We want to train an MLP by minimizing a loss over the training set
\[ \mathcal{L}(w, b) = \sum_n \ell(f(x_n; w, b), y_n). \]

To use gradient descent, we need the expression of the gradient of the per-sample loss \( \ell_n = \ell(f(x_n; w, b), y_n) \) with respect to the parameters, e.g.

\[ \frac{\partial \ell_n}{\partial w_{ij}^{(l)}} \quad \text{and} \quad \frac{\partial \ell_n}{\partial b_i^{(l)}}. \]
For clarity, we consider a single training sample $x$, and introduce $s^{(1)}, \ldots, s^{(L)}$ as the summations before activation functions.

$$
x^{(0)} = \mathbf{x} \xrightarrow{\mathbf{w}^{(1)}, b^{(1)}} s^{(1)} \xrightarrow{\sigma} x^{(1)} \xrightarrow{\mathbf{w}^{(2)}, b^{(2)}} s^{(2)} \xrightarrow{\sigma} \ldots \xrightarrow{\mathbf{w}^{(L)}, b^{(L)}} s^{(L)} \xrightarrow{\sigma} x^{(L)} = f(x; \mathbf{w}, \mathbf{b}).
$$

Formally we set $x^{(0)} = \mathbf{x}$,

$$
\forall l = 1, \ldots, L, \quad \begin{cases} 
    s^{(l)} = \mathbf{w}^{(l)}x^{(l-1)} + b^{(l)}, \\
    x^{(l)} = \sigma(s^{(l)}),
\end{cases}
$$

and we set the output of the network as $f(x; \mathbf{w}, \mathbf{b}) = x^{(L)}$.

This is the **forward pass**.

The core principle of the back-propagation algorithm is the “chain rule” from differential calculus:

$$(g \circ f)' = (g' \circ f)f'.$$

The linear approximation of a composition of mappings is the product of their individual linear approximations.

This generalizes to longer compositions and higher dimensions

$$
J_{f_N \circ f_{N-1} \circ \ldots \circ f_1}(x) = J_{f_N}(f_{N-1}(\ldots(x))) \ldots J_{f_3}(f_2(f_1(x))) J_{f_2}(f_1(x)) J_{f_1}(x)
$$

where $J_f(x)$ is the Jacobian of $f$ at $x$, that is the matrix of the linear approximation of $f$ in the neighborhood of $x$. 
\[ x^{(l-1)} \xrightarrow{w^{(l)},b^{(l)}} s^{(l)} \xrightarrow{\sigma} x^{(l)} \]

Since \( s_i^{(l)} \) influences \( \ell \) only through \( x_i^{(l)} \) with
\[ x_i^{(l)} = \sigma(s_i^{(l)}) , \]
we have
\[ \frac{\partial \ell}{\partial s_i^{(l)}} = \frac{\partial \ell}{\partial x_i^{(l)}} \frac{\partial x_i^{(l)}}{\partial s_i^{(l)}} = \frac{\partial \ell}{\partial x_i^{(l)}} \sigma'(s_i^{(l)}) , \]

And since \( x_j^{(l-1)} \) influences \( \ell \) only through the \( s_i^{(l)} \) with
\[ s_i^{(l)} = \sum_j w_{i,j} x_j^{(l-1)} + b_i^{(l)} , \]
we have
\[ \frac{\partial \ell}{\partial x_j^{(l-1)}} = \sum_i \frac{\partial \ell}{\partial s_i^{(l)}} \frac{\partial s_i^{(l)}}{\partial x_j^{(l-1)}} = \sum_i \frac{\partial \ell}{\partial s_i^{(l)}} w_{i,j}^{(l)} . \]
To summarize: we can compute $\frac{\partial \ell}{\partial x_i^{(l)}}$ from the definition of $\ell$, and recursively propagate backward the derivatives of the loss w.r.t the activations with

$$\frac{\partial \ell}{\partial s_i^{(l)}} = \frac{\partial \ell}{\partial x_j^{(l)}} \sigma'(s_i^{(l)})$$

and

$$\frac{\partial \ell}{\partial x_j^{(l-1)}} = \sum_i \frac{\partial \ell}{\partial s_i^{(l)}} w_{i,j}^{(l)}.$$

And then compute the derivatives w.r.t the parameters with

$$\frac{\partial \ell}{\partial w_{i,j}^{(l)}} = \frac{\partial \ell}{\partial s_i^{(l)}} x_j^{(l-1)},$$

and

$$\frac{\partial \ell}{\partial b_i^{(l)}} = \frac{\partial \ell}{\partial s_i^{(l)}}.$$

To write in tensorial form we will use a notation for the Jacobian to make explicit the variable w.r.t which the derivatives are computed. For $\psi : \mathbb{R}^N \to \mathbb{R}^M$,

$$\begin{bmatrix} \frac{\partial \psi}{\partial x_1} & \cdots & \frac{\partial \psi}{\partial x_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial \psi}{\partial x_1} & \cdots & \frac{\partial \psi}{\partial x_N} \end{bmatrix},$$

and if $\psi : \mathbb{R}^{N \times M} \to \mathbb{R}$, we will use the notation

$$\begin{bmatrix} \frac{\partial \psi}{\partial w_{1,1}} & \cdots & \frac{\partial \psi}{\partial w_{1,M}} \\ \vdots & \ddots & \vdots \\ \frac{\partial \psi}{\partial w_{N,1}} & \cdots & \frac{\partial \psi}{\partial w_{N,M}} \end{bmatrix}.$$
Forward pass

Compute the activations.

\[ x^{(0)} = x, \quad \forall l = 1, \ldots, L, \quad \left\{ \begin{array}{l} s^{(l)} = w^{(l)} x^{(l-1)} + b^{(l)} \\ x^{(l)} = \sigma \left( s^{(l)} \right) \end{array} \right. \]

Backward pass

Compute the derivatives of the loss wrt the activations.

\[
\left\{ \begin{array}{l}
\frac{\partial \ell}{\partial x^{(L)}} 
\text{ from the definition of } \ell \\
\text{if } l < L, \quad \frac{\partial \ell}{\partial x^{(l)}} = \left( w^{(l+1)} \right)^\top \left[ \frac{\partial \ell}{\partial s^{(l+1)}} \right]
\end{array} \right.
\]

Compute the derivatives of the loss wrt the parameters.

\[
\left[ \frac{\partial \ell}{\partial w^{(l)}} \right] = \left[ \frac{\partial \ell}{\partial s^{(l)}} \right] \left( x^{(l-1)} \right)^\top \\
\left[ \frac{\partial \ell}{\partial b^{(l)}} \right] = \left[ \frac{\partial \ell}{\partial s^{(l)}} \right].
\]

Gradient step

Update the parameters.

\[
w^{(l)} \leftarrow w^{(l)} - \eta \left[ \frac{\partial \ell}{\partial w^{(l)}} \right] \\
b^{(l)} \leftarrow b^{(l)} - \eta \left[ \frac{\partial \ell}{\partial b^{(l)}} \right]
\]
In spite of its hairy formalization, the backward pass is a simple algorithm: apply the chain rule again and again.

As for the forward pass, it can be expressed in tensorial form. Heavy computation is concentrated in linear operations, and all the non-linearities go into component-wise operations.

Regarding computation, since the costly operation for the forward pass is

\[ s^{(l)} = w^{(l)}x^{(l-1)} + b^{(l)} \]

and for the backward

\[
\begin{bmatrix}
\frac{\partial \ell}{\partial x^{(l)}}
\end{bmatrix} = \left( w^{(l+1)} \right)^\top \begin{bmatrix}
\frac{\partial \ell}{\partial s^{(l+1)}}
\end{bmatrix}
\]

and

\[
\begin{bmatrix}
\frac{\partial \ell}{\partial w^{(l)}}
\end{bmatrix} = \begin{bmatrix}
\frac{\partial \ell}{\partial s^{(l)}}
\end{bmatrix} \left( x^{(l-1)} \right)^\top,
\]

the rule of thumb is that the backward pass is twice more expensive than the forward one.