Deep learning

3.6. Back-propagation

François Fleuret

https://fleuret.org/dlc/

UNIVERSITÉ DE GENÈVE

We want to train an MLP by minimizing a loss over the training set

$$\mathscr{L}(w, b) = \sum_n \ell(f(x_n; w, b), y_n).$$

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.

$$\frac{\partial \ell_n}{\partial w_{i,j}^{(l)}} \text{ and } \frac{\partial \ell_n}{\partial b_i^{(l)}}.$$

Notes

We saw in 3.5. "Gradient descent" that, as soon as we can compute the gradient of the loss w.r.t. the parameters, we can optimize them through gradient descent.
What we are missing here is, how to compute the derivative of the loss w.r.t. the parameters of a multi-layer perceptron.
The total loss $\mathscr{L}$ is the sum over all the samples of the individual losses $\ell(f(x_n; w, b), y_n)$, where x_n is sample n of the training set, and y_n its label, w are the weights of the multi-layer perceptron, and b the biases, $f(x_n; w, b)$ is the response of the multi-layer perceptron on sample x_n.
The loss $\ell(f(x_n; w, b), y_n)$ on sample n is computed from the response of the predictor. Note that the quantities $f(x_n; w, b)$ and y_n can be of the same type, for instance with the MSE loss, where they both are real numbers, but as we will see with other loss (e.g. "cross-entropy") it may not always be the case.
For clarity, given a layer index l of the multi-layer perceptron: $w^{(l)}$ is the weight matrix of layer l, with as many columns as the layer's input dimension and as many rows as the layer's output dimension, $b^{(l)}$ is the bias of layer l, with as many components as the layer's output dimension.
With the historical view of a layer as being composed of "neurons", the output dimension corresponds to the number of neurons in the layer, and the input dimension is the number of neurons in the previous layer (or to the model's input dimension).

For clarity, we consider a single training sample x, and introduce $s^{(1)}, \dots, s^{(L)}$ as the summations before activation functions.

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

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

$$\forall l = 1, \dots, L, \begin{cases} s^{(l)} = w^{(l)} x^{(l-1)} + b^{(l)} \\ x^{(l)} = \sigma\left(s^{(l)}\right), \end{cases}$$

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

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

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

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.

Derivatives w.r.t the activations

$$x^{(l-1)} \xrightarrow{w^{(l)},\, b^{(l)}} s^{(l)} \xrightarrow{\sigma} x^{(l)}$$

Since $s_i^{(l)}$ influences ℓ only through $x_i^{(l)}$ with

$$x_i^{(l)} = \sigma(s_i^{(l)}),$$

we have

$$\frac{\partial \ell}{\partial s_i^{(l)}} = \frac{\partial \ell}{\partial x_i^{(l)}} \frac{\partial x_i^{(l)}}{\partial s_i^{(l)}} = \frac{\partial \ell}{\partial x_i^{(l)}} \sigma'\left(s_i^{(l)}\right),$$

And since $x_j^{(l-1)}$ influences ℓ only through the $s_i^{(l)}$ with

$$s_i^{(l)} = \sum_j w_{i,j}^{(l)} x_j^{(l-1)} + b_i^{(l)},$$

we have

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

Derivatives w.r.t the weights an biases

$$x^{(l-1)} \xrightarrow{w^{(l)},\, b^{(l)}} s^{(l)} \xrightarrow{\sigma} x^{(l)}$$

Since $w_{i,j}^{(l)}$ and $b_i^{(l)}$ influences ℓ only through $s_i^{(l)}$ with

$$s_i^{(l)} = \sum_j w_{i,j}^{(l)} x_j^{(l-1)} + b_i^{(l)},$$

we have

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

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

Notes

During the forward pass, in each layer l, we keep trace of:

- the pre-non-linearity activations $s_i^{(l)}$, which we need to compute $\frac{\partial \ell}{\partial s_i^{(l)}}$,
- the inputs $x_j^{(l-1)}$ to the layer which we need to compute $\frac{\partial \ell}{\partial w_{i,j}^{(l)}}$.

To summarize: we can compute $\frac{\partial \ell}{\partial x_i^{(L)}}$ from the definition of ℓ, and recursively **propagate backward** the derivatives of the loss w.r.t the activations with

$$\frac{\partial \ell}{\partial s_i^{(l)}} = \frac{\partial \ell}{\partial x_i^{(l)}} \sigma'\left(s_i^{(l)}\right)$$

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

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

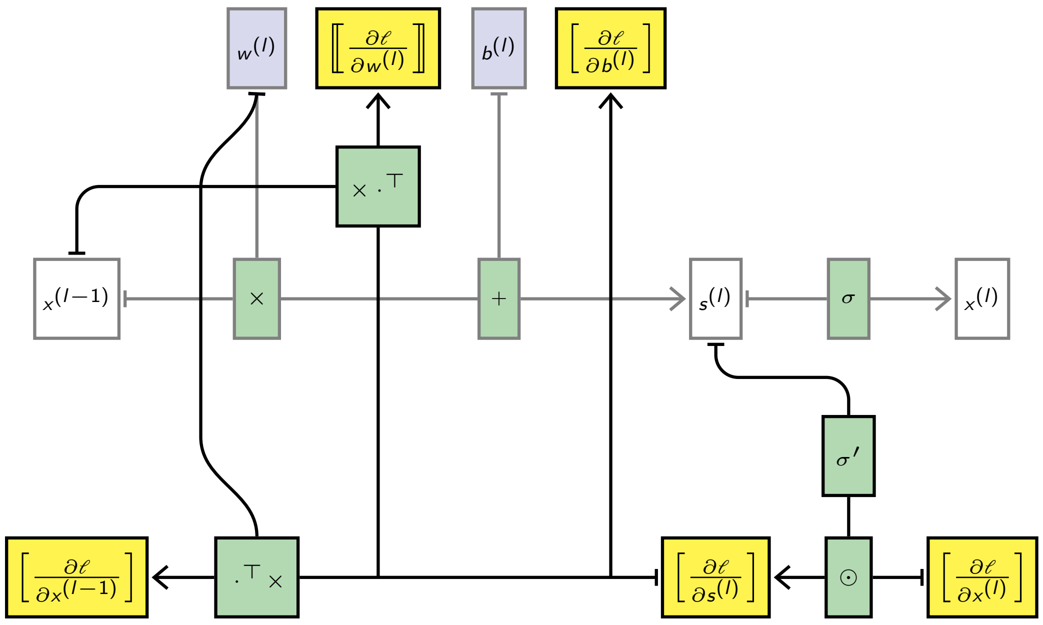
This is the **backward pass**.

To write in tensorial form we will use the following notation for the gradient of a loss $\ell : \mathbb{R}^N \to \mathbb{R}$,

$$\left[\frac{\partial \ell}{\partial x}\right] = \begin{pmatrix} \frac{\partial \ell}{\partial x_1} \\ \vdots \\ \frac{\partial \ell}{\partial x_N} \end{pmatrix},$$

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

$$\left[\!\left[\frac{\partial \psi}{\partial w}\right]\!\right] = \begin{pmatrix} \frac{\partial \psi}{\partial w_{1,1}} & \dots & \frac{\partial \psi}{\partial w_{1,M}} \\ \vdots & \ddots & \vdots \\ \frac{\partial \psi}{\partial w_{N,1}} & \dots & \frac{\partial \psi}{\partial w_{N,M}} \end{pmatrix}.$$



$w^{(l)}$
$\left[\!\left[\frac{\partial \ell}{\partial w^{(l)}}\right]\!\right]$
$b^{(l)}$
$\left[\frac{\partial \ell}{\partial b^{(l)}}\right]$
$\times \cdot^{\top}$
$x^{(l-1)}$
$\times$
$+$
$s^{(l)}$
σ
$x^{(l)}$
σ'
$\left[\frac{\partial \ell}{\partial x^{(l-1)}}\right]$
$\cdot^{\top} \times$
$\left[\frac{\partial \ell}{\partial s^{(l)}}\right]$
$\odot$
$\left[\frac{\partial \ell}{\partial x^{(l)}}\right]$

Forward pass

Compute the activations.

$$x^{(0)} = x, \quad \forall l = 1, \dots, L, \begin{cases} s^{(l)} = w^{(l)} x^{(l-1)} + b^{(l)} \\ x^{(l)} = \sigma\left(s^{(l)}\right) \end{cases}$$

Backward pass

Compute the derivatives of the loss w.r.t. the activations.

$$\begin{cases} \left[\frac{\partial \ell}{\partial x^{(L)}}\right] \text{ from the definition of } \ell \\ \text{if } l < L, \left[\frac{\partial \ell}{\partial x^{(l)}}\right] = \left(w^{(l+1)}\right)^\top \left[\frac{\partial \ell}{\partial s^{(l+1)}}\right] \end{cases} \qquad \left[\frac{\partial \ell}{\partial s^{(l)}}\right] = \left[\frac{\partial \ell}{\partial x^{(l)}}\right] \odot \sigma'\left(s^{(l)}\right)$$

Compute the derivatives of the loss w.r.t. the parameters.

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

Gradient step

Update the parameters.

$$w^{(l)} \leftarrow w^{(l)} - \eta \left[\!\left[\frac{\partial \ell}{\partial w^{(l)}}\right]\!\right] \qquad b^{(l)} \leftarrow b^{(l)} - \eta \left[\frac{\partial \ell}{\partial b^{(l)}}\right]$$

Notes

Here $\odot$ denotes the component-wise multiplication, also known as the Hadamard product.

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.

Without tricks, we have to keep in memory all the activations computed during the forward pass.

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

$$s^{(l)} = w^{(l)} x^{(l-1)} + b^{(l)}$$

and for the backward

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

and

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

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