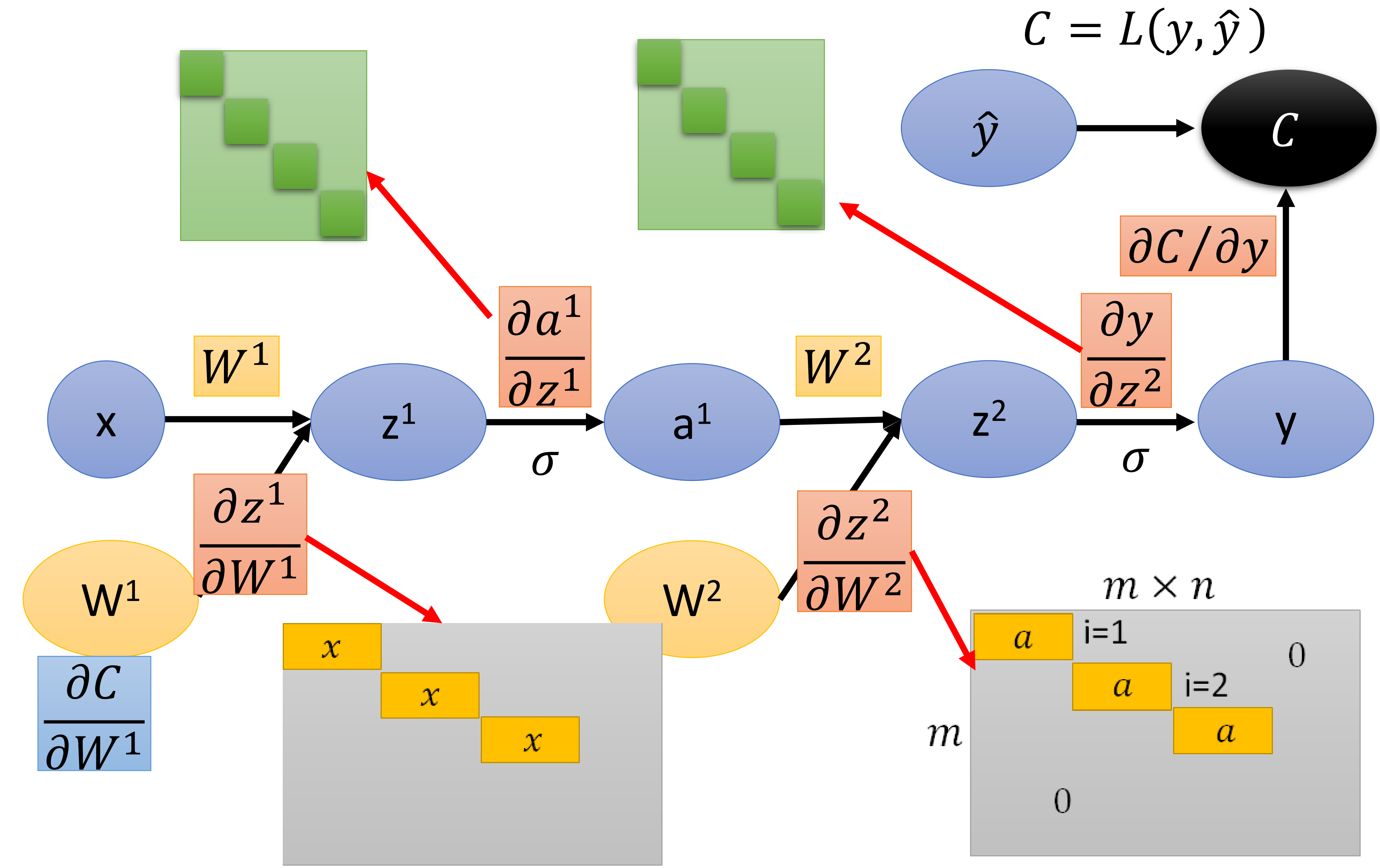
$$i = j: \frac{\partial z_i^2}{\partial W_{ik}^2} = a_k^1$$



Considering W^2 as a mxn **vector**



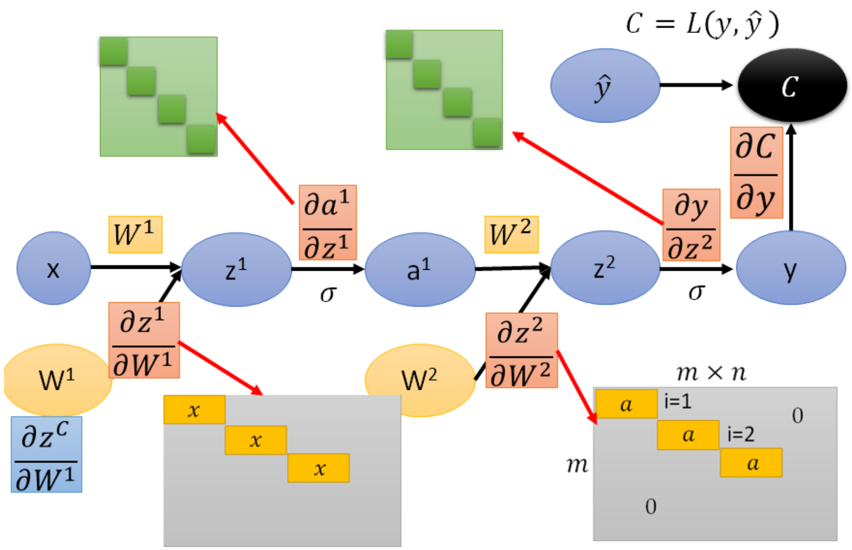
$$\frac{\partial C}{\partial W^1} = \frac{\partial C}{\partial y} \frac{\partial y}{\partial z^2} W^2 \frac{\partial a^1}{\partial z^1} \frac{\partial z^1}{\partial W^1} = \left[\cdots \frac{\partial C}{\partial W^1_{ij}} \cdots\right]$$



Question

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

Error signal



Forward Pass

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

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

……

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

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

Backward Pass

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

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

……

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

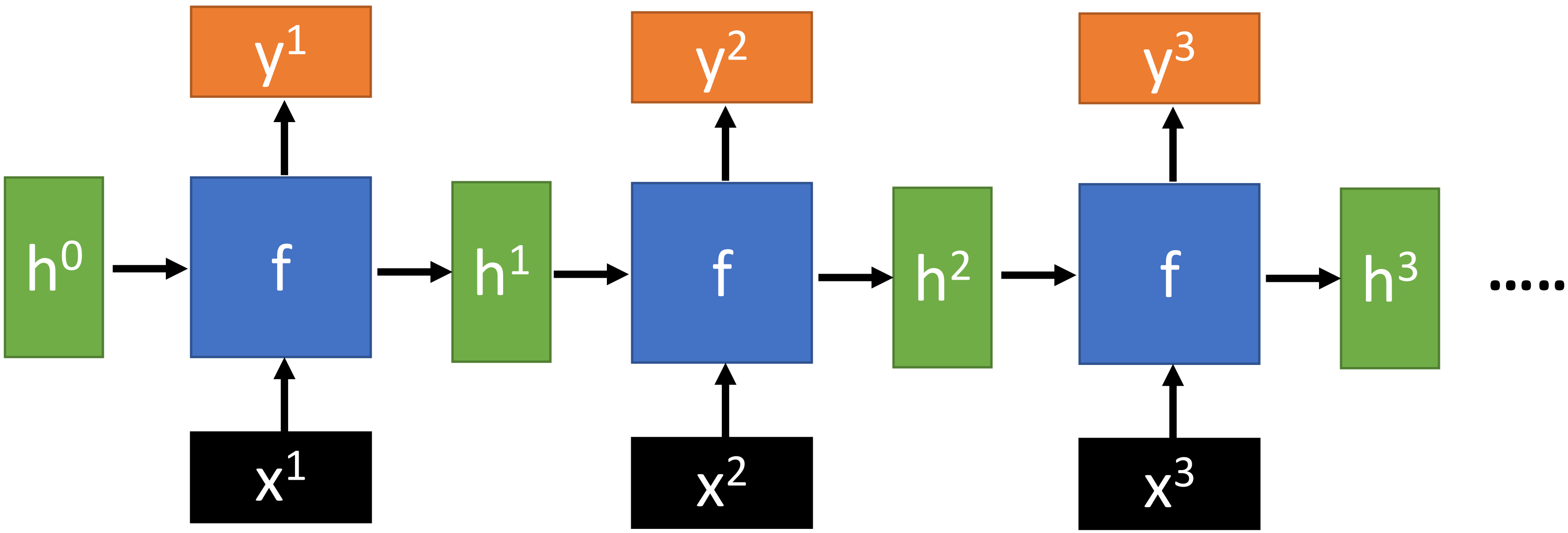
……

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



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

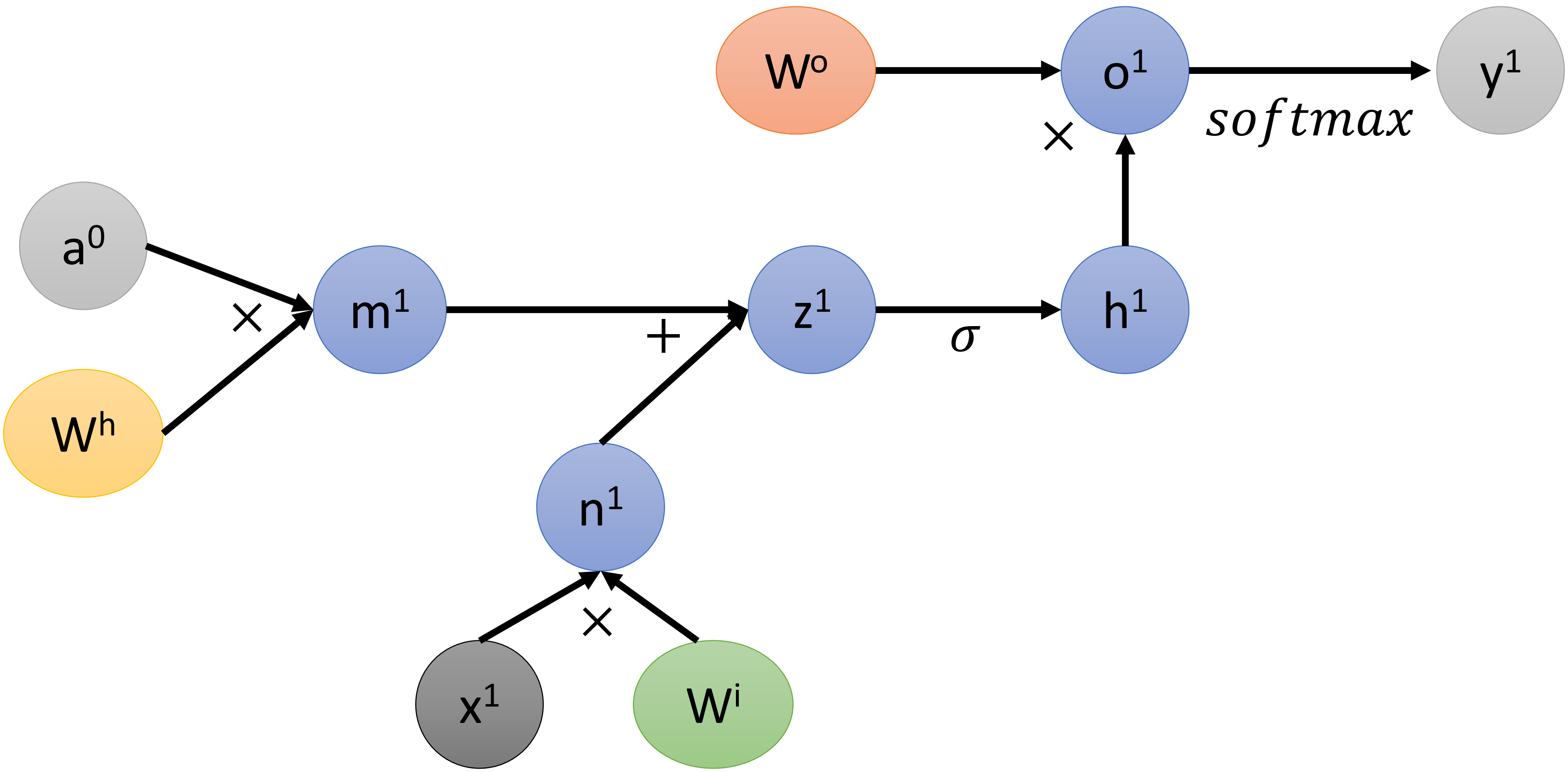
$$h^t = \sigma\left(W^i x^t + W^h h^{t-1}\right)$$

$$y^t = softmax(W^o h^t)$$

(biases are ignored here)

Recurrent Network

$$y^t, h^t = f\left(x^t, h^{t-1}; W^i, W^h, W^o\right)$$
$$a^t = \sigma\left(W^i x^t + W^h h^{t-1}\right)$$
$$y^t = softmax(W^o h^t)$$

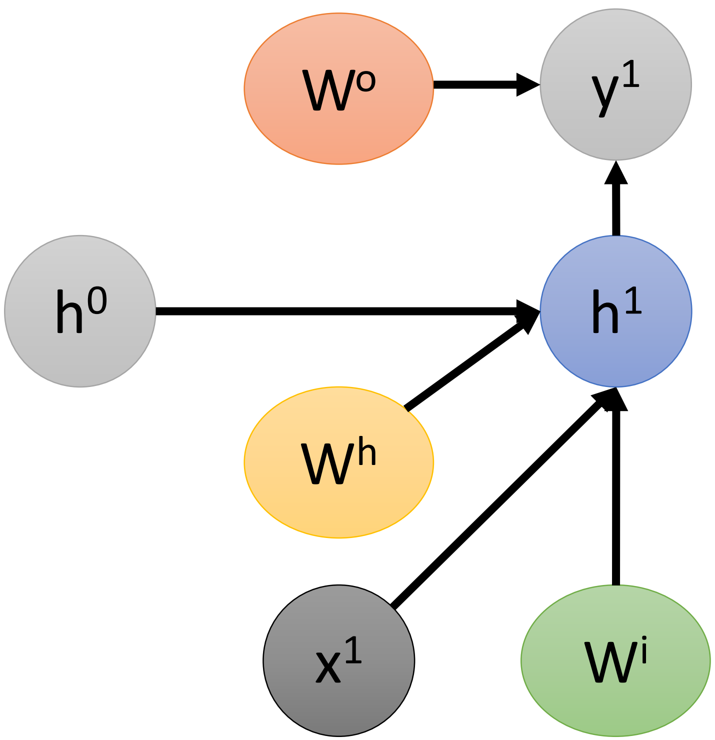


Recurrent Network

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

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

$$y^t = softmax(W^o h^t)$$

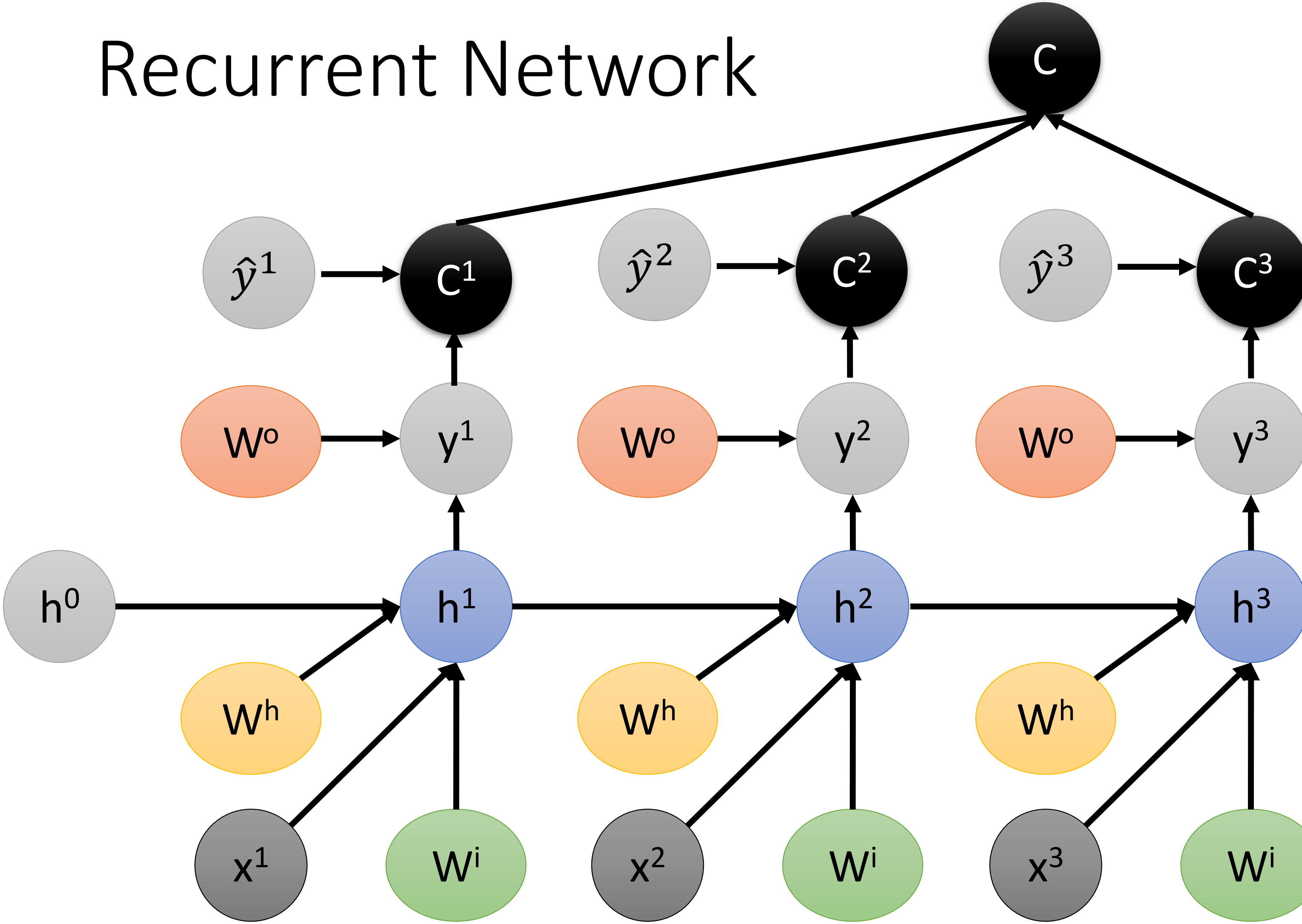


$$y^t = softmax(W^o h^t)$$

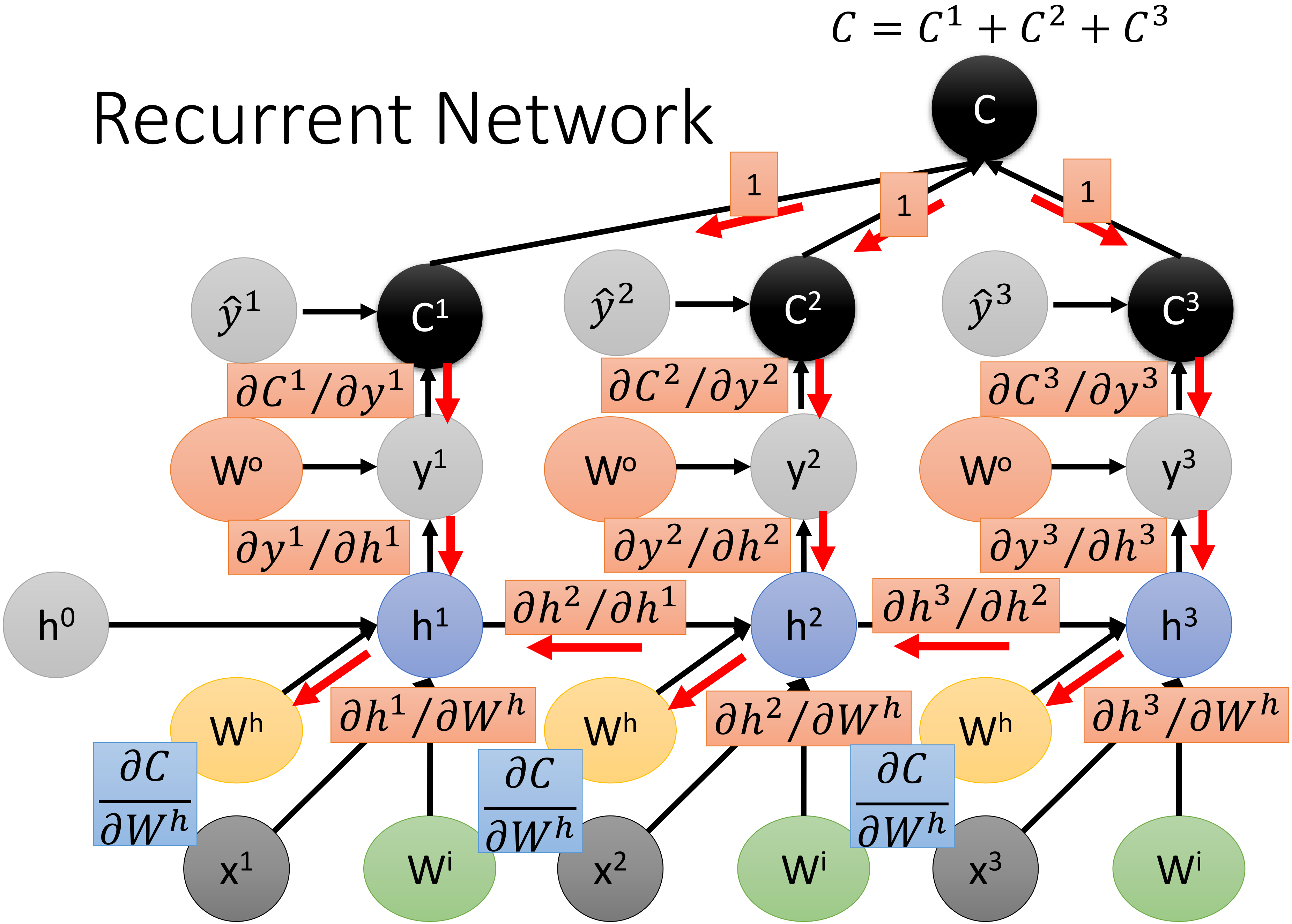
$$h^t = \sigma\left(W^i x^t + W^h h^{t-1}\right)$$

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

Recurrent Network



Recurrent Network

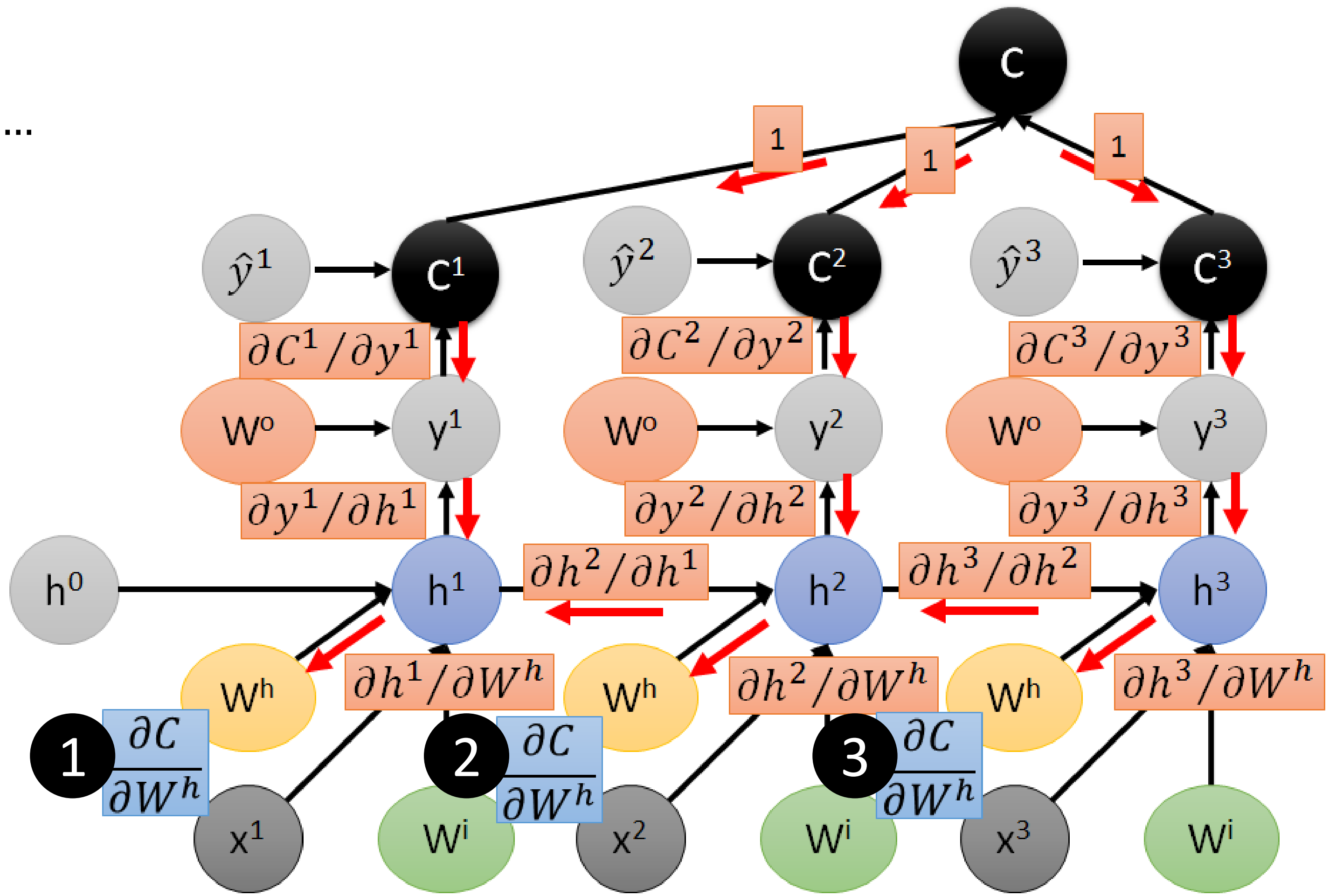


$$\textcircled{1}\ \frac{\partial C}{\partial W^h} = \left[\frac{\partial C^1}{\partial y^1}\frac{\partial y^1}{\partial h^1} + \frac{\partial C^2}{\partial y^2}\frac{\partial y^2}{\partial h^2}\frac{\partial h^2}{\partial h^1} + \frac{\partial C^3}{\partial y^3}\frac{\partial y^3}{\partial h^3}\frac{\partial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

$$\textcircled{2}\ \frac{\partial C}{\partial W^h} = \dots\dots$$

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

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



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