1 What is an Artificial Neural Network (ANN)?

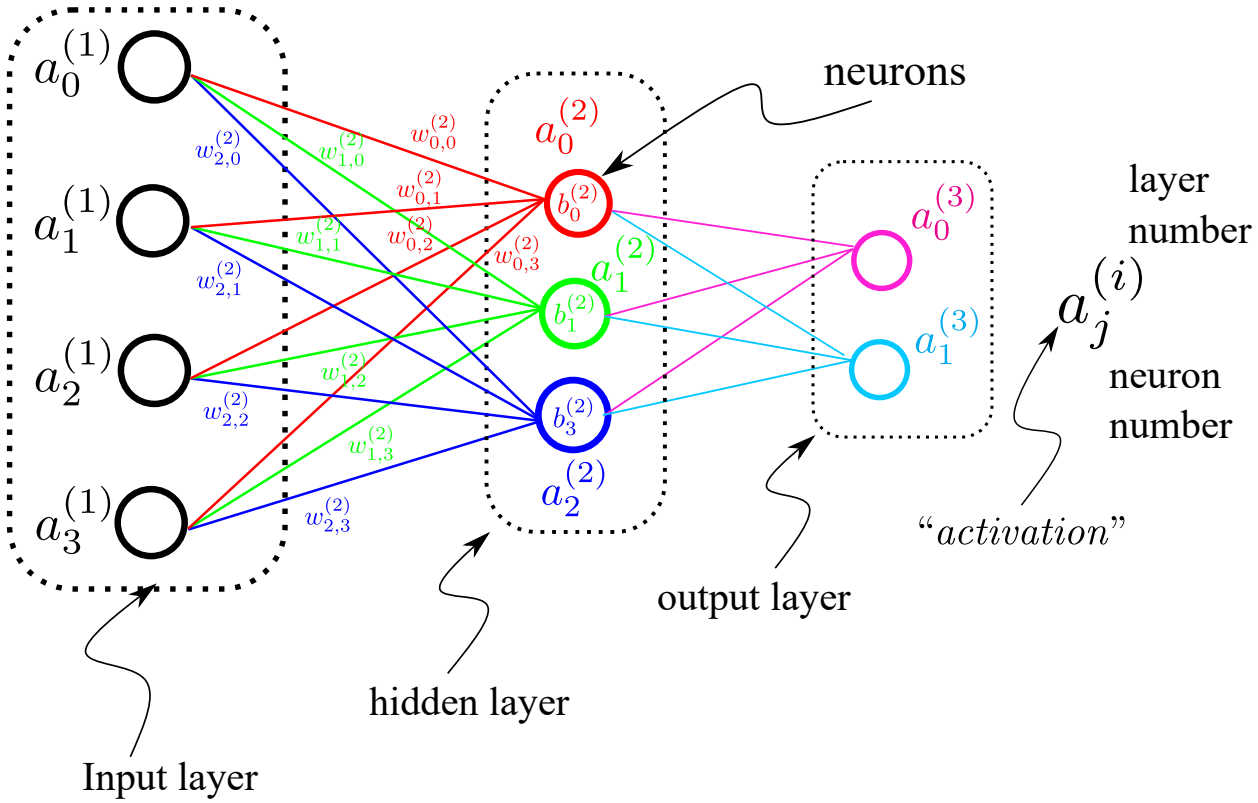


Figure 1: Schematic diagram of a neural network.

An ANN consists of
"Activations" in the second layer can be written as:

$$a_0^{(2)} = \sigma\left(w_{0,0}^{(2)}a_0^{(1)} + w_{0,1}^{(2)}a_1^{(1)} + w_{0,2}^{(2)}a_2^{(1)} + w_{0,3}^{(2)}a_3^{(1)} + b_0^{(2)}\right) \tag{1a}$$

$$a_1^{(2)} = \sigma\left(w_{1,0}^{(2)}a_0^{(1)} + w_{1,1}^{(2)}a_1^{(1)} + w_{1,2}^{(2)}a_2^{(1)} + w_{1,3}^{(2)}a_3^{(1)} + b_1^{(2)}\right) \tag{1b}$$

$$a_2^{(2)} = \sigma\left(w_{2,0}^{(2)}a_0^{(1)} + w_{2,1}^{(2)}a_1^{(1)} + w_{2,2}^{(2)}a_2^{(1)} + w_{2,3}^{(2)}a_3^{(1)} + b_2^{(2)}\right) \tag{1c}$$

In the above equation:

- $w_{i,j}^{(2)} \rightarrow$ are the called the "weights" for the connection between the j^{th} neuron of 1^{st} layer and i^{th} neuron of 2^{nd} layer.
- $b_i^{(2)} \rightarrow$ are called the "biases" for the activation of the i^{th} neuron of 2^{nd} layer.
- The function, σ denotes the sigmoid function.

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{2}$$

Eq. (1) can be written in matrix form as:

$$\begin{pmatrix} a_0^{(2)} \\ a_1^{(2)} \\ a_2^{(2)} \end{pmatrix} = \sigma\left[\begin{pmatrix} w_{0,0}^{(2)} & w_{0,1}^{(2)} & w_{0,2}^{(2)} & w_{0,3}^{(2)} \\ w_{1,0}^{(2)} & w_{1,1}^{(2)} & w_{1,2}^{(2)} & w_{1,3}^{(2)} \\ w_{2,0}^{(2)} & w_{2,1}^{(2)} & w_{2,2}^{(2)} & w_{2,3}^{(2)} \end{pmatrix} \cdot \begin{pmatrix} a_0^{(1)} \\ a_1^{(1)} \\ a_2^{(1)} \\ a_3^{(1)} \end{pmatrix} + \begin{pmatrix} b_0^{(2)} \\ b_1^{(2)} \\ b_2^{(2)} \end{pmatrix}\right] \tag{3}$$

The above equation can be concisely written as:

$$a^{(2)} = \sigma\left[W^{(2)}.a^{(1)} + b^{(2)}\right] \tag{4}$$

Where, $W^{(2)}$ and $b^{(2)}$ are the matrices containing the weights and biases, respectively; corresponding to the connection between neurons of layer no. 2 and 1. Generalizing the above equation for n^{th} and $n-1^{\text{th}}$ layer, we have:

$$a^{(n)} = \sigma\left[W^{(n)}.a^{(n-1)} + b^{(n)}\right] \tag{5}$$

How many weights and biases are there for a given neural network? For the neural network in Fig. 1

- No. of weights $= 3 \times 4 = 12$
- No. of biases $= 3$

In general, for any neural network

- No. of weights $= \sum_{i=1}^{\mathcal{N}} n_i \times n_{i-1}$, ($\mathcal{N} \rightarrow$ Total no. of layers, $n_i \rightarrow$ no. of neurons in the i^{th} layer).
- No. of biases $= \sum_{i=1}^{\mathcal{N}} n_i$.

The total no. of parameters (No. of weights + No. of biases) associated with the neural network,

$$D = \sum_{i=1}^{\mathcal{N}} n_i \times n_{i-1} + \sum_{i=1}^{\mathcal{N}} n_i = \sum_{i=1}^{\mathcal{N}} n_i(1 + n_{i-1}). \tag{6}$$

2 How does a neural network train?

A neural network trains over something called the "training data".

What is training data? A "training data" is large collection of data which consists of pairs of data sets (X_1, Y_1), (X_2, Y_2),....,$(X_{\mathcal{M}}, Y_{\mathcal{M}})$. Where, $\mathcal{M}$ is the total no. of pairs, X_j consists of the activations in the "*input layer*", $a_i^{(in)}$ and Y_i contains the corresponding expected activations in the "*output layer*", $a_i^{(out)}$, for the j^{th} pair, (X_j, Y_j).

How to train your neural network?

1. Randomly initialize all the weights and biases for all the connections between neurons in the neural network.
2. Feed the input activations from a random training data set, say (X_k, Y_k) from your "training data". Recall, that X_k contains the activation of the input layer and Y_k contains the expected activation of the output layer.
3. Calculate something called as the "cost" of this k^{th} data set.

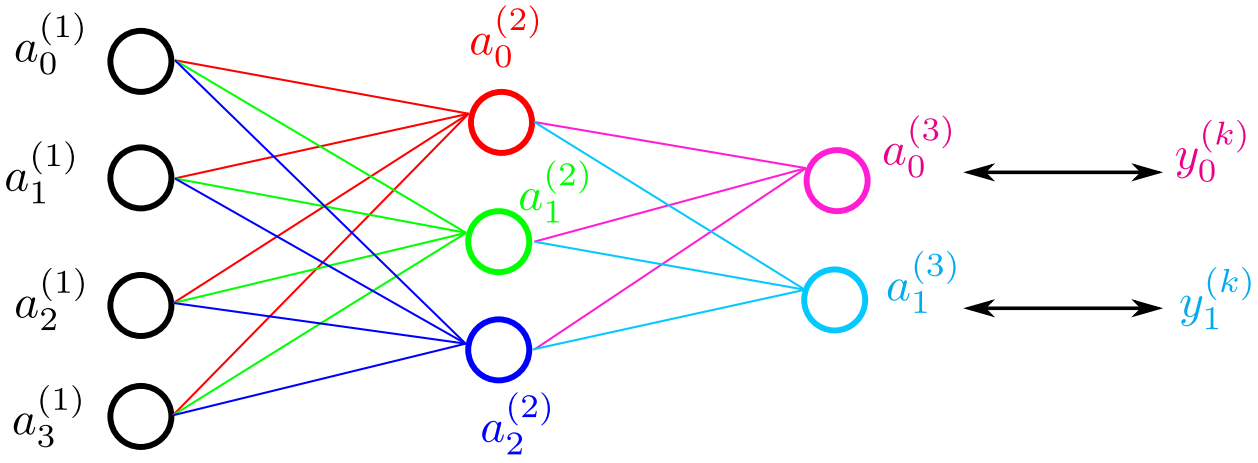


Figure 2: Cost of a particular data set from the "training data".

What is "cost" ?: We first randomly initialize all the weights and biases of our neural network and then feed it the input activations, $a_i^{(1)}$ from the say, k^{th} data set. Let $a_i^{(3)}$ be the corresponding activations of the output layer and $y_i^{(k)}$ be the expected activations of the output layer. The "cost" of the k^{th} training data is then given as:

$$C_k(W^{(2)}, W^{(3)}; b^{(2)}, b^{(3)}) = \left(a_0^{(3)} - e_0^{(3)}\right)^2 + \left(a_1^{(3)} - e_1^{(3)}\right)^2 \tag{7}$$

The generalization of the above equation is:

$$C_k(W^{(2)}, ..., W^{(\mathcal{N})}; b^{(2)}, ..., b^{(\mathcal{N})}) = \sum_{i=0}^{n_{\mathcal{N}}-1} \left(a_i^{(\mathcal{N})} - e_i^{(\mathcal{N})}\right)^2 \tag{8}$$

The above equation gives the cost of the k^{th} training data for a neural network of $\mathcal{N}$ layers. In the above equation $n_{\mathcal{N}}$ is the no. of neurons in the $\mathcal{N}^{\text{th}}$ layer, i.e., the final or output layer. The "total cost" of the neural network is then calculated by taking the average over all the data sets in the "training data", i.e.,

$$C(W^{(2)}, ..., W^{(\mathcal{N})}; b^{(2)}, ..., b^{(\mathcal{N})}) = \frac{1}{\mathcal{M}} \sum_{k=0}^{\mathcal{M}-1} C_k \tag{9}$$

In the above equation, $\mathcal{M}$ is the total no. of data sets in the "training data" and C_k is a shorthand notation for $C_k(W^{(2)}, ..., W^{(\mathcal{N})}; b^{(2)}, ..., b^{(\mathcal{N})})$. This "total cost" $C(W^{(2)}, ..., W^{(\mathcal{N})}; b^{(2)}, ..., b^{(\mathcal{N})})$ is called the "**Cost function**". It is a function of all the weights and biases in a $\sum_{i=1}^{\mathcal{N}} n_i(1 + n_{i-1})$ dimensional space [See, Eq. (6)].

4. Minimise the cost function, i.e., we have to move towards the global minima of the cost function in the $\sum_{i=1}^{\mathcal{N}} n_i(1 + n_{i-1})$ dimensional space of weights and biases. In other words, change the weights and biases along the negative gradient of C until we hit the global minima. This process is called "**Gradient descent**". The weights and biases corresponding to the global minima of C are the desired weights and biases for the neural network.

$$w_{i,j}^{(l)} \rightarrow w_{i,j}^{(l)} - \eta \frac{\partial C}{\partial w_{i,j}^{(l)}}$$
$$b_{i}^{(l)} \rightarrow b_{i}^{(l)} - \eta \frac{\partial C}{\partial b_{i}^{(l)}}$$

In the above equations, η is called the "**learning rate**". It is simply the rate at which we are moving towards the global minima of the C.

5. The averaging over all the data sets in the "training data" would take a lot of time for a real data sets. Therefore, to achieve computational speedup, the entire "training data" is first randomly shuffled and then subdivided into small groups known as a "**mini batch**". Each mini batch consists of a certain no. of data sets and cost function is evaluated mini batch-wise, $C(W^{(2)}, ..., W^{(m)}; b^{(2)}, ..., b^{(m)}) = \frac{1}{m} \sum_{k=0}^{m-1} C_k$ (m is the size of the mini batch). After evaluating the cost function of a particular mini batch, all the weights and biases are updated. The updated weights and biases are then used to recalculate the cost function and its gradient corresponding to another "**mini batch**". The weights and biases are then updated using this gradient to repeat the same process. At each update to the weights and biases, we move towards the global minima of C. Each of these iterations to move towards the global-minima of the cost function is known as an "**epoch**". This process of first randomly shuffling the "training data" and then dividing it into mini batches and applying gradient descent mini batch wise is known as "**stochastic gradient descent**". The method of calculating the gradient of the cost function at each "epoch" (iteration) is known as "**backpropagation**". In the next section, we will discuss the "backpropagation algorithm" in detail.

- **Note**: The parameters, no. of "**epoch**" (or simply iterations), "**mini batch size**", and the "**learning rate**", η are called the "**hyper parameters**" of the neural network.

3 How does back propagation work?

In this section, we will discuss how to calculate the gradient of the cost function using "training data". Consider the simplest neural with only one neuron in each layer, as shown in Fig. 3. As a first step, initialized the weights and biases of the neural network randomly. The neural network is then fed the 0^{th} training data set (i.e., the first data set) of the first "mini batch" [See earlier discussions]. The neural network will then accordingly generate the activation, $a^{(L)}$ corresponding to the 0^{th} training data set of the first "mini batch". Now consider the last two layers of the neural network in Fig. 3. For these two layers, the relevant partial derivatives of the cost corresponding to the 0^{th} training data set are:

$$\frac{\partial C_0}{\partial w^{(L)}} = \frac{\partial z^{(L)}}{\partial w^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C_0}{\partial a^{(L)}} \tag{10a}$$
$$\frac{\partial C_0}{\partial a^{(L-1)}} = \frac{\partial z^{(L)}}{\partial a^{(L-1)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C_0}{\partial a^{(L)}} \tag{10b}$$

The "**chain rule**" in Eq. (10a) is simply saying that a change in $w^{(L)}$ causes a change in $z^{(L)}$, the change in $z^{(L)}$ then causes a change in $a^{(L)}$, which then causes a change in C_0 as illustrated in Fig. 4. Similarly, the chain rule in Eq. (10b) is saying that a change in $a^{(L-1)}$

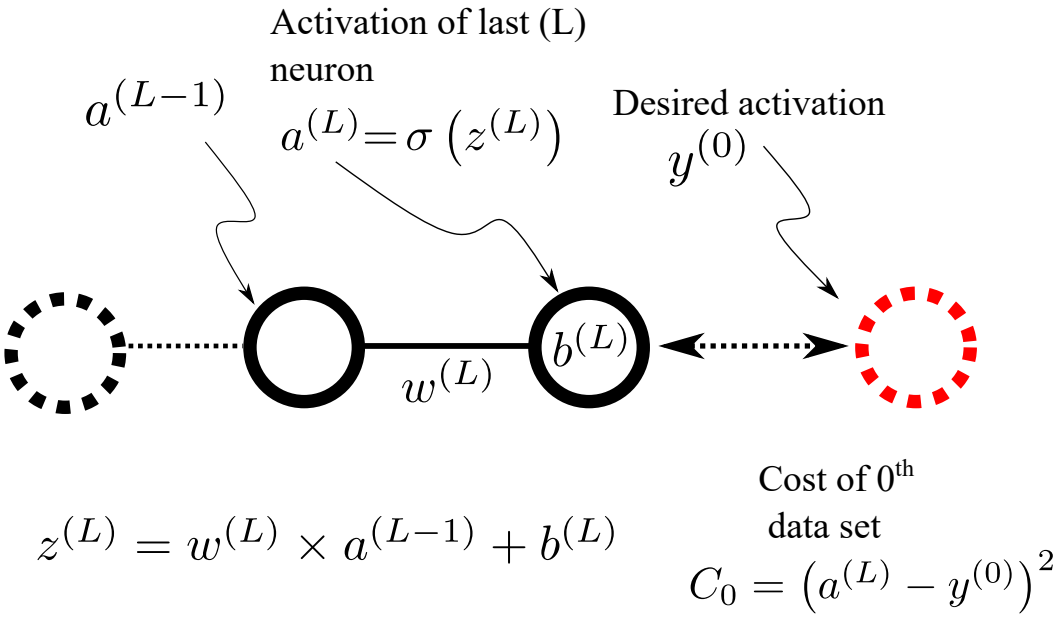


Figure 3: Schematic diagram of a neural network with two layers and two neurons.

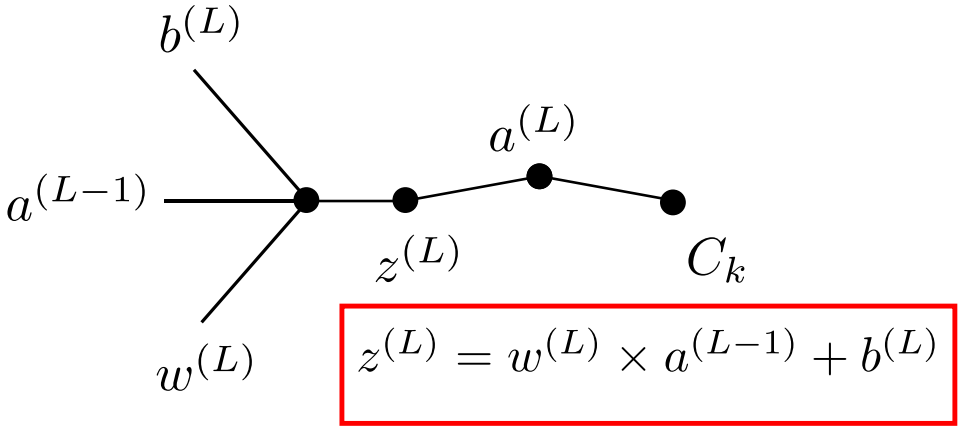


Figure 4: Figure illustrating the chain rule.

causes a change in $z^{(L)}$, the change in $z^{(L)}$ then causes a change in $a^{(L)}$, which then causes a change in C_0 as illustrated in Fig. 4.

$$\frac{\partial C_k}{\partial a^{(L)}} = 2(a^{(L)} - y^{(k)}) \tag{11a}$$

$$\frac{\partial a^{(L)}}{\partial z^{(L)}} = \sigma'(z^{(L)}) \tag{11b}$$

$$\frac{\partial z^{(L)}}{\partial w^{(L)}} = a^{(L-1)} \quad \text{and} \quad \frac{\partial z^{(L)}}{\partial a^{(L-1)}} = w^{(L)} \tag{11c}$$

Therefore, Eq. (10a) and Eq. (10b) implies

$$\frac{\partial C_0}{\partial w^{(L)}} = a^{(L-1)}\sigma'(z^{(L)})2(a^{(L)} - y^{(0)}) \tag{12a}$$

$$\frac{\partial C_0}{\partial a^{(L-1)}} = w^{(L)}\sigma'(z^{(L)})2(a^{(L)} - y^{(0)}) \tag{12b}$$

Similarly,

$$\frac{\partial C_0}{\partial b^{(L)}} = \frac{\partial z^{(L)}}{\partial b^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C_0}{\partial a^{(L)}} \tag{13}$$

$$\frac{\partial C_0}{\partial a^{(L)}} = 2(a^{(L)} - y^{(0)}) \tag{14a}$$

$$\frac{\partial a^{(L)}}{\partial z^{(L)}} = \sigma'(z^{(L)}) \tag{14b}$$

$$\frac{\partial z^{(L)}}{\partial b^{(L)}} = 1 \tag{14c}$$

Therefore,

$$\frac{\partial C_0}{\partial b^{(L)}} = \sigma'(z^{(L)}) 2(a^{(L)} - y^{(0)}) \tag{15}$$

The total cost for a certain mini batch containing m training data sets is then calculated by taking an average over the cost of all the data sets in the mini batch, i.e.,

$$\frac{\partial C}{\partial w^{(L)}} = \frac{1}{m} \sum_{k=0}^{m-1} \frac{\partial C_k}{\partial w^{(L)}} \tag{16a}$$

$$\frac{\partial C}{\partial b^{(L)}} = \frac{1}{m} \sum_{k=0}^{m-1} \frac{\partial C_k}{\partial b^{(L)}} \tag{16b}$$

The above equations are then used for updating the weight, $w^{(L)}$ and bias, $b^{(L)}$ as:

$$w^{(L)}_{new} = w^{(L)} - \eta \frac{\partial C}{\partial w^{(L)}} \tag{17a}$$

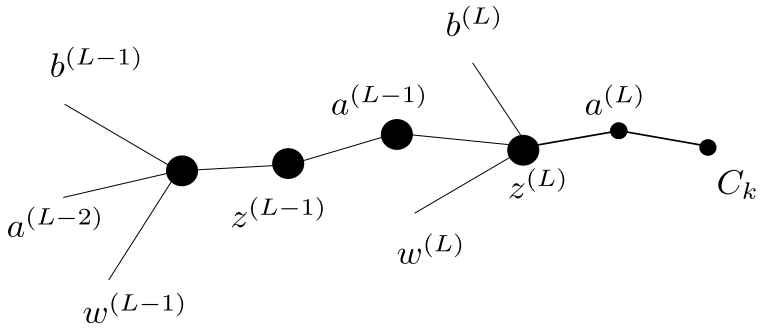
$$b^{(L)}_{new} = b^{(L)} - \eta \frac{\partial C}{\partial b^{(L)}} \tag{17b}$$

This is where the term "back propagation" comes from. As it can be seen, we are updating the weights and biases for the last layer first. Similarly, the weight and biases corresponding to the $L - 1^{\text{th}}$ layer are going to be updated as:

$$w^{(L-1)}_{new} = w^{(L-1)} - \eta \frac{\partial C}{\partial w^{(L-1)}} \tag{18a}$$

$$b^{(L-1)}_{new} = b^{(L-1)} - \eta \frac{\partial C}{\partial b^{(L-1)}} \tag{18b}$$

We then update the weights and biases of $L - 2^{\text{th}}$ layer and so on in the backward direction, until all the weights and biases for the neural network are updated. This marks the end of the 1^{st} "**epoch**". We have now moved a small step towards the global minima of the cost function. Remember, the above calculation was for the 1^{st} mini batch. In the 2^{nd} "epoch", we again the update the weights and biases of the neural network by using the data sets from the 2^{nd} mini batch. With each "epoch", we move a small step toward the global minima of the cost function of the entire neural network. After we have gone through all the "epochs", we would have trained the neural network. Let's now try to evaluate Eq. (18) as an excessive. In Eq. (18a),



$$z^{(L-1)} = w^{(L-1)} \times a^{(L-2)} + b^{(L-1)}$$

$$\frac{\partial C}{\partial w^{(L-1)}} = \frac{1}{m}\sum_{k=0}^{m-1}\frac{\partial C_k}{\partial w^{(L-1)}} \tag{19}$$

Using the concept of chain rule as earlier we have:

$$\frac{\partial C_k}{\partial w^{(L-1)}} = \frac{\partial z^{(L-1)}}{\partial w^{(L-1)}}\frac{\partial a^{(L-1)}}{\partial z^{(L-1)}}\frac{\partial C_k}{\partial a^{(L-1)}} \tag{20a}$$

$$\implies \frac{\partial C_k}{\partial w^{(L-1)}} = a^{(L-2)}\sigma'(z^{(L-1)})\frac{\partial C_k}{\partial a^{(L-1)}}$$

$$\implies \frac{\partial C_k}{\partial w^{(L-1)}} = a^{(L-2)}\sigma'(z^{(L-1)})\left(w^{(L)}\sigma'(z^{(L)})2(a^{(L)} - y^{(0)})\right)$$

$$\frac{\partial C_k}{\partial a^{(L-2)}} = \frac{\partial z^{(L-1)}}{\partial a^{(L-2)}}\frac{\partial a^{(L-1)}}{\partial z^{(L-1)}}\frac{\partial C_k}{\partial a^{(L-1)}}$$

$$\implies \frac{\partial C_k}{\partial a^{(L-2)}} = w^{(L-1)}\sigma'(z^{(L-1)})\left(w^{(L)}\sigma'(z^{(L)})2(a^{(L)} - y^{(0)})\right) \tag{20b}$$

Similarly,

$$\frac{\partial C_k}{\partial w^{(L-2)}} = \frac{\partial z^{(L-2)}}{\partial w^{(L-2)}}\frac{\partial a^{(L-2)}}{\partial z^{(L-2)}}\frac{\partial C_k}{\partial a^{(L-2)}}$$

$$\implies \frac{\partial C_k}{\partial w^{(L-2)}} = a^{(L-3)}\sigma'(z^{(L-2)})\frac{\partial C_k}{\partial a^{(L-2)}}$$

$$\frac{\partial C_k}{\partial a^{(L-3)}} = \frac{\partial z^{(L-2)}}{\partial a^{(L-3)}}\frac{\partial a^{(L-1)}}{\partial z^{(L-2)}}\frac{\partial C_k}{\partial a^{(L-2)}}$$

$$\implies \frac{\partial C_k}{\partial a^{(L-3)}} = w^{(L-2)}\sigma'(z^{(L-2)})\frac{\partial C_k}{\partial a^{(L-2)}}$$

Thus in general,

$$\frac{\partial C_k}{\partial w^{(l)}} = a^{(l-1)}\sigma'\left(z^{(l)}\right)\frac{\partial C_k}{\partial a^{(l)}} \tag{22a}$$

$$\frac{\partial C_k}{\partial a^{(l)}} = w^{(l+1)}\sigma'(z^{(l+1)})\frac{\partial C_k}{\partial a^{(l+1)}} \tag{22b}$$

The above two equations can be generalized for a neural network with multiple neurons and layers as:

$$\frac{\partial C_k}{\partial w_{i,j}^{(l)}} = a_j^{(l-1)} \sigma'(z_i^{(l)}) \frac{\partial C_k}{\partial a_i^{(l)}} \tag{23a}$$

$$\frac{\partial C_k}{\partial a_i^{(l)}} = \sum_{p=0}^{n_{l+1}-1} w_{i,p}^{(l+1)} \sigma'(z_p^{(l+1)}) \frac{\partial C_k}{\partial a_p^{(l+1)}} \tag{23b}$$

The partial derivative, $\frac{\partial C_k}{\partial a_p^{(l+1)}}$ can be eventually traced back to the relevant partial derivative of C_k with respect to the activations in the last two layers of the neural network as:

$$\frac{\partial C_k}{\partial a_i^{(\mathcal{N}-1)}} = \sum_{p=0}^{n_{\mathcal{N}}-1} w_{i,p}^{(\mathcal{N})} \sigma'(z_p^{(\mathcal{N})}) \frac{\partial C_k}{\partial a_p^{(\mathcal{N})}}$$

$$\frac{\partial C_k}{\partial a_p^{(\mathcal{N})}} = 2(a_p^{(\mathcal{N})} - y_p^{(k)})$$

All parameters are known in the above equation for numerical calculation.
Similarly, for partial derivative of C_k with respect to biases can be expressed as:

$$\frac{\partial C}{\partial b_i^{(l)}} = \frac{1}{m} \sum_{k=0}^{m-1} \frac{\partial C_k}{\partial b_i^{(l)}} \tag{25a}$$

$$\frac{\partial C_k}{\partial b_i^{(l)}} = \sigma'(z_i^{(l)}) \frac{\partial C_k}{\partial a_i^{(l)}} \quad \left[\frac{\partial C_k}{\partial a_i^{(l)}} \text{ can be calculated from Eq. 23b} \right] \tag{25b}$$