

The Chain Rule and the Idea of Backpropagation

Group 2 – Ferschl Martin, Reiter Roman, Zenkic Mirza

December 12, 2025

1 Task specification

Let $f(x)$ be a function that depends on two parameters a and b . More precisely, we write

$$f = f(x; a, b),$$

so f is a function in three variables. Furthermore, let

$$F(x) = f(f(x + a; a, b) + b; a, b).$$

The exercise asks us to:

1. Compute the partial derivatives $F_a = \frac{\partial F}{\partial a}$ and $F_b = \frac{\partial F}{\partial b}$ in terms of the corresponding partial derivatives of f (using the chain rule).
2. Observe that the expression for F_b also appears in F_a , so that it is natural to compute F_b first and reuse intermediate results.
3. Write the derivation down precisely and explain the procedure (backpropagation viewpoint).

2 Setup and notation

To apply the chain rule systematically, we introduce intermediate variables describing the “layers” of the composition. This is exactly the computational graph perspective used in backpropagation.

We regard f as a function of three arguments, and we denote its partial derivatives by

$$f_x(x; a, b) := \frac{\partial f}{\partial x}(x; a, b), \quad f_a(x; a, b) := \frac{\partial f}{\partial a}(x; a, b), \quad f_b(x; a, b) := \frac{\partial f}{\partial b}(x; a, b).$$

Now introduce the following intermediate quantities:

$$\begin{aligned} z &= x + a, \\ y_{\text{inner}} &= f(z; a, b), \\ y_{\text{outer}} &= y_{\text{inner}} + b = f(z; a, b) + b, \\ F(x) &= f(y_{\text{outer}}; a, b). \end{aligned}$$

Thus the computational graph is

$$x \xrightarrow{+a} z \xrightarrow{f(\cdot; a, b)} y_{\text{inner}} \xrightarrow{+b} y_{\text{outer}} \xrightarrow{f(\cdot; a, b)} F.$$

We will compute the gradients by starting from the output (outermost layer) and propagating derivatives backwards.

3 Computation of F_b

We first compute the partial derivative of F with respect to b . The parameter b influences F through three distinct paths:

1. directly as a parameter of the outer f ,
2. explicitly through the term $+b$ in $y_{\text{outer}} = y_{\text{inner}} + b$,
3. implicitly as a parameter in the inner $f(z; a, b)$.

Applying the chain rule to the outermost function,

$$F(x) = f(y_{\text{outer}}; a, b),$$

gives

$$F_b = \frac{\partial F}{\partial b} = f_x(y_{\text{outer}}; a, b) \cdot \frac{\partial y_{\text{outer}}}{\partial b} + f_b(y_{\text{outer}}; a, b).$$

We now compute $\frac{\partial y_{\text{outer}}}{\partial b}$. Since

$$y_{\text{outer}} = f(z; a, b) + b$$

with $z = x + a$ independent of b , we obtain

$$\frac{\partial y_{\text{outer}}}{\partial b} = \frac{\partial}{\partial b}(f(z; a, b)) + \frac{\partial b}{\partial b} = f_b(z; a, b) + 1.$$

Substituting back, we obtain

$$F_b = f_x(y_{\text{outer}}; a, b) (f_b(z; a, b) + 1) + f_b(y_{\text{outer}}; a, b).$$

This already has the structure typical for backpropagation: the factor $f_x(y_{\text{outer}}; a, b)$ is the “sensitivity” of the output with respect to its input, and it will be reused in the derivative with respect to other parameters.

4 Computation of F_a

Next we compute $F_a = \frac{\partial F}{\partial a}$. The parameter a influences F through:

1. directly as a parameter of the outer f ,
2. implicitly through its role in the inner function $f(z; a, b)$,
3. implicitly via $z = x + a$.

Again apply the chain rule at the outermost level:

$$F_a = \frac{\partial F}{\partial a} = f_x(y_{\text{outer}}; a, b) \cdot \frac{\partial y_{\text{outer}}}{\partial a} + f_a(y_{\text{outer}}; a, b).$$

We already know that the factor $f_x(y_{\text{outer}}; a, b)$ appears in F_b , so we do not need to recompute it. It is reused here, exactly as in backpropagation.

Now we compute $\frac{\partial y_{\text{outer}}}{\partial a}$. Since $y_{\text{outer}} = f(z; a, b) + b$ and b does not depend on a ,

$$\frac{\partial y_{\text{outer}}}{\partial a} = \frac{\partial}{\partial a} f(z; a, b).$$

Here we must apply the chain rule again, because $z = x + a$ depends on a :

$$\frac{\partial}{\partial a} f(z; a, b) = f_x(z; a, b) \cdot \frac{\partial z}{\partial a} + f_a(z; a, b).$$

Since $\frac{\partial z}{\partial a} = 1$, we obtain

$$\frac{\partial y_{\text{outer}}}{\partial a} = f_x(z; a, b) + f_a(z; a, b).$$

Substituting into the expression for F_a gives

$$F_a = f_x(y_{\text{outer}}; a, b) (f_x(z; a, b) + f_a(z; a, b)) + f_a(y_{\text{outer}}; a, b).$$

5 Summary and backpropagation viewpoint

For compactness, write

$$z = x + a, \quad y = f(z; a, b) + b$$

and, to avoid clutter, suppress the explicit parameters (a, b) in the notation of the derivatives (it is understood that all f_x, f_a, f_b are evaluated at the appropriate argument together with the same parameters (a, b)). Then we can summarise:

$$\begin{aligned} F_b(x; a, b) &= f_x(y) (f_b(z) + 1) + f_b(y), \\ F_a(x; a, b) &= f_x(y) (f_x(z) + f_a(z)) + f_a(y). \end{aligned}$$

The procedure illustrates the principle of *backpropagation*:

- We express the computation as a sequence of simple steps (a computational graph).
- We compute derivatives layer by layer, starting from the output and moving backwards.
- Intermediate “sensitivities” such as $f_x(y)$ are computed once and reused when differentiating with respect to different parameters (here both a and b).

This is exactly how gradient computation is implemented efficiently in neural networks.