Backpropagation Worked Example

Computing $\dfrac{\partial \mathcal{L}}{\partial w_{1,1}^{(1)}}$ in a [3, 2, 2, 1] Network

Bartosz Naskręcki
Adam Mickiewicz University, Poznań

Chapter 16 Supplement · Classical Foundations of Neural Networks

2026

Abstract

This handout develops the complete computation of a single partial derivative in a four-layer feed-forward neural network with architecture [3, 2, 2, 1]. Every algebraic step is shown explicitly and justified by the corresponding backpropagation theorem (BP1–BP4). A finite-difference check confirms the result to a relative error of 9.34×10^{-9}.

Contents

Notation Reference **1**

1 Network Architecture **1**
1.1 Parameters . . . 1
1.2 Active Paths . . . 2

2 The Backpropagation Theorems **2**

3 Forward Pass **4**
3.1 Layer 1 . . . 4
3.2 Layer 2 . . . 4
3.3 Layer 3 (Output) . . . 5
3.4 Loss . . . 5
3.5 Sigmoid Derivatives (for later use) . . . 5

4 Backward Pass **5**
4.1 Step 2a: Output Error $\delta^{(3)}$ BP1 . . . 6
4.2 Step 2b: Hidden Error $\delta^{(2)}$ BP2 . . . 6
4.3 Step 2c: Hidden Error $\delta^{(1)}$ BP2 . . . 6
4.4 Step 2d: The Target Gradient BP3 . . . 7

5 Numerical Verification **8**

6 Summary **8**
6.1 The Complete Chain of Derivatives . . . 8
6.2 Reference Table . . . 9
6.3 Quick-Reference Algorithm . . . 9

7 Exercises **9**

A Derivation Sketches for BP1 and BP2 **10**
A.1 BP1 11
A.2 BP2 11

Notation Reference

The following conventions are used consistently throughout this handout and the companion course materials.

Table 1: Symbol glossary.

Symbol	Meaning
Indices & scalars	
$w_{j,k}^{(l)}$	Weight *from* neuron k in layer $l-1$ *to* neuron j in layer l. First subscript = destination (row of $\mathbf{W}$), second = source (column).
$a_j^{(l)}$	Activation (post-σ) of neuron j in layer l: $a_j^{(l)} = \sigma(z_j^{(l)})$.
$z_j^{(l)}$	Pre-activation (weighted sum + bias) of neuron j in layer l.
$\delta_j^{(l)}$	Error signal: $\delta_j^{(l)} = \partial\mathcal{L}/\partial z_j^{(l)}$. Measures how the loss changes with the pre-activation.
L	Index of the last (output) layer. For $[3, 2, 2, 1]$, $L = 3$.
n_l	Number of neurons in layer l.
Vectors, matrices & operators	
$\mathbf{W}^{(l)}$	Weight matrix for layer l, size $n_l \times n_{l-1}$. **Bold** = matrix or vector.
$\boldsymbol{\delta}^{(l)}$	Error vector for layer l (all n_l error signals stacked).
$\odot$	Hadamard (element-wise) product: $[\mathbf{u} \odot \mathbf{v}]_i = u_i\, v_i$.
$(\cdot)^T$	Matrix transpose. In BP2, $(\mathbf{W}^{(l+1)})^T$ propagates errors backward.
$[\boldsymbol{\delta}^{(l)}]_j$	Bracket notation: the j-th scalar component of the vector $\boldsymbol{\delta}^{(l)}$.
σ, σ'	Sigmoid activation and its derivative: $\sigma'(t) = \sigma(t)(1 - \sigma(t))$.
$\mathcal{L}$	Loss function (MSE): $\mathcal{L} = \frac{1}{2}(a^{(L)} - y)^2$.
Colour code	
orange	Input quantities (x_k).
blue	Weights ($w_{j,k}^{(l)}$).
green	Activations and σ' derivatives ($a_j^{(l)}$).
red	Error signals and gradients ($\delta_j^{(l)}$, $\partial\mathcal{L}/\partial w$).
Superscript & subscript conventions	
(l)	Always a layer index (parenthesised to distinguish from exponents).
j, k	Two subscripts on a weight: j = destination row, k = source column.

1 Network Architecture

We study a feed-forward network with four layers: an input layer of three neurons, two hidden layers of two neurons each, and a single output neuron. The *sigmoid* function

$\sigma(t) = \frac{1}{1+e^{-t}}$ is used at every non-input layer. The loss is the mean-squared-error (MSE) for a single sample:

$$\mathcal{L} = \tfrac{1}{2}\big(a^{(3)} - y\big)^2.$$

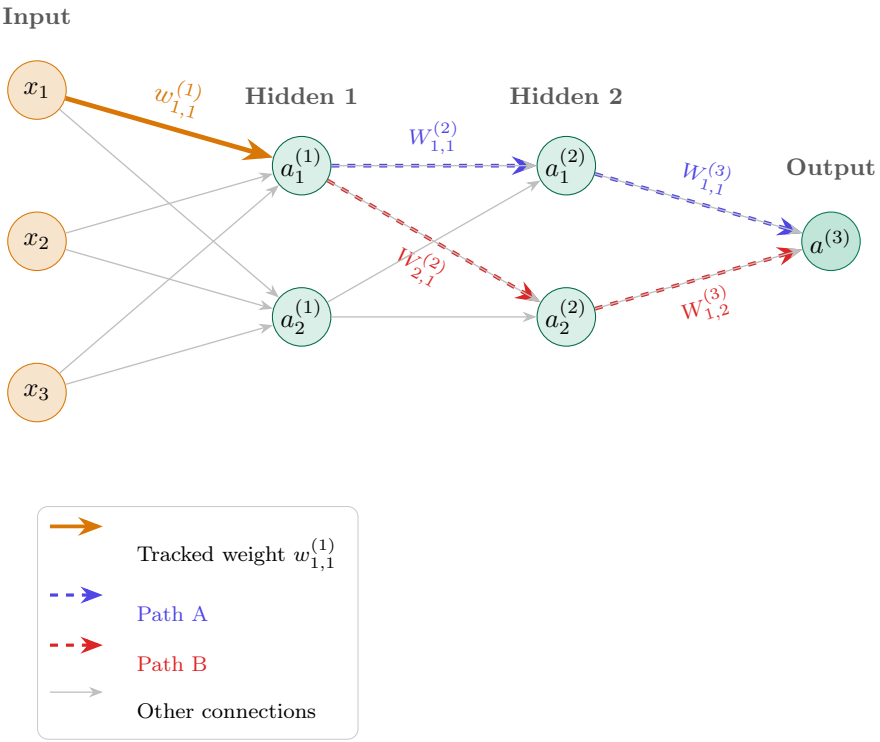

Figure 1: The [3, 2, 2, 1] network. The two dashed paths show every route through which $w^{(1)}_{1,1}$ influences the loss $\mathcal{L}$.

1.1 Parameters

Table 2: Network parameters.

Layer	Weights $W^{(l)}$	Biases $b^{(l)}$
$l = 1$	$W^{(1)} = \begin{pmatrix} 0.1 & 0.2 & 0.3 \\ 0.4 & 0.5 & 0.6 \end{pmatrix}$	$b^{(1)} = \begin{pmatrix} 0.1 \\ 0.2 \end{pmatrix}$
$l = 2$	$W^{(2)} = \begin{pmatrix} 0.7 & 0.8 \\ 0.9 & 1.0 \end{pmatrix}$	$b^{(2)} = \begin{pmatrix} 0.1 \\ 0.2 \end{pmatrix}$
$l = 3$	$W^{(3)} = \begin{pmatrix} 0.3 & 0.4 \end{pmatrix}$	$b^{(3)} = \begin{pmatrix} 0.1 \end{pmatrix}$

The input vector and target are

$$x = \begin{pmatrix} 1.0 \\ 0.5 \\ -1.0 \end{pmatrix}, \qquad y = 1.0.$$

1.2 Active Paths

The weight $w^{(1)}_{1,1}$ connects x_1 to $a^{(1)}_1$. From $a^{(1)}_1$, information flows to the output through *two* routes:

The *multiv*
chain rule t
that each p
contributes
additive ter
total deriva

Path A $x_1 \xrightarrow{w^{(1)}_{1,1}} a^{(1)}_1 \xrightarrow{W^{(2)}_{1,1}} a^{(2)}_1 \xrightarrow{W^{(3)}_{1,1}} a^{(3)} \longrightarrow \mathcal{L}$

Path B $x_1 \xrightarrow{w^{(1)}_{1,1}} a^{(1)}_1 \xrightarrow{W^{(2)}_{2,1}} a^{(2)}_2 \xrightarrow{W^{(3)}_{1,2}} a^{(3)} \longrightarrow \mathcal{L}$

Backpropagation automatically sums the contributions of both paths via the matrix–vector products in BP2.

2 The Backpropagation Theorems

The four theorems below are stated for a network with L layers, sigmoid activation σ, and MSE loss $\mathcal{L} = \frac{1}{2}\|a^{(L)} - y\|^2$.

Theorem BP1 — Output Error

$$\delta^{(L)} = \nabla_a \mathcal{L} \odot \sigma'\big(z^{(L)}\big) \tag{1}$$

For MSE loss with a single output: $\nabla_a \mathcal{L} = \big(a^{(L)} - y\big)$.

Theorem BP2 — Error Propagation

$$\delta^{(l)} = \Big(\big(W^{(l+1)}\big)^\top \delta^{(l+1)}\Big) \odot \sigma'\big(z^{(l)}\big) \tag{2}$$

Theorem BP3 — Weight Gradient

$$\frac{\partial \mathcal{L}}{\partial w^{(l)}_{jk}} = a^{(l-1)}_k \cdot \delta^{(l)}_j \tag{3}$$

Theorem BP4 — Bias Gradient

$$\frac{\partial \mathcal{L}}{\partial b^{(l)}_j} = \delta^{(l)}_j \tag{4}$$

Recall. The sigmoid derivative satisfies the elegant identity $\sigma'(t) = \sigma(t)\big(1 - \sigma(t)\big)$, which we use repeatedly.

3 Forward Pass

We propagate the input x through the network, layer by layer, recording every intermediate quantity.

3.1 Layer 1

Pre-activations. $z^{(1)} = W^{(1)}\, x + b^{(1)}$.

$z^{(1)}$ and $a^{(1)}$ are reused in Steps 4.3 and 4.4.

$$\begin{aligned} z^{(1)}_1 &= 0.1 \cdot 1.0 + 0.2 \cdot 0.5 + 0.3 \cdot (-1.0) + 0.1 \\ &= 0.1 + 0.1 - 0.3 + 0.1 = 0.0 \end{aligned} \tag{5}$$

$$\begin{aligned} z^{(1)}_2 &= 0.4 \cdot 1.0 + 0.5 \cdot 0.5 + 0.6 \cdot (-1.0) + 0.2 \\ &= 0.4 + 0.25 - 0.6 + 0.2 = 0.25 \end{aligned} \tag{6}$$

$$\boxed{z^{(1)} = \begin{pmatrix} 0.0 \\ 0.25 \end{pmatrix}} \tag{7}$$

Activations. $a^{(l)} = \sigma(z^{(l)})$. Since $\sigma(0) = 0.5$ exactly:

$$a_1^{(1)} = \sigma(0.0) = 0.5 \tag{8}$$

$$a_2^{(1)} = \sigma(0.25) = \frac{1}{1 + e^{-0.25}} = 0.5622 \tag{9}$$

$$\boxed{a^{(1)} = \begin{pmatrix} 0.5 \\ 0.5622 \end{pmatrix}} \tag{10}$$

3.2 Layer 2

Pre-activations. $z^{(2)} = W^{(2)}\, a^{(1)} + b^{(2)}$.

$z^{(2)}$ and $a^{(2)}$ are reused in Step 4.2.

$$\begin{aligned} z_1^{(2)} &= 0.7 \cdot 0.5 + 0.8 \cdot 0.5622 + 0.1 \\ &= 0.35 + 0.4498 + 0.1 = 0.8998 \approx 0.8997 \end{aligned} \tag{11}$$

$$\begin{aligned} z_2^{(2)} &= 0.9 \cdot 0.5 + 1.0 \cdot 0.5622 + 0.2 \\ &= 0.45 + 0.5622 + 0.2 = 1.2122 \end{aligned} \tag{12}$$

$$\boxed{z^{(2)} = \begin{pmatrix} 0.8997 \\ 1.2122 \end{pmatrix}} \tag{13}$$

Activations.

$$a_1^{(2)} = \sigma(0.8997) = 0.7109 \tag{14}$$

$$a_2^{(2)} = \sigma(1.2122) = 0.7707 \tag{15}$$

$$\boxed{a^{(2)} = \begin{pmatrix} 0.7109 \\ 0.7707 \end{pmatrix}} \tag{16}$$

3.3 Layer 3 (Output)

Pre-activation. $z^{(3)} = W^{(3)}\, a^{(2)} + b^{(3)}$.

$z^{(3)}$ and $a^{(3)}$ BP1 in Step

$$\begin{aligned} z^{(3)} &= 0.3 \cdot 0.7109 + 0.4 \cdot 0.7707 + 0.1 \\ &= 0.2133 + 0.3083 + 0.1 = 0.6216 \approx 0.6215 \end{aligned} \tag{17}$$

$$\boxed{z^{(3)} = 0.6215} \tag{18}$$

Activation.

$$a^{(3)} = \sigma(0.6215) = 0.6506 \tag{19}$$

$$\boxed{a^{(3)} = 0.6506} \tag{20}$$

3.4 Loss

$$\begin{aligned}\mathcal{L} &= \tfrac{1}{2}\big(a^{(3)} - y\big)^2 = \tfrac{1}{2}\big(0.6506 - 1.0\big)^2 \\ &= \tfrac{1}{2}\,(-0.3494)^2 = \tfrac{1}{2}\,(0.1221) = 0.0611 \end{aligned} \tag{21}$$

$$\boxed{\mathcal{L} = 0.0611} \tag{22}$$

3.5 Sigmoid Derivatives (for later use)

Using $\sigma'(t) = \sigma(t)(1 - \sigma(t))$:

$$\sigma'(z_1^{(1)}) = 0.5\,(1 - 0.5) \qquad = 0.2500 \tag{23}$$
$$\sigma'(z_2^{(1)}) = 0.5622\,(1 - 0.5622) = 0.2461 \tag{24}$$
$$\sigma'(z_1^{(2)}) = 0.7109\,(1 - 0.7109) = 0.2055 \tag{25}$$
$$\sigma'(z_2^{(2)}) = 0.7707\,(1 - 0.7707) = 0.1767 \tag{26}$$
$$\sigma'(z^{(3)}) = 0.6506\,(1 - 0.6506) = 0.2273 \tag{27}$$

4 Backward Pass

We now propagate errors from the output back to the input, computing every $\delta^{(l)}$ vector and finally the target gradient.

4.1 Step 2a: Output Error $\delta^{(3)}$ BP1

Apply Theorem BP1 (Eq. 1):

$$\begin{aligned}\delta^{(3)} &= \big(a^{(3)} - y\big) \;\cdot\; \sigma'\big(z^{(3)}\big) \\ &= \big(0.6506 - 1.0\big) \;\cdot\; 0.2273 \\ &= (-0.3494) \;\cdot\; (0.2273) \\ &= \boxed{-0.0794}\end{aligned} \tag{BP1}$$

$$\tag{28}$$

Negative sign: the output is *below* the target, so the loss decreases if activations increase.

4.2 Step 2b: Hidden Error $\delta^{(2)}$ BP2

Apply Theorem BP2 (Eq. 2) with $l = 2$:

$$\delta^{(2)} = \Big(\big(W^{(3)}\big)^{\top}\,\delta^{(3)}\Big) \;\odot\; \sigma'\big(z^{(2)}\big).$$

Step (i): transpose–multiply.

$$\begin{aligned}\big(W^{(3)}\big)^{\top}\,\delta^{(3)} &= \begin{pmatrix}0.3\\0.4\end{pmatrix} \cdot (-0.0794) \\ &= \begin{pmatrix}0.3 \times (-0.0794)\\0.4 \times (-0.0794)\end{pmatrix} = \begin{pmatrix}-0.0238\\-0.0318\end{pmatrix}\end{aligned} \tag{29}$$

Step (ii): Hadamard product.

$$\delta^{(2)} = \begin{pmatrix} -0.0238 \\ -0.0318 \end{pmatrix} \odot \begin{pmatrix} 0.2055 \\ 0.1767 \end{pmatrix}$$
$$= \begin{pmatrix} -0.0238 \times 0.2055 \\ -0.0318 \times 0.1767 \end{pmatrix}$$
$$= \boxed{\delta^{(2)} = \begin{pmatrix} -0.00490 \\ -0.00562 \end{pmatrix}} \tag{30}$$

4.3 Step 2c: Hidden Error $\delta^{(1)}$

BP2

Apply BP2 again with $l = 1$:

$$\delta^{(1)} = \left(\left(W^{(2)}\right)^{\top} \delta^{(2)} \right) \odot \sigma'\left(z^{(1)}\right).$$

Step (i): transpose–multiply.

$$\left(W^{(2)}\right)^{\top} = \begin{pmatrix} 0.7 & 0.9 \\ 0.8 & 1.0 \end{pmatrix} \tag{31}$$

$$\left(W^{(2)}\right)^{\top} \delta^{(2)} = \begin{pmatrix} 0.7 \times (-0.00490) + 0.9 \times (-0.00562) \\ 0.8 \times (-0.00490) + 1.0 \times (-0.00562) \end{pmatrix}$$
$$= \begin{pmatrix} -0.00343 - 0.00506 \\ -0.00392 - 0.00562 \end{pmatrix} = \begin{pmatrix} -0.00849 \\ -0.00954 \end{pmatrix} \tag{32}$$

Step (ii): Hadamard product.

$$\delta^{(1)} = \begin{pmatrix} -0.00849 \\ -0.00954 \end{pmatrix} \odot \begin{pmatrix} 0.2500 \\ 0.2461 \end{pmatrix}$$
$$= \begin{pmatrix} -0.00849 \times 0.25 \\ -0.00954 \times 0.2461 \end{pmatrix}$$
$$= \boxed{\delta^{(1)} = \begin{pmatrix} -0.00212 \\ -0.00235 \end{pmatrix}} \tag{33}$$

Key observation

The quantity $\delta_1^{(1)} = -0.00212$ is precisely the error signal that has been back-propagated through *both* paths (A and B) from the output down to the first neuron of hidden layer 1. BP2 automatically summed the two path contributions via the matrix multiplication in Eq. (32).

4.4 Step 2d: The Target Gradient

BP3

Apply Theorem BP3 (Eq. 3) with $l = 1$, $j = 1$, $k = 1$. The "activation from the previous layer" is the input itself: $a_1^{(0)} = x_1 = 1.0$.

$$\frac{\partial \mathcal{L}}{\partial w_{1,1}^{(1)}} = \delta_1^{(1)} \cdot a_1^{(0)} = (-0.00212) \cdot 1.0 \tag{34}$$

$$\boxed{\frac{\partial \mathcal{L}}{\partial w_{1,1}^{(1)}} = -0.00212} \tag{35}$$

Interpretation. The negative sign tells us that *increasing* $w_{1,1}^{(1)}$ will *decrease* the loss. This is consistent with our observation that the network's output (0.6506) is below the target (1.0): the gradient points in the direction that pushes the output upward.

The magnitude $|\partial \mathcal{L}/\partial w_{1,1}^{(1)}| = 0.00212$ is small because the signal must traverse three layers of sigmoid squashing, each of which attenuates the gradient (the "vanishing gradient" effect in embryonic form).

5 Numerical Verification

We verify the analytic gradient by a centred finite difference. Define the perturbed loss

$$\mathcal{L}^{\pm} = \mathcal{L}\Big(w_{1,1}^{(1)} \pm \varepsilon\Big)$$

where only $w_{1,1}^{(1)}$ is perturbed and all other parameters remain fixed. The finite-difference approximation is

$$\frac{\partial \mathcal{L}}{\partial w_{1,1}^{(1)}} \approx \frac{\mathcal{L}^{+} - \mathcal{L}^{-}}{2\,\varepsilon}. \tag{36}$$

Choosing $\varepsilon = 10^{-5}$, a full forward pass with $w_{1,1}^{(1)} = 0.1 + 10^{-5}$ and another with $w_{1,1}^{(1)} = 0.1 - 10^{-5}$ yield:

Quantity	Value
$\mathcal{L}^{+}$	$0.06108938\ldots$
$\mathcal{L}^{-}$	$0.06109363\ldots$
Finite difference	$-0.002124\ldots$
Analytic (BP)	-0.00212
Relative error $\dfrac{\lvert \text{FD} - \text{BP} \rvert}{\lvert \text{BP} \rvert}$	$= 9.34 \times 10^{-9}$

Verification Passed ✓

The relative error between the analytic gradient and the centred finite difference is 9.34×10^{-9}, well below the typical threshold of 10^{-5}. This confirms that our backward-pass computation is correct.

Why centred differences? The centred formula (36) has error $O(\varepsilon^2)$, compared to $O(\varepsilon)$ for the one-sided difference. With $\varepsilon = 10^{-5}$ the truncation error is of order 10^{-10}, leaving floating-point round-off as the dominant error source.

6 Summary

6.1 The Complete Chain of Derivatives

The total derivative decomposes as a sum over the two active paths. Writing $s_j^{(l)} = \sigma'(z_j^{(l)})$ for brevity:

$$\frac{\partial \mathcal{L}}{\partial w_{1,1}^{(1)}} = \underbrace{\frac{\partial \mathcal{L}}{\partial a^{(3)}}}_{a^{(3)}-y} \cdot s^{(3)} \cdot \Big[\underbrace{W_{1,1}^{(3)}\, s_1^{(2)}\, W_{1,1}^{(2)}}_{\text{Path A}} + \underbrace{W_{1,2}^{(3)}\, s_2^{(2)}\, W_{2,1}^{(2)}}_{\text{Path B}} \Big] \cdot s_1^{(1)} \cdot x_1 \tag{37}$$

Substituting numerical values:

$$\begin{aligned}
&= (-0.3494) \times (0.2273) \times \big[0.3 \times 0.2055 \times 0.7 \;+\; 0.4 \times 0.1767 \times 0.9\big] \times 0.25 \times 1.0 \\
&= (-0.3494) \times (0.2273) \times \big[0.04316 + 0.06361\big] \times 0.25 \\
&= (-0.3494) \times (0.2273) \times (0.10677) \times (0.25) \\
&= -0.00212 \;\checkmark
\end{aligned}$$

6.2 Reference Table

Table 3: All intermediate values.

Layer	z_1	z_2	a_1	a_2	σ'
$l = 0$ (input)	—	—	1.0	0.5 / −1.0	—
$l = 1$	0.0	0.25	0.5	0.5622	0.2500 / 0.2461
$l = 2$	0.8997	1.2122	0.7109	0.7707	0.2055 / 0.1767
$l = 3$	0.6215	—	0.6506	—	0.2273
	δ_1		δ_2		
$l = 3$	−0.0794		—		
$l = 2$	−0.00490		−0.00562		
$l = 1$	−0.00212		−0.00235		

6.3 Quick-Reference Algorithm

Backpropagation — Algorithmic Summary

1. **Forward pass:** Compute all $z^{(l)} = W^{(l)}\, a^{(l-1)} + b^{(l)}$ and $a^{(l)} = \sigma(z^{(l)})$ for $l = 1, \ldots, L$.
2. **BP1:** Compute the output error $\delta^{(L)} = \nabla_a \mathcal{L} \odot \sigma'(z^{(L)})$.
3. **BP2:** For $l = L-1, L-2, \ldots, 1$, propagate $\delta^{(l)} = \left((W^{(l+1)})^\top \delta^{(l+1)}\right) \odot \sigma'(z^{(l)})$.
4. **BP3:** $\dfrac{\partial \mathcal{L}}{\partial w_{jk}^{(l)}} = a_k^{(l-1)}\, \delta_j^{(l)}$.
5. **BP4:** $\dfrac{\partial \mathcal{L}}{\partial b_j^{(l)}} = \delta_j^{(l)}$.

7 Exercises

Exercise 1. BP4 Application. Using the $\delta^{(l)}$ values computed in Section 4, calculate *all* bias gradients $\dfrac{\partial \mathcal{L}}{\partial b_j^{(l)}}$ for every layer $l = 1, 2, 3$ and every neuron j in that layer.

Hint: By BP4, each bias gradient equals the corresponding δ component. Organise your answer as three vectors.

Exercise 2. Another Weight. Compute $\dfrac{\partial \mathcal{L}}{\partial w_{2,3}^{(1)}}$, the gradient with respect to the weight from x_3 to the *second* neuron in hidden layer 1.

(a) Which path(s) connect $w_{2,3}^{(1)}$ to the output? Draw them on a copy of Figure 1.

(b) Apply BP3. Compare the sign and magnitude with $\partial \mathcal{L}/\partial w_{1,1}^{(1)}$. Explain any differences.

Exercise 3. Layer 2 Weight. Compute $\dfrac{\partial \mathcal{L}}{\partial w_{1,2}^{(2)}}$.

(a) Identify the unique active path from $w_{1,2}^{(2)}$ to the loss.

(b) Apply BP3 at layer $l = 2$. You may reuse the $\delta^{(2)}$ values from Section 4.

(c) Is this gradient larger or smaller in magnitude than the layer-1 gradient? Explain why in terms of the number of sigmoid squashings.

Exercise 4. Full Gradient Matrix. Write out the complete gradient matrix $\dfrac{\partial \mathcal{L}}{\partial W^{(1)}} \in \mathbb{R}^{2\times 3}$.

(a) Compute each of the six entries using BP3:

$$\left(\frac{\partial \mathcal{L}}{\partial W^{(1)}}\right)_{jk} = x_k \cdot \delta_j^{(1)}, \qquad j \in \{1,2\}, \quad k \in \{1,2,3\}.$$

(b) Observe that the matrix can be written as the outer product $\delta^{(1)}\, x^\top$. Verify this factorisation.

(c) Verify one entry by finite differences ($\varepsilon = 10^{-5}$).

Exercise 5. Weight Update and Loss Reduction. Suppose the learning rate is $\eta = 0.5$.

(a) Compute the updated weight matrices for all three layers:

$$W^{(l)}_{\text{new}} = W^{(l)} - \eta\, \frac{\partial \mathcal{L}}{\partial W^{(l)}}, \qquad l = 1,2,3.$$

(You will also need the δ vectors for layers 2 and 3. For $l = 3$, note that $\partial \mathcal{L}/\partial W^{(3)} = \delta^{(3)}\,(a^{(2)})^\top$.)

(b) Similarly update all biases using BP4.

(c) Perform a complete forward pass with the updated parameters and compute the new loss $\mathcal{L}_{\text{new}}$.

(d) Verify that $\mathcal{L}_{\text{new}} < \mathcal{L}$. By how much did the loss decrease?

A Derivation Sketches for BP1 and BP2

For completeness we outline the proofs. Full details are in Chapter 16 of the lecture notes.

A.1 BP1

By definition, $\delta_j^{(L)} = \partial \mathcal{L}/\partial z_j^{(L)}$. Since $a_j^{(L)} = \sigma(z_j^{(L)})$, the chain rule gives

$$\delta_j^{(L)} = \frac{\partial \mathcal{L}}{\partial a_j^{(L)}} \cdot \sigma'(z_j^{(L)}).$$

For MSE loss, $\partial \mathcal{L}/\partial a_j^{(L)} = a_j^{(L)} - y_j$. Writing this component-wise relation in vector form, with the Hadamard product replacing scalar multiplication, gives

$$\delta^{(L)} = \nabla_a \mathcal{L} \odot \sigma'(z^{(L)}). \qquad \square$$

A.2 BP2

We have $z_j^{(l+1)} = \sum_k w_{jk}^{(l+1)}\, a_k^{(l)} + b_j^{(l+1)}$ and $a_k^{(l)} = \sigma(z_k^{(l)})$. Hence

$$\frac{\partial z_j^{(l+1)}}{\partial z_k^{(l)}} = w_{jk}^{(l+1)}\, \sigma'(z_k^{(l)}).$$

By the chain rule,

$$\begin{aligned}\delta_k^{(l)} = \frac{\partial \mathcal{L}}{\partial z_k^{(l)}} &= \sum_j \frac{\partial \mathcal{L}}{\partial z_j^{(l+1)}} \cdot \frac{\partial z_j^{(l+1)}}{\partial z_k^{(l)}} \\ &= \sum_j \delta_j^{(l+1)}\, w_{jk}^{(l+1)}\, \sigma'(z_k^{(l)}) \\ &= \Big(\sum_j w_{jk}^{(l+1)}\, \delta_j^{(l+1)}\Big)\sigma'(z_k^{(l)}).\end{aligned}$$

The sum is the k-th component of $(W^{(l+1)})^\top \delta^{(l+1)}$, so in vector form

$$\delta^{(l)} = \big((W^{(l+1)})^\top \delta^{(l+1)}\big) \odot \sigma'(z^{(l)}). \qquad \square$$

End of handout.