Backpropagation

Chun-Hsiang Chan

Undergraduate Program in Intelligent Computing and Big Data, Chung Yuan Christian University
Master Program in Intelligent Computing and Big Data, Chung Yuan Christian University

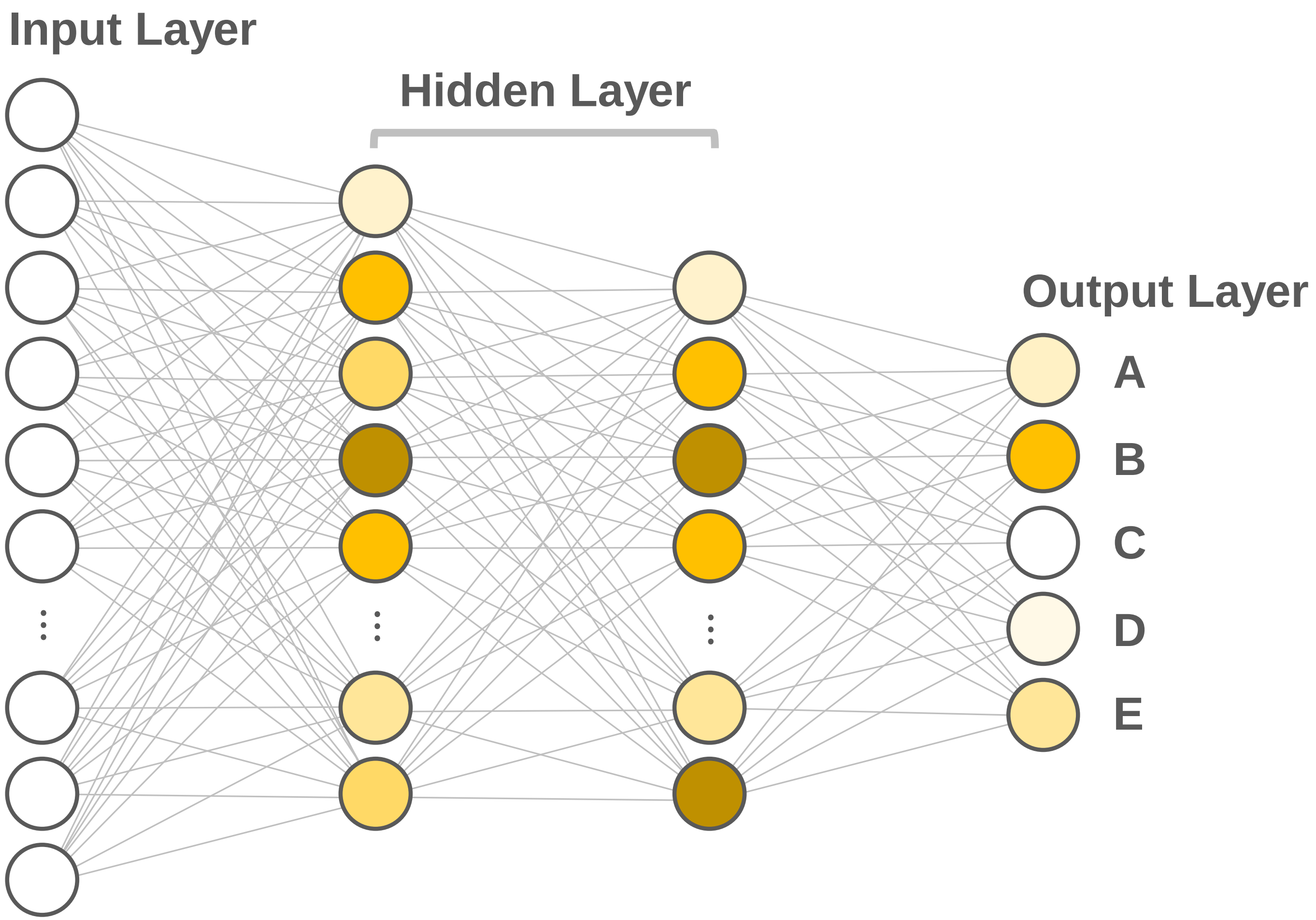
Slide credit: Prof. Hung-Yi Lee & Prof. Yun-Nung Chen

Outline

- Forward and Back Propagation
- Chain Rule
- $\partial C(\theta)/\partial w_{ij}^{l}$
- $\partial C(\theta)/\partial z_{i}^{l}$
- $\partial C(\theta)/\partial z_{i}^{l} = \delta_{i}^{l}$
- Back Propagation
- Summary

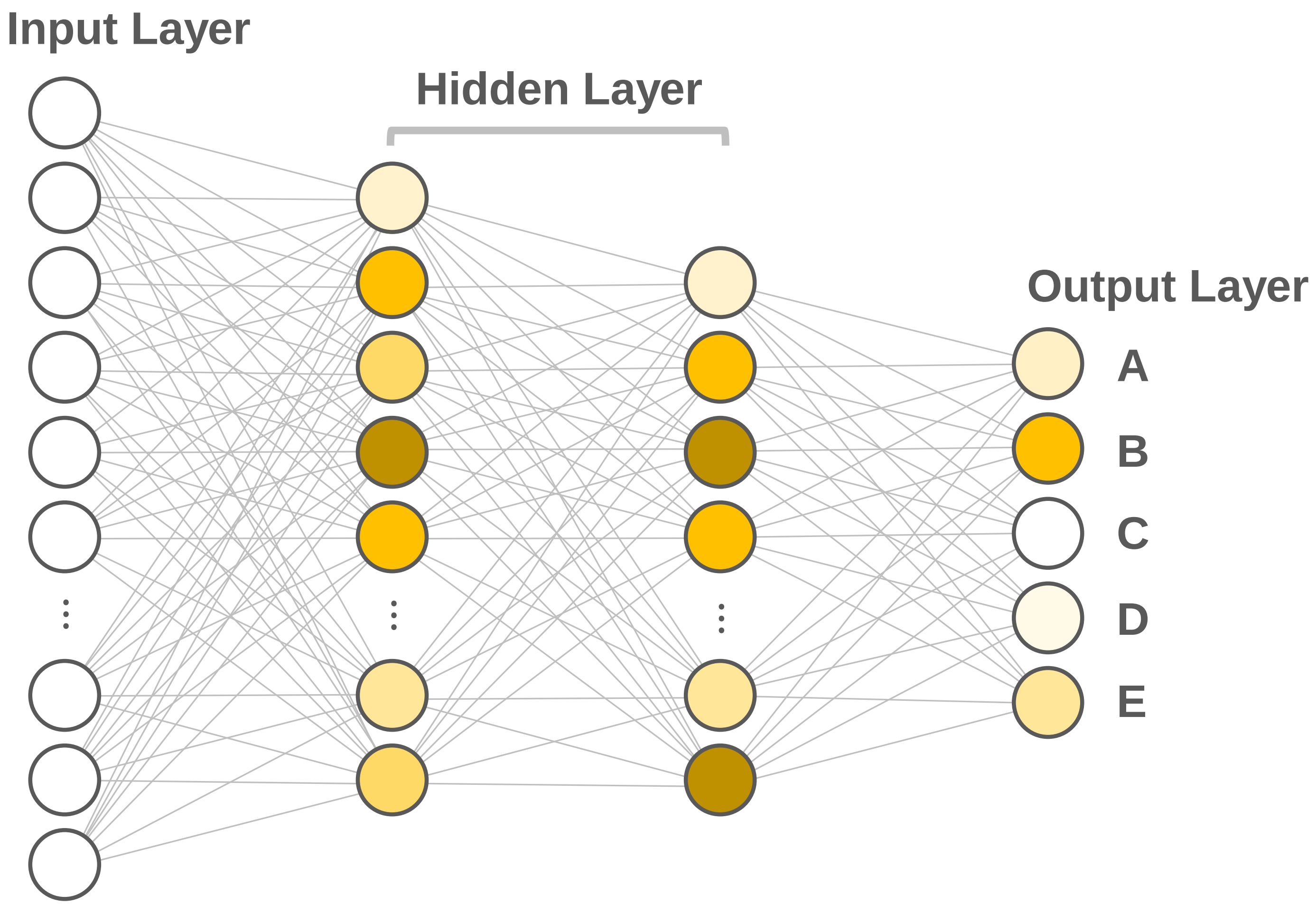
Forward Propagation

- Forward propagation is the process of calculating the output of a neural network given an input. In other words, it is the process of feeding input data through the network's layers in a forward direction to produce an output.
- During forward propagation, each neuron in a layer receives input from the previous layer and applies an activation function to produce an output, which is then passed to the next layer. The output of the final layer is the predicted output of the neural network.



Back Propagation

- Backpropagation computes the gradient of the loss function with respect to the weights of the network for a single input–output example, and does so efficiently, unlike a naive direct computation of the gradient with respect to each weight individually.



Chain Rule

$$\Delta w \to \Delta x \to \Delta y \to \Delta z$$

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

$$= f'(y)f'(x)f'(w)$$

Forward propagation for computing the **cost**

$$= f'\Big(f\big(f(w)\big)\Big)f'\big(f(w)\big)f'(w)$$

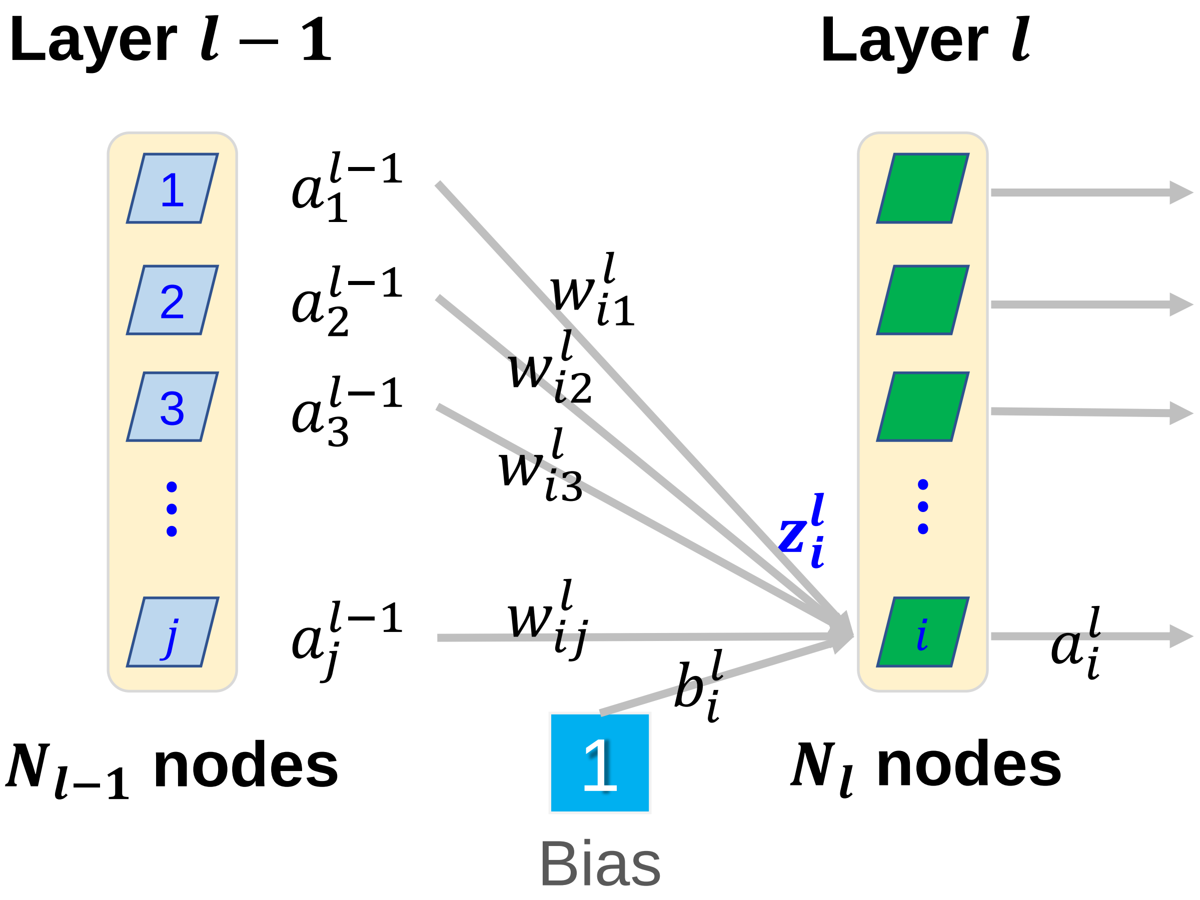
Back propagation for computing the **gradient**

Looking Back – Gradient Decsent

$$y = f(x) = \sigma(W^L \ldots \sigma(W^2 \sigma(W^1 x + b^1) + b^2) \ldots + b^L)$$
$$\theta = \{W^1, b^1, W^2, b^2, \ldots, W^L, b^L\}$$
$$W^l = \begin{bmatrix} w_{11}^l & w_{12}^l & \cdots \\ w_{21}^l & w_{22}^l & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}, b^l = \begin{bmatrix} \vdots \\ b_i^l \\ \vdots \end{bmatrix}$$
$$\nabla C(\theta) = \begin{bmatrix} \vdots \\ \frac{\partial C(\theta)}{\partial w_{ij}^l} \\ \vdots \\ \frac{\partial C(\theta)}{\partial b_i^l} \end{bmatrix}$$

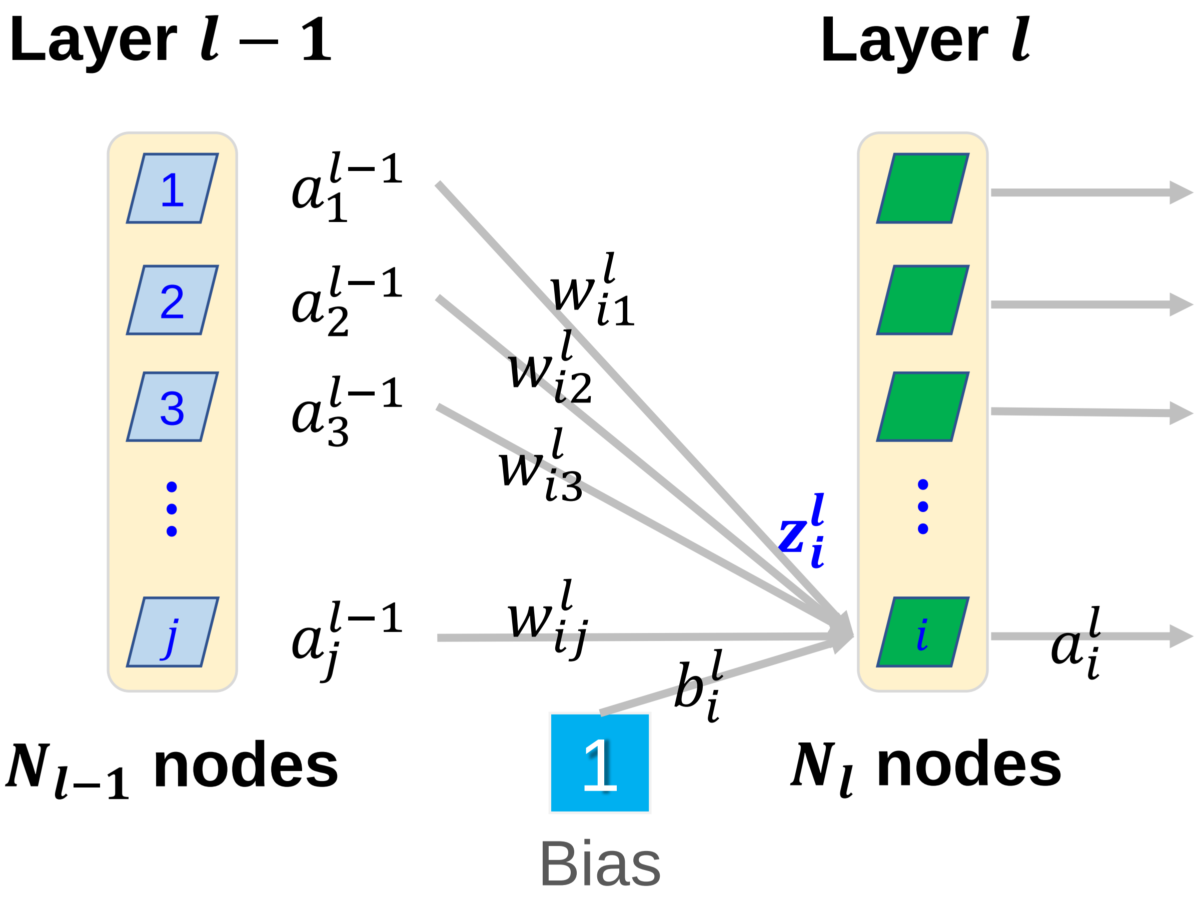
Algorithm
Initialization: start at θ^0
while $(\theta^{(i+1)} \neq \theta^i)$
{
compute the gradients at θ^i
update parameters
$\theta^{i+1} \leftarrow \theta^i - \eta \nabla_\theta C(\theta^i)$
}

$\partial C(\theta)/\partial w_{ij}^{l}$



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

$\partial z_i^l / \partial w_{ij}^l \ (l > 1)$

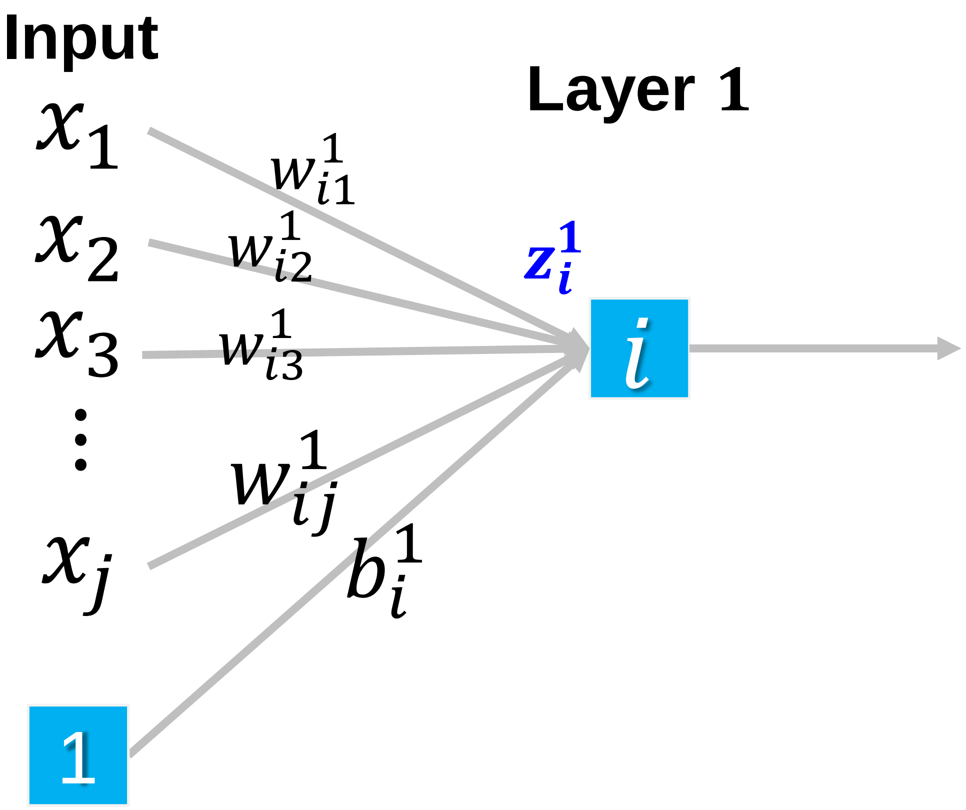


$$z^l = W^l a^{l-1} + b^l$$

$$z_i^l = \sum_j w_{ij}^l a_j^{l-1} + b_i^l$$

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

$\partial z_i^l / \partial w_{ij}^l \ (l = 1)$

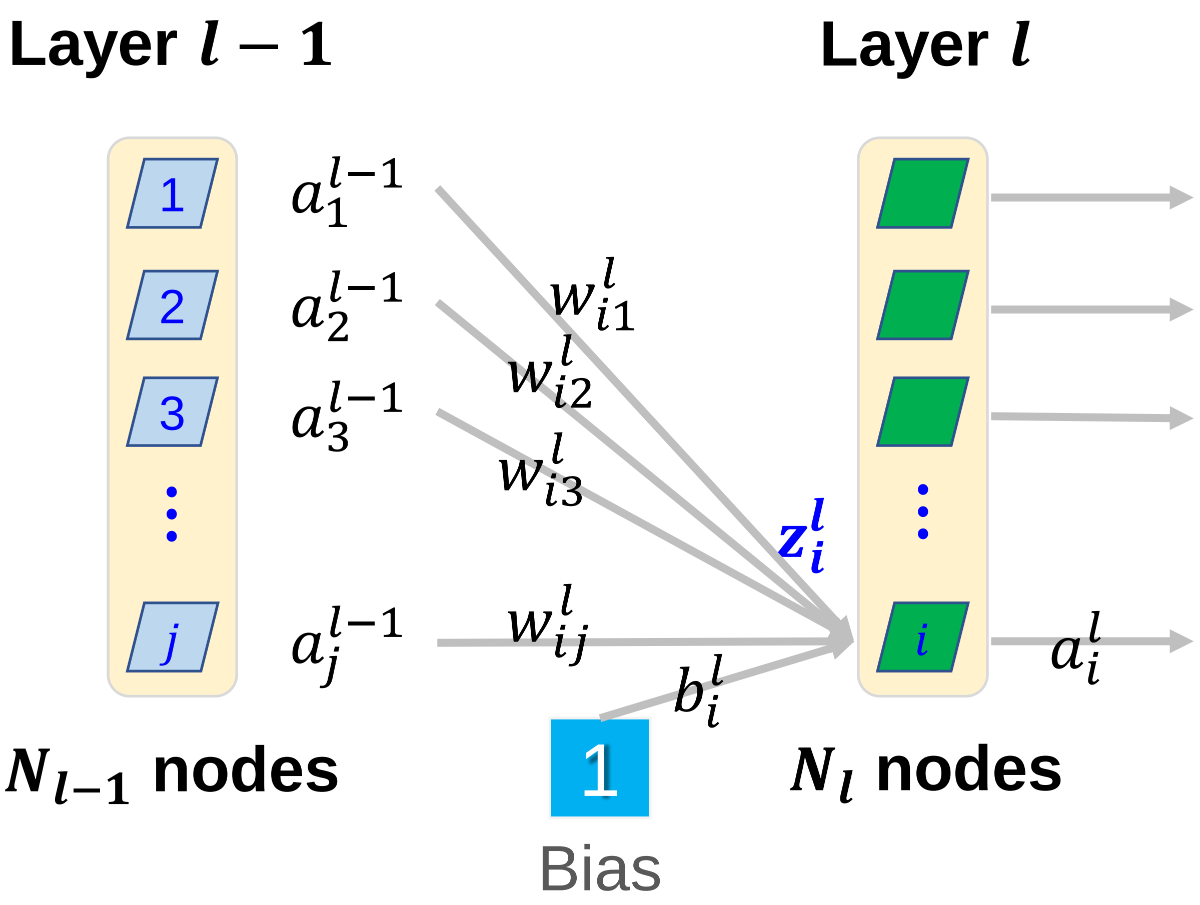


$$z^1 = W^1 x + b^1$$

$$z_i^1 = \sum_j w_{ij}^1 x_j + b_i^1$$

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

$\partial C(\theta)/\partial w_{ij}^{l}$

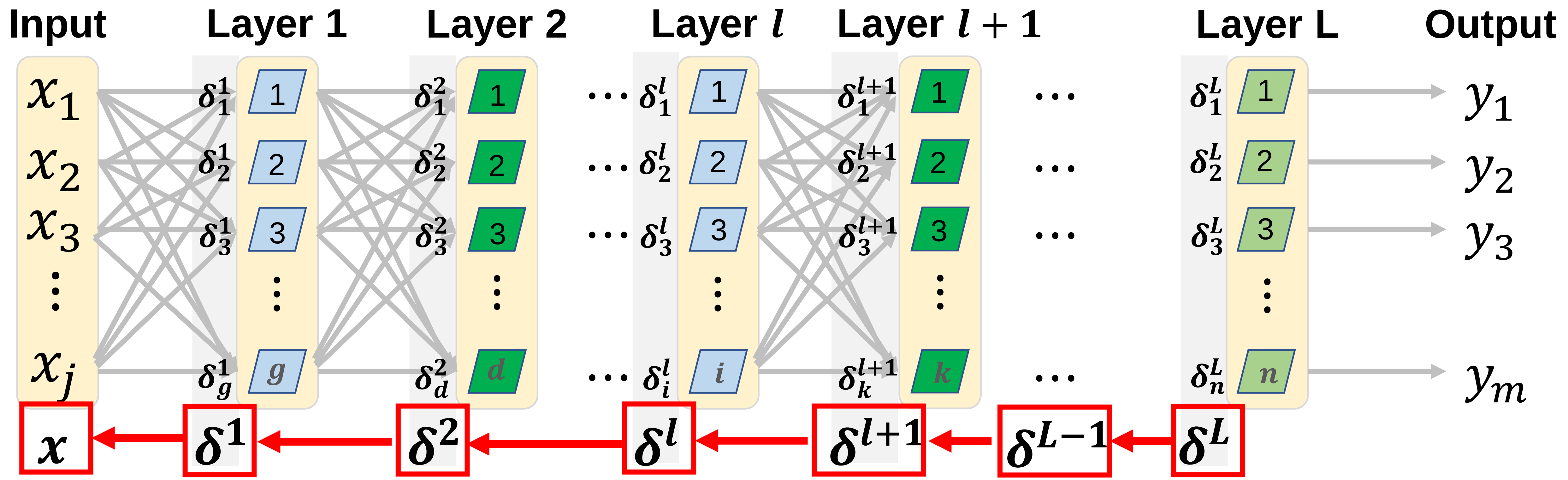


$$\frac{\partial C(\theta)}{\partial w_{ij}^{l}} = \frac{\partial C(\theta)}{\partial z_i^l}\frac{\partial z_i^l}{\partial w_{ij}^{l}}$$

$$\frac{\partial z_i^l}{\partial w_{ij}^{l}} = \begin{cases} a_j^{l-1} & , l > 1 \\ x_j & , l = 1 \end{cases}$$

$$\partial C(\theta)/\partial z_i^l \qquad \frac{\partial C(\theta)}{\partial w_{ij}^l} = \frac{\partial C(\theta)}{\partial z_i^l}\frac{\partial z_i^l}{\partial w_{ij}^l}$$

δ_i^l: *the propagated gradient corresponding to the* $l-th$ *layer*



$\partial C(\theta)/\partial z_i^l = \delta_i^l$

- **Procedure: from layer L to layer 1**

1. Initialization: compute δ^L

$$\delta_i^L = \frac{\partial C}{\partial z_i^L} = \frac{\partial C}{\partial y_i}\frac{\partial y_i}{\partial z_i^L}, \quad \frac{\partial C}{\partial y_i}\ \textit{depends on the loss function}$$

$$\Delta z_i^L \to a_i^L = \Delta y_i \to \Delta C$$

2. Compute δ^L based on δ^{l+1}

$\partial C(\theta)/\partial z_i^l = \delta_i^l$

- **Procedure: from layer L to layer 1**

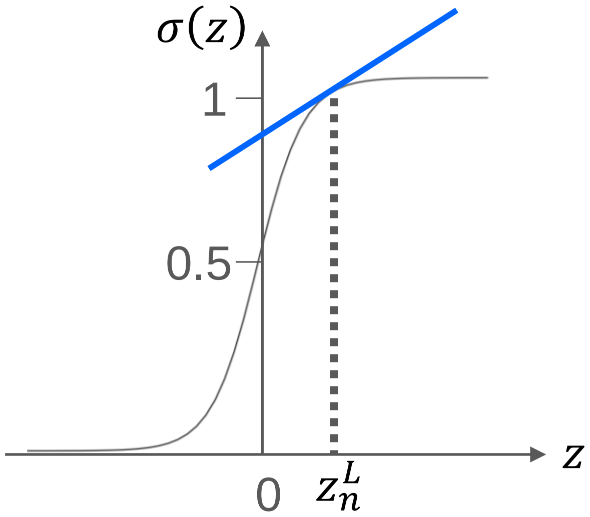
1. Initialization: compute δ^L
2. Compute δ^L based on δ^{l+1}

$$\Delta z_i^L \rightarrow \Delta a_i^L = \Delta y_i \rightarrow \Delta C$$

$$\delta_i^L = \frac{\partial C}{\partial z_i^L}$$

$$\delta_i^L = \frac{\partial C}{\partial y_i}\frac{\partial y_i}{\partial z_i^L} = \frac{\partial C}{\partial y_i}\sigma'(z_i^L),$$

$$where\ y_i = a_i^L = \sigma(z_i^L)$$



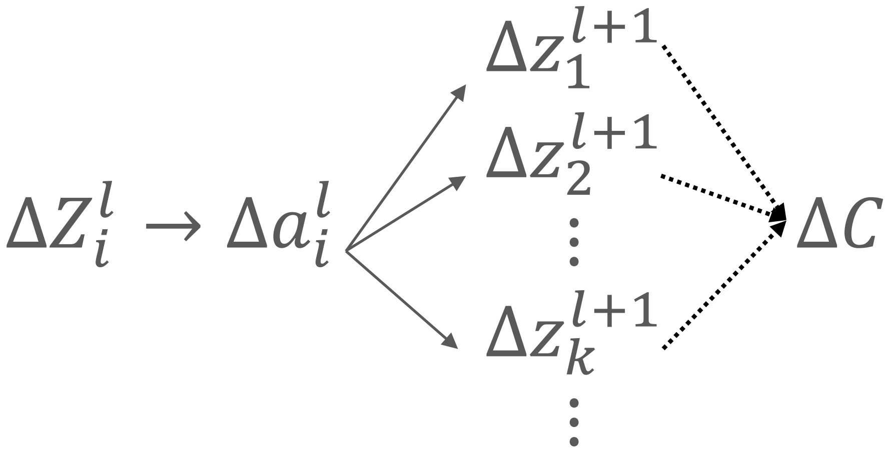
$$\sigma'(z^L) = \begin{bmatrix} \sigma'(z_1^L) \\ \sigma'(z_2^L) \\ \vdots \\ \sigma'(z_i^L) \\ \vdots \end{bmatrix}, \nabla C(y) = \begin{bmatrix} \frac{\partial C}{\partial y_1} \\ \frac{\partial C}{\partial y_2} \\ \vdots \\ \frac{\partial C}{\partial y_i} \\ \vdots \end{bmatrix}$$

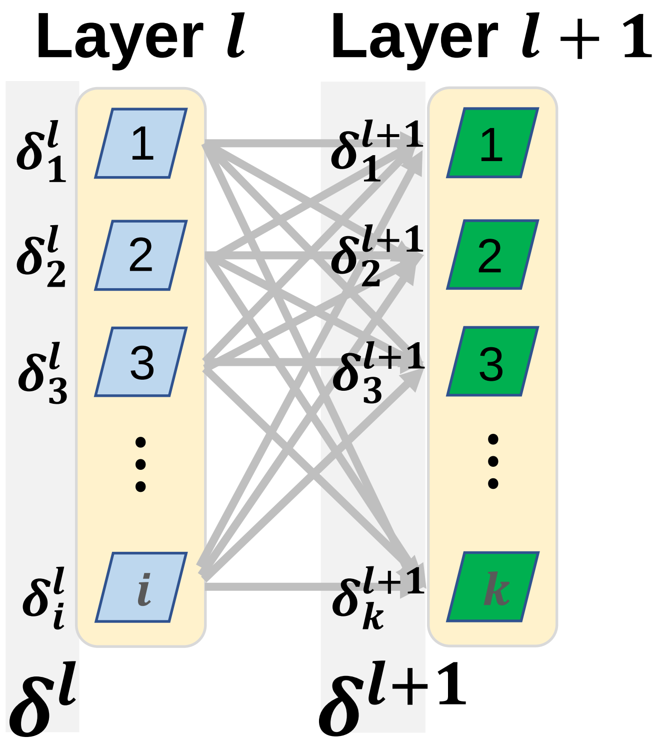
$$\boldsymbol{\delta^L = \sigma'(z^L) \odot \nabla C(y)}$$

$\partial C(\boldsymbol{\theta})/\partial z_i^l = \boldsymbol{\delta}_i^l$

- **Procedure: from layer L to layer 1**

1. Initialization: compute δ^L
2. Compute δ^L based on δ^{l+1}





$$\delta_i^l = \frac{\partial C}{\partial z_i^l} = \sum_k \left(\frac{\partial C}{\partial z_k^{l+1}} \frac{\partial z_k^{l+1}}{\partial a_i^l} \frac{\partial a_i^l}{\partial z_i^l} \right) = \frac{\partial a_i^l}{\partial z_i^l} \sum_k \left(\boxed{\frac{\partial C}{\partial z_k^{l+1}}} \frac{\partial z_k^{l+1}}{\partial a_i^l} \right)$$

$\partial C(\theta)/\partial z_i^l = \delta_i^l$

- **Procedure: from layer L to layer 1**

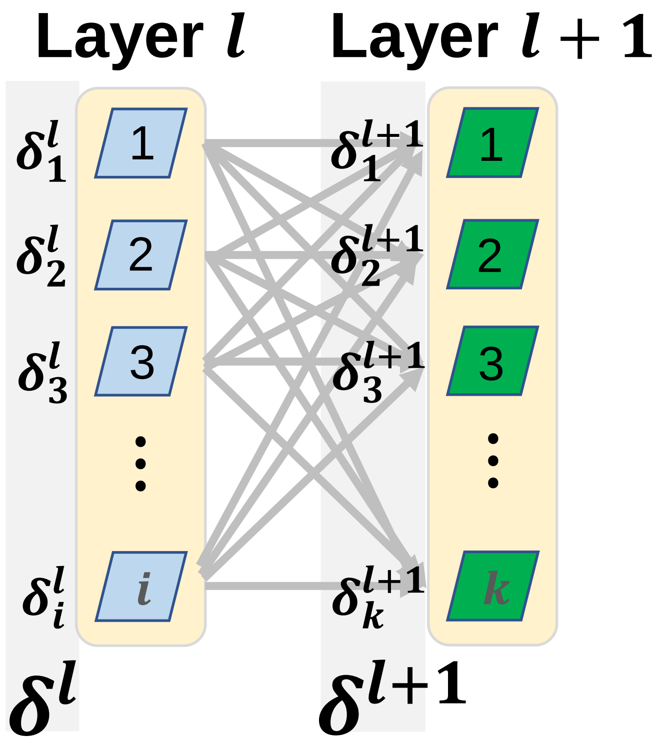
1. Initialization: compute δ^L
2. Compute δ^L based on δ^{l+1}

$$\delta_i^l = \frac{\partial a_i^l}{\partial z_i^l} \sum_k \frac{\partial z_k^{l+1}}{\partial a_i^l} \delta_k^{l+1}$$

$$\delta_i^l = \sigma'(z_i) \sum_k \frac{\partial z_k^{l+1}}{\partial a_i^l} \delta_k^{l+1}$$

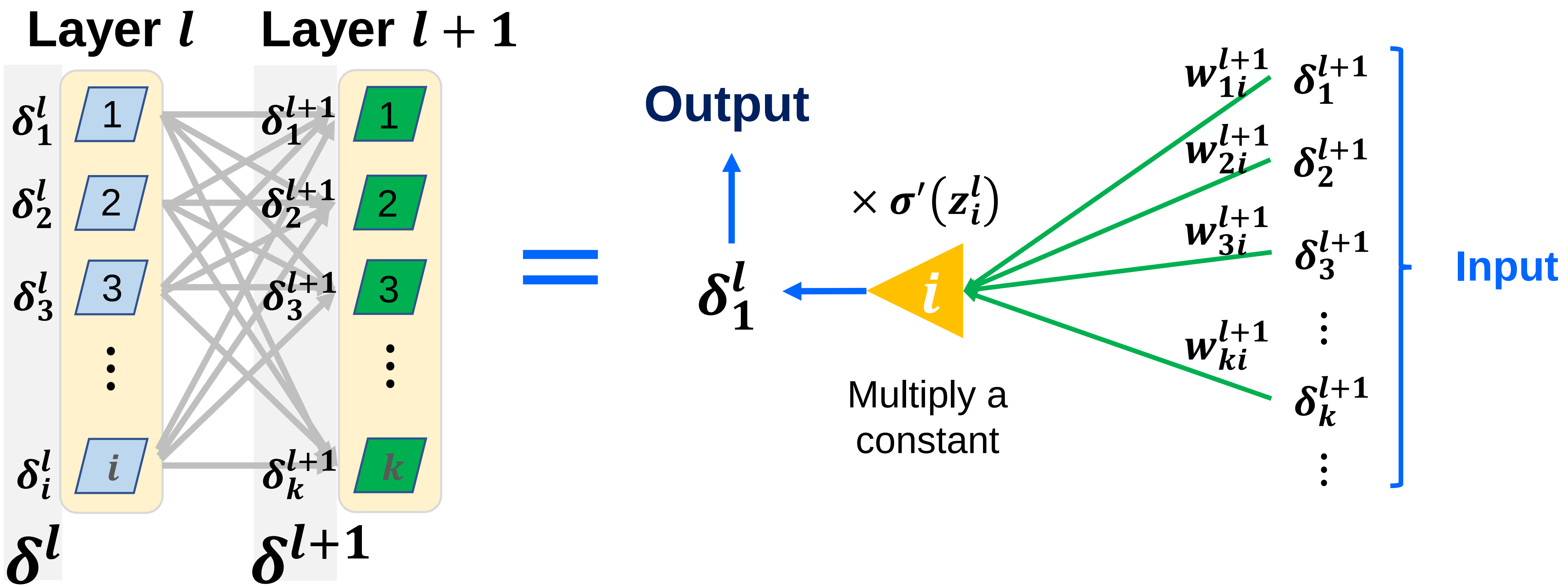
$$\frac{\partial z_k^{l+1}}{\partial a_i^l} \rightarrow = \sum_i w_{ki}^{i+1} a_i^l + b_k^{l+1}$$

$$\delta_i^l = \sigma'(z_i) \sum_k w_{ki}^{l+1} \delta_k^{l+1}$$



$$\partial C(\theta)/\partial z_i^l = \delta_i^l$$

- The propagation: $\delta_i^l = \sigma'(z_i) \sum_k w_{ki}^{l+1} \delta_k^{l+1}$

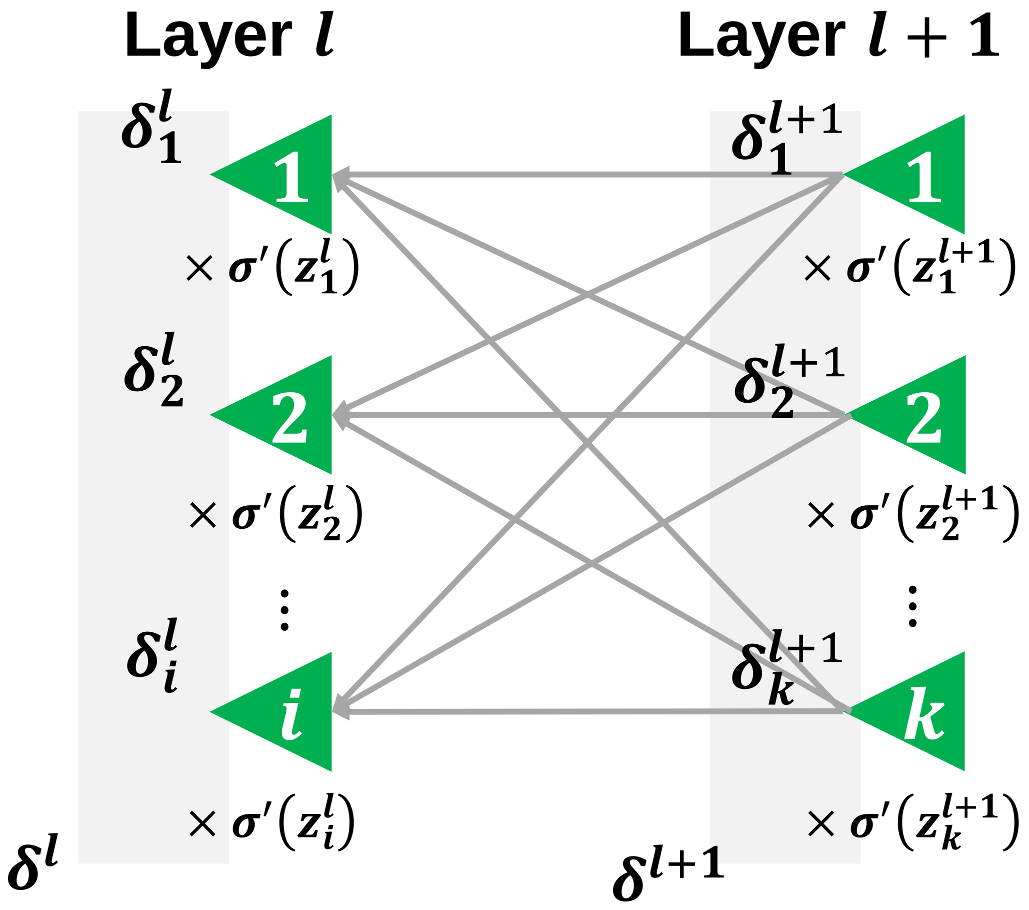


$$\partial C(\theta)/\partial z_i^l = \delta_i^l$$

$$\delta_i^l = \sigma'(z_i) \sum_k w_{ki}^{l+1} \delta_k^{l+1}$$

$$\sigma'(z^l) = \begin{bmatrix} \sigma'(z_1^l) \\ \sigma'(z_2^l) \\ \vdots \\ \sigma'(z_i^l) \\ \vdots \end{bmatrix}$$

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

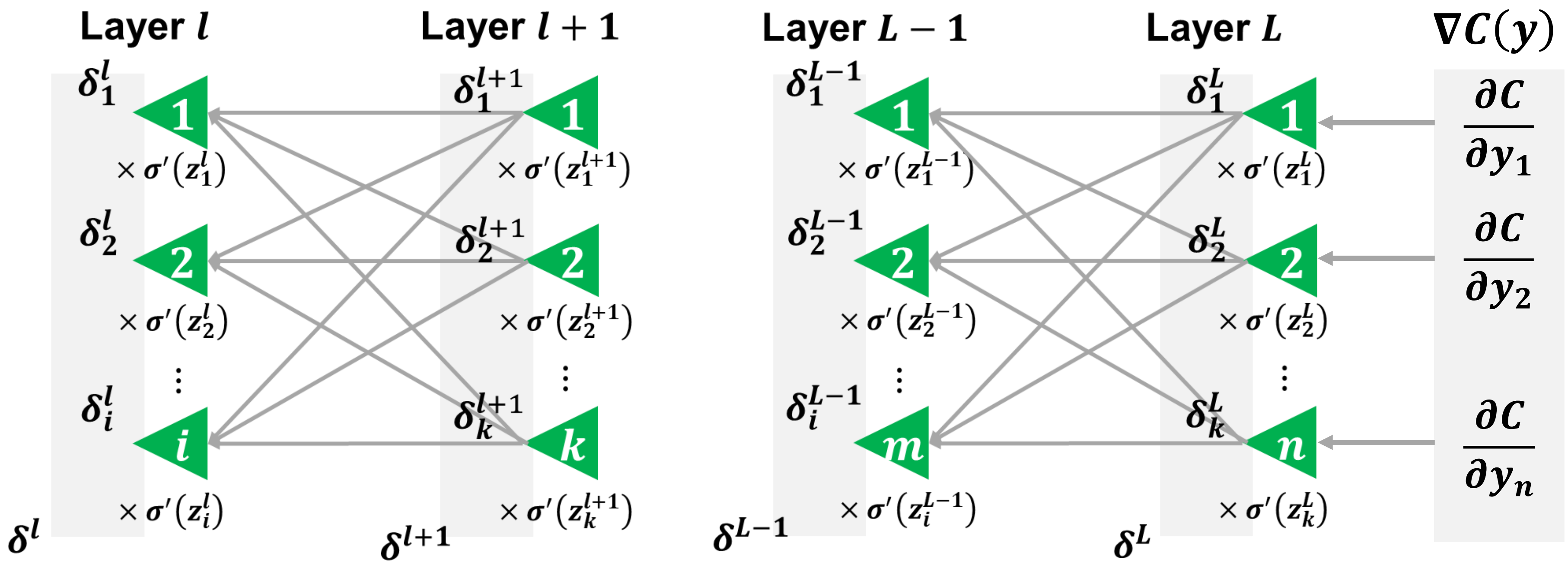


$$\partial C(\theta)/\partial z_i^l = \delta_i^l$$

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

- **Procedure: from layer L to layer 1**

1. Initialization: compute δ^L $\quad \delta^L = \sigma'(z^L) \odot \nabla C(y)$
2. Compute δ^{l-1} based on δ^l $\quad \delta^l = \sigma'(z^l) \odot (W^{l+1})^T \delta^{l+1}$



Back Propagation

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

$$\frac{\delta z_i^i}{\partial w_{ij}^l} = \begin{cases} a_j^{l-1} & ,l > 1 \\ x_j & ,l = 1 \end{cases}$$

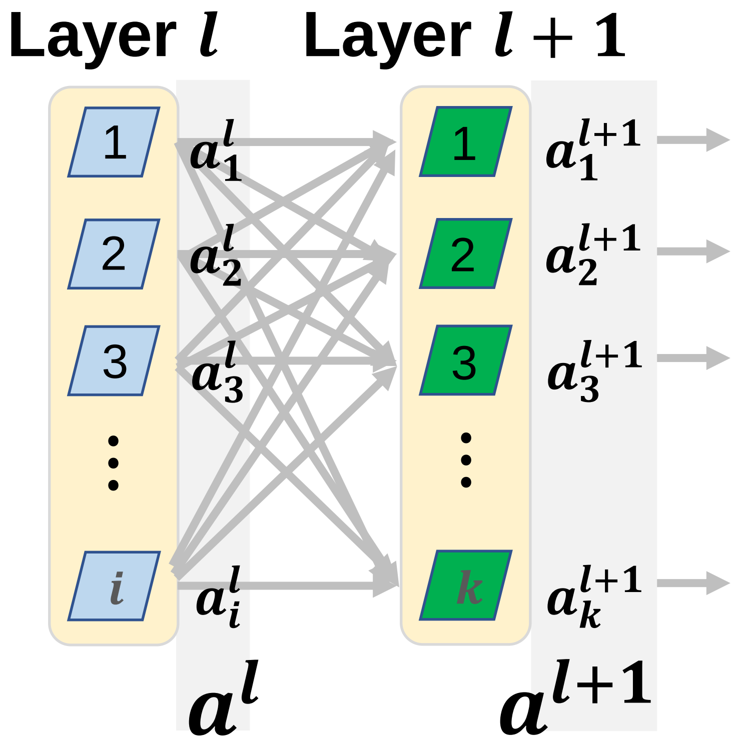
Forward Direction

$$z^1 = W^1 x + b^1 \qquad a^1 = \sigma(z^1)$$

$$\vdots \qquad \vdots$$

$$z^l = W^l x^{l-1} + b^l \qquad a^l = \sigma(z^l)$$

$$\vdots \qquad \vdots$$



Back Propagation

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

$$\frac{\partial C(\theta)}{\partial z_i^{l}} = \delta_i^l$$

Backward Direction

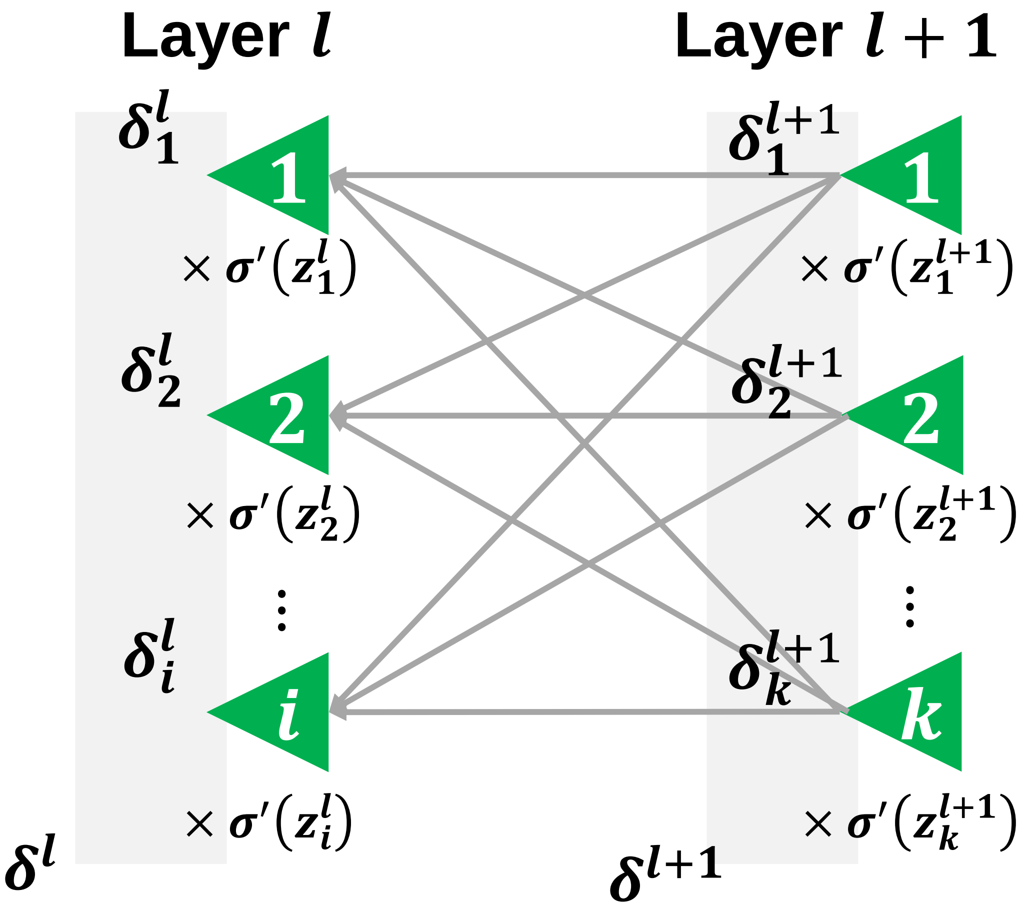
$$\boldsymbol{\delta}^L = \sigma'(\mathbf{z}^L) \odot \nabla C(\mathbf{y})$$

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

$\vdots$

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

$\vdots$



Gradient Descent

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

$$\theta = \{W^1, b^1, W^2, b^2, \ldots, W^L, b^L\}$$

$$W^l = \begin{bmatrix} w_{11}^l & w_{12}^l & \cdots \\ w_{21}^l & w_{22}^l & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}, b^l = \begin{bmatrix} \vdots \\ b_i^l \\ \vdots \end{bmatrix}$$

$$\nabla C(\theta) = \begin{bmatrix} \vdots \\ \frac{\partial C(\theta)}{\partial w_{ij}^l} \\ \vdots \\ \frac{\partial C(\theta)}{\partial b_i^l} \end{bmatrix}$$

Algorithm

Initialization: start at θ^0

while $(\theta^{(i+1)} \neq \theta^i)$

{

compute the gradients at θ^i

update parameters

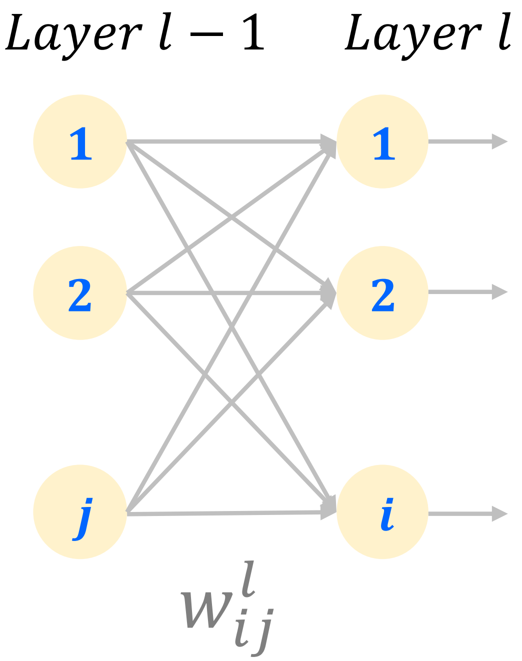
$\theta^{i+1} \leftarrow \theta^i - \eta\nabla_\theta C(\theta^i)$

}

Gradient descent in neural network requires to update millions of parameters …
However, it is time-consuming and an inefficient process…
Therefore, we will introduce another solution – **backpropagation**.

Summary

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



$\frac{\partial C(\theta)}{\partial z_i^l} = \delta_i^l$

$$\frac{\partial z_i^l}{\partial w_{ij}^l} = \begin{cases} a_j^{l-1} & , l > 1 \\ x_j & , l = 1 \end{cases}$$

Forward Direction

$$z^1 = W^1 x + b^1 \qquad a^1 = \sigma(z^1)$$
$$\vdots \qquad \vdots$$
$$z^l = W^l x^{l-1} + b^l \qquad a^l = \sigma(z^l)$$
$$\vdots \qquad \vdots$$

Backward Direction

$$\delta^L = \sigma'(z^L) \odot \nabla C(y)$$
$$\delta^{L-1} = \sigma'(z^{L-1}) \odot (W^L)^T \delta^L$$
$$\vdots$$
$$\delta^l = \sigma'(z^l) \odot (W^{l+1})^T \delta^{l+1}$$
$$\vdots$$

References

- 臺大電機系李宏毅教授講義
- 臺大資工系陳縕儂教授講義
- NVIDIA Deep Learning Tutorial

Thank you for your attention!