Hints for Backpropagation

Furuzuki Lab

March, 2025

Hint 1:

In previous lessons and assignments, we discussed how to use computational graphs to demonstrate how deep learning frameworks utilize the **Chain Rule** to implement automatic differentiation for gradient descent.

Notice that there are two major modes of automatic differentiation:

1. **Forward Mode**: The data flow forward through computational graph.
2. **Reverse Mode**: The gradients are calculated using the **Chain Rule** and flow backward through the computational graph.

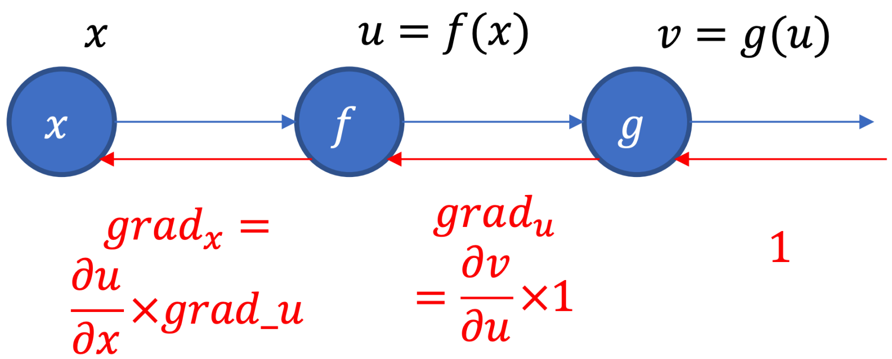

Fig. 1: Two modes of automatic differentiation on the computational graph.
Image credit: Dr. Hangyu DENG

And in deep learning or other machine learning methods, there are three fundamental steps:

1. **Define a set of functions**: $f(\omega_1, b_1)$, $f(\omega_2, b_2)$ ....

2. **Define the goodness of the functions**: This step requires designing a loss function to evaluate the performance of the set of functions. A simple loss function is shown below:

$$L(\omega, b) = \frac{1}{n}\sum_{i=0}^{n}[\hat{y}^i - f^i(\omega, b)]$$

3. **Find the best function** $f^* = \underset{f}{argmin} L(f)$: This requires finding the optimal parameters $\omega^*, b^* = \underset{\omega, b}{argmin} L(\omega, b)$. This step involves using **Gradient Descent** to update the parameters of the function set.

To apply **Gradient Descent** to update parameters, suppose the set of parameters in the function sets is $\theta = \{\omega_1, \omega_2, \cdots, b_1, b_2, \cdots\}$.

We first calculate the gradient of the set of parameters:

$$\nabla L(\theta) = \begin{bmatrix} \frac{\partial L(\theta)}{\partial \omega_1} \\ \frac{\partial L(\theta)}{\partial \omega_2} \\ \vdots \\ \frac{\partial L(\theta)}{\partial b_1} \\ \frac{\partial L(\theta)}{\partial b_2} \\ \vdots \end{bmatrix}$$

Suppose the initial parameters of the function set are θ^0, we can compute $\nabla L(\theta^0)$ and use it to update the parameters as follows:

$$\theta^1 = \theta^0 - \eta \nabla L(\theta^0)$$

Where η is the learning rate.

We repeat this process to obtain $\theta_2, \theta_3, \cdots$, until the loss function $L(\theta)$ is minimized.

Notice that we can calculate this process manually for one simple function set, but manual calculation is impractical for a function set with millions of parameters, like a deep neural network.

Therefore, **BP algorithm (Backpropagation)** was designed specifically for deep neural networks to calculate the gradients of all parameters efficiently.

Hint 2 *(mainly refer these references: [1, 2, 3]):*

For **BP algorithm (Backpropagation)**, because we calculate the loss in each iteration, we solve for a batch of data at each iteration.
Suppose we have a batch of data:

$$\{(x^1, \hat{y}^1), \cdots, (x^t, \hat{y}^t), \cdots, (x^N, \hat{y}^N)\}$$

For the $t-th$ data point, we have:

$$\begin{aligned} x^t &= \left[x_1^t, \cdots, x_k^t\right] \\ \hat{y}^t &= \left[\hat{y}_1^t, \cdots, \hat{y}_m^t\right] \end{aligned}$$

Let the parameters of the neural network be:

$$\theta = \{\omega_1, \omega_2, \cdots, b_1, b_2, \cdots\}$$

We calculate the loss over a batch; the loss function of the neural network can be expressed as:

$$\begin{aligned} L(\theta) &= \frac{1}{N}\sum_{t=1}^{N} C^t(\theta) \\ &= \frac{1}{N}\sum_{t=1}^{N} \left\| f(x^t; \theta) - \hat{y}^t \right\| \end{aligned}$$

Here, we calculate the distance between each output of the neural network $f(x^t; \theta)$ and its corresponding label $\hat{y}^t$. In other words, $C^t(\theta)$ represents the loss for each data point, and by summing these losses over the batch, we obtain the total loss function $L(\theta)$.

Similar to gradient descent, our target is to calculate the gradient $\nabla_{\omega,b} L(\theta)$ with respect to all parameters ω and b of the neural network.
Since:

$$L(\theta) = \frac{1}{N}\sum_{t=1}^{N} C^t(\theta)$$

Our objective becomes:

$$\nabla_{\omega,b} L(\theta) = \frac{1}{N}\sum_{t=1}^{N} \nabla_{\omega,b} C^t(\theta)$$

This means that for given ω_{ij}^l and b_i^l, we need to find $\frac{\partial C^t}{\partial \omega_{ij}^l}$ and $\frac{\partial C^t}{\partial b_i^l}$.
Here ω_{ij}^l represents the $i-th$ neuron's $j-th$ ω in layer l, and b_i^l is the $i-th$ neuron's b in layer l, as shown in Fig. 2.

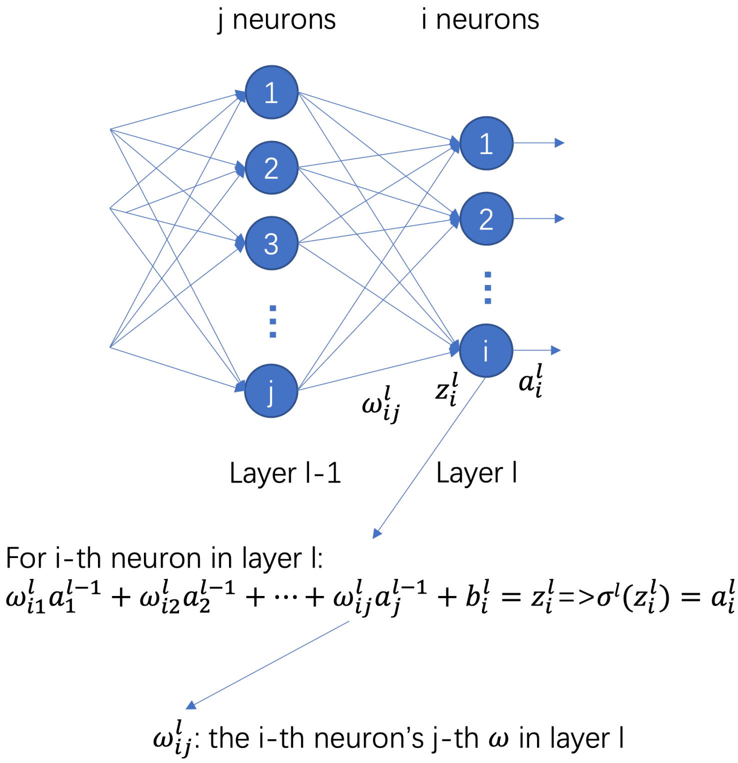

Fig. 2: The figure of w and b in layer l.

Here we take $\frac{\partial C^t}{\partial \omega_{ij}^l}$ as an example:

As shown in Fig. 2, because the change of the sepecific parameter ω_{ij}^l can influence the value of z_i^l, then the change of z_i^l can influence the value of C^t, in other words:

$$\Delta\omega_{ij}^l \to \Delta z_i^l \to \cdots \to \Delta C^t$$

Therefore, according to the **Chain Rule**, we have:

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

Therefore, $\frac{\partial C^t}{\partial \omega_{ij}^l}$ can be regarded as two terms:

$\frac{\partial z_i^l}{\partial \omega_{ij}^l}$ – First Term

For the first term $\frac{\partial z_i^l}{\partial \omega_{ij}^l}$, because:

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

Therefore:

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

Specially, when $l = 1$, then layer $l - 1$ is the input layer, then:

$$z_i^l = \sum_k \omega_{ik}^l x_k^t + b_i^l$$

$$\frac{\partial z_i^l}{\partial \omega_{ij}^l} = x_k^t$$

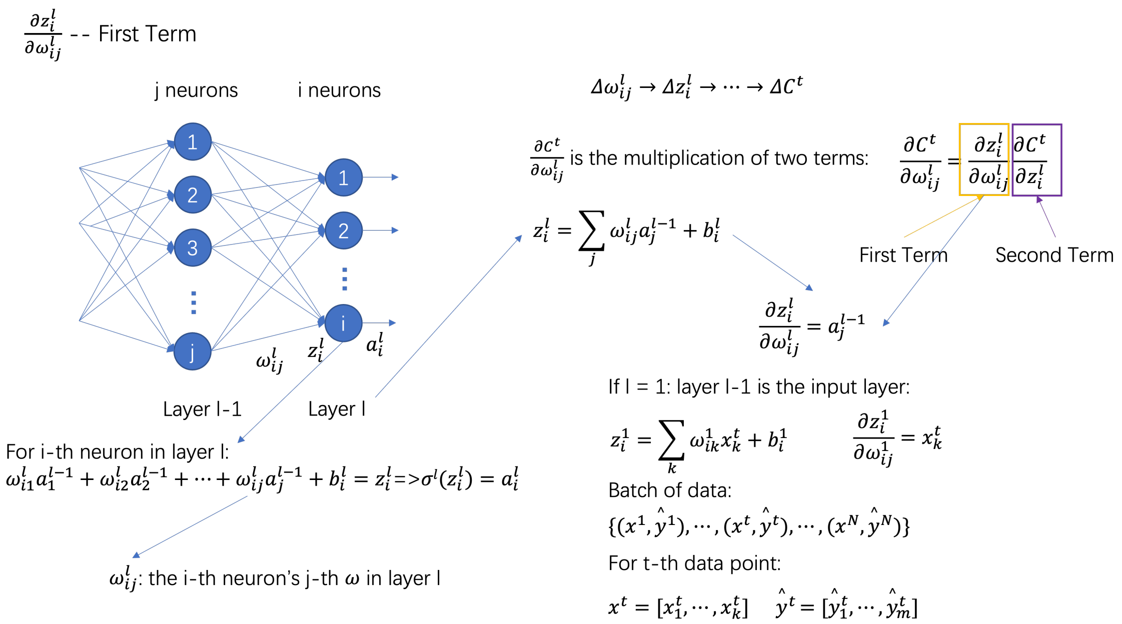

Fig. 3: The calculation of first term.

$\frac{\partial C^t}{\partial z_i^l}$ – Second Term

Here the $\frac{\partial C^t}{\partial z_i^l}$ means the partial derivative of C^t with respect to the $i-th$ neuron's output (before through activation function) of layer l.

To make clear representation, we name the second term $\frac{\partial C^t}{\partial z_i^l}$ as δ_i^l, indicates the $i-th$ neuron's δ in layer l. And we name all δ_i^l in layer l as δ^l.

Suppose we name the output layer as the layer L, then if we can compute δ^L, and we can find the relation between δ^l and δ^{l+1}, then as shown in fig. 4, we can gradually get all δ^l.

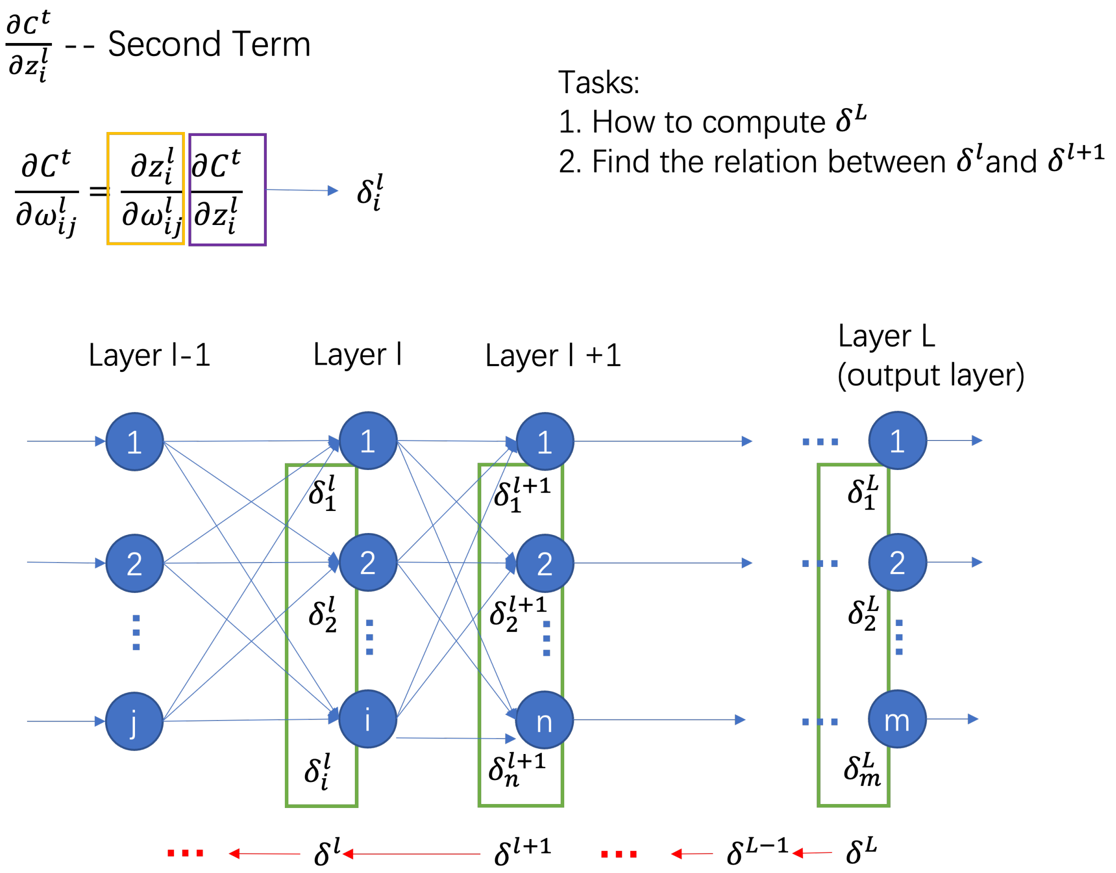

Fig. 4: Name the second term $\frac{\partial C^t}{\partial z_i^l}$ as δ_i^l, and transfer the calculation of δ_i^l to 2 tasks: 1. Compute δ^L and 2. find the relation between δ^l and δ^{l+1}.

Therefore, our task has been transferred as two tasks: 1. Compute δ^L and 2. find the relation between δ^l and δ^{l+1}.

$\frac{\partial C^t}{\partial z_i^l}$ – Second Term – Task 1. Compute δ^L

As we defined before:

$$\delta_i^l = \frac{\partial C^t}{\partial z_i^l}$$

Therefore, for the δ of $m-th$ neuron in output layer L:

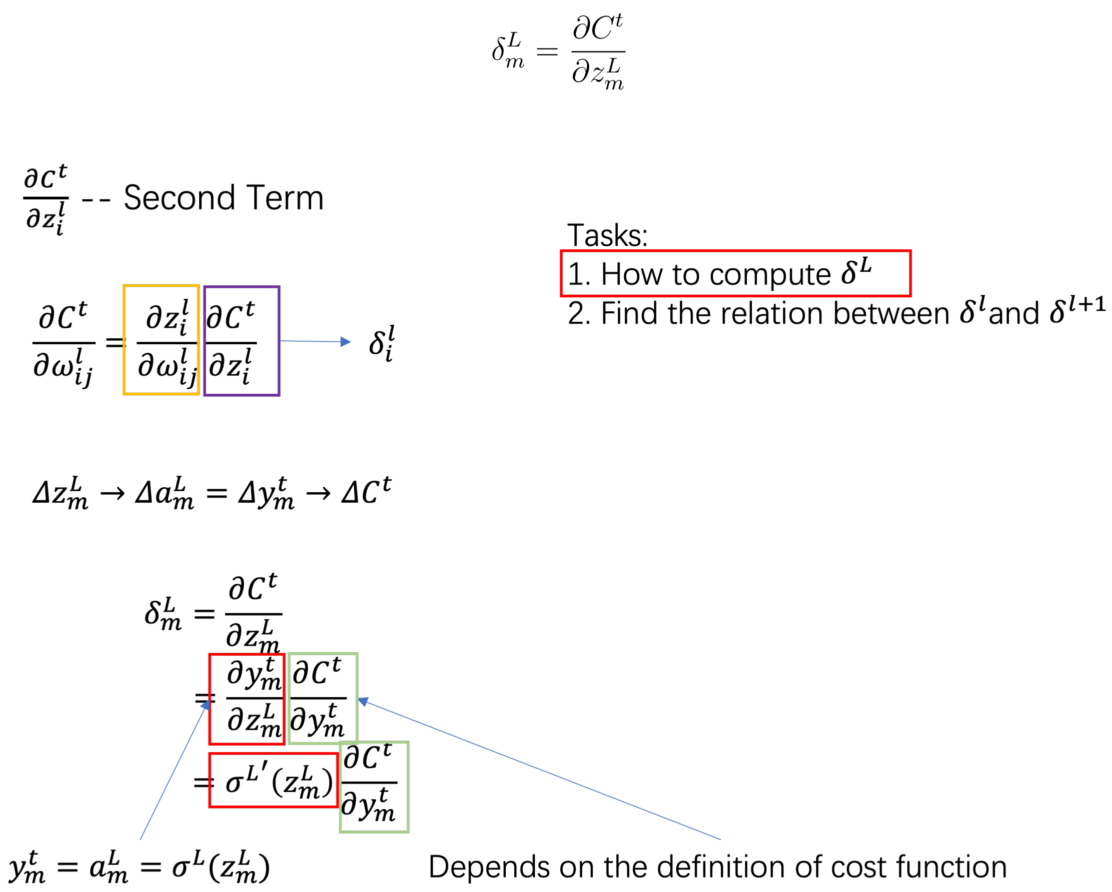

Fig. 5: How to compute δ^L.

Similar to our previous discussion, as shown in Fig. 5, the change of z_m^L can influence the value of a_m^L, which is the output of neural network y_m^t. Then the change of y_m^t can influence the value of C^t, in other words:

$$\Delta z_m^L \rightarrow \Delta a_m^L = \Delta y_m^t \rightarrow \Delta C^t$$

Therefore, according to the chain rule:

$$\begin{aligned}\delta_m^L &= \frac{\partial C^t}{\partial z_m^L} \\ &= \frac{\partial y_m^t}{\partial z_m^L} \frac{\partial C^t}{\partial y_m^t}\end{aligned}$$

Because:

$$y_m^t = a_m^L = \sigma^L(z_m^L)$$

where $\sigma^L(x)$ is the activation function of the output layer.
Therefore, **for the first part** $\frac{\partial y_m^t}{\partial z_m^L}$ of $\delta_m^L = \frac{\partial y_m^t}{\partial z_m^L}\frac{\partial C^t}{\partial y_m^t}$, the result is the derivative of $\sigma^L(x)$ on z_m^L:

$$\frac{\partial y_m^t}{\partial z_m^L} = \sigma^{L'}(z_m^L)$$

As **for the second part** $\frac{\partial C^t}{\partial y_m^t}$ of $\delta_m^L = \frac{\partial y_m^t}{\partial z_m^L}\frac{\partial C^t}{\partial y_m^t}$, the value of it depends on the definition of cost function.
For example, for $t-th$ data point's loss function $C^t(\theta)$, we have defined it as the distance between the output of the neural network $f(x^t;\theta)$ and its corresponding label $\hat{y}^t$:

$$C^t(\theta) = \left\| f(x^t;\theta) - \hat{y}^t \right\|$$

If we further define it as:

$$\begin{aligned} C^t(\theta) &= \left\| f(x^t;\theta) - \hat{y}^t \right\| \\ &= \frac{1}{2m}\sum_m (y_m^t - \hat{y_m}^t)^2 \end{aligned}$$

Then the second part $\frac{\partial C^t}{\partial y_m^t}$ is:

$$\begin{aligned} \frac{\partial C^t}{\partial y_m^t} &= \frac{\partial \frac{1}{2m}\sum_m (y_m^t - \hat{y_m}^t)^2}{\partial y_m^t} \\ &= \frac{\partial \frac{1}{2m}(y_m^t - \hat{y_m}^t)^2}{\partial y_m^t} \\ &= \frac{1}{m}(y_m^t - \hat{y_m}^t) \end{aligned}$$

similarly we can calculate the other δ_m^L of output layer L and get the δ^L

$\frac{\partial C^t}{\partial z_i^l}$ – Second Term – Task 2. Find the relation between δ^l and δ^{l+1}

As shown in Fig. 6, similar to the previous discussion, the change of z_i^l (the input of activation function of layer l) can influence the value of a_i^l (the output of activation function of layer l). Then the change of a_i^l will influence each z_n^{l+1} (the values of the input of activation function of layer $l+1$). Then, these z_n^{l+1} will finally change the value of C^t.

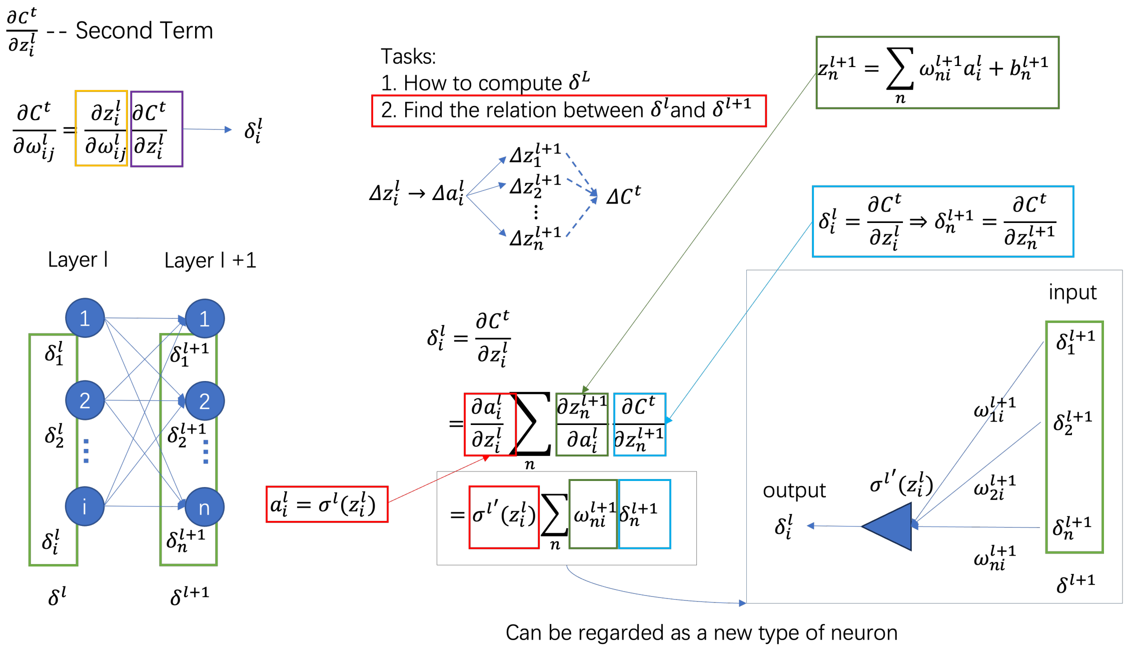

Fig. 6: Find the relation between δ^l and δ^{l+1}.

Therefore, according to the chain rule:

$$\begin{aligned}\delta_i^l &= \frac{\partial C^t}{\partial z_i^l} \\ &= \frac{\partial a_i^l}{\partial z_i^l} \sum_n \frac{\partial z_n^{l+1}}{\partial a_i^l} \frac{\partial C^t}{\partial z_n^{l+1}}\end{aligned}$$

For the first part $\frac{\partial a_i^l}{\partial z_i^l}$ of $\delta_i^l = \frac{\partial a_i^l}{\partial z_i^l} \sum_n \frac{\partial z_n^{l+1}}{\partial a_i^l} \frac{\partial C^t}{\partial z_n^{l+1}}$, because:

$$a_i^l = \sigma^l(z_i^l)$$

Therefore:

$$\frac{\partial a_i^l}{\partial z_i^l} = \sigma^{l'}(z_i^l)$$

For the second part $\frac{\partial z_n^{l+1}}{\partial a_i^l}$ of $\delta_i^l = \frac{\partial a_i^l}{\partial z_i^l} \sum_n \frac{\partial z_n^{l+1}}{\partial a_i^l} \frac{\partial C^t}{\partial z_n^{l+1}}$, because:

$$z_n^{l+1} = \sum_n \omega_{ni}^{l+1} a_i^l + b_n^{l+1}$$

Therefore:

$$\frac{\partial z_n^{l+1}}{\partial a_i^l} = \omega_{ni}^{l+1}$$

And **for the third part** $\frac{\partial C^t}{\partial z_n^{l+1}}$ of $\delta_i^l = \frac{\partial a_i^l}{\partial z_i^l} \sum_n \frac{\partial z_n^{l+1}}{\partial a_i^l} \frac{\partial C^t}{\partial z_n^{l+1}}$, because we defined the second term $\frac{\partial C^t}{\partial z_i^l}$ as:

$$\delta_i^l = \frac{\partial C^t}{\partial z_i^l}$$

Therefore:

$$\frac{\partial C^t}{\partial z_n^{l+1}} = \delta_n^{l+1}$$

Therefore:

$$\begin{aligned}
\delta_i^l &= \frac{\partial C^t}{\partial z_i^l} \\
&= \frac{\partial a_i^l}{\partial z_i^l} \sum_n \frac{\partial z_n^{l+1}}{\partial a_i^l} \frac{\partial C^t}{\partial z_n^{l+1}} \\
&= \sigma^{l'}(z_i^l) \sum_n \omega_{ni}^{l+1} \delta_n^{l+1}
\end{aligned}$$

And such formula can be regarded as a new type of neuron, for such neuron, the input of it are δ^{l+1}, the parameters of it are ω_{ni}^{l+1}, the activation function of it is a constant value $\sigma^{l'}(z_i^l)$, and the output of it is δ_i^l

Therefore, we can use each δ_n^{l+1} of δ^{l+1} to get each δ_i^l, thus we have found the relation between δ^l and δ^{l+1}.

Once we have the values of δ^{l+1}, we can use this formula to calculate the values of δ^l. And because we have already computed the values of δ^L, therefore we can calculate the values of δ^{L-1}, δ^{L-2}, ... and finally get all the value of δ.

In summary

As shown in Fig. 7, our target is to update all the parameters $\theta = \{\omega_1, \omega_2, \cdots, b_1, b_2, \cdots\}$ in the neural network.

This means that for given ω_{ij}^l and b_i^l, we need to find $\frac{\partial C^t}{\partial \omega_{ij}^l}$ and $\frac{\partial C^t}{\partial b_i^l}$.

And we take $\frac{\partial C^t}{\partial \omega_{ij}^l}$ as an example. According to chain rule, $\frac{\partial C^t}{\partial \omega_{ij}^l}$ can be regarded as two terms: $\frac{\partial z_i^l}{\partial \omega_{ij}^l}$ and $\frac{\partial C^t}{\partial z_i^l}$.

For the first term:

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

Specially, when $l = 1$, then layer $l - 1$ is the input layer, then:

$$\frac{\partial z_i^l}{\partial \omega_{ij}^l} = x_k^t$$

For the second term:

we name the second term $\frac{\partial C^t}{\partial z_i^l}$ as δ_i^l, and we transfer the calculation of δ_i^l to 2 tasks: ***1. Compute*** δ^L and ***2. find the relation between*** δ^l ***and*** δ^{l+1}.

1. For the first task, compute δ^L:

$$\begin{aligned}
\delta_m^L &= \frac{\partial C^t}{\partial z_m^L} \\
&= \frac{\partial y_m^t}{\partial z_m^L} \frac{\partial C^t}{\partial y_m^t} \\
&= \sigma^{L'}(z_m^L) \frac{\partial C^t}{\partial y_m^t}
\end{aligned}$$

2. For the second task, find the relation between δ^l and δ^{l+1} can be described as below:

$$\begin{aligned}
\delta_i^l &= \frac{\partial C^t}{\partial z_i^l} \\
&= \frac{\partial a_i^l}{\partial z_i^l} \sum_n \frac{\partial z_n^{l+1}}{\partial a_i^l} \frac{\partial C^t}{\partial z_n^{l+1}} \\
&= \sigma^{l'}(z_i^l) \sum_n \omega_{ni}^{l+1} \delta_n^{l+1}
\end{aligned}$$

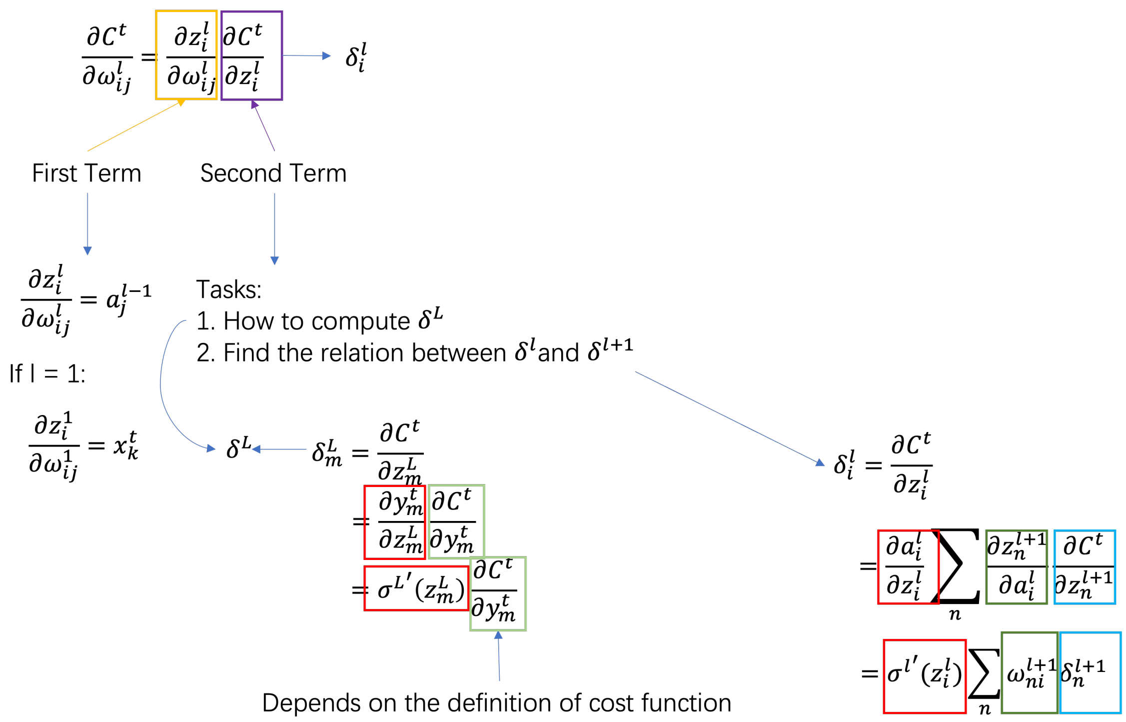

Fig. 7: Summary of BP algorithm.

As shown in Fig. 8. It can be observed that the process of **Backpropagation** is actually the same as that of **Automatic Differentiation**, and **Backpropagation** is a method specifically designed for DNN.

Same as **Automatic Differentiation**, we can forward the data through **Forward Pass** to calculate each a_j^{l-1}, which is the first term of $\frac{\partial C^t}{\partial \omega_{ij}^l}$. Then we can calculate δ^L and pass the data through **Backward Pass** to calculate each δ, which is the second term of $\frac{\partial C^t}{\partial \omega_{ij}^l}$. From this, we calculated the gradient of the loss function C^t with respect to each ω_{ij}^l.

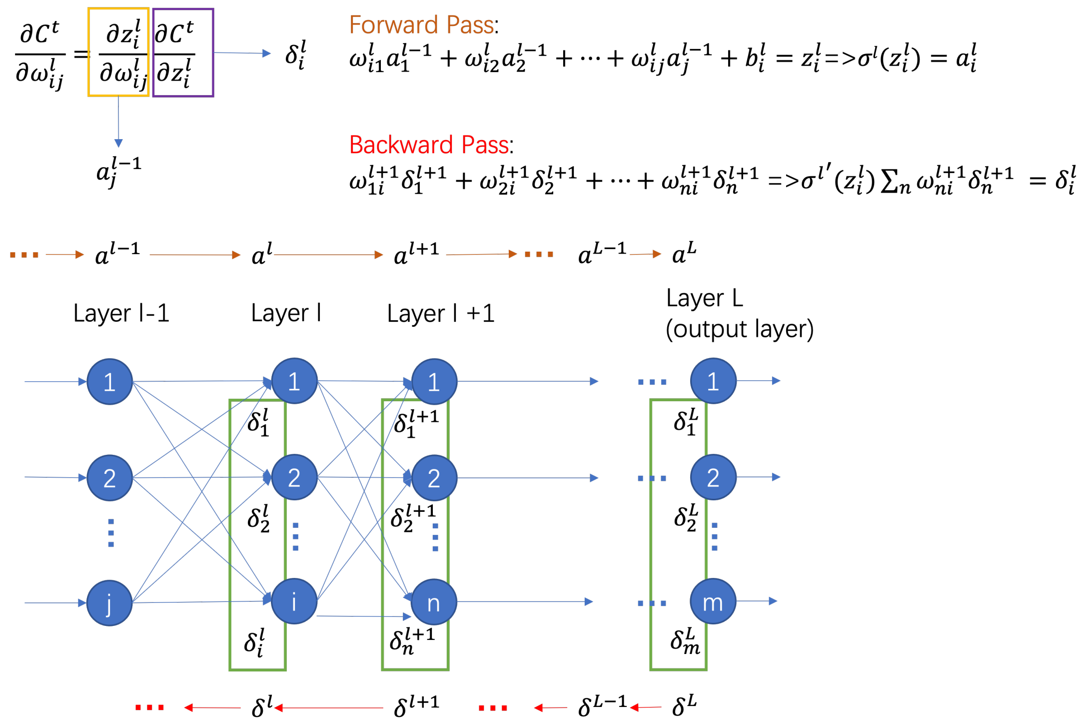

Fig. 8: The forward pass and backward pass of Backpropagation.

Similar to $\frac{\partial C^t}{\partial \omega_{ij}^l}$, we can also calculate the result of $\frac{\partial C^t}{\partial b_i^l}$.

If you are still confused about **Backpropagation**, you may review what you have learned:

Chapter 3: Neural Networks – Training; 3.2 BP Training Algorithm I

Also, you may refer to these references: [4, 5]

References

[1] "Machine learning and having it deep and structured 2015 spring." [Online]. Available: https://speech.ee.ntu.edu.tw/~hylee/mlds/2015-fall.php

[2] "Mlds2015, backpropagation, ppt." [Online]. Available: https://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/Lecture/DNN%20backprop.pdf

[3] "Mlds2015, backpropagation, video." [Online]. Available: https://speech.ee.ntu.edu.tw/~tlkagk/courses/MLDS_2015_2/Lecture/DNN%20backprop.ecm.mp4/index.html

[4] "Ml lecture 7: Backpropagation." [Online]. Available: https://www.youtube.com/watch?v=ibJpTrp5mcE&ab_channel=Hung-yiLee

[5] "Backpropagation calculus." [Online]. Available: https://www.3blue1brown.com/lessons/backpropagation-calculus