Understanding the mathematics of artificial neural networks

Preprint · January 2020

1 author:

Hamid Bougourzi
Independent
34 PUBLICATIONS **551** CITATIONS

SEE PROFILE

Understanding the mathematics of artificial neural networks

A. Hamid Bougourzi, PhD *

January 2020

Résumé

The goal of the following notes is to build a non-linear model based on a specific neural network, with the aim of generalizing the results to any neural network.

1 Introduction to Artificial Neural Networks

Artificial neural networks are a major branch in the larger field of machine learning techniques that try to simulate the mechanism of learning in natural biological brains. The understand how biological brains inspired (a) artificial neural network let us review how the human brain works. It consists of nervous cells called neurons. As Figure 1.(a) illustrates each neuron has a cell body, dendrites and an axon.

*e-mail: hamid.bougourzi@outlook.com

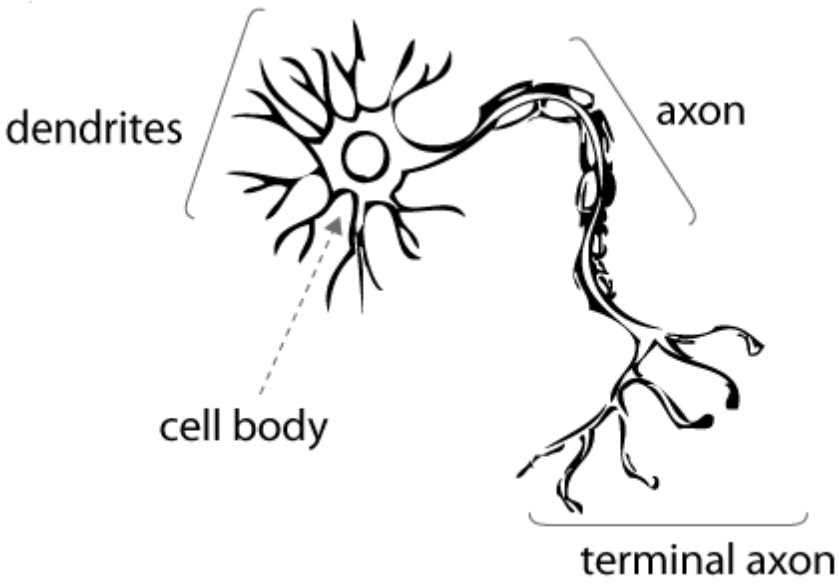

Figure 1.(a) : a biological neuron.

A neuron's axon is connected to other neurons dendrites and these connecting regions between axons and dendrides are called synpases as shown in Figure 1.(b). The strengths of synaptic connections often change in response to external stimuli. This change is how learning takes place in natural brains.

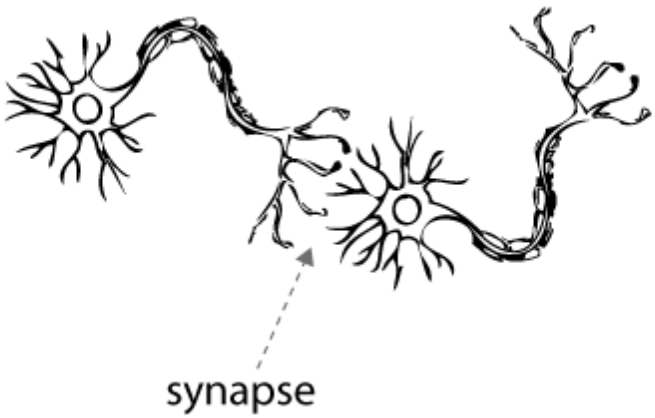

Figure 1.(b) : how neurons are connected to each other.

By analogy to their biological neural networks counterpart, the way artificial neural networks work is that they contain computational units that are referred to as neurons or nodes. These nodes are connected to one another through weights. See the following Figure 2. which illustrates how a typical neural network is architected.

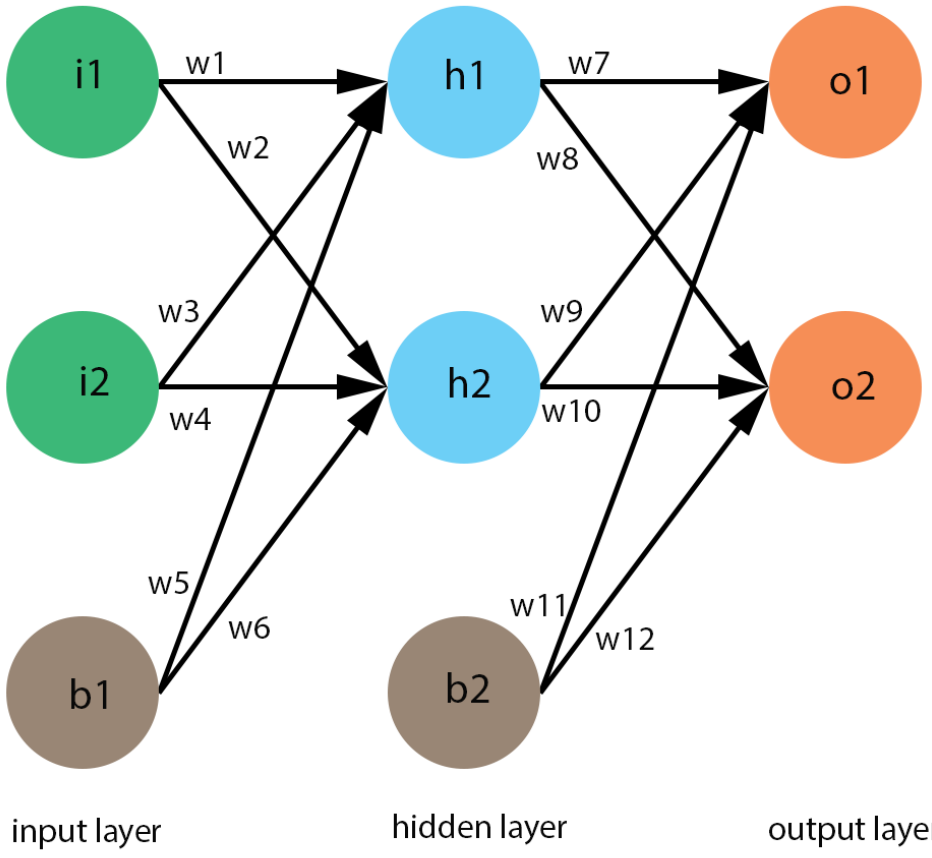

Figure 2. : an example of a three layered neural network.

A neuron computes a function of the inputs by propagating the computed values from the input neurons on the left to the output neurons on the right. It does so by using the weights as intermediate parameters along with a judicious choice of what is called an activation function. In fact using the weights alone will lead to neural

network that simualtes a linear function between input and output. What makes this function nonlinear and hence more general in simulating any complicated and nontrivial function is really the usage of a nonlinear activation function along with the weights. We will see examples of activation functions that are typicalled used in neural networks in the next section.

The main challenge to resolve a neural network is the determination of these weights once we decide on an approriate choise of the activation function. Usually we start from a set of random weights and we adjust them by making the neural network learn from the set of training data containing input-output pairs. The training data provides feedback to the correcteness of the weights depending on how well the predicted output for a particular input matches the corresponding known output. One can think of the neural network as a nonlinear function that maps the input to the ouput. The way the weights will be adjusted is so that it minimizes the error which is the difference between the known output and the predicted ouput. This improvment of the accuracy of the neural network is done iteratively as new training data is fed to it. Once the best set of weights is found the neural network can then be used to predict new ouput corresponding to new unseen input that is not found in the training data. This is in a nutshell the main goal and purpose of any machine learning algorithm such as neural networks, that is, we solve them through known traning data and then we use them to make predictions using input from new unseen data. The real power of neural networks shines when we use many computational units which then allow for the simulation of any complicated and nonlinear function that maps input to output. Moreover, in this case we also need sufficient training data in order to find the larger number of weights.

In Figure 2. The nodes labeled $i1$ and $i2$ are called input nodes and the node labled $b1$ is called a bias node. Both the input nodes and the bias node belong to the input layer. The nodes labeled $h1$ and $h2$ are called hidden nodes and along with the bias node labeled $b2$ belong to the hidden layer. The nodes labeled $o1$ and $o2$ are called output nodes and they belog to the output layer. Every neural network model consists of exactly one input layer and exactly one output layer and any number of hidden layers. An input layer consists of one or more input nodes plus a bias node. An output layer consists of one or more output nodes. Finally, any hidden layer consists of one or more hidden nodes plus a bias node. The set weights of this particular neural network model is $\{w1, ..., w12\}$ whose elements are real numbers. Note the direction of the arrows in Figure 2. which shows that we start training the model by feeding it data into its input layer on the far left and get the predicted result from its output layer on the far right in

process called forward propagation. Once we get the predicted results we compare them to the known output from the training data and we try to adjust the weights that minimize this difference starting from the far right and moving to the left in a process called backpropagation. We will desribe in details these two processes of forward propagation and backpropagation which play a key role in neural networks. (Question to hamid What is the purpose of the bias node ?). A more throurough mathematical resolution of a general neural network model will be described in the next section when we show how to solve for all the weights using the training data.

2 Training a neural network

Let's start with the specific neural network shown in Figure 3. It has four layers consisting of one input layer denoted with $\ell = 1$, two hidden layers denoted with $\ell = 2$ and $\ell = 3$ and one output layer denoted with $\ell = 4$. It has connections between an input layer and a hidden layer, and connections between a hidden layer and another hidden layer, and connections between a hidden layer and an output layer. Therefore it has all of the features of the most general neural network possible. Therefore we will derive from it all necessary formulas that describe how to train this specific neural network and then we show to easily generalize these formulas to most general neural network case.

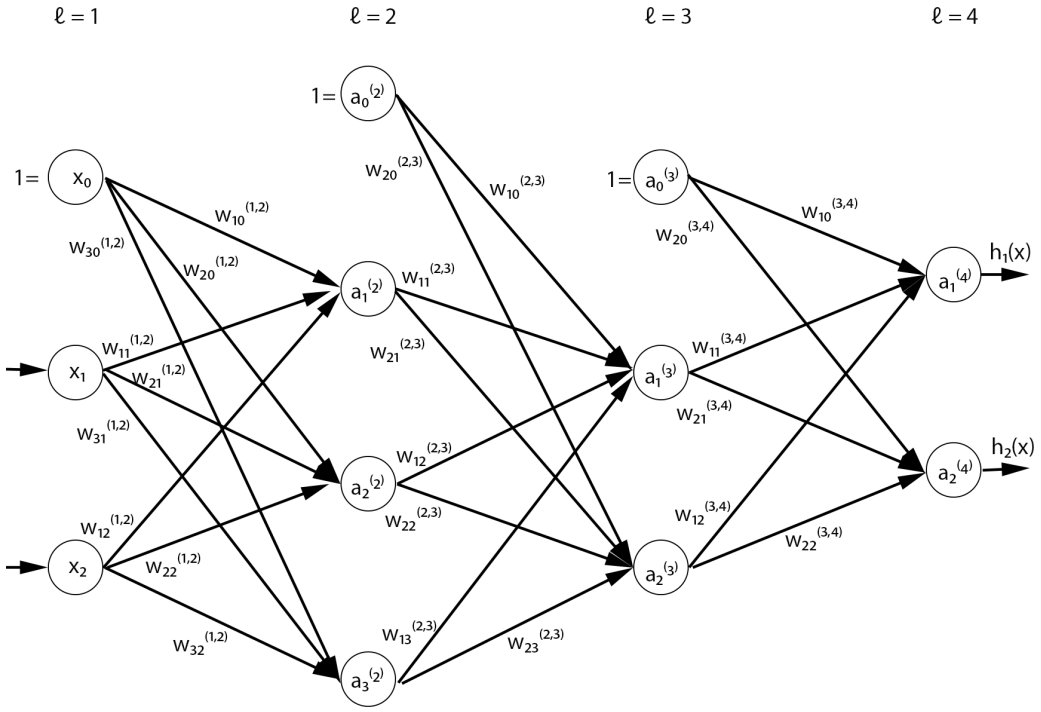

Figure 3. : a neural nework with one input layer, two hidden layers and one output layer

Let's now describe the notations introduced in Figure 3. The symbol $\ell = 1, ..., N$ denotes a layer and N denotes the number of layers. In our case $N = 4$. The node at the top of each layer except the output layer is called a bias node. The output layer does not have a bias node. The symbol n_ℓ denotes the number of nodes contained in layer ℓ not including the bias node. In our case $n_1 = 2$, $n_2 = 3$, $n_3 = 2$, and $n_4 = 2$. The set of weights are denoted by $w_{ij}^{\ell,\ell+1)}$ where $\ell = 1, ..., N-1$, $i = 1, ..., n_{\ell+1}$, and $j = 0, ..., n_\ell$. The indices i and j increase going from the nodes at the top to nodes at the bottom. For example $w_{13}^{(2,3)}$ is the weight of the connection between node 3 of layer 2 and node 1 of layer 3. In the general case $w_{ij}^{\ell,\ell+1)}$ is the weight of the connection between node j of layer ℓ and node i of layer $\ell + 1$. x_1 and x_2 represent the input data that is fed to the neural network from the far left. $h_1(x)$ and $h_2(x)$ represent the output values coming out of the output layer of the neural network. The symbol a_i^ℓ represents the activation at node i of layer ℓ. We have the following mathematical relations that define these

symbols in terms of the weights :

$$\begin{array}{rcl} a_i^{(\ell+1)} & = & f(z_i^{\ell+1}), = \quad \ell = 1, .., N-1, \quad i = 1, .., n_{\ell+1} \\ a_i^{(1)} & = & x_j \quad i = 1, .., n_1 \\ a_0^{\ell} & = & 1 \quad \ell = 1, .., N-1 \\ h_i(x) & = & a_i^N \quad i = 1, ..., n_N \\ z_i^{\ell+1} & = & \sum_{j=0}^{n_\ell} w_{ij}^{(\ell,\ell+1)} a_j^{\ell} \qquad \ell = 1, .., N-1, \quad i = 1, .., n_{\ell+1} \end{array} \tag{1}$$

Here the symbols $z_i^{\ell+1}$ are called ? ? ? and $f(z)$ is called an activation function. The choice of a nonlinear activation function is an important aspect in the functionnig of a neural network. Two examples of an activation function are the sigmoid function and the hyperbolic tangent function. The sigmoid function is defined by :

$$f(z) = \frac{1}{1 + e^{-z}} \tag{2}$$

and the hyperbolic function is defined by

$$f(z) = \frac{e^{2z} - 1}{e^{2z} + 1} \tag{3}$$

The sigmoid activation function outputs a value in the range $(0, 1)$ which thus can be interpreted as a probability. The hyperbolic tangent function outputs a value in the range $(-1, 1)$ and thus is used when the output values can be both positive and negative. Figures 4(a). and 4(b). represent the shapes of the sigmoid function and the hyperbolic tangent function respectively :

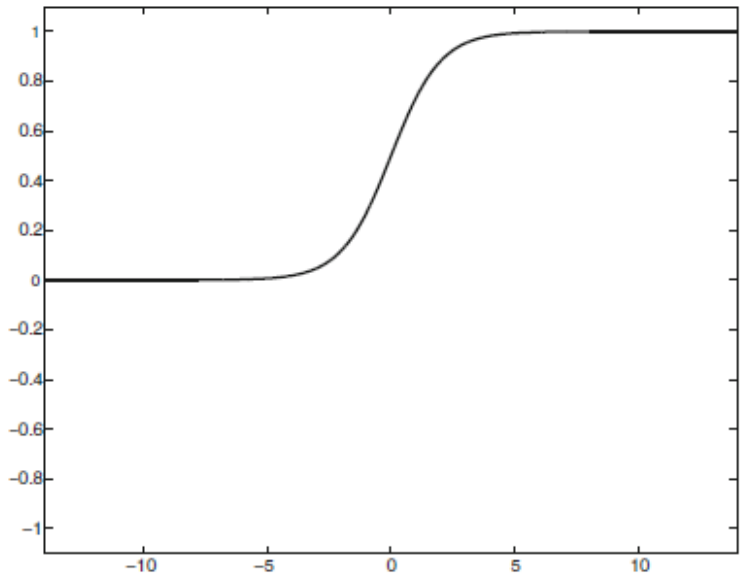

Figure 4.(a) : Sigmoid function

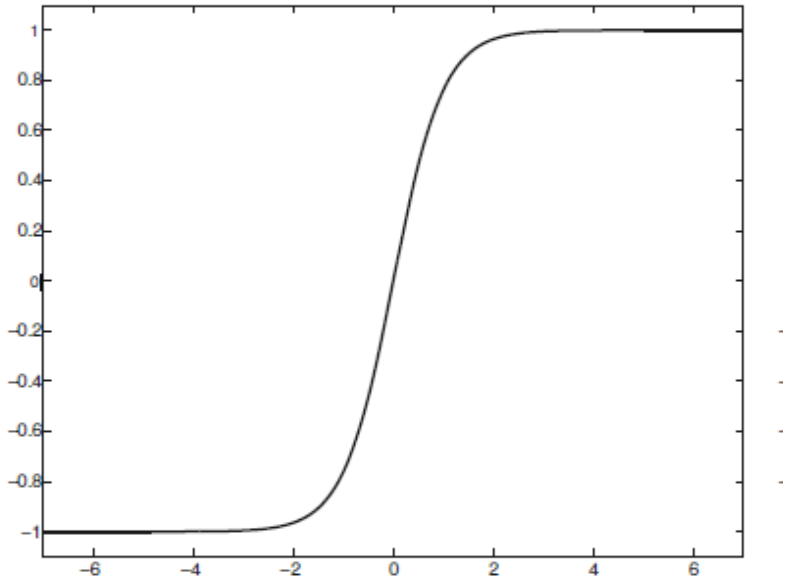

Figure 4.(b) : Hyperbolic tangent function

It is the nonlinearity characteristic of these two functions that makes the neural network nontrivial and able to simulate any nonlinear function describing the shape of the data. Of course neural networks can also model linear functions if we choose the activation function to be the identity, that is $f(z) = z$. One can find more examples of activation functions in modern litterature. For our purposes henceforth we choose the activation function to be the sigmoid function and we set out to solving the specific neural network of Figure 3.

Let $E(a_1^{(4)}, a_2^{(4)}, y_1, y_2)$ be the error function that depends on the observed values y_i and the activations $a_i^{(4)}$. We will give an explicit formula for E later on but for the moment we keep it general. Solving this model means that we would like to determine the set of all weights $w_{ij}^{(\ell,\ell+1)}$ such that the model fits the data as best as possible, or in other words it minimizes the error E. For this purpose and as an example let's determine the effect of varying the specific weight $w_{10}^{(3,4)}$ on E while fixing all other weights. From the Taylor expansion of E in terms of $w_{10}^{(3,4)}$ and

keeping only the first term in this expansion we have :

$$\Delta E = \frac{\partial E}{\partial w_{10}^{(3,4)}} \Delta w_{10}^{(3,4)} \tag{4}$$

From the latter formula we can deduce that one possible change $\Delta w_{10}^{(3,4)}$ that garantees that the error decreases after this change (i.e. it results in $\Delta E \leq 0$) is given by :

$$\Delta w_{10}^{(3,4)} = -\eta \frac{\partial E}{\partial w_{10}^{(3,4)}} \tag{5}$$

where $\eta > 0$ is some small positive number that is called the learning rate. Using the latter formula we find the change in E is indeed negative and given by :

$$\Delta E = -\eta (\frac{\partial E}{\partial w_{10}^{(3,4)}})^2 \tag{6}$$

We can express $\Delta w_{10}^{(3,4)}$ as

$$\Delta w_{10}^{(3,4)} = -\eta \frac{\partial E}{\partial w_{10}^{(3,4)}} = -\eta \sum_{i=1}^{2} \frac{\partial E}{\partial z_i^{(4)}} \frac{\partial z_i^{(4)}}{\partial w_{10}^{(3,4)}} \tag{7}$$

Let's recall the explicit expressions for $z_i^{(4)}$ to be

$$\begin{array}{rcl} z_1^{(4)} & = & w_{10}^{(3,4)} a_0^{(3)} + w_{11}^{(3,4)} a_1^{(3)} + w_{12}^{(3,4)} a_2^{(3)} \\ z_2^{(4)} & = & w_{20}^{(3,4)} a_0^{(3)} + w_{21}^{(3,4)} a_1^{(3)} + w_{22}^{(3,4)} a_2^{(3)} \end{array} \tag{8}$$

This means

$$\begin{array}{rcl} \frac{\partial z_1^{(4)}}{\partial w_{10}^{(3,4)}} & = & a_0^{(3)} \\ \frac{\partial z_2^{(4)}}{\partial w_{10}^{(3,4)}} & = & 0 \end{array} \tag{9}$$

Substituting the latter results back in (7) we find :

$$\Delta w_{10}^{(3,4)} = -\eta \frac{\partial E}{\partial w_{10}^{(3,4)}} = -\eta \frac{\partial E}{\partial z_1^{(4)}} a_0^{(3)} \tag{10}$$

If we denote $-\frac{\partial E}{\partial z_1^{(4)}}$ by

$$-\frac{\partial E}{\partial z_1^{(4)}} = \delta_1^{(4)} \tag{11}$$

then we get

$$\Delta w_{10}^{(3,4)} = -\eta \frac{\partial E}{\partial w_{10}^{(3,4)}} = \eta \delta_1^{(4)} a_0^{(3)} \tag{12}$$

Let's repeat a similar exercise to calculate the change in a different weight, for example, $\Delta w_{21}^{(3,4)}$. We have

$$\Delta w_{21}^{(3,4)} = -\eta \frac{\partial E}{\partial w_{21}^{(3,4)}} = -\eta \sum_{i=1}^{2} \frac{\partial E}{\partial z_i^{(4)}} \frac{\partial z_i^{(4)}}{\partial w_{21}^{(3,4)}} \tag{13}$$

From (8) we find :

$$\begin{array}{rcl} \frac{\partial z_1^{(4)}}{\partial w_{21}^{(3,4)}} & = & 0 \\ \frac{\partial z_2^{(4)}}{\partial w_{21}^{(3,4)}} & = & a_1^{(3)} \end{array} \tag{14}$$

Therefore

$$\Delta w_{21}^{(3,4)} = -\eta \frac{\partial E}{\partial w_{21}^{(3,4)}} = -\eta \frac{\partial E}{\partial z_2^{(4)}} a_1^{(3)} = \eta \delta_2^{(4)} a_1^{(3)} \tag{15}$$

where again we have introduced the notation

$$-\frac{\partial E}{\partial z_2^{(4)}} = \delta_2^{(4)} \tag{16}$$

By similar reasoning we arrive to the following general formula for the changes in the weights $w_{ij}^{(3,4)}$ that map the hidden layer 3 to output layer 4 :

$$\Delta w_{ij}^{(3,4)} = \eta \delta_i^{(4)} a_j^{(3)}, \quad i = 1, 2 \qquad j = 0, 1, 2 \tag{17}$$

Let's now consider the change in a weight mapping a hidden layer to a hidden layer such as $w_{21}^{(2,3)}$. Again in order for this change to lower the error we have :

$$\Delta w_{21}^{(2,3)} = -\eta \frac{\partial E}{\partial w_{21}^{(2,3)}} \tag{18}$$

It can be written as :

$$\Delta w_{21}^{(2,3)} = -\eta \frac{\partial E}{\partial w_{21}^{(2,3)}} = -\eta \sum_{i=1}^{2} \frac{\partial E}{\partial z_i^{(3)}} \frac{\partial z_i^{(3)}}{\partial w_{21}^{(2,3)}} \tag{19}$$

Let's now recall the explicit expressions for $z_i^{(3)}$:

$$\begin{array}{rcl} z_1^{(3)} & = & w_{10}^{(2,3)} a_0^{(2)} + w_{11}^{(2,3)} a_1^{(2)} + w_{12}^{(2,3)} a_2^{(2)} + w_{13}^{(2,3)} a_3^{(2)} \\ z_2^{(3)} & = & w_{20}^{(2,3)} a_0^{(2)} + w_{21}^{(2,3)} a_1^{(2)} + w_{22}^{(2,3)} a_2^{(2)} + w_{23}^{(2,3)} a_3^{(2)} \end{array} \tag{20}$$

which lead to :

$$\begin{array}{rcl} \frac{\partial z_1^{(3)}}{\partial w_{21}^{(2,3)}} & = & 0 \\ \frac{\partial z_2^{(3)}}{\partial w_{21}^{(2,3)}} & = & a_1^{(2)} \end{array} \tag{21}$$

Substituting the results of (21) in (19) we get :

$$\Delta w_{21}^{(2,3)} = -\eta \frac{\partial E}{\partial w_{21}^{(2,3)}} = -\eta \frac{\partial E}{\partial z_2^{(3)}} a_1^{(2)} = \eta \delta_2^{(3)} a_1^{(2)} \tag{22}$$

where once more we have defined

$$\delta_2^{(3)} = -\frac{\partial E}{\partial z_2^{(3)}} \tag{23}$$

For $\delta_2^{(3)}$ we can go even further and express it as :

$$\delta_2^{(3)} = -\frac{\partial E}{\partial z_2^{(3)}} = -\frac{\partial E}{\partial z_1^{(4)}} \frac{\partial z_1^{(4)}}{\partial z_2^{(3)}} - \frac{\partial E}{\partial z_2^{(4)}} \frac{\partial z_2^{(4)}}{\partial z_2^{(3)}} = \delta_1^{(4)} \frac{\partial z_1^{(4)}}{\partial z_2^{(3)}} + \delta_2^{(4)} \frac{\partial z_2^{(4)}}{\partial z_2^{(3)}} \tag{24}$$

In order to calculate the latter expression let's first calculate $\frac{\partial z_1^{(4)}}{\partial z_2^{(3)}}$ as :

$$\frac{\partial z_1^{(4)}}{\partial z_2^{(3)}} = \frac{\partial z_1^{(4)}}{\partial a_1^{(3)}} \frac{\partial a_1^{(3)}}{\partial z_2^{(3)}} + \frac{\partial z_1^{(4)}}{\partial a_2^{(3)}} \frac{\partial a_2^{(3)}}{\partial z_2^{(3)}} \tag{25}$$

Here we didn't include a term corresponding to $a_0^{(3)}$ since it's a constant and not a variable. Since $a_1^{(3)}$ depends only on $z_1^{(3)}$ we therefore have

$$\frac{\partial a_1^{(3)}}{\partial z_2^{(3)}} = 0 \tag{26}$$

and from relations (8) we find :

$$\frac{\partial z_1^{(4)}}{\partial a_2^{(3)}} = w_{12}^{(3,4)} \tag{27}$$

Since by defintiion $a_2^{(3)} = f(z_2^{(3)})$ where f is the activation function, therefore we arrive at :

$$\frac{\partial a_2^{(3)}}{\partial z_2^{(3)}} = f'(z_2^{(3)}) \tag{28}$$

Where $f'(z_2^{(3)})$ denotes the derivative of $f(z)$ with respect to $z = z_2^{(3)}$. Puting everything together we get :

$$\frac{\partial z_1^{(4)}}{\partial z_2^{(3)}} = w_{12}^{(3,4)} f'(z_2^{(3)}) \tag{29}$$

Let's now calculate $\frac{\partial z_2^{(4)}}{\partial z_2^{(3)}}$. We have

$$\frac{\partial z_2^{(4)}}{\partial z_2^{(3)}} = \frac{\partial z_2^{(4)}}{\partial a_1^{(3)}} \frac{\partial a_1^{(3)}}{\partial z_2^{(3)}} + \frac{\partial z_2^{(4)}}{\partial a_2^{(3)}} \frac{\partial a_2^{(3)}}{\partial z_2^{(3)}} \tag{30}$$

For similar reasons as before we get

$$\frac{\partial z_2^{(4)}}{\partial z_2^{(3)}} = \frac{\partial z_2^{(4)}}{\partial a_2^{(3)}} \frac{\partial a_2^{(3)}}{\partial z_2^{(3)}} = w_{22}^{(3,4)} f'(z_2^{(3)}) \tag{31}$$

Substituting the results (29) and (31) in (24) we finally arrive at :

$$\delta_2^{(3)} = (\delta_1^{(4)} w_{12}^{(3,4)} + \delta_2^{(4)} w_{22}^{(3,4)}) f'(z_2^{(3)}) \tag{32}$$

Let's now consider the change in a weight mapping an input layer to a hidden layer such as $w_{31}^{(1,2)}$. Again we have the general formula :

$$\Delta w_{31}^{(1,2)} = -\eta \frac{\partial E}{\partial w_{31}^{(1,2)}} \tag{33}$$

We can rewrite it as

$$\Delta w_{31}^{(1,2)} = -\eta \frac{\partial E}{\partial w_{31}^{(1,2)}} = -\eta \sum_{i=1}^{3} \frac{\partial E}{\partial z_i^{(2)}} \frac{\partial z_i^{(2)}}{\partial w_{31}^{(1,2)}} \tag{34}$$

Let's recall the explicit expressions of $z_i^{(2)}$ which are given by

$$\begin{array}{rcl} z_1^{(2)} & = & w_{10}^{(1,2)} a_0^{(1)} + w_{11}^{(1,2)} a_1^{(1)} + w_{12}^{(1,2)} a_2^{(1)} \\ z_2^{(2)} & = & w_{20}^{(1,2)} a_0^{(1)} + w_{21}^{(1,2)} a_1^{(1)} + w_{22}^{(1,2)} a_2^{(1)} \\ z_3^{(2)} & = & w_{30}^{(1,2)} a_0^{(1)} + w_{31}^{(1,2)} a_1^{(1)} + w_{32}^{(1,2)} a_2^{(1)} \end{array} \tag{35}$$

where we have set

$$\begin{array}{rcll} a_0^{(1)} & = & x_0 = 1 & \\ a_i^{(1)} & = & x_i & i = 1, 2 \end{array} \tag{36}$$

Using relations (35) we find :

$$\begin{array}{rcl} \frac{\partial z_1^{(2)}}{\partial w_{31}^{(1,2)}} & = & 0 \\ \frac{\partial z_2^{(2)}}{\partial w_{31}^{(1,2)}} & = & 0 \\ \frac{\partial z_3^{(2)}}{\partial w_{31}^{(1,2)}} & = & a_1^{(1)} \end{array} \tag{37}$$

Then

$$\Delta w_{31}^{(1,2)} = -\eta \frac{\partial E}{\partial w_{31}^{(1,2)}} = -\eta \frac{\partial E}{\partial z_3^{(2)}} a_1^{(1)} = \eta \delta_3^{(2)} a_1^{(1)} \tag{38}$$

Where again we have defined

$$\delta_3^{(2)} = -\frac{\partial E}{\partial z_3^{(2)}} \tag{39}$$

Let's use the chain rule for the last expression as :

$$\begin{aligned} \delta_3^{(2)} &= -\frac{\partial E}{\partial z_3^{(2)}} = -\frac{\partial E}{\partial z_1^{(3)}} \frac{\partial z_1^{(3)}}{\partial z_3^{(2)}} - \frac{\partial E}{\partial z_2^{(3)}} \frac{\partial z_2^{(3)}}{\partial z_3^{(2)}} = \delta_1^{(3)} \frac{\partial z_1^{(3)}}{\partial z_3^{(2)}} + \delta_2^{(3)} \frac{\partial z_2^{(3)}}{\partial z_3^{(2)}} \\ &= \delta_1^{(3)} \frac{\partial z_1^{(3)}}{\partial a_3^{(2)}} \frac{\partial a_3^{(2)}}{\partial z_3^{(2)}} + \delta_2^{(3)} \frac{\partial z_2^{(3)}}{\partial a_3^{(2)}} \frac{\partial a_3^{(2)}}{\partial z_3^{(2)}} \\ &= \delta_1^{(3)} w_{13}^{(2,3)} f'(z_3^{(2)}) + \delta_2^{(3)} w_{23}^{(2,3)} f'(z_3^{(2)}) = (\delta_1^{(3)} w_{13}^{(2,3)} + \delta_2^{(3)} w_{23}^{(2,3)}) f'(z_3^{(2)}) \end{aligned} \tag{40}$$

From all the above formulas we can infer the following generalizations for any neural network. Note that these generalizations can be explicitly proven using proof by induction.

$$\begin{aligned} \Delta w_{ij}^{(\ell,\ell+1)} &= \eta \delta_i^{(\ell+1)} a_j^{(\ell)}, \quad \ell = 1, .., N-1 \quad , i = 1, ..., n_{\ell+1}, \quad j = 0, ..., n_\ell \\ a_i^{(\ell+1)} &= f(z_i^{\ell+1}), = \quad \ell = 1, .., N-1, \quad i = 1, .., n_{\ell+1} \\ a_i^{(1)} &= x_j \quad i = 1, .., n_1 \\ a_0^{\ell} &= 1 \quad \ell = 1, .., N-1 \\ z_i^{\ell+1} &= \textstyle\sum_{j=0}^{n_\ell} w_{ij}^{(\ell,\ell+1)} a_j^{\ell} \qquad \ell = 1, .., N-1, \quad i = 1, .., n_{\ell+1} \\ \delta_i^{(\ell)} &= (\textstyle\sum_{j=1}^{n_{\ell+1}} \delta_j^{(\ell+1)} w_{ji}^{(\ell,\ell+1)}) f'(z_i^{\ell}), \quad \ell = 2, .., N-1, \quad i = 1, .., n_\ell \\ \delta_i^{(N)} &= -\frac{\partial E}{\partial z_i^{(N)}} \quad i = 1, .., n_N \end{aligned} \tag{41}$$

where N is the number of layers consisting of the input layer and the output layer and all the hidden layers. n_ℓ is the number of nodes at the layer ℓ not including the bias node. We can vectorize the latter formulas as :

$$\begin{aligned} z^{(\ell+1)} &= W^{(\ell,\ell+1)} a^{\ell} \quad z^{\ell+1} \in R^{n_{\ell+1}}, \quad W^{(\ell,\ell+1)} \in R^{n_{\ell+1} \times (n_\ell+1)}, \quad a^{\ell} \in R^{n_\ell+1}, \quad \ell = 1, .., N-1 \\ a^{(\ell)} &= f(z^{\ell}) \quad z^{\ell}, a^{\ell} \in R^{n_\ell} \quad \ell = 2, .., N \\ a^{(1)} &= x, \quad a^1, x \in R^{n_1+1}, \qquad a_0^1 = x_0 = 1 \\ \delta^{\ell} &= (W^{\ell,\ell+1})^T \delta^{\ell+1} .* f'(z^{\ell}) \quad \delta^{\ell} \in R^{n_\ell}, \quad \ell = 2, .., N-1 \\ \delta^N &= -\nabla_{z^N} E \quad \delta^N \in R^{n_N} \\ \Delta W^{\ell,\ell+1} &= \eta \delta^{(\ell+1)} (a^{\ell})^T, \quad \delta^{(\ell+1)} \in R^{n_{\ell+1} \times 1} \quad (a^{\ell})^T \in R^{1 \times (n_\ell+1)}, \quad W^{\ell,\ell+1} \in R^{n_{\ell+1} \times (n_\ell+1)} \end{aligned} \tag{42}$$

Here $W^{\ell,\ell+1}$ is a matrix, and the symbol $.*$ means the element wise multiplication operator between two vectors, and T means the transpose operator, and finally ∇_z is the gradient operator with respect to the vector z. Let's now present an example

of an error function E and an example of an activation function f. For E we use the least mean square error as defined by the following formula :

$$E = \frac{1}{2}\sum_{i=1}^{n_N}(y_i - a_i^{(N)})^2 \tag{43}$$

where y_i are the observed outputs. From relation (41) we get

$$\delta_i^{(N)} = -\frac{\partial E}{\partial z_i^{(N)}} = -\frac{\partial E}{\partial a_i^{(N)}}\frac{\partial a_i^{(N)}}{\partial z_i^{(N)}} = (y_i - a_i^{(N)})f'(z_i^{(N)}) \tag{44}$$

For the activation function we can use the sigmoid function defined by :

$$f(z) = \frac{1}{1+e^{-z}} \tag{45}$$

The activation function part in (44) can be explicitly calculated as

$$\begin{array}{rcl} f'(z_i^{(N)}) & = & \frac{\partial}{\partial z_i^{(N)}}\frac{1}{1+e^{-z_i^{(N)}}} \\ & = & \frac{1}{(1+e^{-z_i^{(N)}})^2}(e^{-z_i^{(N)}}) \\ & = & \frac{1}{(1+e^{-z_i^{(N)}})}\frac{e^{-z_i^{(N)}}}{(1+e^{-z_i^{(N)}})} \\ & = & f(z_i^{(N)})(1-f(z_i^{(N)})) \\ & = & a_i^{(N)}(1-a_i^{(N)}) \end{array} \tag{46}$$

substituing this expression back in (44) we get

$$\delta_i^{(N)} = (y_i - a_i^{(N)})a_i^{(N)}(1-a_i^{(N)}) \tag{47}$$

Similar result applied to relation (41) gives

$$\delta_i^{(\ell)} = (\sum_{j=1}^{n_{\ell+1}}\delta_j^{(\ell+1)}w_{ji}^{(\ell,\ell+1)})f'(z_i^{\ell}) = a_i^{(\ell)}(1-a_i^{(\ell)})(\sum_{j=1}^{n_{\ell+1}}\delta_j^{(\ell+1)}w_{ji}^{(\ell,\ell+1)}), \quad \ell = 2,..,N-1, \quad i = 1,..,n_\ell \tag{48}$$

All of the above formula show how to find $\Delta w_{i,j}^{(\ell,\ell+1)}$ from using just one particular sample data (x, y). However, updating $w_{i,j}^{(\ell,\ell+1)}$ one sample at a time is a very slow process. We would like to calculate $\Delta w_{i,j}^{(\ell,\ell+1)}$ by using a set or batch of of training data samples $(x(1), y^{(1)}), ..., (x^{(s)}, y^{(s)})$. The error function E corresponding to this set becomes

$$\begin{array}{rcl} E & = & \sum_{m=1}^{s} E^{(m)} \\ E^{(m)} & = & \frac{1}{2}\sum_{i=1}^{n_N}(y_i^{(m)} - a_i^{(m)})^2 \end{array} \tag{49}$$

Relation (41) generalizes for the training set as

$$\Delta w_{ij}^{(\ell,\ell+1)} = -\eta \frac{\partial E}{\partial w_{ij}^{(\ell,\ell+1)}} = -\eta \sum_{m=1}^{s} \frac{\partial E^{(m)}}{\partial w_{ij}^{(\ell,\ell+1)}} = \eta \sum_{m=1}^{s} \delta_i^{(\ell+1,m)} a_j^{(\ell,m)} \tag{50}$$

We can therefore implement an algorithm to find all the weights $w_{ij}^{(\ell,\ell+1)}$ using the training set $\{(x^{(1)}, y^{(1)}), ..., (x^{(s)}, y^{(s)})\}$ as :
start from random values of $w_{ij}^{(\ell,\ell+1)}$
initialize the accumulators $\Delta_{ij}^{(\ell,\ell+1)} = 0$
then implement the following loop :
for $(m = 1$ to $s)$
set $a^{(1,m)} = x^{(m)}$
use forward propagation and relation (42) to calculate $a^{(\ell,m)}$ for $\ell = 2, ..., N$
use $y^{(m)}$ and relation (47) to compute $\delta^{(N,m)}$
use $\delta^{(N,m)}$ and relation (48) to compute $\delta^{(N-1,m)}$, $\delta^{(N-2,m)}$,...., $\delta^{(2,m)}$

$$\Delta_{ij}^{(\ell,\ell+1)} = \Delta_{ij}^{(\ell,\ell+1)} + \delta_i^{(2,m)} a_j^{(\ell,m)} \tag{51}$$

end for loop

$$w_{ij}^{(\ell,\ell+1)} = w_{ij}^{(\ell,\ell+1)} + \eta \Delta_{ij}^{(\ell,\ell+1)} \tag{52}$$

And now starting from the latter weights that we have just found from the first batch we can now repeat the same algorithm using the next batch of data samples and so on until we use all data samples. In a nutshell this is then how we determine all the weights $w_{ij}^{(\ell,\ell+1)}$ that define the neural network model which best fits all the data and minimizes the error E.And of course once we have the set of weights we can use the model to predict the output for new input data not used for the training of the model.