Neural-Network Backpropagation Notes

anomitroid

June 9, 2025

1 Gradient Descent

1.1 Finite Difference Definition

The derivative of a cost function $C(w)$ with respect to parameter w is defined via the limit of the finite-difference quotient. Explicitly, we write:

$$C'(w) = \lim_{\epsilon \to 0} \frac{C(w+\epsilon) - C(w)}{\epsilon} \tag{1}$$

Here:

- $C(w+\epsilon)$ is the cost evaluated at $w+\epsilon$.
- $C(w)$ is the cost at w.
- Dividing by ϵ gives the average rate of change over the interval $[w,\, w+\epsilon]$.
- Taking $\epsilon \to 0$ yields the instantaneous rate of change, i.e. the derivative.

1.2 Linear Regression Cost and Its Derivative

Consider a simple linear model with weight w, input x_i, and true output y_i. The mean-squared error (MSE) cost over n examples is:

$$C(w) = \frac{1}{n} \sum_{i=1}^{n} \left(x_i\, w \;-\; y_i\right)^2 \tag{2}$$

We want to compute $\frac{dC}{dw}$. To do so, we proceed step by step.

1.2.1 Step 1: Distribute the Derivative over the Sum

First note that

$$C(w) = \frac{1}{n} \sum_{i=1}^{n} f_i(w), \quad \text{where} \quad f_i(w) = \left(x_i w - y_i\right)^2$$

By linearity of differentiation,

$$\frac{dC}{dw} = \frac{d}{dw}\left(\frac{1}{n}\sum_{i=1}^{n} f_i(w)\right) = \frac{1}{n}\sum_{i=1}^{n} \frac{d}{dw}\left[f_i(w)\right] \tag{3}$$

Here, we have used:

- $\frac{d}{dw}\left(\sum_{i=1}^{n} f_i(w)\right) = \sum_{i=1}^{n} \frac{d}{dw}[f_i(w)]$
- The constant factor $1/n$ can be pulled outside the derivative.

1.2.2 Step 2: Differentiate Each Term

We now focus on a single term:

$$f_i(w) = \left(x_i w - y_i\right)^2$$

By the chain rule, if $g(w) = \left(x_i w - y_i\right)^2$, then

$$\frac{d}{dw} g(w) = 2\left(x_i w - y_i\right) \cdot \frac{d}{dw}\left(x_i w - y_i\right)$$

Next, since $\frac{d}{dw}(x_i w - y_i) = x_i$, we have

$$\frac{d}{dw}\left(x_i w - y_i\right)^2 = 2\left(x_i w - y_i\right)\left(x_i\right) \tag{4}$$

Substituting back into(3), we get:

$$\frac{dC}{dw} = \frac{1}{n}\sum_{i=1}^{n}\Big[2\left(x_i w - y_i\right) x_i\Big] \tag{5}$$

Finally, we can factor out the 2 if desired:

$$\frac{dC}{dw} = \frac{2}{n}\sum_{i=1}^{n}\left(x_i w - y_i\right) x_i \tag{6}$$

This expression gives the gradient of the MSE cost with respect to w.

2 One-Neuron Model

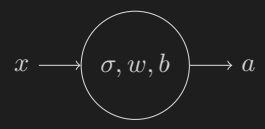
In this section, we analyze a single neuron (sigmoid activation) with first one input, then two inputs. We present:

1. Forward-pass equations
2. Detailed derivation of $\sigma(z)$ and $\sigma'(z)$
3. Cost function and step-by-step computation of gradients

2.1 Single Input Neuron

2.1.1 Network Diagram and Forward Pass

A single input x is multiplied by weight w, added to bias b, and passed through a sigmoid activation to produce output a. The diagram is:



Therefore, the forward-pass equation is:

$$z = x\,w + b, \quad a \;=\; \sigma(z) \tag{7}$$

We will omit the intermediate z in later notation and write $a = \sigma(xw + b)$.

2.1.2 Sigmoid Activation and Its Derivative

The sigmoid activation function is defined by:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{8}$$

To find $\sigma'(z)$, we apply the chain rule. Let

$$h(z) = \frac{1}{1 + e^{-z}} = (1 + e^{-z})^{-1}$$

Then

$$\frac{dh}{dz} = -1 \cdot (1 + e^{-z})^{-2} \;\cdot\; \frac{d}{dz}[\,1 + e^{-z}\,]$$

Since $\frac{d}{dz}[\,1 + e^{-z}\,] = -\,e^{-z}$, we obtain:

$$\sigma'(z) = -(1 + e^{-z})^{-2} \;\cdot\; \big(-\,e^{-z}\big) = \frac{e^{-z}}{(1 + e^{-z})^2} \tag{9}$$

Next, we rewrite $\frac{e^{-z}}{(1+e^{-z})^2}$ in terms of $\sigma(z)$. Observe:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \quad \Longrightarrow \quad 1 - \sigma(z) = 1 - \frac{1}{1 + e^{-z}} = \frac{(1 + e^{-z}) - 1}{1 + e^{-z}} = \frac{e^{-z}}{1 + e^{-z}}$$

Hence,

$$\sigma(z)\,\big(1 - \sigma(z)\big) = \frac{1}{1 + e^{-z}} \;\cdot\; \frac{e^{-z}}{1 + e^{-z}} = \frac{e^{-z}}{(1 + e^{-z})^2}$$

Comparing with(9), we conclude:

$$\sigma'(z) = \sigma(z)\,\big[1 - \sigma(z)\big] \tag{10}$$

2.1.3 Cost Function and Gradients

Given a dataset $\{(x_i,\, y_i)\}_{i=1}^{n}$, the neuron's output for the ith example is:

$$a_i = \sigma(x_i w + b)$$

We define the MSE cost as:

$$C(w,b) = \frac{1}{n}\sum_{i=1}^{n}\big(a_i - y_i\big)^2 \tag{11}$$

To apply gradient descent, we need $\partial C/\partial w$ and $\partial C/\partial b$. We compute these in detail.

Gradient with respect to w

First, note that

$$C(w,b) = \frac{1}{n}\sum_{i=1}^{n}\big(a_i - y_i\big)^2, \quad a_i = \sigma(x_i w + b)$$

We differentiate C with respect to w using the chain rule:

$$\frac{\partial C}{\partial w} = \frac{\partial}{\partial w}\left(\frac{1}{n}\sum_{i=1}^{n}(a_i - y_i)^2\right) = \frac{1}{n}\sum_{i=1}^{n}\frac{\partial}{\partial w}\big(a_i - y_i\big)^2 \tag{12}$$

Focus on a single summand:

$$\frac{\partial}{\partial w}\big(a_i - y_i\big)^2 = 2\,\big(a_i - y_i\big)\;\cdot\;\frac{\partial}{\partial w}\big(a_i - y_i\big)$$

Since y_i is a constant (true label), $\frac{\partial}{\partial w}(a_i - y_i) = \frac{\partial a_i}{\partial w}$. Therefore,

$$\frac{\partial}{\partial w}\big(a_i - y_i\big)^2 = 2\,\big(a_i - y_i\big)\;\cdot\;\frac{\partial a_i}{\partial w} \tag{13}$$

Next, compute $\frac{\partial a_i}{\partial w}$. Recall $a_i = \sigma(z_i)$ with $z_i = x_i w + b$. By the chain rule:

$$\frac{\partial a_i}{\partial w} = \sigma'(z_i)\;\cdot\;\frac{\partial}{\partial w}(z_i) = \sigma'(x_i w + b)\;\cdot\;x_i \tag{14}$$

Using(10), $\sigma'(x_i w + b) = a_i\,\big(1 - a_i\big)$. Hence,

$$\frac{\partial a_i}{\partial w} = a_i\,\big(1 - a_i\big)\,x_i \tag{15}$$

Substitute(15) into(13):

$$\frac{\partial}{\partial w}\big(a_i - y_i\big)^2 = 2\,(a_i - y_i)\;\big[a_i\,\big(1 - a_i\big)\,x_i\big] \tag{16}$$

Finally, plug back into(12):

$$\frac{\partial C}{\partial w} = \frac{1}{n}\sum_{i=1}^{n}\Big[\,2\,(a_i - y_i)\,a_i\,(1 - a_i)\,x_i\Big] = \frac{2}{n}\sum_{i=1}^{n}(a_i - y_i)\,a_i\,(1 - a_i)\,x_i \tag{17}$$

Gradient with respect to b

Similarly, to find $\partial C/\partial b$, start with(11):

$$\frac{\partial C}{\partial b} = \frac{\partial}{\partial b}\left(\frac{1}{n}\sum_{i=1}^{n}(a_i - y_i)^2\right) = \frac{1}{n}\sum_{i=1}^{n}\frac{\partial}{\partial b}(a_i - y_i)^2$$

By the same reasoning as before,

$$\frac{\partial}{\partial b}(a_i - y_i)^2 = 2\,(a_i - y_i)\;\cdot\;\frac{\partial}{\partial b}(a_i - y_i), \tag{18}$$

and since $\frac{\partial}{\partial b}(a_i - y_i) = \frac{\partial a_i}{\partial b}$, we compute:

$$\frac{\partial a_i}{\partial b} = \sigma'(x_i w + b)\;\cdot\;\frac{\partial}{\partial b}(x_i w + b) = a_i\,(1 - a_i)\;\cdot\;1 = a_i\,(1 - a_i) \tag{19}$$

Thus,

$$\frac{\partial}{\partial b}(a_i - y_i)^2 = 2\,(a_i - y_i)\;\big[a_i\,(1 - a_i)\big],$$

and finally,

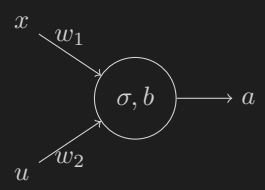
$$\frac{\partial C}{\partial b} = \frac{1}{n}\sum_{i=1}^{n}\Big[\,2\,(a_i - y_i)\,a_i\,(1 - a_i)\Big] = \frac{2}{n}\sum_{i=1}^{n}(a_i - y_i)\,a_i\,(1 - a_i) \tag{20}$$

3 One-Neuron Model with Two Inputs

Now we extend the single-input neuron to accept two inputs x and u, with weights w_1 and w_2, and bias b.

3.1 Network Diagram and Forward Pass

The architecture is:



The forward-pass is:

$$z = x\,w_1 \;+\; u\,w_2 \;+\; b, \quad a = \sigma(z) \tag{21}$$

Again, we abbreviate $a = \sigma(xw_1 + uw_2 + b)$

Recall from(8) and(10):

$$\sigma(z) = \frac{1}{1 + e^{-z}}, \quad \sigma'(z) = \sigma(z)\,(1 - \sigma(z))$$

3.1.1 Cost Function and Gradient Details

Given data $\{(x_i,\, u_i,\, z_i)\}_{i=1}^{n}$, the predicted output is:

$$a_i = \sigma(x_i w_1 + u_i w_2 + b)$$

The cost is:

$$C(w_1,\, w_2,\, b) = \frac{1}{n}\sum_{i=1}^{n}\left(a_i - z_i\right)^2 \tag{22}$$

We compute each partial derivative step by step.

Gradient with respect to w_1
Start with:

$$\frac{\partial C}{\partial w_1} = \frac{\partial}{\partial w_1}\left(\frac{1}{n}\sum_{i=1}^{n}\left(a_i - z_i\right)^2\right) = \frac{1}{n}\sum_{i=1}^{n}\frac{\partial}{\partial w_1}\left(a_i - z_i\right)^2 \tag{23}$$

For each i, apply the chain rule:

$$\frac{\partial}{\partial w_1}\left(a_i - z_i\right)^2 = 2\left(a_i - z_i\right)\,\cdot\,\frac{\partial}{\partial w_1}\left(a_i - z_i\right) = 2\left(a_i - z_i\right)\,\cdot\,\frac{\partial a_i}{\partial w_1} \tag{24}$$

Next, $\frac{\partial a_i}{\partial w_1}$ is found by chain rule on $a_i = \sigma(z_i)$ with $z_i = x_i w_1 + u_i w_2 + b$:

$$\frac{\partial a_i}{\partial w_1} = \sigma'(z_i)\,\cdot\,\frac{\partial}{\partial w_1}(z_i) = \sigma'(x_i w_1 + u_i w_2 + b)\,\cdot\,x_i \tag{25}$$

$$= a_i\left(1 - a_i\right)x_i \tag{26}$$

Combining(24) and(26):

$$\frac{\partial}{\partial w_1}\left(a_i - z_i\right)^2 = 2\left(a_i - z_i\right)\left[a_i\left(1 - a_i\right)x_i\right]$$

Therefore, from(23):

$$\frac{\partial C}{\partial w_1} = \frac{1}{n}\sum_{i=1}^{n}\Big[\,2\left(a_i - z_i\right)a_i\left(1 - a_i\right)x_i\Big] = \frac{2}{n}\sum_{i=1}^{n}\left(a_i - z_i\right)a_i\left(1 - a_i\right)x_i \tag{27}$$

Gradient with respect to w_2
Analogously,

$$\frac{\partial C}{\partial w_2} = \frac{\partial}{\partial w_2}\left(\frac{1}{n}\sum_{i=1}^{n}\left(a_i - z_i\right)^2\right) = \frac{1}{n}\sum_{i=1}^{n}\frac{\partial}{\partial w_2}\left(a_i - z_i\right)^2 \tag{28}$$

For each term:

$$\frac{\partial}{\partial w_2}\left(a_i - z_i\right)^2 = 2\left(a_i - z_i\right)\,\cdot\,\frac{\partial a_i}{\partial w_2}, \tag{29}$$

and

$$\frac{\partial a_i}{\partial w_2} = \sigma'(x_i w_1 + u_i w_2 + b) \cdot \frac{\partial}{\partial w_2}(x_i w_1 + u_i w_2 + b) = a_i\,(1 - a_i) \cdot u_i \quad (30)$$

Thus,

$$\frac{\partial}{\partial w_2}(a_i - z_i)^2 = 2\,(a_i - z_i)\,\big[a_i\,(1 - a_i)\,u_i\big]$$

Hence from(28):

$$\frac{\partial C}{\partial w_2} = \frac{1}{n}\sum_{i=1}^{n}\Big[\,2\,(a_i - z_i)\,a_i\,(1 - a_i)\,u_i\Big] = \frac{2}{n}\sum_{i=1}^{n}(a_i - z_i)\,a_i\,(1 - a_i)\,u_i \quad (31)$$

Gradient with respect to b

Finally, for b:

$$\frac{\partial C}{\partial b} = \frac{\partial}{\partial b}\left(\frac{1}{n}\sum_{i=1}^{n}(a_i - z_i)^2\right) = \frac{1}{n}\sum_{i=1}^{n}\frac{\partial}{\partial b}(a_i - z_i)^2 \quad (32)$$

Each term:

$$\frac{\partial}{\partial b}(a_i - z_i)^2 = 2\,(a_i - z_i) \cdot \frac{\partial a_i}{\partial b}, \quad (33)$$

and

$$\frac{\partial a_i}{\partial b} = \sigma'(x_i w_1 + u_i w_2 + b) \cdot \frac{\partial}{\partial b}(x_i w_1 + u_i w_2 + b) = a_i\,(1 - a_i) \cdot 1 = a_i\,(1 - a_i) \quad (34)$$

Hence,

$$\frac{\partial}{\partial b}(a_i - z_i)^2 = 2\,(a_i - z_i)\,\big[a_i\,(1 - a_i)\big],$$

and from(32):

$$\frac{\partial C}{\partial b} = \frac{1}{n}\sum_{i=1}^{n}\Big[\,2\,(a_i - z_i)\,a_i\,(1 - a_i)\Big] = \frac{2}{n}\sum_{i=1}^{n}(a_i - z_i)\,a_i\,(1 - a_i) \quad (35)$$

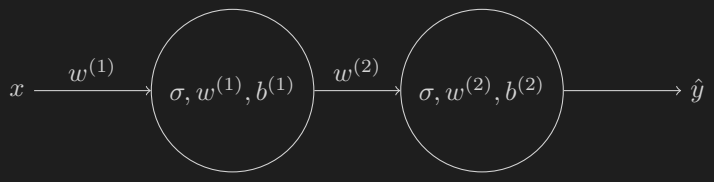
With these detailed intermediate steps, the derivations for both the single-input neuron and the two-input neuron are now fully transparent.

4 Two-Neuron Model with One Input

In this section, we consider a network with one input x that feeds into two sequential neurons (a hidden layer of size 1 and an output neuron). We provide a fully detailed derivation of the forward pass, cost function, and backpropagation steps, showing every derivative calculation and explaining the role of each term (including the concept of "error").

4.1 Network Architecture and Forward Pass

The architecture can be drawn as:



We index examples by $i = 1, 2, \ldots, n$. For the i-th example:

- The input is x_i.
- The hidden layer pre-activation ("net input") is

$$z_i^{(1)} = x_i\, w^{(1)} + b^{(1)}$$

- The hidden layer activation (output of neuron 1) is

$$a_i^{(1)} = \sigma\big(z_i^{(1)}\big) \;=\; \sigma\big(x_i\, w^{(1)} + b^{(1)}\big)$$

- The output layer pre-activation is

$$z_i^{(2)} = a_i^{(1)}\, w^{(2)} + b^{(2)}$$

- The final prediction (activation of neuron 2) is

$$a_i^{(2)} = \hat{y}_i = \sigma\big(z_i^{(2)}\big) \;=\; \sigma\big(a_i^{(1)}\, w^{(2)} + b^{(2)}\big)$$

In summary:

$$z_i^{(1)} = x_i\, w^{(1)} + b^{(1)}, \tag{36}$$

$$a_i^{(1)} = \sigma\big(z_i^{(1)}\big), \tag{37}$$

$$z_i^{(2)} = a_i^{(1)}\, w^{(2)} + b^{(2)}, \tag{38}$$

$$a_i^{(2)} = \sigma\big(z_i^{(2)}\big). \tag{39}$$

Here, $\sigma(z) = 1/(1 + e^{-z})$ is the sigmoid activation (see Sections 2.1.2 for its definition and derivative). Note the superscripts (1) and (2) indicate layer index.

4.2 Cost Function

Given target labels y_i for each example, we define the mean-squared error (MSE) over the training set:

$$C\big(w^{(1)}, b^{(1)}, w^{(2)}, b^{(2)}\big) = \frac{1}{n} \sum_{i=1}^{n} \big(a_i^{(2)} - y_i\big)^2 \tag{40}$$

Our goal is to compute the four gradients: $\partial C/\partial w^{(1)}$, $\partial C/\partial b^{(1)}$, $\partial C/\partial w^{(2)}$, $\partial C/\partial b^{(2)}$. These tell us how to update each parameter to decrease the cost.

4.3 Backpropagation: Definitions and "Error" Terms

The key idea of backpropagation is to compute derivatives of the cost C with respect to each parameter by recursively applying the chain rule from the output layer back toward the input.

Output-Layer "Error" ($\delta_i^{(2)}$)

We define the error at the output neuron for example i as:

$$\delta_i^{(2)} \;=\; \frac{\partial C}{\partial z_i^{(2)}}$$

Intuitively, $\delta_i^{(2)}$ measures how much the cost changes if we perturb the pre-activation $z_i^{(2)}$ at the output neuron. A large $|\delta_i^{(2)}|$ means that a small change in $z_i^{(2)}$ will cause a large change in cost.

Using the chain rule, we can write:

$$\delta_i^{(2)} = \frac{\partial C}{\partial a_i^{(2)}} \cdot \frac{\partial a_i^{(2)}}{\partial z_i^{(2)}}$$

We compute each factor separately:

1. $\dfrac{\partial C}{\partial a_i^{(2)}}$: From(40),

$$C = \frac{1}{n} \sum_{j=1}^{n} \left(a_j^{(2)} - y_j\right)^2$$

Differentiating with respect to $a_i^{(2)}$ (only the $j = i$ term survives):

$$\frac{\partial C}{\partial a_i^{(2)}} = \frac{1}{n} \cdot 2 \left(a_i^{(2)} - y_i\right) = \frac{2}{n} \left(a_i^{(2)} - y_i\right)$$

This term represents the discrepancy between the network's output $a_i^{(2)}$ and the true label y_i. We sometimes call $\left(a_i^{(2)} - y_i\right)$ the *prediction error* at the output.

2. $\dfrac{\partial a_i^{(2)}}{\partial z_i^{(2)}}$: Since $a_i^{(2)} = \sigma(z_i^{(2)})$ and $\sigma'(z) = \sigma(z)(1 - \sigma(z))$ (see(10)),

$$\frac{\partial a_i^{(2)}}{\partial z_i^{(2)}} = \sigma'\left(z_i^{(2)}\right) = a_i^{(2)}\left(1 - a_i^{(2)}\right)$$

This factor tells us how sensitive the activation is to changes in the pre-activation.

Combining these:

$$\delta_i^{(2)} = \frac{\partial C}{\partial a_i^{(2)}} \cdot \frac{\partial a_i^{(2)}}{\partial z_i^{(2)}} \tag{41}$$

$$= \left[\tfrac{2}{n}\,(a_i^{(2)} - y_i)\right] \cdot \left[a_i^{(2)}(1 - a_i^{(2)})\right] \tag{42}$$

$$= \frac{2}{n}\,(a_i^{(2)} - y_i)\; a_i^{(2)}(1 - a_i^{(2)}) \tag{43}$$

In many implementations, one omits the factor $2/n$ by absorbing it into the learning rate; however, for full transparency, we keep it here.

Hidden-Layer "Error" ($\delta_i^{(1)}$)

We similarly define the error for the hidden neuron:

$$\delta_i^{(1)} \;=\; \frac{\partial C}{\partial z_i^{(1)}}$$

To relate $\delta_i^{(1)}$ back to $\delta_i^{(2)}$, we apply the chain rule through the output neuron:

$$\delta_i^{(1)} = \frac{\partial C}{\partial z_i^{(1)}} = \frac{\partial C}{\partial a_i^{(2)}} \cdot \frac{\partial a_i^{(2)}}{\partial z_i^{(2)}} \cdot \frac{\partial z_i^{(2)}}{\partial a_i^{(1)}} \cdot \frac{\partial a_i^{(1)}}{\partial z_i^{(1)}}$$

Notice that C depends on $z_i^{(1)}$ only through $a_i^{(1)}$ and $z_i^{(2)}$. We can group these into known terms:

- We already computed $\dfrac{\partial C}{\partial a_i^{(2)}} = \dfrac{2}{n}(a_i^{(2)} - y_i)$
- We already computed $\dfrac{\partial a_i^{(2)}}{\partial z_i^{(2)}} = a_i^{(2)}(1 - a_i^{(2)})$
- $\dfrac{\partial z_i^{(2)}}{\partial a_i^{(1)}} = w^{(2)}$, since $z_i^{(2)} = a_i^{(1)} w^{(2)} + b^{(2)}$
- $\dfrac{\partial a_i^{(1)}}{\partial z_i^{(1)}} = \sigma'(z_i^{(1)}) = a_i^{(1)}(1 - a_i^{(1)})$

Therefore,

$$\delta_i^{(1)} = \left[\frac{2}{n}\,(a_i^{(2)} - y_i)\right] \cdot \left[a_i^{(2)}(1 - a_i^{(2)})\right] \cdot w^{(2)} \cdot \left[a_i^{(1)}(1 - a_i^{(1)})\right] \tag{44}$$

$$= \left[\frac{2}{n}(a_i^{(2)} - y_i)\, a_i^{(2)}(1 - a_i^{(2)})\, w^{(2)}\right] \cdot \left[a_i^{(1)}(1 - a_i^{(1)})\right] \tag{45}$$

Here, $\delta_i^{(1)}$ quantifies the contribution of the hidden neuron's pre-activation $z_i^{(1)}$ to the overall cost. It is the "backpropagated" error from the output neuron, scaled by the derivative of the hidden activation.

4.4 Gradients for All Parameters

Once we have $\delta_i^{(2)}$ and $\delta_i^{(1)}$, we can obtain parameter gradients by noting how each parameter appears in the pre-activations.

4.4.1 Gradient w.r.t. Output-Layer Parameters

Gradient of C w.r.t. $w^{(2)}$
Since $z_i^{(2)} = a_i^{(1)} w^{(2)} + b^{(2)}$, we have

$$\frac{\partial z_i^{(2)}}{\partial w^{(2)}} = a_i^{(1)}$$

By definition of $\delta_i^{(2)} = \partial C / \partial z_i^{(2)}$, the chain rule gives:

$$\frac{\partial C}{\partial w^{(2)}} = \sum_{i=1}^{n} \frac{\partial C}{\partial z_i^{(2)}} \cdot \frac{\partial z_i^{(2)}}{\partial w^{(2)}} = \sum_{i=1}^{n} \delta_i^{(2)} \left[a_i^{(1)} \right]$$

Since we included a factor of $1/n$ in $\delta_i^{(2)}$ (see(43)), we write:

$$\frac{\partial C}{\partial w^{(2)}} = \sum_{i=1}^{n} \delta_i^{(2)} \, a_i^{(1)} \tag{46}$$

Gradient of C w.r.t. $b^{(2)}$
Since $z_i^{(2)} = a_i^{(1)} w^{(2)} + b^{(2)}$,

$$\frac{\partial z_i^{(2)}}{\partial b^{(2)}} = 1$$

Hence,

$$\frac{\partial C}{\partial b^{(2)}} = \sum_{i=1}^{n} \delta_i^{(2)} \cdot 1 = \sum_{i=1}^{n} \delta_i^{(2)} \tag{47}$$

4.4.2 Gradient w.r.t. Hidden-Layer Parameters

Gradient of C w.r.t. $w^{(1)}$
Since $z_i^{(1)} = x_i w^{(1)} + b^{(1)}$, we have

$$\frac{\partial z_i^{(1)}}{\partial w^{(1)}} = x_i$$

By the chain rule,

$$\frac{\partial C}{\partial w^{(1)}} = \sum_{i=1}^{n} \frac{\partial C}{\partial z_i^{(1)}} \cdot \frac{\partial z_i^{(1)}}{\partial w^{(1)}} = \sum_{i=1}^{n} \delta_i^{(1)} \left[x_i \right]$$

Thus,

$$\frac{\partial C}{\partial w^{(1)}} = \sum_{i=1}^{n} \delta_i^{(1)} \, x_i \tag{48}$$

Gradient of C w.r.t. $b^{(1)}$
Since $z_i^{(1)} = x_i w^{(1)} + b^{(1)}$,

$$\frac{\partial z_i^{(1)}}{\partial b^{(1)}} = 1$$

Therefore,

$$\frac{\partial C}{\partial b^{(1)}} = \sum_{i=1}^{n} \delta_i^{(1)} \tag{49}$$

4.5 Step-by-Step Derivative Computations

1. Compute $\partial C/\partial a_i^{(2)}$
From $C = \frac{1}{n}\sum_j \left(a_j^{(2)} - y_j\right)^2$,

$$\frac{\partial C}{\partial a_i^{(2)}} = \frac{1}{n} \cdot 2\left(a_i^{(2)} - y_i\right) = \frac{2}{n}\left(a_i^{(2)} - y_i\right)$$

2. Compute $\partial a_i^{(2)}/\partial z_i^{(2)}$
Since $a_i^{(2)} = \sigma(z_i^{(2)})$ with $\sigma'(z) = \sigma(z)(1 - \sigma(z))$,

$$\frac{\partial a_i^{(2)}}{\partial z_i^{(2)}} = a_i^{(2)}\left(1 - a_i^{(2)}\right)$$

3. Form $\delta_i^{(2)} = \partial C/\partial z_i^{(2)}$
By chain rule,

$$\delta_i^{(2)} = \frac{\partial C}{\partial a_i^{(2)}} \cdot \frac{\partial a_i^{(2)}}{\partial z_i^{(2)}} = \left[\frac{2}{n}(a_i^{(2)} - y_i)\right] \cdot \left[a_i^{(2)}(1 - a_i^{(2)})\right]$$

4. Compute $\partial C/\partial w^{(2)}$ and $\partial C/\partial b^{(2)}$
Since $z_i^{(2)} = a_i^{(1)} w^{(2)} + b^{(2)}$:

$$\frac{\partial z_i^{(2)}}{\partial w^{(2)}} = a_i^{(1)}, \quad \frac{\partial z_i^{(2)}}{\partial b^{(2)}} = 1$$

Hence,

$$\frac{\partial C}{\partial w^{(2)}} = \sum_i \delta_i^{(2)} a_i^{(1)}, \quad \frac{\partial C}{\partial b^{(2)}} = \sum_i \delta_i^{(2)}$$

5. Compute $\partial z_i^{(2)}/\partial a_i^{(1)}$
Since $z_i^{(2)} = a_i^{(1)} w^{(2)} + b^{(2)}$,

$$\frac{\partial z_i^{(2)}}{\partial a_i^{(1)}} = w^{(2)}$$

6. Compute $\partial a_i^{(1)}/\partial z_i^{(1)}$
Since $a_i^{(1)} = \sigma(z_i^{(1)})$,

$$\frac{\partial a_i^{(1)}}{\partial z_i^{(1)}} = a_i^{(1)}\big(1 - a_i^{(1)}\big)$$

7. Form $\delta_i^{(1)} = \partial C/\partial z_i^{(1)}$
By combining steps 1–6 using chain rule:

$$\begin{aligned}\delta_i^{(1)} &= \frac{\partial C}{\partial a_i^{(2)}} \cdot \frac{\partial a_i^{(2)}}{\partial z_i^{(2)}} \cdot \frac{\partial z_i^{(2)}}{\partial a_i^{(1)}} \cdot \frac{\partial a_i^{(1)}}{\partial z_i^{(1)}} \\ &= \Big[\frac{2}{n}(a_i^{(2)} - y_i)\Big] \;\cdot\; \big[a_i^{(2)}(1 - a_i^{(2)})\big] \;\cdot\; w^{(2)} \;\cdot\; \big[a_i^{(1)}(1 - a_i^{(1)})\big].\end{aligned}$$

8. Compute $\partial C/\partial w^{(1)}$ and $\partial C/\partial b^{(1)}$
Since $z_i^{(1)} = x_i w^{(1)} + b^{(1)}$:

$$\frac{\partial z_i^{(1)}}{\partial w^{(1)}} = x_i, \quad \frac{\partial z_i^{(1)}}{\partial b^{(1)}} = 1$$

Therefore:

$$\frac{\partial C}{\partial w^{(1)}} = \sum_i \delta_i^{(1)} x_i, \quad \frac{\partial C}{\partial b^{(1)}} = \sum_i \delta_i^{(1)}$$

4.6 Interpretation of "Error" Terms

- **Output error $\delta_i^{(2)}$:** Measures how much changing the output neuron's pre-activation $z_i^{(2)}$ affects the cost. It combines:
  1. The prediction error $(a_i^{(2)} - y_i)$, which tells us how far the network's output is from the target.
  2. The factor $a_i^{(2)}(1 - a_i^{(2)})$, which is the derivative of the sigmoid and scales the error by the local slope of the activation.
  3. The factor $2/n$, which normalizes by dataset size and accounts for the derivative of $(\cdot)^2$.
- **Hidden error $\delta_i^{(1)}$:** Measures how much changing the hidden neuron's pre-activation $z_i^{(1)}$ affects the final cost. It arises by "backpropagating" the output error through:
  1. The weight $w^{(2)}$, since the hidden neuron's activation $a_i^{(1)}$ contributes to $z_i^{(2)}$ via $w^{(2)}$.
  2. The sigmoid slope at the hidden layer $a_i^{(1)}(1 - a_i^{(1)})$, since that is $\partial a_i^{(1)}/\partial z_i^{(1)}$.

In effect, $\delta_i^{(1)}$ redistributes the output-layer error back into the hidden layer, scaled by how sensitive the hidden activation is.

By computing these errors $\delta^{(2)}$ and $\delta^{(1)}$, we efficiently obtain all parameter gradients using(46)–(49). This is the essence of backpropagation: local gradients at each neuron (the "error" terms) are combined with the derivatives of pre-activations to give global parameter gradients.

5 One-Input, n-Layer Neural Network

In this section, we generalize from the specific "two-layer' (one hidden layer + one output layer) case to an arbitrary number n of layers, all with a single scalar input x. We will:

1. Introduce notation for n layers.
2. Describe the forward pass in full detail.
3. Define the cost function (mean-squared error) over a dataset.
4. Derive the backpropagation formulas layer by layer, showing every intermediate term.
5. Explain the meaning of each symbol and each mathematical step.

Throughout, we keep the dark-background / light-text style established earlier.

5.1 Architecture and Notation

Consider a neural network with one input $x \in \mathbb{R}$ (a single scalar), followed by $n-1$ hidden layers and one final output layer. We index layers by superscript (ℓ), where $\ell = 1, 2, \ldots, n$. Layer $\ell = 1$ is the very first layer that directly receives the input x, and layer $\ell = n$ is the output layer whose activation we compare to the true label.

For each layer ℓ, we have:

- A *weight* $w^{(\ell)} \in \mathbb{R}$ (since each layer receives exactly one scalar input from the previous layer, there is exactly one weight per layer).
- A *bias* $b^{(\ell)} \in \mathbb{R}$.
- A *pre-activation* $z^{(\ell)} \in \mathbb{R}$, defined just before applying the activation function.
- An *activation* $a^{(\ell)} = \sigma\big(z^{(\ell)}\big)$ (we will use the sigmoid nonlinearity by default, though any differentiable activation could be used).

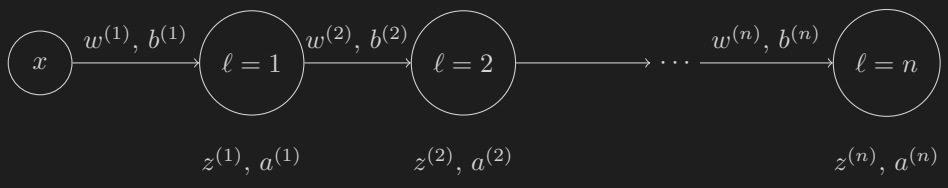
In summary:

$$\underbrace{x}_{\text{input}} \xrightarrow{w^{(1)},\, b^{(1)}} \underbrace{z^{(1)}}_{\text{pre-act. 1}} \xrightarrow{\sigma} \underbrace{a^{(1)}}_{\text{act. 1}} \xrightarrow{w^{(2)},\, b^{(2)}} z^{(2)} \xrightarrow{\sigma} a^{(2)} \;\cdots\; \xrightarrow{w^{(n)},\, b^{(n)}} z^{(n)} \xrightarrow{\sigma} a^{(n)} = \hat{y}$$

Here:

- $z^{(1)} = x\, w^{(1)} + b^{(1)}$.
- For $\ell = 2, 3, \ldots, n$,
$$z^{(\ell)} = a^{(\ell-1)}\, w^{(\ell)} + b^{(\ell)}.$$
- The final network output (prediction) is $a^{(n)} = \hat{y}$.
- The true label for example i is y_i.

Illustration of a One-Input, n-Layer Network



In this diagram:

- The single input x feeds into layer $\ell = 1$, producing pre-activation $z^{(1)}$ and activation $a^{(1)}$.
- That activation $a^{(1)}$ becomes the input to layer $\ell = 2$, producing $z^{(2)}$, $a^{(2)}$.
- This process continues through $\ell = 3, 4, \ldots, n$.
- The final output $a^{(n)}$ is the network's prediction $\hat{y}$.

We now detail the forward pass, cost, and backward pass (backpropagation).

We assume a dataset of N examples $\{(x_i, y_i)\}_{i=1}^{N}$. We will drop the index i during the derivation, writing (x, y) for a generic single example, and re-insert the sum at the end to define the full cost.

5.2 Forward Pass

For a single example (x, y), the forward-pass proceeds layer by layer:

1. **Layer $\ell = 1$:**

$$z^{(1)} = w^{(1)}\, x \;+\; b^{(1)}, \qquad a^{(1)} = \sigma\big(z^{(1)}\big).$$

   Here σ is the sigmoid function:

$$\sigma(u) \;=\; \frac{1}{1 + e^{-\,u}}.$$

2. **Layer $\ell = 2$:**

$$z^{(2)} = w^{(2)}\, a^{(1)} \;+\; b^{(2)}, \qquad a^{(2)} = \sigma\big(z^{(2)}\big).$$

3. $\vdots$

4. **Layer $\ell = n$ (Output Layer):**

$$z^{(n)} = w^{(n)}\, a^{(n-1)} \;+\; b^{(n)}, \qquad a^{(n)} = \sigma\big(z^{(n)}\big) \;=\; \hat{y}.$$

Each $z^{(\ell)}$ is called the *pre-activation*, and each $a^{(\ell)}$ is called the *activation* of layer ℓ. Note that $a^{(0)} := x$ if we wish to unify notation, but we will keep x as a special symbol.

In full, for our one-input n-layer network,

$$\begin{aligned}
z^{(1)} &= w^{(1)}\, x + b^{(1)}, & a^{(1)} &= \sigma\big(z^{(1)}\big),\\
z^{(2)} &= w^{(2)}\, a^{(1)} + b^{(2)}, & a^{(2)} &= \sigma\big(z^{(2)}\big),\\
&\;\;\vdots & &\;\;\vdots\\
z^{(n)} &= w^{(n)}\, a^{(n-1)} + b^{(n)}, & a^{(n)} &= \sigma\big(z^{(n)}\big) = \hat{y}.
\end{aligned}$$

After computing $a^{(n)}$, we measure the discrepancy against the true label y via a loss function. We choose the mean-squared error (MSE) for a single example:

$$L\big(x, y; \{w^{(\ell)}, b^{(\ell)}\}\big) = \big(a^{(n)} - y\big)^2.$$

5.3 Cost over the Entire Dataset

To train, we average the loss over all N examples:

$$\begin{aligned}
C &= \frac{1}{N} \sum_{i=1}^{N} L\big(x_i,\, y_i;\, \{w^{(\ell)}, b^{(\ell)}\}\big)\\
&= \frac{1}{N} \sum_{i=1}^{N} \big(\hat{y}_i - y_i\big)^2 = \frac{1}{N} \sum_{i=1}^{N} \Big[a_i^{(n)} - y_i\Big]^2. \qquad (50)
\end{aligned}$$

Here $\hat{y}_i = a_i^{(n)}$ is the network's prediction on input x_i. The goal of training is to find weights and biases $\{w^{(\ell)}, b^{(\ell)}\}_{\ell=1}^{n}$ that minimize C.

—

5.4 Backpropagation: Detailed Derivation

We now derive how to compute $\dfrac{\partial C}{\partial w^{(\ell)}}$, $\dfrac{\partial C}{\partial b^{(\ell)}}$ for $\ell = 1, 2, \ldots, n$. Because the network is built by repeated compositions, we apply the chain rule systematically from the output layer backward to the input layer. For clarity, we focus on a single example (x, y) and then re-insert the sum over i at the end—this is equivalent to computing $\partial L/\partial\theta$ for one example, then averaging.

Step 1: Define "Error' at the Output Layer.

Recall the single-example loss

$$L = \big(a^{(n)} - y\big)^2, \qquad a^{(n)} = \sigma\big(z^{(n)}\big).$$

We introduce the shorthand

$$\delta^{(n)} \;=\; \frac{\partial L}{\partial z^{(n)}}.$$

That is, $\delta^{(n)}$ measures how much the loss L changes if we nudge the pre-activation $z^{(n)}$ at the output layer. By the chain rule:

$$\delta^{(n)} = \frac{\partial L}{\partial a^{(n)}} \;\cdot\; \frac{\partial a^{(n)}}{\partial z^{(n)}}.$$

We compute each factor:

- $\dfrac{\partial L}{\partial a^{(n)}} = \dfrac{\partial}{\partial a^{(n)}}(a^{(n)} - y)^2 = 2\left(a^{(n)} - y\right)$
- $\dfrac{\partial a^{(n)}}{\partial z^{(n)}} = \sigma'\big(z^{(n)}\big) = \sigma\big(z^{(n)}\big)\left[1 - \sigma\big(z^{(n)}\big)\right] = a^{(n)}\left(1 - a^{(n)}\right)$

Therefore,

$$\delta^{(n)} = \Big[2\left(a^{(n)} - y\right)\Big] \times \Big[a^{(n)}\left(1 - a^{(n)}\right)\Big] = 2\left(a^{(n)} - y\right) a^{(n)}\left(1 - a^{(n)}\right). \qquad (51)$$

Step 2: Gradients at the Output Layer.

For layer $\ell = n$, the pre-activation was

$$z^{(n)} = w^{(n)}\, a^{(n-1)} + b^{(n)}.$$

Hence:

- $\dfrac{\partial z^{(n)}}{\partial w^{(n)}} = a^{(n-1)}$,
- $\dfrac{\partial z^{(n)}}{\partial b^{(n)}} = 1$

By the chain rule for the loss:

$$\frac{\partial L}{\partial w^{(n)}} = \frac{\partial L}{\partial z^{(n)}} \cdot \frac{\partial z^{(n)}}{\partial w^{(n)}} = \delta^{(n)} \times a^{(n-1)}, \tag{52}$$

$$\frac{\partial L}{\partial b^{(n)}} = \frac{\partial L}{\partial z^{(n)}} \cdot \frac{\partial z^{(n)}}{\partial b^{(n)}} = \delta^{(n)} \times 1 = \delta^{(n)}. \tag{53}$$

Step 3: Propagate Error Back to Hidden Layers.

For a generic hidden layer ℓ (where $1 \le \ell \le n-1$), we define

$$\delta^{(\ell)} = \frac{\partial L}{\partial z^{(\ell)}}.$$

We will derive a recurrence expressing $\delta^{(\ell)}$ in terms of $\delta^{(\ell+1)}$. First, observe that

$$a^{(\ell)} = \sigma\big(z^{(\ell)}\big), \qquad z^{(\ell+1)} = w^{(\ell+1)}\, a^{(\ell)} + b^{(\ell+1)}.$$

Hence, by the multivariate chain rule:

$$\delta^{(\ell)} = \frac{\partial L}{\partial z^{(\ell)}} = \frac{\partial L}{\partial z^{(\ell+1)}} \cdot \frac{\partial z^{(\ell+1)}}{\partial a^{(\ell)}} \cdot \frac{\partial a^{(\ell)}}{\partial z^{(\ell)}}.$$

We already know:

$$\frac{\partial L}{\partial z^{(\ell+1)}} = \delta^{(\ell+1)}, \qquad \frac{\partial z^{(\ell+1)}}{\partial a^{(\ell)}} = w^{(\ell+1)}, \qquad \frac{\partial a^{(\ell)}}{\partial z^{(\ell)}} = \sigma'\big(z^{(\ell)}\big) = a^{(\ell)}\,(1 - a^{(\ell)}).$$

Therefore, for $\ell = n-1,\, n-2,\, \ldots,\, 1$:

$$\delta^{(\ell)} = \delta^{(\ell+1)} \times w^{(\ell+1)} \times \big[a^{(\ell)}\,\big(1 - a^{(\ell)}\big)\big] = \big[w^{(\ell+1)}\,\delta^{(\ell+1)}\big]\; a^{(\ell)}\big(1 - a^{(\ell)}\big). \tag{54}$$

In words:

$$\delta^{(\ell)} = \big(\text{next-layer error } \delta^{(\ell+1)}\big) \times \big(\text{weight } w^{(\ell+1)}\big) \times \big(\text{activation slope } a^{(\ell)}(1 - a^{(\ell)})\big).$$

Geometrically, $\delta^{(\ell)}$ measures how much the final loss L would change if you nudge the pre-activation at layer ℓ. We compute these $\delta^{(\ell)}$ in a backward sweep from $\ell = n$ down to $\ell = 1$.

Step 4: Gradients for Hidden Layers.

Once $\delta^{(\ell)}$ is known, the gradient of L with respect to $w^{(\ell)}$ and $b^{(\ell)}$ is straightforward, because

$$z^{(\ell)} = w^{(\ell)}\, a^{(\ell-1)} + b^{(\ell)} \quad (\text{with } a^{(0)} := x).$$

Thus:

$$\frac{\partial L}{\partial w^{(\ell)}} = \frac{\partial L}{\partial z^{(\ell)}} \cdot \frac{\partial z^{(\ell)}}{\partial w^{(\ell)}} = \delta^{(\ell)} \times a^{(\ell-1)}, \tag{55}$$

$$\frac{\partial L}{\partial b^{(\ell)}} = \frac{\partial L}{\partial z^{(\ell)}} \cdot \frac{\partial z^{(\ell)}}{\partial b^{(\ell)}} = \delta^{(\ell)} \times 1 = \delta^{(\ell)}. \tag{56}$$

Here $a^{(\ell-1)}$ is the activation coming *into* layer ℓ. For $\ell = 1$, $a^{(0)} = x$ (the network's input).

Step 5: Summing / Averaging Over All Examples.

All of the above was for a single example (x, y). To get the gradient of the total cost C in(50), we average the single-example gradients:

$$\frac{\partial C}{\partial w^{(\ell)}} = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial L_i}{\partial w^{(\ell)}}, \qquad \frac{\partial C}{\partial b^{(\ell)}} = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial L_i}{\partial b^{(\ell)}},$$

where $L_i = \left(a_i^{(n)} - y_i\right)^2$ is the loss on example i. In practice, one often uses mini-batch gradients, but the principle is identical: compute $\delta_i^{(\ell)}$ for each example i, then average across that batch.

Putting it all together:

1. **Forward Pass (per example i):**

$$\begin{aligned} z_i^{(1)} &= w^{(1)}\, x_i \;+\; b^{(1)}, & a_i^{(1)} &= \sigma\big(z_i^{(1)}\big), \\ z_i^{(2)} &= w^{(2)}\, a_i^{(1)} \;+\; b^{(2)}, & a_i^{(2)} &= \sigma\big(z_i^{(2)}\big), \\ &\vdots & &\vdots \\ z_i^{(n)} &= w^{(n)}\, a_i^{(n-1)} \;+\; b^{(n)}, & a_i^{(n)} &= \sigma\big(z_i^{(n)}\big). \end{aligned}$$

   Then $L_i = \left(a_i^{(n)} - y_i\right)^2$

2. **Compute Output-Layer Error $\delta_i^{(n)}$:**

$$\delta_i^{(n)} = 2\left(a_i^{(n)} - y_i\right)\, a_i^{(n)}\big(1 - a_i^{(n)}\big).$$

3. **Backward Pass (for $\ell = n-1, n-2, \ldots, 1$):**

$$\delta_i^{(\ell)} = \left[w^{(\ell+1)}\, \delta_i^{(\ell+1)}\right] \times a_i^{(\ell)}\big(1 - a_i^{(\ell)}\big).$$

4. **Gradients (per example i):** For each $\ell = 1, 2, \ldots, n$,

$$\begin{aligned} \frac{\partial L_i}{\partial w^{(\ell)}} &= \delta_i^{(\ell)} \;\times\; a_i^{(\ell-1)}, \quad (\text{where}) a_i^{(0)} = x_i; \\ \frac{\partial L_i}{\partial b^{(\ell)}} &= \delta_i^{(\ell)}. \end{aligned}$$

5. **Average Over All N Examples:**

$$\frac{\partial C}{\partial w^{(\ell)}} = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial L_i}{\partial w^{(\ell)}}, \qquad \frac{\partial C}{\partial b^{(\ell)}} = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial L_i}{\partial b^{(\ell)}}.$$

—

5.5 Key Explanations and Interpretations

We summarize the conceptual meaning of each term:

- **Pre-activation $z^{(\ell)}$:** This is the weighted input plus bias for layer ℓ. It is the argument to the activation function at that layer.
- **Activation $a^{(\ell)}$:** Defined by $a^{(\ell)} = \sigma\big(z^{(\ell)}\big)$. This is what layer ℓ passes forward to layer $\ell + 1$. For $\ell = 0$, we set $a^{(0)} = x$.
- **Single-example loss $L = \big(a^{(n)} - y\big)^2$:** Measures how far the network's prediction $a^{(n)}$ is from the true label y.
- **Error term $\delta^{(n)}$ at the output:**

  $$\delta^{(n)} = \frac{\partial L}{\partial z^{(n)}} = 2\,(a^{(n)} - y)\,\sigma'\big(z^{(n)}\big).$$

  This quantity tells us how sensitive the loss is to a small change in $z^{(n)}$. Since $z^{(n)}$ is the input of the output neuron, $\delta^{(n)}$ measures the "blame" placed on that neuron's pre-activation.
- **Error term $\delta^{(\ell)}$ at a hidden layer $\ell < n$:**

  $$\delta^{(\ell)} = w^{(\ell+1)}\,\delta^{(\ell+1)} \;\times\; \sigma'\big(z^{(\ell)}\big).$$

  Here, $w^{(\ell+1)}\,\delta^{(\ell+1)}$ carries back how much layer $\ell + 1$ "blamed" layer ℓ. Multiplying by $\sigma'(z^{(\ell)})$ captures how a small perturbation in $z^{(\ell)}$ changes $a^{(\ell)}$, and thus the final loss.
- **Gradients $\partial L/\partial w^{(\ell)}$ and $\partial L/\partial b^{(\ell)}$:** Once $\delta^{(\ell)}$ is known, we simply note that

  $$z^{(\ell)} = w^{(\ell)}\,a^{(\ell-1)} + b^{(\ell)} \quad \Longrightarrow \quad \frac{\partial z^{(\ell)}}{\partial w^{(\ell)}} = a^{(\ell-1)}, \quad \frac{\partial z^{(\ell)}}{\partial b^{(\ell)}} = 1.$$

  Hence

  $$\frac{\partial L}{\partial w^{(\ell)}} = \delta^{(\ell)}\,a^{(\ell-1)}, \qquad \frac{\partial L}{\partial b^{(\ell)}} = \delta^{(\ell)}.$$

  In other words, $\delta^{(\ell)}$ is the "rate of change of loss with respect to a small change in $z^{(\ell)}$," and $a^{(\ell-1)}$ is the input to $z^{(\ell)}$.
- **Averaging over examples:** When using the mean-squared error over N examples, each example yields its own $\delta_i^{(\ell)}$. We average the per-example gradients to find $\partial C/\partial w^{(\ell)}$.

Because every step uses only quantities computed in the forward pass (activations $a^{(\ell)}$) and previously computed δ values (one from the layer above), backpropagation requires exactly one forward sweep and one backward sweep per mini-batch. No repeated forward passes are needed per parameter, which is what makes deep network training computationally feasible even when n is large.

—

6 Arbitrary n Inputs and L Layers

In this section, we extend our analysis to a fully general feedforward neural network with:

- n *input features* $x \in \mathbb{R}^n$.
- L *layers* (numbered $\ell = 1, 2, \ldots, L$), each of which may contain an arbitrary number of neurons.

We will:

1. Introduce notation for arbitrary input dimension and a sequence of hidden/output layers.
2. Describe the forward pass in a vectorized, scalable form.
3. Define the dataset cost (mean-squared error) and explain how it depends on the network's parameters.
4. Derive the backpropagation formulas in full detail, layer by layer, showing every intermediate step.
5. Clarify the dimensions and interpretations of all vectors and matrices.

Throughout, we continue using a dark background with light text; new notation is bolded for clarity.

6.1 Network Architecture and Notation

Let $x \in \mathbb{R}^n$ be the column vector of n input features:

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}.$$

We build an L-layer feedforward network as follows:

- $\ell = 0$ denotes the *input layer*, whose "activation" is simply $a^{(0)} = x \in \mathbb{R}^n$.
- For each layer $\ell = 1, 2, \ldots, L$, we denote:
  - m_ℓ = number of neurons in layer ℓ. We allow $m_1, m_2, \ldots, m_L$ to be arbitrary positive integers.
  - $\mathbf{W}^{(\ell)} \in \mathbb{R}^{m_\ell \times m_{\ell-1}}$ = weight matrix connecting layer $\ell - 1$ to layer ℓ.
  - $\mathbf{b}^{(\ell)} \in \mathbb{R}^{m_\ell}$ = bias vector for layer ℓ.
  - $\mathbf{z}^{(\ell)} \in \mathbb{R}^{m_\ell}$ = pre-activation vector for layer ℓ.

– $\mathbf{a}^{(\ell)} \in \mathbb{R}^{m_\ell}$ = activation vector for layer ℓ, with $\mathbf{a}^{(\ell)} = \sigma\big(\mathbf{z}^{(\ell)}\big)$ applied elementwise.

- The final layer $\ell = L$ is the *output layer*. If the task is regression with a single scalar output, then $m_L = 1$. If it is multiclass classification, m_L can be larger; here we assume m_L may be arbitrary.

Concretely, for each layer $\ell = 1, 2, \ldots, L$:

$$\mathbf{z}^{(\ell)} \;=\; \mathbf{W}^{(\ell)}\,\mathbf{a}^{(\ell-1)} \;+\; \mathbf{b}^{(\ell)}, \qquad \mathbf{a}^{(\ell)} \;=\; \sigma\big(\mathbf{z}^{(\ell)}\big),$$

where $\sigma(\cdot)$ acts component-wise:

$$\sigma(u) = \frac{1}{1+e^{-\,u}} \quad\Longrightarrow\quad \big[\sigma(\mathbf{z})\big]_i = \frac{1}{1+e^{-\,z_i}}, \quad \mathbf{z} \in \mathbb{R}^m.$$

For the input layer we set $\mathbf{a}^{(0)} = x \in \mathbb{R}^n$, so:

$$\mathbf{z}^{(1)} = \mathbf{W}^{(1)} \underbrace{x}_{\mathbf{a}^{(0)}} + \mathbf{b}^{(1)}, \quad \mathbf{a}^{(1)} = \sigma\big(\mathbf{z}^{(1)}\big),$$

and so on until the output:

$$\mathbf{z}^{(L)} = \mathbf{W}^{(L)}\,\mathbf{a}^{(L-1)} + \mathbf{b}^{(L)}, \quad \mathbf{a}^{(L)} = \sigma\big(\mathbf{z}^{(L)}\big).$$

If $m_L = 1$, then $\mathbf{a}^{(L)}$ is a single scalar $\hat{y}$. In general, we denote the prediction vector by:

$$\hat{\mathbf{y}} \;=\; \mathbf{a}^{(L)} \;\in\; \mathbb{R}^{m_L}.$$

Dimensions at a Glance:

$$\begin{aligned} &\mathbf{a}^{(0)} \in \mathbb{R}^{m_0}, && m_0 = n, \\ &\mathbf{W}^{(\ell)} \in \mathbb{R}^{m_\ell \times m_{\ell-1}}, && \mathbf{b}^{(\ell)} \in \mathbb{R}^{m_\ell}, \\ &\mathbf{z}^{(\ell)},\, \mathbf{a}^{(\ell)} \in \mathbb{R}^{m_\ell}, && \ell = 1, 2, \ldots, L. \end{aligned}$$

Hence $\mathbf{W}^{(1)}$ is $m_1 \times n$, $\mathbf{W}^{(2)}$ is $m_2 \times m_1$, …, $\mathbf{W}^{(L)}$ is $m_L \times m_{L-1}$. Bias $\mathbf{b}^{(\ell)}$ is a column vector of length m_ℓ.

6.2 Forward Pass (Vectorized Form)

For a single input vector x, the forward pass computes:

(1) Input Activation:

$$\mathbf{a}^{(0)} \;=\; x \;\in\; \mathbb{R}^{m_0 = n}.$$

(2) Layer-by-Layer Computations:

$$\begin{aligned} \mathbf{z}^{(1)} &= \mathbf{W}^{(1)}\,\mathbf{a}^{(0)} + \mathbf{b}^{(1)}, && \mathbf{a}^{(1)} = \sigma\big(\mathbf{z}^{(1)}\big); \\ \mathbf{z}^{(2)} &= \mathbf{W}^{(2)}\,\mathbf{a}^{(1)} + \mathbf{b}^{(2)}, && \mathbf{a}^{(2)} = \sigma\big(\mathbf{z}^{(2)}\big); \\ &\vdots && \vdots \\ \mathbf{z}^{(L)} &= \mathbf{W}^{(L)}\,\mathbf{a}^{(L-1)} + \mathbf{b}^{(L)}, && \mathbf{a}^{(L)} = \sigma\big(\mathbf{z}^{(L)}\big). \end{aligned}$$

The final activation $\mathbf{a}^{(L)} \in \mathbb{R}^{m_L}$ is the network's prediction $\hat{\mathbf{y}}$. In a regression setting with a single real output, $m_L = 1$. In a classification setting, m_L may be equal to the number of classes.

6.3 Cost Function Over a Dataset

Assume we have a training set $\{(x_i, \mathbf{y}_i)\}_{i=1}^N$, where $\mathbf{y}_i \in \mathbb{R}^{m_L}$ is the true label vector for example i. We use the mean-squared error (MSE) over the entire dataset:

$$C\big(\{\mathbf{W}^{(\ell)}, \mathbf{b}^{(\ell)}\}\big) = \frac{1}{N}\sum_{i=1}^{N} \big\| \hat{\mathbf{y}}_i - \mathbf{y}_i \big\|_2^2 \;=\; \frac{1}{N}\sum_{i=1}^{N} \big\| \mathbf{a}_i^{(L)} - \mathbf{y}_i \big\|_2^2. \tag{57}$$

Here $\hat{\mathbf{y}}_i = \mathbf{a}_i^{(L)}$ is the network's output when the input is x_i. If $m_L = 1$, this reduces to

$$C \;=\; \frac{1}{N}\sum_{i=1}^{N} \big(a_i^{(L)} - y_i\big)^2.$$

Our objective is to adjust all weight matrices $\{\mathbf{W}^{(\ell)}\}$ and bias vectors $\{\mathbf{b}^{(\ell)}\}$ to minimize C.

6.4 Backpropagation: Detailed Derivation

We derive formulas for the gradients:

$$\frac{\partial C}{\partial \mathbf{W}^{(\ell)}} \quad \text{and} \quad \frac{\partial C}{\partial \mathbf{b}^{(\ell)}}, \quad \ell = 1, 2, \ldots, L.$$

The derivation proceeds in two nested loops:

1. Compute the *error vectors* $\delta_i^{(\ell)} = \partial L_i / \partial \mathbf{z}_i^{(\ell)}$ for each training example i, layer by layer from $\ell = L$ down to $\ell = 1$.
2. Sum or average these per-example gradients to form $\partial C/\partial \mathbf{W}^{(\ell)}$ and $\partial C/\partial \mathbf{b}^{(\ell)}$.

Notation: - For example i, we denote by $\mathbf{a}_i^{(\ell)}, \mathbf{z}_i^{(\ell)}, \delta_i^{(\ell)}$ the activations, pre-activations, and error vectors at layer ℓ. - The single-example loss is

$$L_i = \big\| \mathbf{a}_i^{(L)} - \mathbf{y}_i \big\|_2^2 = \sum_{k=1}^{m_L} \big[a_{i,k}^{(L)} - y_{i,k}\big]^2.$$

- The network parameters at layer ℓ are $\mathbf{W}^{(\ell)} \in \mathbb{R}^{m_\ell \times m_{\ell-1}}$ and $\mathbf{b}^{(\ell)} \in \mathbb{R}^{m_\ell}$.

—

Step 1 : Compute Output-Layer Error $\delta_i^{(L)}$.

For each example $i = 1, \ldots, N$, the output layer's pre-activation is

$$\mathbf{z}_i^{(L)} = \mathbf{W}^{(L)}\, \mathbf{a}_i^{(L-1)} \;+\; \mathbf{b}^{(L)}, \quad \mathbf{a}_i^{(L)} = \sigma\big(\mathbf{z}_i^{(L)}\big).$$

The per-example loss is

$$L_i = \left\|\mathbf{a}_i^{(L)} - \mathbf{y}_i\right\|_2^2 = \sum_{k=1}^{m_L} \left[a_{i,k}^{(L)} - y_{i,k}\right]^2.$$

We define

$$\delta_i^{(L)} \;=\; \frac{\partial L_i}{\partial \mathbf{z}_i^{(L)}} \;\in\; \mathbb{R}^{\,m_L}.$$

By the chain rule (applied elementwise across the m_L outputs):

$$\delta_{i,k}^{(L)} = \frac{\partial L_i}{\partial a_{i,k}^{(L)}} \;\cdot\; \frac{\partial a_{i,k}^{(L)}}{\partial z_{i,k}^{(L)}}, \quad k = 1, \ldots, m_L.$$

Explicitly:

$$\frac{\partial L_i}{\partial a_{i,k}^{(L)}} = 2\left[a_{i,k}^{(L)} - y_{i,k}\right], \qquad \frac{\partial a_{i,k}^{(L)}}{\partial z_{i,k}^{(L)}} = \sigma'\big(z_{i,k}^{(L)}\big) = a_{i,k}^{(L)}\left(1 - a_{i,k}^{(L)}\right).$$

Therefore,

$$\delta_i^{(L)} = \big(\mathbf{a}_i^{(L)} - \mathbf{y}_i\big) \;\circ\; \sigma'\big(\mathbf{z}_i^{(L)}\big) \;=\; \big(\mathbf{a}_i^{(L)} - \mathbf{y}_i\big) \;\circ\; \big[\mathbf{a}_i^{(L)} \circ \big(\mathbf{1} - \mathbf{a}_i^{(L)}\big)\big], \tag{58}$$

where "$\circ$" denotes element-wise (Hadamard) product and $\mathbf{1}$ is the all-ones vector of dimension m_L.

—

Step 2 : **Backpropagate to Hidden Layers**.
For each $\ell = L-1, L-2, \ldots, 1$, we compute

$$\delta_i^{(\ell)} \;=\; \frac{\partial L_i}{\partial \mathbf{z}_i^{(\ell)}} \;\in\; \mathbb{R}^{\,m_\ell}.$$

Since

$$\mathbf{z}_i^{(\ell+1)} = \mathbf{W}^{(\ell+1)}\,\mathbf{a}_i^{(\ell)} + \mathbf{b}^{(\ell+1)}, \quad \mathbf{a}_i^{(\ell)} = \sigma\big(\mathbf{z}_i^{(\ell)}\big),$$

the chain rule gives:

$$\delta_i^{(\ell)} = \big(\mathbf{W}^{(\ell+1)}\big)^T\,\delta_i^{(\ell+1)} \;\circ\; \sigma'\big(\mathbf{z}_i^{(\ell)}\big) \;=\; \Big[\big(\mathbf{W}^{(\ell+1)}\big)^T\,\delta_i^{(\ell+1)}\Big] \;\circ\; \big[\mathbf{a}_i^{(\ell)} \circ \big(\mathbf{1} - \mathbf{a}_i^{(\ell)}\big)\big]. \tag{59}$$

Here:

- $\big(\mathbf{W}^{(\ell+1)}\big)^T\,\delta_i^{(\ell+1)}$ computes the "weighted sum of errors" from layer $\ell+1$ back to layer ℓ.
- $\sigma'\big(\mathbf{z}_i^{(\ell)}\big)$ is computed elementwise as $\mathbf{a}_i^{(\ell)} \circ \big(\mathbf{1} - \mathbf{a}_i^{(\ell)}\big)$.

This backward sweep continues until $\ell = 1$, yielding $\delta_i^{(1)} \in \mathbb{R}^{m_1}$.

—

Step 3 : **Compute Per-Example Parameter Gradients**.
Once $\delta_i^{(\ell)}$ is known for all layers ℓ, we find the gradient of the single-example loss L_i with respect to each weight matrix $\mathbf{W}^{(\ell)}$ and bias vector $\mathbf{b}^{(\ell)}$. Recall

$$\mathbf{z}_i^{(\ell)} = \mathbf{W}^{(\ell)}\, \mathbf{a}_i^{(\ell-1)} \; + \; \mathbf{b}^{(\ell)}.$$

Therefore:

$$\frac{\partial L_i}{\partial \mathbf{W}^{(\ell)}} = \delta_i^{(\ell)} \, \big(\mathbf{a}_i^{(\ell-1)}\big)^T \; \in \; \mathbb{R}^{\,m_\ell \times m_{\ell-1}}, \tag{60}$$

$$\frac{\partial L_i}{\partial \mathbf{b}^{(\ell)}} = \delta_i^{(\ell)} \; \in \; \mathbb{R}^{\,m_\ell}. \tag{61}$$

- In(60), $\delta_i^{(\ell)}$ is $m_\ell \times 1$ and $\mathbf{a}_i^{(\ell-1)}$ is $m_{\ell-1} \times 1$, so $\delta_i^{(\ell)}(\mathbf{a}_i^{(\ell-1)})^T$ is an $m_\ell \times m_{\ell-1}$ matrix, matching the dimensions of $\mathbf{W}^{(\ell)}$. - In(61), $\delta_i^{(\ell)}$ and $\mathbf{b}^{(\ell)}$ are both $m_\ell \times 1$ vectors.

—

Step 4 : **Average Across the Entire Dataset**.
The total cost C from(57) is the average of L_i over $i = 1, \ldots, N$. Hence, the gradient of C is the average of per-example gradients:

$$\frac{\partial C}{\partial \mathbf{W}^{(\ell)}} = \frac{1}{N} \sum_{i=1}^{N} \frac{\partial L_i}{\partial \mathbf{W}^{(\ell)}} = \frac{1}{N} \sum_{i=1}^{N} \Big[\, \delta_i^{(\ell)} \, \big(\mathbf{a}_i^{(\ell-1)}\big)^T \Big], \tag{62}$$

$$\frac{\partial C}{\partial \mathbf{b}^{(\ell)}} = \frac{1}{N} \sum_{i=1}^{N} \frac{\partial L_i}{\partial \mathbf{b}^{(\ell)}} = \frac{1}{N} \sum_{i=1}^{N} \delta_i^{(\ell)}. \tag{63}$$

In practice, one often uses *mini-batches* of size $B \leq N$, summing over i in that batch and dividing by B.

Algorithmic Summary (Vectorized Pseudocode):

A. *Forward Pass (all N examples):*

Input: $\{x_i\}_{i=1}^N$, $\{\mathbf{W}^{(\ell)}, \mathbf{b}^{(\ell)}\}_{\ell=1}^L$.
For $i = 1 \ldots N$: $\quad \mathbf{a}_i^{(0)} = x_i$.
For $\ell = 1 \ldots L$: $\quad \mathbf{z}_i^{(\ell)} = \mathbf{W}^{(\ell)}\, \mathbf{a}_i^{(\ell-1)} + \mathbf{b}^{(\ell)}, \quad \mathbf{a}_i^{(\ell)} = \sigma\big(\mathbf{z}_i^{(\ell)}\big)$.

B. *Backward Pass (all N examples):*

For $i = 1 \ldots N$: $\quad$ Compute output-layer error: $\delta_i^{(L)} = \big(\mathbf{a}_i^{(L)} - \mathbf{y}_i\big) \; \circ \; \big[\mathbf{a}_i^{(L)} \circ (\mathbf{1} - \mathbf{a}_i^{(L)})\big]$.
For $\ell = L-1, \ldots, 1$: $\quad \delta_i^{(\ell)} = \Big[\big(\mathbf{W}^{(\ell+1)}\big)^T \, \delta_i^{(\ell+1)}\Big] \; \circ \; \big[\mathbf{a}_i^{(\ell)} \circ (\mathbf{1} - \mathbf{a}_i^{(\ell)})\big]$.

C. *Accumulate Gradients (all N examples):*

$$\text{Initialize } \frac{\partial C}{\partial \mathbf{W}^{(\ell)}} = \mathbf{0}, \quad \frac{\partial C}{\partial \mathbf{b}^{(\ell)}} = \mathbf{0}, \quad \ell = 1, \dots, L.$$

$$\text{For } i = 1 \dots N,\ \ell = 1 \dots L: \quad \frac{\partial C}{\partial \mathbf{W}^{(\ell)}} \mathrel{+}= \delta_i^{(\ell)} \left(\mathbf{a}_i^{(\ell-1)}\right)^T, \quad \frac{\partial C}{\partial \mathbf{b}^{(\ell)}} \mathrel{+}= \delta_i^{(\ell)}.$$

$$\text{Then divide each by } N: \quad \frac{\partial C}{\partial \mathbf{W}^{(\ell)}} \leftarrow \frac{1}{N} \frac{\partial C}{\partial \mathbf{W}^{(\ell)}}, \quad \frac{\partial C}{\partial \mathbf{b}^{(\ell)}} \leftarrow \frac{1}{N} \frac{\partial C}{\partial \mathbf{b}^{(\ell)}}.$$

After these steps, $\partial C/\partial \mathbf{W}^{(\ell)}$ and $\partial C/\partial \mathbf{b}^{(\ell)}$ are ready for a gradient-descent update:

$$\mathbf{W}^{(\ell)} \leftarrow \mathbf{W}^{(\ell)} - \eta \frac{\partial C}{\partial \mathbf{W}^{(\ell)}}, \quad \mathbf{b}^{(\ell)} \leftarrow \mathbf{b}^{(\ell)} - \eta \frac{\partial C}{\partial \mathbf{b}^{(\ell)}}, \quad \ell = 1, \dots, L,$$

where $\eta > 0$ is the learning rate.

6.5 Interpretations and Dimensional Checks

1. Interpretation of $\delta_i^{(\ell)}$.

$$\delta_i^{(\ell)} = \frac{\partial L_i}{\partial \mathbf{z}_i^{(\ell)}} \in \mathbb{R}^{m_\ell}.$$

Each component $\delta_{i,k}^{(\ell)}$ measures how much a small change in the pre-activation $z_{i,k}^{(\ell)}$ of neuron k in layer ℓ affects the single-example loss L_i. Concretely:

$$\delta_{i,k}^{(\ell)} = \sum_{j=1}^{m_{\ell+1}} \frac{\partial L_i}{\partial z_{i,j}^{(\ell+1)}} \frac{\partial z_{i,j}^{(\ell+1)}}{\partial a_{i,k}^{(\ell)}} \frac{\partial a_{i,k}^{(\ell)}}{\partial z_{i,k}^{(\ell)}},$$

which collapses to $\left(\mathbf{W}^{(\ell+1)}\right)_{j,k} \delta_{i,j}^{(\ell+1)} \times \sigma'(z_{i,k}^{(\ell)})$, summed over j. In vector form, this is exactly equation(59).

2. Dimensional Consistency.

$$\delta_i^{(\ell)} \in \mathbb{R}^{m_\ell}, \quad \mathbf{a}_i^{(\ell-1)} \in \mathbb{R}^{m_{\ell-1}},$$
$$\delta_i^{(\ell)} \left(\mathbf{a}_i^{(\ell-1)}\right)^T \in \mathbb{R}^{m_\ell \times m_{\ell-1}}, \quad \text{matches } \mathbf{W}^{(\ell)} \in \mathbb{R}^{m_\ell \times m_{\ell-1}}.$$

Elementwise:

$$\left[\delta_i^{(\ell)} \left(\mathbf{a}_i^{(\ell-1)}\right)^T\right]_{k,j} = \delta_{i,k}^{(\ell)} a_{i,j}^{(\ell-1)}, \quad 1 \le k \le m_\ell,\ 1 \le j \le m_{\ell-1}.$$

Thus $\partial L_i/\partial \mathbf{W}^{(\ell)}$ has the correct shape.

3. Activation Derivative. For any layer ℓ, the derivative of $\mathbf{a}_i^{(\ell)} = \sigma(\mathbf{z}_i^{(\ell)})$ with respect to $\mathbf{z}_i^{(\ell)}$ is the vector:

$$\sigma'\big(\mathbf{z}_i^{(\ell)}\big) = \mathbf{a}_i^{(\ell)} \circ \big(\mathbf{1} - \mathbf{a}_i^{(\ell)}\big) \ \in \ \mathbb{R}^{\,m_\ell}.$$

We multiply this elementwise by $\big(\mathbf{W}^{(\ell+1)}\big)^T \delta_i^{(\ell+1)}$ to propagate errors backward.

4. Summary of Backpropagation in Matrix-Vector Form. Concise formulas for each example i:

$$\begin{aligned}
\mathbf{a}_i^{(0)} &= x_i, \\
\mathbf{z}_i^{(\ell)} &= \mathbf{W}^{(\ell)}\, \mathbf{a}_i^{(\ell-1)} + \mathbf{b}^{(\ell)}, \quad \mathbf{a}_i^{(\ell)} = \sigma\big(\mathbf{z}_i^{(\ell)}\big), \quad \ell = 1, \dots, L. \\
\delta_i^{(L)} &= \big(\mathbf{a}_i^{(L)} - \mathbf{y}_i\big) \ \circ \ \sigma'\big(\mathbf{z}_i^{(L)}\big), \\
\delta_i^{(\ell)} &= \Big[\big(\mathbf{W}^{(\ell+1)}\big)^T \delta_i^{(\ell+1)}\Big] \ \circ \ \sigma'\big(\mathbf{z}_i^{(\ell)}\big), \quad \ell = L-1, \dots, 1. \\
\frac{\partial L_i}{\partial \mathbf{W}^{(\ell)}} &= \delta_i^{(\ell)} \big(\mathbf{a}_i^{(\ell-1)}\big)^T, \quad \frac{\partial L_i}{\partial \mathbf{b}^{(\ell)}} = \delta_i^{(\ell)}, \quad \ell = 1, \dots, L.
\end{aligned}$$

Finally,

$$\frac{\partial C}{\partial \mathbf{W}^{(\ell)}} = \frac{1}{N} \sum_{i=1}^{N} \frac{\partial L_i}{\partial \mathbf{W}^{(\ell)}}, \quad \frac{\partial C}{\partial \mathbf{b}^{(\ell)}} = \frac{1}{N} \sum_{i=1}^{N} \frac{\partial L_i}{\partial \mathbf{b}^{(\ell)}}.$$

With these detailed, vectorized expressions, the network can be implemented in a highly scalable way for arbitrary n-dimensional inputs and L layers of arbitrary widths $m_1, m_2, \dots, m_L$. Each forward pass requires $O\big(\sum_{\ell=1}^{L} m_\ell\, m_{\ell-1}\big)$ operations, and each backward pass (to compute all $\delta_i^{(\ell)}$) also requires $O\big(\sum_{\ell=1}^{L} m_\ell\, m_{\ell-1}\big)$. Thus training scales linearly in the total number of parameters, making this derivation suitable for deep, wide networks with high-dimensional inputs.