Backpropagation

Jérémie Cabessa

Laboratoire DAVID, UVSQ

Chain Rule

On rappelle le **théorème des fonctions composées** (chain rule).

Soient les fonctions suivantes:

$$
\begin{array}{ccccc}
\mathbb{R} & \xrightarrow{f} & \mathbb{R} & \xrightarrow{g} & \mathbb{R} \\
x & \xmapsto{f} & y = f(x) & \xmapsto{g} & z = g(y) \\
 & & & & \quad = g\left(f(x)\right)
\end{array}
$$

alors on a:

$$
\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x}
$$

Chain Rule

On rappelle le **théorème des fonctions composées** (chain rule).

Soient les fonctions suivantes:

$$\begin{array}{ccccc} \mathbb{R} & \xrightarrow{f} & \mathbb{R} & \xrightarrow{g} & \mathbb{R} \\ x & \xmapsto{f} & y = f(x) & \xmapsto{g} & z = g(y) \\ & & & & \quad = g\left(f(x)\right) \end{array}$$

alors on a:

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x}$$

Chain Rule

On rappelle le **théorème des fonctions composées** (chain rule).

Soient les fonctions suivantes:

$$
\begin{array}{ccccc}
\mathbb{R} & \xrightarrow{f} & \mathbb{R} & \xrightarrow{g} & \mathbb{R} \\
x & \overset{f}{\longmapsto} & y = f(x) & \overset{g}{\longmapsto} & z = g(y) \\
 & & & & = g\left(f(x)\right)
\end{array}
$$

alors on a:

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x}$$

Chain Rule

Exemple:

$$\begin{array}{ccccc} \mathbb{R} & \xrightarrow{f} & \mathbb{R} & \xrightarrow{g} & \mathbb{R} \\ x & \overset{f}{\longmapsto} & y = x^2 + 1 & \overset{g}{\longmapsto} & z = 5y \\ & & & & \quad = 5(x^2+1) \end{array}$$

On a:

$$\frac{\partial z}{\partial x} = \frac{\partial \left[5(x^2+1)\right]}{\partial x} = 10x = 5 \cdot 2x = \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x}$$

Chain Rule

Exemple:

$$\begin{array}{ccccc} \mathbb{R} & \xrightarrow{f} & \mathbb{R} & \xrightarrow{g} & \mathbb{R} \\ x & \overset{f}{\longmapsto} & y = x^2 + 1 & \overset{g}{\longmapsto} & z = 5y \\ & & & & \quad = 5(x^2+1) \end{array}$$

On a:

$$\frac{\partial z}{\partial x} = \frac{\partial \left[5(x^2+1)\right]}{\partial x} = 10x = 5 \cdot 2x = \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x}$$

Chain Rule

Généralisation multidimensionnelle

Soient les fonctions suivantes:

$$
\begin{array}{ccccc}
\mathbb{R}^m & \xrightarrow{f} & \mathbb{R}^n & \xrightarrow{g} & \mathbb{R} \\
\boldsymbol{x} & \xmapsto{f} & \boldsymbol{y} = f(\boldsymbol{x}) & \xmapsto{g} & z = g(\boldsymbol{y}) \\
 & & & & \quad = g\left(f(\boldsymbol{x})\right)
\end{array}
$$

alors on a:

$$
\frac{\partial z}{\partial x_i} = \sum_{j=1}^{n} \frac{\partial z}{\partial y_j} \cdot \frac{\partial y_j}{\partial x_i}, \quad i = 1, \ldots, m
$$

Chain Rule

Généralisation multidimensionnelle

Soient les fonctions suivantes:

$$
\begin{array}{ccccc}
\mathbb{R}^m & \xrightarrow{f} & \mathbb{R}^n & \xrightarrow{g} & \mathbb{R} \\
\boldsymbol{x} & \xmapsto{f} & \boldsymbol{y} = f(\boldsymbol{x}) & \xmapsto{g} & z = g(\boldsymbol{y}) \\
 & & & & \quad = g\left(f(\boldsymbol{x})\right)
\end{array}
$$

alors on a:

$$
\frac{\partial z}{\partial x_i} = \sum_{j=1}^{n} \frac{\partial z}{\partial y_j} \cdot \frac{\partial y_j}{\partial x_i}, \quad i = 1, \ldots, m
$$

Chain Rule

Généralisation multidimensionnelle

Soient les fonctions suivantes:

$$\begin{array}{ccccc} \mathbb{R}^m & \xrightarrow{f} & \mathbb{R}^n & \xrightarrow{g} & \mathbb{R} \\ \boldsymbol{x} & \xmapsto{f} & \boldsymbol{y} = f(\boldsymbol{x}) & \xmapsto{g} & z = g(\boldsymbol{y}) \\ & & & & = g\left(f(\boldsymbol{x})\right) \end{array}$$

alors on a:

$$\frac{\partial z}{\partial x_i} = \sum_{j=1}^{n} \frac{\partial z}{\partial y_j} \cdot \frac{\partial y_j}{\partial x_i}, \quad i = 1, \ldots, m$$

Chain Rule

Formulation vectorielle

$$
\begin{array}{ccccl}
\mathbb{R}^m & \xrightarrow{f} & \mathbb{R}^n & \xrightarrow{g} & \mathbb{R} \\
\boldsymbol{x} & \xmapsto{f} & \boldsymbol{y} = f(\boldsymbol{x}) & \xmapsto{g} & z = g(\boldsymbol{y}) \\
 & & & & \quad = g\left(f(\boldsymbol{x})\right)
\end{array}
$$

Soient $\nabla_{\boldsymbol{x}} z$ et le *gradient* de z par rapport à $\boldsymbol{x}$, $\nabla_{\boldsymbol{y}} z$ et le gradient de z par rapport à $\boldsymbol{y}$, et $\boldsymbol{J}_f := \left[\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}\right]$ le *jacobien* de la fonction f:

$$\nabla_{\boldsymbol{x}} z = \left[\left[\nabla_{\boldsymbol{y}} z\right]^T \left[\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}\right] \right]^T = \boldsymbol{J}_f^T \nabla_{\boldsymbol{y}} z$$

$$\begin{pmatrix} \frac{\partial z}{\partial x_1} \\ \vdots \\ \frac{\partial z}{\partial x_m} \end{pmatrix} = \left[\begin{pmatrix} \frac{\partial z}{\partial y_1} & \cdots & \frac{\partial z}{\partial y_n} \end{pmatrix} \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_n}{\partial x_1} & \cdots & \frac{\partial y_n}{\partial x_m} \end{pmatrix} \right]^T$$

Chain Rule

Formulation vectorielle

$$\begin{array}{ccccc} \mathbb{R}^m & \xrightarrow{f} & \mathbb{R}^n & \xrightarrow{g} & \mathbb{R} \\ \boldsymbol{x} & \xmapsto{f} & \boldsymbol{y} = f(\boldsymbol{x}) & \xmapsto{g} & z = g(\boldsymbol{y}) \\ & & & & = g\left(f(\boldsymbol{x})\right) \end{array}$$

Soient $\nabla_{\boldsymbol{x}} z$ et le *gradient* de z par rapport à $\boldsymbol{x}$, $\nabla_{\boldsymbol{y}} z$ et le gradient de z par rapport à $\boldsymbol{y}$, et $\boldsymbol{J}_f := \left[\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}\right]$ le *jacobien* de la fonction f:

$$\nabla_{\boldsymbol{x}} z = \left[\left[\nabla_{\boldsymbol{y}} z\right]^T \left[\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}\right] \right]^T = \boldsymbol{J}_f^T \nabla_{\boldsymbol{y}} z$$

$$\begin{pmatrix} \frac{\partial z}{\partial x_1} \\ \vdots \\ \frac{\partial z}{\partial x_m} \end{pmatrix} = \left[\begin{pmatrix} \frac{\partial z}{\partial y_1} & \cdots & \frac{\partial z}{\partial y_n} \end{pmatrix} \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_n}{\partial x_1} & \cdots & \frac{\partial y_n}{\partial x_m} \end{pmatrix} \right]^T$$

Chain Rule

Formulation vectorielle

$$
\begin{array}{ccccc}
\mathbb{R}^m & \xrightarrow{f} & \mathbb{R}^n & \xrightarrow{g} & \mathbb{R} \\
\boldsymbol{x} & \xmapsto{f} & \boldsymbol{y} = f(\boldsymbol{x}) & \xmapsto{g} & z = g(\boldsymbol{y}) \\
 & & & & \quad = g\left(f(\boldsymbol{x})\right)
\end{array}
$$

Soient $\nabla_{\boldsymbol{x}} z$ et le *gradient* de z par rapport à $\boldsymbol{x}$, $\nabla_{\boldsymbol{y}} z$ et le gradient de z par rapport à $\boldsymbol{y}$, et $\boldsymbol{J}_f := \left[\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}\right]$ le *jacobien* de la fonction f:

$$
\nabla_{\boldsymbol{x}} z = \left[\left[\nabla_{\boldsymbol{y}} z\right]^T \left[\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}\right] \right]^T = \boldsymbol{J}_f^T \nabla_{\boldsymbol{y}} z
$$

$$
\begin{pmatrix} \frac{\partial z}{\partial x_1} \\ \vdots \\ \frac{\partial z}{\partial x_m} \end{pmatrix} = \left[\begin{pmatrix} \frac{\partial z}{\partial y_1} & \cdots & \frac{\partial z}{\partial y_n} \end{pmatrix} \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_n}{\partial x_1} & \cdots & \frac{\partial y_n}{\partial x_m} \end{pmatrix} \right]^T
$$

Chain Rule

Formulation vectorielle

$$
\begin{array}{ccccc}
\mathbb{R}^m & \xrightarrow{f} & \mathbb{R}^n & \xrightarrow{g} & \mathbb{R} \\
\boldsymbol{x} & \overset{f}{\longmapsto} & \boldsymbol{y} = f(\boldsymbol{x}) & \overset{g}{\longmapsto} & z = g(\boldsymbol{y}) \\
 & & & & = g\left(f(\boldsymbol{x})\right)
\end{array}
$$

Soient $\nabla_{\boldsymbol{x}} z$ et le *gradient* de z par rapport à $\boldsymbol{x}$, $\nabla_{\boldsymbol{y}} z$ et le gradient de z par rapport à $\boldsymbol{y}$, et $\boldsymbol{J}_f := \left[\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}\right]$ le *jacobien* de la fonction f:

$$
\nabla_{\boldsymbol{x}} z = \left[[\nabla_{\boldsymbol{y}} z]^T \left[\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}\right] \right]^T = \boldsymbol{J}_f^T \nabla_{\boldsymbol{y}} z
$$

$$
\begin{pmatrix} \frac{\partial z}{\partial x_1} \\ \vdots \\ \frac{\partial z}{\partial x_m} \end{pmatrix} = \left[\begin{pmatrix} \frac{\partial z}{\partial y_1} & \cdots & \frac{\partial z}{\partial y_n} \end{pmatrix} \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_n}{\partial x_1} & \cdots & \frac{\partial y_n}{\partial x_m} \end{pmatrix} \right]^T
$$

Chain Rule

Exemple:

$$
\begin{array}{ccccc}
\mathbb{R}^m & \xrightarrow{f} & \mathbb{R}^n & \xrightarrow{g} & \mathbb{R} \\
\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} & \xmapsto{f} & \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} x_1 x_2 \\ x_1^2 \\ x_2^3 \end{pmatrix} & \xmapsto{g} & z = y_1 y_2 + y_3 \\
 & & & & = (x_1 x_2) x_1^2 + x_2^3
\end{array}
$$

On a:

$$
\begin{aligned}
\frac{\partial z}{\partial x_1} = \frac{\partial \left[(x_1 x_2) x_1^2 + x_2^3 \right]}{\partial x_1} &= x_1^2 x_2 + x_1 x_2 2 x_1 + 0 \\
&= \frac{\partial z}{\partial y_1} \cdot \frac{\partial y_1}{\partial x_1} + \frac{\partial z}{\partial y_2} \cdot \frac{\partial y_2}{\partial x_1} + \frac{\partial z}{\partial y_3} \cdot \frac{\partial y_3}{\partial x_1} \\
\frac{\partial z}{\partial x_2} = \frac{\partial \left[(x_1 x_2) x_1^2 + x_2^3 \right]}{\partial x_2} &= x_1^2 x_1 + 3 x_2^2 + 0 \\
&= \frac{\partial z}{\partial y_1} \cdot \frac{\partial y_1}{\partial x_2} + \frac{\partial z}{\partial y_2} \cdot \frac{\partial y_2}{\partial x_2} + \frac{\partial z}{\partial y_3} \cdot \frac{\partial y_3}{\partial x_2}
\end{aligned}
$$

Chain Rule

Exemple:

$$\mathbb{R}^m \xrightarrow{f} \mathbb{R}^n \xrightarrow{g} \mathbb{R}$$

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \xmapsto{f} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} x_1 x_2 \\ x_1^2 \\ x_2^3 \end{pmatrix} \xmapsto{g} z = y_1 y_2 + y_3 = (x_1 x_2) x_1^2 + x_2^3$$

On a:

$$\frac{\partial z}{\partial x_1} = \frac{\partial \left[(x_1 x_2) x_1^2 + x_2^3 \right]}{\partial x_1} = x_1^2 x_2 + x_1 x_2 2 x_1 + 0 = \frac{\partial z}{\partial y_1} \cdot \frac{\partial y_1}{\partial x_1} + \frac{\partial z}{\partial y_2} \cdot \frac{\partial y_2}{\partial x_1} + \frac{\partial z}{\partial y_3} \cdot \frac{\partial y_3}{\partial x_1}$$

$$\frac{\partial z}{\partial x_2} = \frac{\partial \left[(x_1 x_2) x_1^2 + x_2^3 \right]}{\partial x_2} = x_1^2 x_1 + 3 x_2^2 + 0 = \frac{\partial z}{\partial y_1} \cdot \frac{\partial y_1}{\partial x_2} + \frac{\partial z}{\partial y_2} \cdot \frac{\partial y_2}{\partial x_2} + \frac{\partial z}{\partial y_3} \cdot \frac{\partial y_3}{\partial x_2}$$

Chain Rule

Exemple:

$$
\begin{array}{ccccc}
\mathbb{R}^m & \xrightarrow{f} & \mathbb{R}^n & \xrightarrow{g} & \mathbb{R} \\
\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} & \xmapsto{f} & \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} x_1 x_2 \\ x_1^2 \\ x_2^3 \end{pmatrix} & \xmapsto{g} & z = y_1 y_2 + y_3 \\
 & & & & = (x_1 x_2) x_1^2 + x_2^3
\end{array}
$$

On a:

$$
\begin{aligned}
\frac{\partial z}{\partial x_1} = \frac{\partial \left[(x_1 x_2) x_1^2 + x_2^3 \right]}{\partial x_1} &= x_1^2 x_2 + x_1 x_2 2 x_1 + 0 \\
&= \frac{\partial z}{\partial y_1} \cdot \frac{\partial y_1}{\partial x_1} + \frac{\partial z}{\partial y_2} \cdot \frac{\partial y_2}{\partial x_1} + \frac{\partial z}{\partial y_3} \cdot \frac{\partial y_3}{\partial x_1} \\
\frac{\partial z}{\partial x_2} = \frac{\partial \left[(x_1 x_2) x_1^2 + x_2^3 \right]}{\partial x_2} &= x_1^2 x_1 + 3 x_2^2 + 0 \\
&= \frac{\partial z}{\partial y_1} \cdot \frac{\partial y_1}{\partial x_2} + \frac{\partial z}{\partial y_2} \cdot \frac{\partial y_2}{\partial x_2} + \frac{\partial z}{\partial y_3} \cdot \frac{\partial y_3}{\partial x_2}
\end{aligned}
$$

Chain Rule

Exemple (suite):

$$
\begin{array}{ccccc}
\mathbb{R}^m & \xrightarrow{f} & \mathbb{R}^n & \xrightarrow{g} & \mathbb{R} \\
\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} & \xmapsto{f} & \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} x_1 x_2 \\ x_1^2 \\ x_2^3 \end{pmatrix} & \xmapsto{g} & z = y_1 y_2 + y_3 \\
 & & & & = (x_1 x_2) x_1^2 + x_2^3
\end{array}
$$

On a:

$$
\nabla_{\boldsymbol{x}} z = \begin{pmatrix} \frac{\partial z}{\partial x_1} \\ \frac{\partial z}{\partial x_2} \end{pmatrix} = \begin{pmatrix} 3x_1^2 x_2 \\ x_1^3 + 3x_2^2 \end{pmatrix} = \begin{pmatrix} y_2 & y_1 & 1 \end{pmatrix} \begin{pmatrix} x_2 & x_1 \\ 2x_1 & 0 \\ 0 & 3x_2^2 \end{pmatrix} = [\nabla_{\boldsymbol{y}} z]^T \left[\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}} \right]
$$

Chain Rule

Exemple (suite):

$$
\begin{array}{ccccc}
\mathbb{R}^m & \xrightarrow{f} & \mathbb{R}^n & \xrightarrow{g} & \mathbb{R} \\
\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} & \xmapsto{f} & \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} x_1 x_2 \\ x_1^2 \\ x_2^3 \end{pmatrix} & \xmapsto{g} & z = y_1 y_2 + y_3 \\
 & & & & = (x_1 x_2) x_1^2 + x_2^3
\end{array}
$$

On a:

$$
\nabla_{\boldsymbol{x}} z = \begin{pmatrix} \frac{\partial z}{\partial x_1} \\ \frac{\partial z}{\partial x_2} \end{pmatrix} = \begin{pmatrix} 3x_1^2 x_2 \\ x_1^3 + 3x_2^2 \end{pmatrix} = \begin{pmatrix} y_2 & y_1 & 1 \end{pmatrix} \begin{pmatrix} x_2 & x_1 \\ 2x_1 & 0 \\ 0 & 3x_2^2 \end{pmatrix} = [\nabla_{\boldsymbol{y}} z]^T \left[\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}} \right]
$$

Neural Network as a Function

- Soit $S = \left\{(\boldsymbol{x_i}, \boldsymbol{y_i}) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \ldots, N\right\}$ un dataset.
- Soit $\mathcal{N}_{\boldsymbol{\Theta}}$ un réseau de neurones (MLP) à L couches donné par les *paramètres* (poids et biais)

$$\boldsymbol{\Theta} := \left\{\left(\boldsymbol{W}^{[l]}, \boldsymbol{b}^{[l]}\right) : l = 1, \ldots, L\right\}$$

et par la dynamique

$$\begin{cases} \boldsymbol{a}^{[0]} &= \boldsymbol{x} \\ \boldsymbol{z}^{[l]} &= \boldsymbol{W}^{[l]}\boldsymbol{a}^{[l-1]} + \boldsymbol{b}^{[l]}, \\ \boldsymbol{a}^{[l]} &= \boldsymbol{\sigma}\left(\boldsymbol{z}^{[l]}\right) \end{cases} \quad l = 1, \ldots, L$$

Neural Network as a Function

- Soit $S = \left\{(\boldsymbol{x_i}, \boldsymbol{y_i}) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \dots, N\right\}$ un dataset.
- Soit $\mathcal{N}_{\boldsymbol{\Theta}}$ un réseau de neurones (MLP) à L couches donné par les *paramètres* (poids et biais)

$$\boldsymbol{\Theta} := \left\{\left(\boldsymbol{W}^{[l]}, \boldsymbol{b}^{[l]}\right) : l = 1, \dots, L\right\}$$

et par la dynamique

$$\begin{cases} \boldsymbol{a}^{[0]} & = & \boldsymbol{x} \\ \boldsymbol{z}^{[l]} & = & \boldsymbol{W}^{[l]}\boldsymbol{a}^{[l-1]} + \boldsymbol{b}^{[l]}, \\ \boldsymbol{a}^{[l]} & = & \boldsymbol{\sigma}\left(\boldsymbol{z}^{[l]}\right) \end{cases} \quad l = 1, \dots, L$$

Neural Network as a Function

- **Remarque:** le réseau $\mathcal{N}_{\boldsymbol{\Theta}}$ peut-être naturellement associé à la fonction

$$\begin{aligned} f_{\boldsymbol{\Theta}} : \mathbb{R}^{d_1} & \longrightarrow \mathbb{R}^{d_2} \\ \boldsymbol{x} & \longmapsto f_{\boldsymbol{\Theta}}(\boldsymbol{x}) := \boldsymbol{a}^{[\boldsymbol{L}]} \end{aligned}$$

- $f_{\boldsymbol{\Theta}}(x)$ est la *prédiction* (output) de $\mathcal{N}_{\boldsymbol{\Theta}}$ associée à l'input x.
- Chaque jeu de paramètres $\boldsymbol{\Theta}$ donne lieu à une fonction $f_{\boldsymbol{\Theta}}$ différente.

Neural Network as a Function

- **Remarque:** le réseau $\mathcal{N}_{\mathbf{\Theta}}$ peut-être naturellement associé à la fonction

$$\begin{aligned} f_{\mathbf{\Theta}} : \mathbb{R}^{d_1} &\longrightarrow \mathbb{R}^{d_2} \\ \boldsymbol{x} &\longmapsto f_{\mathbf{\Theta}}(\boldsymbol{x}) := \boldsymbol{a}^{[\boldsymbol{L}]} \end{aligned}$$

- $f_{\mathbf{\Theta}}(\boldsymbol{x})$ est la *prédiction* (output) de $\mathcal{N}_{\mathbf{\Theta}}$ associée à l'input $\boldsymbol{x}$.
- Chaque jeu de paramètres $\mathbf{\Theta}$ donne lieu à une fonction $f_{\mathbf{\Theta}}$ différente.

Neural Network as a Function

- **Remarque:** le réseau $\mathcal{N}_{\boldsymbol{\Theta}}$ peut-être naturellement associé à la fonction

$$\begin{aligned} f_{\boldsymbol{\Theta}} : \mathbb{R}^{d_1} & \longrightarrow \mathbb{R}^{d_2} \\ \boldsymbol{x} & \longmapsto f_{\boldsymbol{\Theta}}(\boldsymbol{x}) := \boldsymbol{a}^{[L]} \end{aligned}$$

- $f_{\boldsymbol{\Theta}}(\boldsymbol{x})$ est la *prédiction* (output) de $\mathcal{N}_{\boldsymbol{\Theta}}$ associée à l'input $\boldsymbol{x}$.
- Chaque jeu de paramètres $\boldsymbol{\Theta}$ donne lieu à une fonction $f_{\boldsymbol{\Theta}}$ différente.

Loss Function

- Soit une *fonction de coût* (cost or loss function) qui mesure l'erreur entre la *prédiction* $\hat{\boldsymbol{y}}_{\boldsymbol{i}}$ et la *réalité* $\boldsymbol{y}_{\boldsymbol{i}}$:

$$\begin{aligned} \ell : \mathbb{R}^{d_2} \times \mathbb{R}^{d_2} &\longrightarrow \mathbb{R} \\ (\hat{\boldsymbol{y}}_{\boldsymbol{i}}, \boldsymbol{y}_{\boldsymbol{i}}) &\longmapsto \ell\left(\hat{\boldsymbol{y}}_{\boldsymbol{i}}, \boldsymbol{y}_{\boldsymbol{i}}\right) \end{aligned}$$

- Typiquement, la fonction de coût pourrait être l'erreur quadratique (distance Euclidienne au carré)

$$\ell\left(\hat{\boldsymbol{y}}_{\boldsymbol{i}}, \boldsymbol{y}_{\boldsymbol{i}}\right) = \frac{1}{2}\left\|\hat{\boldsymbol{y}}_{\boldsymbol{i}} - \boldsymbol{y}_{\boldsymbol{i}}\right\|_2^2$$

Loss Function

- Soit une *fonction de coût* (cost or loss function) qui mesure l'erreur entre la *prédiction* $\hat{\boldsymbol{y}}_{\boldsymbol{i}}$ et la *réalité* $\boldsymbol{y}_{\boldsymbol{i}}$:

$$\begin{aligned} \ell : \mathbb{R}^{d_2} \times \mathbb{R}^{d_2} &\longrightarrow \mathbb{R} \\ (\hat{\boldsymbol{y}}_{\boldsymbol{i}}, \boldsymbol{y}_{\boldsymbol{i}}) &\longmapsto \ell\left(\hat{\boldsymbol{y}}_{\boldsymbol{i}}, \boldsymbol{y}_{\boldsymbol{i}}\right) \end{aligned}$$

- Typiquement, la fonction de coût pourrait être l'erreur quadratique (distance Euclidienne au carré)

$$\ell\left(\hat{\boldsymbol{y}}_{\boldsymbol{i}}, \boldsymbol{y}_{\boldsymbol{i}}\right) = \frac{1}{2} \left\| \hat{\boldsymbol{y}}_{\boldsymbol{i}} - \boldsymbol{y}_{\boldsymbol{i}} \right\|_2^2$$

Loss Function

- La *fonction de coût* peut être naturellement généralisée à un ensemble de *prédictions* et de *réalités*:

$$\begin{aligned}\mathcal{L} : \mathbb{R}^{d_2} \times \cdots\cdots \times \mathbb{R}^{d_2} &\longrightarrow \mathbb{R} \\ (\hat{\boldsymbol{y}}_{\mathbf{1}}, \ldots, \hat{\boldsymbol{y}}_{\boldsymbol{N}}, \boldsymbol{y}_{\boldsymbol{i}} \ldots, \boldsymbol{y}_{\boldsymbol{N}}) &\longmapsto \mathcal{L}(\hat{\boldsymbol{y}}_{\mathbf{1}}, \ldots, \hat{\boldsymbol{y}}_{\boldsymbol{N}}, \boldsymbol{y}_{\boldsymbol{i}} \ldots, \boldsymbol{y}_{\boldsymbol{N}})\end{aligned}$$

- Typiquement, la fonction de coût pourrait être l'erreur quadratique moyenne (mean squared error MSE)

$$\begin{aligned}\mathcal{L}(\hat{\boldsymbol{y}}_{\mathbf{1}}, \ldots, \hat{\boldsymbol{y}}_{\boldsymbol{N}}, \boldsymbol{y}_{\boldsymbol{i}}, \ldots, \boldsymbol{y}_{\boldsymbol{N}}) &= \frac{1}{N}\sum_{i=1}^{N} \ell(\hat{\boldsymbol{y}}_{\boldsymbol{i}}, \boldsymbol{y}_{\boldsymbol{i}}) \\ &= \frac{1}{2N}\sum_{i=1}^{N} \|\hat{\boldsymbol{y}}_{\boldsymbol{i}} - \boldsymbol{y}_{\boldsymbol{i}}\|_2^2\end{aligned}$$

Loss Function

- La *fonction de coût* peut être naturellement généralisée à un ensemble de *prédictions* et de *réalités*:

$$\begin{aligned} \mathcal{L} : \mathbb{R}^{d_2} \times \cdots\cdots \times \mathbb{R}^{d_2} &\longrightarrow \mathbb{R} \\ (\hat{\boldsymbol{y}}_{\boldsymbol{1}}, \dots, \hat{\boldsymbol{y}}_{\boldsymbol{N}}, \boldsymbol{y}_{\boldsymbol{i}} \dots, \boldsymbol{y}_{\boldsymbol{N}}) &\longmapsto \mathcal{L}(\hat{\boldsymbol{y}}_{\boldsymbol{1}}, \dots, \hat{\boldsymbol{y}}_{\boldsymbol{N}}, \boldsymbol{y}_{\boldsymbol{i}} \dots, \boldsymbol{y}_{\boldsymbol{N}}) \end{aligned}$$

- Typiquement, la fonction de coût pourrait être l'erreur quadratique moyenne (mean squared error MSE)

$$\begin{aligned} \mathcal{L}(\hat{\boldsymbol{y}}_{\boldsymbol{1}}, \dots, \hat{\boldsymbol{y}}_{\boldsymbol{N}}, \boldsymbol{y}_{\boldsymbol{i}}, \dots, \boldsymbol{y}_{\boldsymbol{N}}) &= \frac{1}{N} \sum_{i=1}^{N} \ell(\hat{\boldsymbol{y}}_{\boldsymbol{i}}, \boldsymbol{y}_{\boldsymbol{i}}) \\ &= \frac{1}{2N} \sum_{i=1}^{N} \|\hat{\boldsymbol{y}}_{\boldsymbol{i}} - \boldsymbol{y}_{\boldsymbol{i}}\|_2^2 \end{aligned}$$

Loss Function

- Pour un réseau de neurones $\mathcal{N}_{\boldsymbol{\Theta}}$, l'erreur entre les prédictions et les réalités est

$$\mathcal{L}\left(f_{\boldsymbol{\Theta}}\left(\boldsymbol{x_1}\right), \ldots, f_{\boldsymbol{\Theta}}\left(\boldsymbol{x_N}\right), \boldsymbol{y_1}, \ldots, \boldsymbol{y_N}\right).$$

- Pour différents paramètres $\boldsymbol{\Theta}$, on aura différentes prédictions $f_{\boldsymbol{\Theta}}\left(\boldsymbol{x_1}\right), \ldots, f_{\boldsymbol{\Theta}}\left(\boldsymbol{x_N}\right)$, et donc différentes erreurs $\mathcal{L}\left(\ldots\right)$.
- **Ainsi, $\mathcal{L}$ est une fonction des paramètres $\boldsymbol{\Theta}$ du réseau:**

$$\begin{aligned} \mathcal{L} : \mathbb{R}^{|\boldsymbol{\Theta}|} &\longrightarrow \mathbb{R} \\ \boldsymbol{\Theta} &\longmapsto \mathcal{L}\left(f_{\boldsymbol{\Theta}}\left(\boldsymbol{x_1}\right), \ldots, f_{\boldsymbol{\Theta}}\left(\boldsymbol{x_N}\right), \boldsymbol{y_1}, \ldots, \boldsymbol{y_N}\right). \end{aligned}$$

où $|\boldsymbol{\Theta}|$ est le nombre de paramètres $\boldsymbol{\Theta}$ (poids et biais, souvent plusieurs millions).

Loss Function

- Pour un réseau de neurones $\mathcal{N}_{\boldsymbol{\Theta}}$, l'erreur entre les prédictions et les réalités est

$$\mathcal{L}\left(f_{\boldsymbol{\Theta}}\left(\boldsymbol{x_1}\right), \ldots, f_{\boldsymbol{\Theta}}\left(\boldsymbol{x_N}\right), \boldsymbol{y_1}, \ldots, \boldsymbol{y_N}\right).$$

- Pour différents paramètres $\boldsymbol{\Theta}$, on aura différentes prédictions $f_{\boldsymbol{\Theta}}\left(\boldsymbol{x_1}\right), \ldots, f_{\boldsymbol{\Theta}}\left(\boldsymbol{x_N}\right)$, et donc différentes erreurs $\mathcal{L}\left(\ldots\right)$.
- Ainsi, $\mathcal{L}$ est une fonction des paramètres $\boldsymbol{\Theta}$ du réseau:

$$\begin{aligned} \mathcal{L} : \mathbb{R}^{|\boldsymbol{\Theta}|} &\longrightarrow \mathbb{R} \\ \boldsymbol{\Theta} &\longmapsto \mathcal{L}\left(f_{\boldsymbol{\Theta}}\left(\boldsymbol{x_1}\right), \ldots, f_{\boldsymbol{\Theta}}\left(\boldsymbol{x_N}\right), \boldsymbol{y_1}, \ldots, \boldsymbol{y_N}\right). \end{aligned}$$

  où $|\boldsymbol{\Theta}|$ est le nombre de paramètres $\boldsymbol{\Theta}$ (poids et biais, souvent plusieurs millions).

Loss Function

- Pour un réseau de neurones $\mathcal{N}_{\boldsymbol{\Theta}}$, l'erreur entre les prédictions et les réalités est

$$\mathcal{L}\left(f_{\boldsymbol{\Theta}}\left(\boldsymbol{x_1}\right), \ldots, f_{\boldsymbol{\Theta}}\left(\boldsymbol{x_N}\right), \boldsymbol{y_1}, \ldots, \boldsymbol{y_N}\right).$$

- Pour différents paramètres $\boldsymbol{\Theta}$, on aura différentes prédictions $f_{\boldsymbol{\Theta}}\left(\boldsymbol{x_1}\right), \ldots, f_{\boldsymbol{\Theta}}\left(\boldsymbol{x_N}\right)$, et donc différentes erreurs $\mathcal{L}\left(\ldots\right)$.
- **Ainsi, $\mathcal{L}$ est une fonction des paramètres $\boldsymbol{\Theta}$ du réseau**:

$$\begin{aligned} \mathcal{L} : \mathbb{R}^{|\boldsymbol{\Theta}|} &\longrightarrow \mathbb{R} \\ \boldsymbol{\Theta} &\longmapsto \mathcal{L}\left(f_{\boldsymbol{\Theta}}\left(\boldsymbol{x_1}\right), \ldots, f_{\boldsymbol{\Theta}}\left(\boldsymbol{x_N}\right), \boldsymbol{y_1}, \ldots, \boldsymbol{y_N}\right). \end{aligned}$$

où $|\boldsymbol{\Theta}|$ est le nombre de paramètres $\boldsymbol{\Theta}$ (poids et biais, souvent plusieurs millions).

Training

- *L'entraînement* du réseau $\mathcal{N}_{\boldsymbol{\Theta}}$ consiste à déterminer des paramètres $\boldsymbol{\Theta}$ qui minimisent l'erreur

$$\mathcal{L}\big(f_{\boldsymbol{\Theta}}(\boldsymbol{x_1}), \ldots, f_{\boldsymbol{\Theta}}(\boldsymbol{x_N}), \boldsymbol{y_1}, \ldots, \boldsymbol{y_N}\big).$$

- Pour cela, on utilise une descente de gradient: *mini-batch stochastic gradient descent*.
- *Backpropagation* est un algorithme qui permet de calculer les gradients $\nabla_{\boldsymbol{\Theta}}\mathcal{L}$ de manière efficiente.

Training

- *L'entraînement* du réseau $\mathcal{N}_{\boldsymbol{\Theta}}$ consiste à déterminer des paramètres $\boldsymbol{\Theta}$ qui minimisent l'erreur

$$\mathcal{L}\big(f_{\boldsymbol{\Theta}}\left(\boldsymbol{x_1}\right), \ldots, f_{\boldsymbol{\Theta}}\left(\boldsymbol{x_N}\right), \boldsymbol{y_1}, \ldots, \boldsymbol{y_N}\big).$$

- Pour cela, on utilise une descente de gradient: *mini-batch stochastic gradient descent*.
- *Backpropagation* est un algorithme qui permet de calculer les gradients $\nabla_{\boldsymbol{\Theta}}\mathcal{L}$ de manière efficiente.

Training

- *L'entraînement* du réseau $\mathcal{N}_{\boldsymbol{\Theta}}$ consiste à déterminer des paramètres $\boldsymbol{\Theta}$ qui minimisent l'erreur

$$\mathcal{L}\big(f_{\boldsymbol{\Theta}}\left(\boldsymbol{x_1}\right), \ldots, f_{\boldsymbol{\Theta}}\left(\boldsymbol{x_N}\right), \boldsymbol{y_1}, \ldots, \boldsymbol{y_N}\big).$$

- Pour cela, on utilise une descente de gradient: *mini-batch stochastic gradient descent*.
- *Backpropagation* est un algorithme qui permet de calculer les gradients $\nabla_{\boldsymbol{\Theta}}\mathcal{L}$ de manière efficiente.

Training

$$\mathcal{L}\left(f_{\Theta}\left(\boldsymbol{x_1}\right), \ldots, f_{\Theta}\left(\boldsymbol{x_N}\right), \boldsymbol{y_1}, \ldots, \boldsymbol{y_N}\right) = \mathcal{L}\left(\Theta, \ldots\right)$$

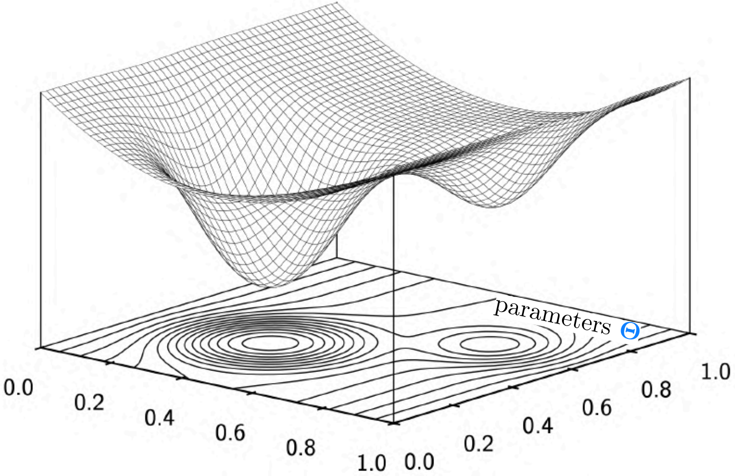


Figure adapted from [Fleuret, 2022]

Training

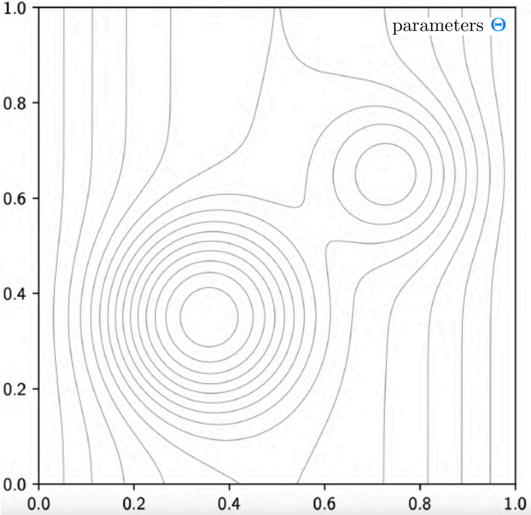


Figure adapted from [Fleuret, 2022]

Training

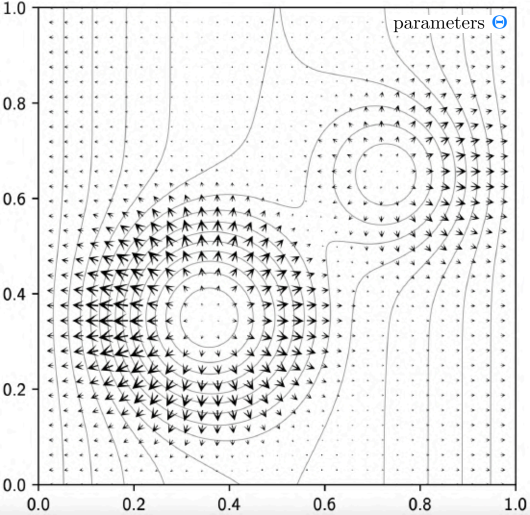


Figure adapted from [Fleuret, 2022]

Training

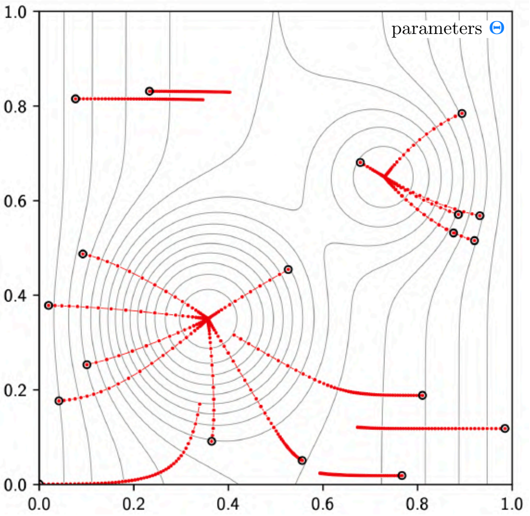


Figure adapted from [Fleuret, 2022]

Training

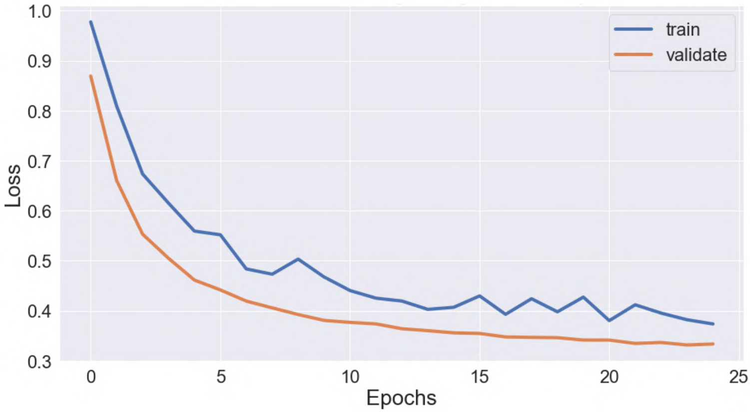
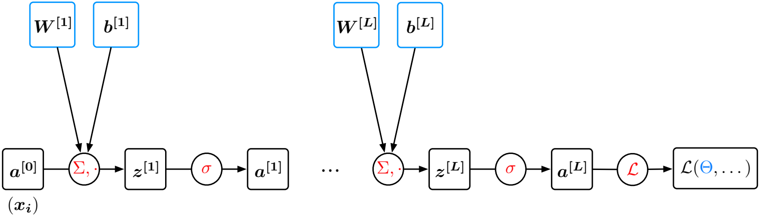


Figure taken from towardsdatascience.com

Graphe computationnel d'un réseau de neurones (forward pass)

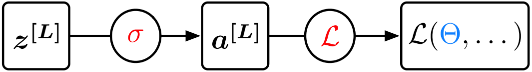


- On veut calculer les gradients:

$$\nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}(\boldsymbol{\Theta}) := \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial \boldsymbol{W}^{[l]}} \quad \text{et} \quad \nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}(\boldsymbol{\Theta}) := \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial \boldsymbol{b}^{[l]}}$$

pour $l = 1, \dots, M$

Calcul des gradients: équation 1

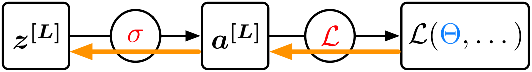


$$\delta_j^{[L]} := \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_j^{[L]}} = \sum_{k=1}^{|\boldsymbol{a}^{[L]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial a_k^{[L]}} \cdot \frac{\partial a_k^{[L]}}{\partial z_j^{[L]}}$$

$$= \sum_{k=1}^{|\boldsymbol{a}^{[L]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial a_k^{[L]}} \cdot \frac{\partial \sigma(z_k^{[L]})}{\partial z_j^{[L]}}$$

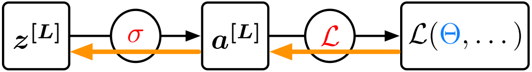
$$\left(\frac{\partial \sigma(z_k^{[L]})}{\partial z_j^{[L]}} = 0 \text{ for } k \neq j \right) = \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial a_j^{[L]}} \cdot \sigma'(z_j^{[L]})$$

Calcul des gradients: équation 1



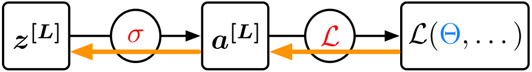
$$
\begin{aligned}
\delta_j^{[L]} \;:=\; \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_j^{[L]}} \quad & = \quad \sum_{k=1}^{|\boldsymbol{a}^{[L]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial a_k^{[L]}} \cdot \frac{\partial a_k^{[L]}}{\partial z_j^{[L]}} \\
& = \quad \sum_{k=1}^{|\boldsymbol{a}^{[L]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial a_k^{[L]}} \cdot \frac{\partial \sigma(z_k^{[L]})}{\partial z_j^{[L]}} \\
\left(\frac{\partial \sigma(z_k^{[L]})}{\partial z_j^{[L]}} = 0 \text{ for } k \neq j \right) \quad & = \quad \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial a_j^{[L]}} \cdot \sigma'(z_j^{[L]})
\end{aligned}
$$

Calcul des gradients: équation 1



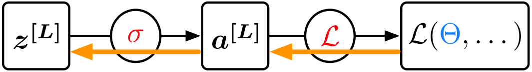
$$
\begin{aligned}
\delta_j^{[L]} \quad := \quad \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_j^{[L]}} \quad & = \quad \sum_{k=1}^{|\boldsymbol{a}^{[L]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial a_k^{[L]}} \cdot \frac{\partial a_k^{[L]}}{\partial z_j^{[L]}} \\
& = \quad \sum_{k=1}^{|\boldsymbol{a}^{[L]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial a_k^{[L]}} \cdot \frac{\partial \sigma(z_k^{[L]})}{\partial z_j^{[L]}} \\
\left(\frac{\partial \sigma(z_k^{[L]})}{\partial z_j^{[L]}} = 0 \text{ for } k \neq j \right) \quad & = \quad \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial a_j^{[L]}} \cdot \sigma'(z_j^{[L]})
\end{aligned}
$$

Calcul des gradients: équation 1



$$
\begin{aligned}
\delta_j^{[L]} := \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_j^{[L]}} &= \sum_{k=1}^{|\boldsymbol{a}^{[L]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial a_k^{[L]}} \cdot \frac{\partial a_k^{[L]}}{\partial z_j^{[L]}} \\
&= \sum_{k=1}^{|\boldsymbol{a}^{[L]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial a_k^{[L]}} \cdot \frac{\partial \sigma(z_k^{[L]})}{\partial z_j^{[L]}} \\
\left(\frac{\partial \sigma(z_k^{[L]})}{\partial z_j^{[L]}} = 0 \text{ for } k \neq j \right) &= \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial a_j^{[L]}} \cdot \sigma'(z_j^{[L]})
\end{aligned}
$$

Calcul des gradients: équation 1



$$
\begin{aligned}
\delta_j^{[L]} := \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_j^{[L]}} &= \sum_{k=1}^{|\boldsymbol{a}^{[L]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial a_k^{[L]}} \cdot \frac{\partial a_k^{[L]}}{\partial z_j^{[L]}} \\
&= \sum_{k=1}^{|\boldsymbol{a}^{[L]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial a_k^{[L]}} \cdot \frac{\partial \sigma(z_k^{[L]})}{\partial z_j^{[L]}} \\
\left(\frac{\partial \sigma(z_k^{[L]})}{\partial z_j^{[L]}} = 0 \text{ for } k \neq j \right) \quad &= \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial a_j^{[L]}} \cdot \sigma'(z_j^{[L]})
\end{aligned}
$$

Calcul des gradients: équation 1

Formulation vectorielle:

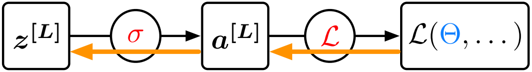
$$\boldsymbol{z}^{[\boldsymbol{L}]} \xrightarrow{\sigma} \boldsymbol{a}^{[\boldsymbol{L}]} \xrightarrow{\mathcal{L}} \mathcal{L}(\Theta, \dots)$$

$$\delta^{[L]} := \nabla_{z^{[L]}} \mathcal{L}(\Theta) = \nabla_{a^{[L]}} \mathcal{L}(\Theta) \odot \sigma'(z^{[L]})$$

où $\odot$ est le produit de Hadamard (composante par composante).

Calcul des gradients: équation 1

Formulation vectorielle:



$$\boldsymbol{\delta}^{[L]} := \nabla_{\boldsymbol{z}^{[L]}} \mathcal{L}(\boldsymbol{\Theta}) = \nabla_{\boldsymbol{a}^{[L]}} \mathcal{L}(\boldsymbol{\Theta}) \odot \boldsymbol{\sigma}'(\boldsymbol{z}^{[L]})$$

où $\odot$ est le produit de Hadamard (composante par composante).

Calcul des gradients: équation 1

Formulation vectorielle:

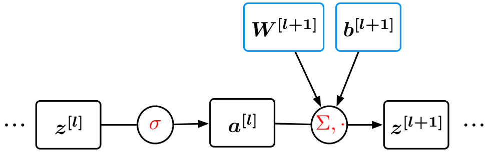
$$\boldsymbol{z}^{[L]} \xrightarrow{\sigma} \boldsymbol{a}^{[L]} \xrightarrow{\mathcal{L}} \mathcal{L}(\Theta, \dots)$$

$$\boldsymbol{\delta}^{[L]} := \nabla_{\boldsymbol{z}^{[L]}} \mathcal{L}(\boldsymbol{\Theta}) = \nabla_{\boldsymbol{a}^{[L]}} \mathcal{L}(\boldsymbol{\Theta}) \odot \boldsymbol{\sigma}'(\boldsymbol{z}^{[L]})$$

où $\odot$ est le produit de Hadamard (composante par composante).

Calcul des gradients: équation 2

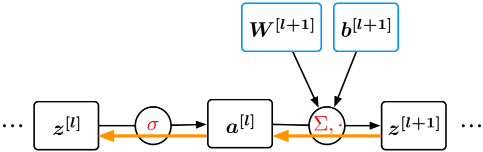
Supposons que les $\delta_k^{[l+1]}$ ont été calculés pour tous k:



$$
\begin{aligned}
\delta_j^{[l]} := \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_j^{[l]}} &= \sum_{k=1}^{|\boldsymbol{a}^{[l]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial a_k^{[l]}} \cdot \frac{\partial a_k^{[l]}}{\partial z_j^{[l]}} = \sum_{k=1}^{|\boldsymbol{a}^{[l]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial a_k^{[l]}} \cdot \frac{\partial \sigma(z_k^{[l]})}{\partial z_j^{[l]}} \\
&= \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial a_j^{[l]}} \cdot \sigma'(z_j^{[l]}) = \sum_{k=1}^{|\boldsymbol{z}^{[l+1]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_k^{[l+1]}} \cdot \frac{\partial z_k^{[l+1]}}{\partial a_j^{[l]}} \cdot \sigma'(z_j^{[l]}) \\
&= \sum_{k=1}^{|\boldsymbol{z}^{[l+1]}|} \delta_k^{[l+1]} \cdot \frac{\partial \left(\sum_{k'} (w_{kk'}^{[l+1]} \cdot a_{k'}^{[l]}) + b_k^{[l+1]} \right)}{\partial a_j^{[l]}} \cdot \sigma'(z_j^{[l]}) \\
&= \sum_{k=1}^{|\boldsymbol{z}^{[l+1]}|} \delta_k^{[l+1]} \cdot w_{kj}^{[l+1]} \cdot \sigma'(z_j^{[l]})
\end{aligned}
$$

Calcul des gradients: équation 2

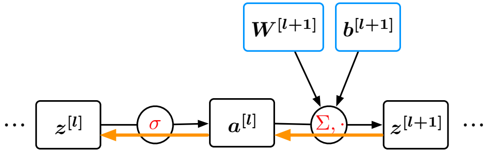
Supposons que les $\delta_k^{[l+1]}$ ont été calculés pour tous k:



$$
\begin{aligned}
\delta_j^{[l]} := \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_j^{[l]}} &= \sum_{k=1}^{|\boldsymbol{a}^{[l]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial a_k^{[l]}} \cdot \frac{\partial a_k^{[l]}}{\partial z_j^{[l]}} = \sum_{k=1}^{|\boldsymbol{a}^{[l]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial a_k^{[l]}} \cdot \frac{\partial \sigma(z_k^{[l]})}{\partial z_j^{[l]}} \\
&= \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial a_j^{[l]}} \cdot \sigma'(z_j^{[l]}) = \sum_{k=1}^{|\boldsymbol{z}^{[l+1]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_k^{[l+1]}} \cdot \frac{\partial z_k^{[l+1]}}{\partial a_j^{[l]}} \cdot \sigma'(z_j^{[l]}) \\
&= \sum_{k=1}^{|\boldsymbol{z}^{[l+1]}|} \delta_k^{[l+1]} \cdot \frac{\partial \left(\sum_{k'} (w_{kk'}^{[l+1]} \cdot a_{k'}^{[l]}) + b_k^{[l+1]} \right)}{\partial a_j^{[l]}} \cdot \sigma'(z_j^{[l]}) \\
&= \sum_{k=1}^{|\boldsymbol{z}^{[l+1]}|} \delta_k^{[l+1]} \cdot w_{kj}^{[l+1]} \cdot \sigma'(z_j^{[l]})
\end{aligned}
$$

Calcul des gradients: équation 2

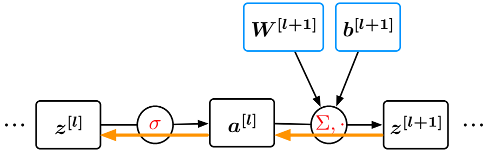
Supposons que les $\delta_k^{[l+1]}$ ont été calculés pour tous k:



$$
\begin{aligned}
\delta_j^{[l]} := \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_j^{[l]}} &= \sum_{k=1}^{|\boldsymbol{a}^{[l]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial a_k^{[l]}} \cdot \frac{\partial a_k^{[l]}}{\partial z_j^{[l]}} = \sum_{k=1}^{|\boldsymbol{a}^{[l]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial a_k^{[l]}} \cdot \frac{\partial \sigma(z_k^{[l]})}{\partial z_j^{[l]}} \\
&= \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial a_j^{[l]}} \cdot \sigma'(z_j^{[l]}) = \sum_{k=1}^{|\boldsymbol{z}^{[l+1]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_k^{[l+1]}} \cdot \frac{\partial z_k^{[l+1]}}{\partial a_j^{[l]}} \cdot \sigma'(z_j^{[l]}) \\
&= \sum_{k=1}^{|\boldsymbol{z}^{[l+1]}|} \delta_k^{[l+1]} \cdot \frac{\partial \left(\sum_{k'} (w_{kk'}^{[l+1]} \cdot a_{k'}^{[l]}) + b_k^{[l+1]}\right)}{\partial a_j^{[l]}} \cdot \sigma'(z_j^{[l]}) \\
&= \sum_{k=1}^{|\boldsymbol{z}^{[l+1]}|} \delta_k^{[l+1]} \cdot w_{kj}^{[l+1]} \cdot \sigma'(z_j^{[l]})
\end{aligned}
$$

Calcul des gradients: équation 2

Supposons que les $\delta_k^{[l+1]}$ ont été calculés pour tous k:



$$\delta_j^{[l]} := \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_j^{[l]}} = \sum_{k=1}^{|\boldsymbol{a}^{[l]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial a_k^{[l]}} \cdot \frac{\partial a_k^{[l]}}{\partial z_j^{[l]}} = \sum_{k=1}^{|\boldsymbol{a}^{[l]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial a_k^{[l]}} \cdot \frac{\partial \sigma(z_k^{[l]})}{\partial z_j^{[l]}}$$

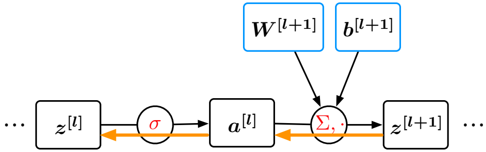
$$= \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial a_j^{[l]}} \cdot \sigma'(z_j^{[l]}) = \sum_{k=1}^{|\boldsymbol{z}^{[l+1]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_k^{[l+1]}} \cdot \frac{\partial z_k^{[l+1]}}{\partial a_j^{[l]}} \cdot \sigma'(z_j^{[l]})$$

$$= \sum_{k=1}^{|\boldsymbol{z}^{[l+1]}|} \delta_k^{[l+1]} \cdot \frac{\partial \left(\sum_{k'} (w_{kk'}^{[l+1]} \cdot a_{k'}^{[l]}) + b_k^{[l+1]}\right)}{\partial a_j^{[l]}} \cdot \sigma'(z_j^{[l]})$$

$$= \sum_{k=1}^{|\boldsymbol{z}^{[l+1]}|} \delta_k^{[l+1]} \cdot w_{kj}^{[l+1]} \cdot \sigma'(z_j^{[l]})$$

Calcul des gradients: équation 2

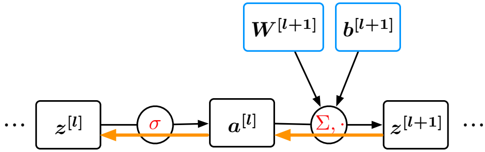
Supposons que les $\delta_k^{[l+1]}$ ont été calculés pour tous k:



$$
\begin{aligned}
\delta_j^{[l]} := \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_j^{[l]}} &= \sum_{k=1}^{|\boldsymbol{a}^{[l]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial a_k^{[l]}} \cdot \frac{\partial a_k^{[l]}}{\partial z_j^{[l]}} = \sum_{k=1}^{|\boldsymbol{a}^{[l]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial a_k^{[l]}} \cdot \frac{\partial \sigma(z_k^{[l]})}{\partial z_j^{[l]}} \\
&= \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial a_j^{[l]}} \cdot \sigma'(z_j^{[l]}) = \sum_{k=1}^{|\boldsymbol{z}^{[l+1]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_k^{[l+1]}} \cdot \frac{\partial z_k^{[l+1]}}{\partial a_j^{[l]}} \cdot \sigma'(z_j^{[l]}) \\
&= \sum_{k=1}^{|\boldsymbol{z}^{[l+1]}|} \delta_k^{[l+1]} \cdot \frac{\partial \left(\sum_{k'} (w_{kk'}^{[l+1]} \cdot a_{k'}^{[l]}) + b_k^{[l+1]} \right)}{\partial a_j^{[l]}} \cdot \sigma'(z_j^{[l]}) \\
&= \sum_{k=1}^{|\boldsymbol{z}^{[l+1]}|} \delta_k^{[l+1]} \cdot w_{kj}^{[l+1]} \cdot \sigma'(z_j^{[l]})
\end{aligned}
$$

Calcul des gradients: équation 2

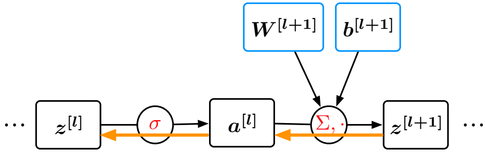
Supposons que les $\delta_k^{[l+1]}$ ont été calculés pour tous k:



$$
\begin{aligned}
\delta_j^{[l]} := \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_j^{[l]}} &= \sum_{k=1}^{|\boldsymbol{a}^{[l]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial a_k^{[l]}} \cdot \frac{\partial a_k^{[l]}}{\partial z_j^{[l]}} = \sum_{k=1}^{|\boldsymbol{a}^{[l]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial a_k^{[l]}} \cdot \frac{\partial \sigma(z_k^{[l]})}{\partial z_j^{[l]}} \\
&= \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial a_j^{[l]}} \cdot \sigma'(z_j^{[l]}) = \sum_{k=1}^{|\boldsymbol{z}^{[l+1]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_k^{[l+1]}} \cdot \frac{\partial z_k^{[l+1]}}{\partial a_j^{[l]}} \cdot \sigma'(z_j^{[l]}) \\
&= \sum_{k=1}^{|\boldsymbol{z}^{[l+1]}|} \delta_k^{[l+1]} \cdot \frac{\partial \left(\sum_{k'} (w_{kk'}^{[l+1]} \cdot a_{k'}^{[l]}) + b_k^{[l+1]} \right)}{\partial a_j^{[l]}} \cdot \sigma'(z_j^{[l]}) \\
&= \sum_{k=1}^{|\boldsymbol{z}^{[l+1]}|} \delta_k^{[l+1]} \cdot w_{kj}^{[l+1]} \cdot \sigma'(z_j^{[l]})
\end{aligned}
$$

Calcul des gradients: équation 2

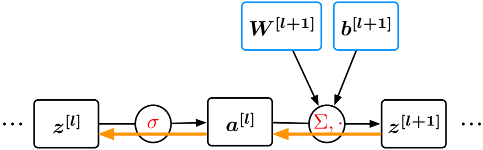
Supposons que les $\delta_k^{[l+1]}$ ont été calculés pour tous k:



$$
\begin{aligned}
\delta_j^{[l]} := \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_j^{[l]}} &= \sum_{k=1}^{|\boldsymbol{a}^{[l]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial a_k^{[l]}} \cdot \frac{\partial a_k^{[l]}}{\partial z_j^{[l]}} = \sum_{k=1}^{|\boldsymbol{a}^{[l]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial a_k^{[l]}} \cdot \frac{\partial \sigma(z_k^{[l]})}{\partial z_j^{[l]}} \\
&= \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial a_j^{[l]}} \cdot \sigma'(z_j^{[l]}) = \sum_{k=1}^{|\boldsymbol{z}^{[l+1]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_k^{[l+1]}} \cdot \frac{\partial z_k^{[l+1]}}{\partial a_j^{[l]}} \cdot \sigma'(z_j^{[l]}) \\
&= \sum_{k=1}^{|\boldsymbol{z}^{[l+1]}|} \delta_k^{[l+1]} \cdot \frac{\partial \left(\sum_{k'} (w_{kk'}^{[l+1]} \cdot a_{k'}^{[l]}) + b_k^{[l+1]} \right)}{\partial a_j^{[l]}} \cdot \sigma'(z_j^{[l]}) \\
&= \sum_{k=1}^{|\boldsymbol{z}^{[l+1]}|} \delta_k^{[l+1]} \cdot w_{kj}^{[l+1]} \cdot \sigma'(z_j^{[l]})
\end{aligned}
$$

Calcul des gradients: équation 2

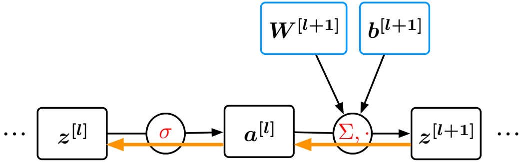
Supposons que les $\delta_k^{[l+1]}$ ont été calculés pour tous k:



$$
\begin{aligned}
\delta_j^{[l]} := \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_j^{[l]}} &= \sum_{k=1}^{|\boldsymbol{a}^{[l]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial a_k^{[l]}} \cdot \frac{\partial a_k^{[l]}}{\partial z_j^{[l]}} = \sum_{k=1}^{|\boldsymbol{a}^{[l]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial a_k^{[l]}} \cdot \frac{\partial \sigma(z_k^{[l]})}{\partial z_j^{[l]}} \\
&= \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial a_j^{[l]}} \cdot \sigma'(z_j^{[l]}) = \sum_{k=1}^{|\boldsymbol{z}^{[l+1]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_k^{[l+1]}} \cdot \frac{\partial z_k^{[l+1]}}{\partial a_j^{[l]}} \cdot \sigma'(z_j^{[l]}) \\
&= \sum_{k=1}^{|\boldsymbol{z}^{[l+1]}|} \delta_k^{[l+1]} \cdot \frac{\partial \left(\sum_{k'} (w_{kk'}^{[l+1]} \cdot a_{k'}^{[l]}) + b_k^{[l+1]} \right)}{\partial a_j^{[l]}} \cdot \sigma'(z_j^{[l]}) \\
&= \sum_{k=1}^{|\boldsymbol{z}^{[l+1]}|} \delta_k^{[l+1]} \cdot w_{kj}^{[l+1]} \cdot \sigma'(z_j^{[l]})
\end{aligned}
$$

Calcul des gradients: équation 2

Formulation vectorielle:

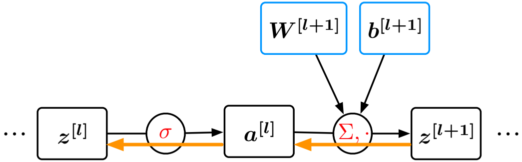


$$
\begin{aligned}
\delta^{[l]} &:= \nabla_{z^{[l]}} \mathcal{L}(\Theta) \\
&= \left[\delta^{[l+1]}\right]^T W^{[l+1]} \odot \sigma'(z^{[l]}) \\
&= \left[W^{[l+1]}\right]^T \delta^{[l+1]} \odot \sigma'(z^{[l]})
\end{aligned}
$$

où $\odot$ est le produit de Hadamard (composante par composante).

Calcul des gradients: équation 2

Formulation vectorielle:

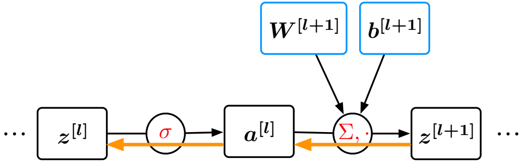


$$
\begin{aligned}
\boldsymbol{\delta}^{[l]} \quad &:= \quad \nabla_{\boldsymbol{z}^{[l]}} \mathcal{L}(\boldsymbol{\Theta}) \\
&= \quad \left[\boldsymbol{\delta}^{[l+1]}\right]^T \boldsymbol{W}^{[l+1]} \odot \boldsymbol{\sigma}'(\boldsymbol{z}^{[l]}) \\
&= \quad \left[\boldsymbol{W}^{[l+1]}\right]^T \boldsymbol{\delta}^{[l+1]} \odot \boldsymbol{\sigma}'(\boldsymbol{z}^{[l]})
\end{aligned}
$$

où $\odot$ est le produit de Hadamard (composante par composante).

Calcul des gradients: équation 2

Formulation vectorielle:

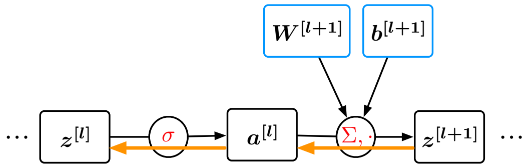


$$
\begin{aligned}
\boldsymbol{\delta}^{[l]} &:= \nabla_{\boldsymbol{z}^{[l]}} \mathcal{L}(\boldsymbol{\Theta}) \\
&= \left[\boldsymbol{\delta}^{[l+1]}\right]^T \boldsymbol{W}^{[l+1]} \odot \boldsymbol{\sigma}'(\boldsymbol{z}^{[l]}) \\
&= \left[\boldsymbol{W}^{[l+1]}\right]^T \boldsymbol{\delta}^{[l+1]} \odot \boldsymbol{\sigma}'(\boldsymbol{z}^{[l]})
\end{aligned}
$$

où $\odot$ est le produit de Hadamard (composante par composante).

Calcul des gradients: équation 2

Formulation vectorielle:

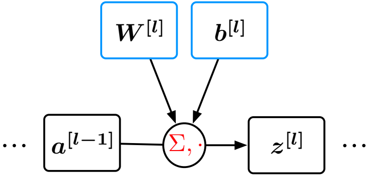


$$\begin{aligned}\boldsymbol{\delta}^{[l]} &:= \nabla_{\boldsymbol{z}^{[l]}} \mathcal{L}(\boldsymbol{\Theta}) \\ &= \left[\boldsymbol{\delta}^{[l+1]}\right]^T \boldsymbol{W}^{[l+1]} \odot \boldsymbol{\sigma}'(\boldsymbol{z}^{[l]}) \\ &= \left[\boldsymbol{W}^{[l+1]}\right]^T \boldsymbol{\delta}^{[l+1]} \odot \boldsymbol{\sigma}'(\boldsymbol{z}^{[l]})\end{aligned}$$

où $\odot$ est le produit de Hadamard (composante par composante).

Calcul des gradients: équation 3

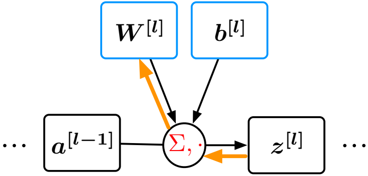
Calcul des gradients proprement dits en utilisant les erreurs δ_j^l:



$$
\begin{aligned}
\frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial w_{jk}^{[l]}} &= \sum_{k'=1}^{|\boldsymbol{z}^{[l]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_{k'}^{[l]}} \cdot \frac{\partial z_{k'}^{[l]}}{\partial w_{jk}^{[l]}} = \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_{j}^{[l]}} \cdot \frac{\partial z_{j}^{[l]}}{\partial w_{jk}^{[l]}} \\
&= \delta_j^{[l]} \cdot \frac{\partial \left(\sum_{k'} (w_{jk'}^{[l]} \cdot a_{k'}^{[l-1]}) + b_j^{[l]} \right)}{\partial w_{jk}^{[l]}} = \delta_j^{[l]} \cdot a_k^{[l-1]}
\end{aligned}
$$

Calcul des gradients: équation 3

Calcul des gradients proprement dits en utilisant les erreurs δ_j^l:

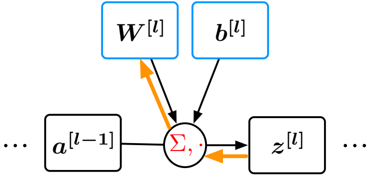


$$\frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial w_{jk}^{[l]}} = \sum_{k'=1}^{|\boldsymbol{z}^{[l]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_{k'}^{[l]}} \cdot \frac{\partial z_{k'}^{[l]}}{\partial w_{jk}^{[l]}} = \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_{j}^{[l]}} \cdot \frac{\partial z_{j}^{[l]}}{\partial w_{jk}^{[l]}}$$

$$= \delta_j^{[l]} \cdot \frac{\partial \left(\sum_{k'} (w_{jk'}^{[l]} \cdot a_{k'}^{[l-1]}) + b_j^{[l]} \right)}{\partial w_{jk}^{[l]}} = \delta_j^{[l]} \cdot a_k^{[l-1]}$$

Calcul des gradients: équation 3

Calcul des gradients proprement dits en utilisant les erreurs δ_j^l:

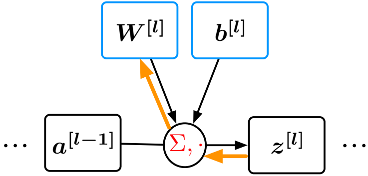


$$\frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial w_{jk}^{[l]}} = \sum_{k'=1}^{|\boldsymbol{z}^{[l]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_{k'}^{[l]}} \cdot \frac{\partial z_{k'}^{[l]}}{\partial w_{jk}^{[l]}} = \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_{j}^{[l]}} \cdot \frac{\partial z_{j}^{[l]}}{\partial w_{jk}^{[l]}}$$

$$= \delta_j^{[l]} \cdot \frac{\partial \left(\sum_{k'} (w_{jk'}^{[l]} \cdot a_{k'}^{[l-1]}) + b_j^{[l]} \right)}{\partial w_{jk}^{[l]}} = \delta_j^{[l]} \cdot a_k^{[l-1]}$$

Calcul des gradients: équation 3

Calcul des gradients proprement dits en utilisant les erreurs δ_j^l:

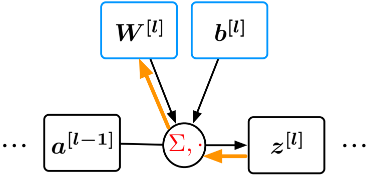


$$\frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial w_{jk}^{[l]}} = \sum_{k'=1}^{|\boldsymbol{z}^{[l]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_{k'}^{[l]}} \cdot \frac{\partial z_{k'}^{[l]}}{\partial w_{jk}^{[l]}} = \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_j^{[l]}} \cdot \frac{\partial z_j^{[l]}}{\partial w_{jk}^{[l]}}$$

$$= \delta_j^{[l]} \cdot \frac{\partial \left(\sum_{k'} (w_{jk'}^{[l]} \cdot a_{k'}^{[l-1]}) + b_j^{[l]} \right)}{\partial w_{jk}^{[l]}} = \delta_j^{[l]} \cdot a_k^{[l-1]}$$

Calcul des gradients: équation 3

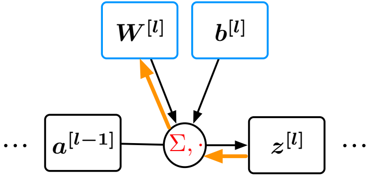
Calcul des gradients proprement dits en utilisant les erreurs δ_j^l:



$$
\begin{aligned}
\frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial w_{jk}^{[l]}} &= \sum_{k'=1}^{|\boldsymbol{z}^{[l]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_{k'}^{[l]}} \cdot \frac{\partial z_{k'}^{[l]}}{\partial w_{jk}^{[l]}} = \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_{j}^{[l]}} \cdot \frac{\partial z_{j}^{[l]}}{\partial w_{jk}^{[l]}} \\
&= \delta_j^{[l]} \cdot \frac{\partial \left(\sum_{k'} (w_{jk'}^{[l]} \cdot a_{k'}^{[l-1]}) + b_j^{[l]} \right)}{\partial w_{jk}^{[l]}} = \delta_j^{[l]} \cdot a_k^{[l-1]}
\end{aligned}
$$

Calcul des gradients: équation 3

Calcul des gradients proprement dits en utilisant les erreurs δ_j^l:

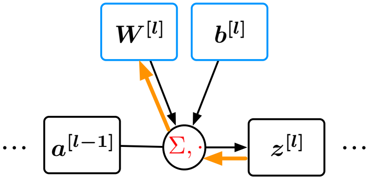


$$\frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial w_{jk}^{[l]}} = \sum_{k'=1}^{|\boldsymbol{z}^{[l]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_{k'}^{[l]}} \cdot \frac{\partial z_{k'}^{[l]}}{\partial w_{jk}^{[l]}} = \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_j^{[l]}} \cdot \frac{\partial z_j^{[l]}}{\partial w_{jk}^{[l]}}$$

$$= \delta_j^{[l]} \cdot \frac{\partial \left(\sum_{k'} (w_{jk'}^{[l]} \cdot a_{k'}^{[l-1]}) + b_j^{[l]} \right)}{\partial w_{jk}^{[l]}} = \delta_j^{[l]} \cdot a_k^{[l-1]}$$

Calcul des gradients: équation 3

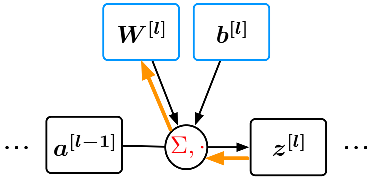
Formulation vectorielle:



$$\nabla_{W^{[l]}} \mathcal{L}(\Theta) = \delta^{[l]} \left[a^{[l-1]} \right]^T$$

Calcul des gradients: équation 3

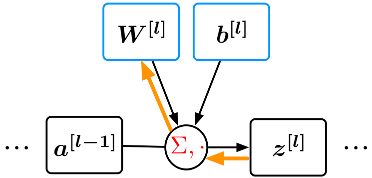
Formulation vectorielle:



$$\nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}(\boldsymbol{\Theta}) = \delta^{[l]} \left[a^{[l-1]} \right]^T$$

Calcul des gradients: équation 3

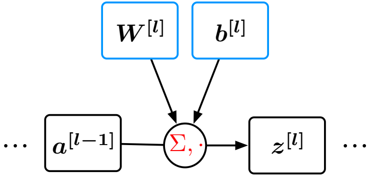
Formulation vectorielle:



$$\nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}(\boldsymbol{\Theta}) = \boldsymbol{\delta}^{[l]} \left[\boldsymbol{a}^{[l-1]}\right]^T$$

Calcul des gradients: équation 4

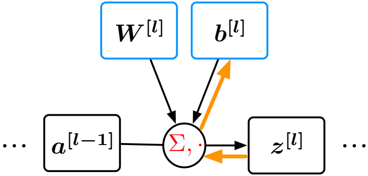
Calcul des gradients proprement dits en utilisant les erreurs δ_j^l:



$$\begin{aligned}
\frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial b_j^{[l]}} &= \sum_{k'=1}^{|\boldsymbol{z}^{[l]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_{k'}^{[l]}} \cdot \frac{\partial z_{k'}^{[l]}}{\partial b_j^{[l]}} = \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_j^{[l]}} \cdot \frac{\partial z_j^{[l]}}{\partial b_j^{[l]}} \\
&= \delta_j^{[l]} \cdot \frac{\partial \left(\sum_{k'} (w_{jk'}^{[l]} \cdot a_{k'}^{[l-1]}) + b_j^{[l]} \right)}{\partial b_j^{[l]}} = \delta_j^{[l]}
\end{aligned}$$

Calcul des gradients: équation 4

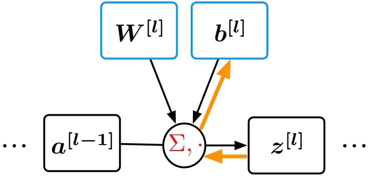
Calcul des gradients proprement dits en utilisant les erreurs δ_j^l:



$$\begin{aligned}
\frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial b_j^{[l]}} &= \sum_{k'=1}^{|\boldsymbol{z}^{[l]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_{k'}^{[l]}} \cdot \frac{\partial z_{k'}^{[l]}}{\partial b_j^{[l]}} = \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_j^{[l]}} \cdot \frac{\partial z_j^{[l]}}{\partial b_j^{[l]}} \\
&= \delta_j^{[l]} \cdot \frac{\partial \left(\sum_{k'} (w_{jk'}^{[l]} \cdot a_{k'}^{[l-1]}) + b_j^{[l]} \right)}{\partial b_j^{[l]}} = \delta_j^{[l]}
\end{aligned}$$

Calcul des gradients: équation 4

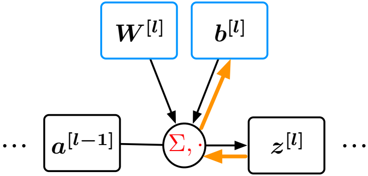
Calcul des gradients proprement dits en utilisant les erreurs δ_j^l:



$$
\begin{aligned}
\frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial b_j^{[l]}} &= \sum_{k'=1}^{|\boldsymbol{z}^{[l]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_{k'}^{[l]}} \cdot \frac{\partial z_{k'}^{[l]}}{\partial b_j^{[l]}} = \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_j^{[l]}} \cdot \frac{\partial z_j^{[l]}}{\partial b_j^{[l]}} \\
&= \delta_j^{[l]} \cdot \frac{\partial \left(\sum_{k'} (w_{jk'}^{[l]} \cdot a_{k'}^{[l-1]}) + b_j^{[l]} \right)}{\partial b_j^{[l]}} = \delta_j^{[l]}
\end{aligned}
$$

Calcul des gradients: équation 4

Calcul des gradients proprement dits en utilisant les erreurs δ_j^l:

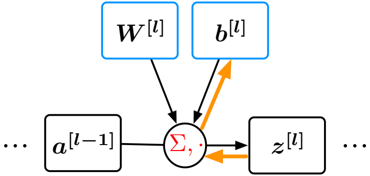


$$\frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial b_j^{[l]}} = \sum_{k'=1}^{|\boldsymbol{z}^{[l]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_{k'}^{[l]}} \cdot \frac{\partial z_{k'}^{[l]}}{\partial b_j^{[l]}} = \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_j^{[l]}} \cdot \frac{\partial z_j^{[l]}}{\partial b_j^{[l]}}$$

$$= \delta_j^{[l]} \cdot \frac{\partial \left(\sum_{k'} (w_{jk'}^{[l]} \cdot a_{k'}^{[l-1]}) + b_j^{[l]} \right)}{\partial b_j^{[l]}} = \delta_j^{[l]}$$

Calcul des gradients: équation 4

Calcul des gradients proprement dits en utilisant les erreurs δ_j^l:

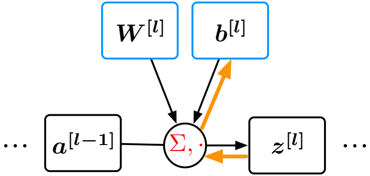


$$\frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial b_j^{[l]}} = \sum_{k'=1}^{|\boldsymbol{z}^{[l]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_{k'}^{[l]}} \cdot \frac{\partial z_{k'}^{[l]}}{\partial b_j^{[l]}} = \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_j^{[l]}} \cdot \frac{\partial z_j^{[l]}}{\partial b_j^{[l]}}$$

$$= \delta_j^{[l]} \cdot \frac{\partial \left(\sum_{k'} (w_{jk'}^{[l]} \cdot a_{k'}^{[l-1]}) + b_j^{[l]} \right)}{\partial b_j^{[l]}} = \delta_j^{[l]}$$

Calcul des gradients: équation 4

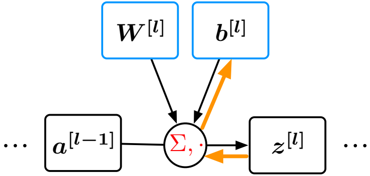
Calcul des gradients proprement dits en utilisant les erreurs δ_j^l:



$$\begin{aligned}
\frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial b_j^{[l]}} &= \sum_{k'=1}^{|\boldsymbol{z}^{[l]}|} \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_{k'}^{[l]}} \cdot \frac{\partial z_{k'}^{[l]}}{\partial b_j^{[l]}} = \frac{\partial \mathcal{L}(\boldsymbol{\Theta})}{\partial z_j^{[l]}} \cdot \frac{\partial z_j^{[l]}}{\partial b_j^{[l]}} \\
&= \delta_j^{[l]} \cdot \frac{\partial \left(\sum_{k'} (w_{jk'}^{[l]} \cdot a_{k'}^{[l-1]}) + b_j^{[l]} \right)}{\partial b_j^{[l]}} = \delta_j^{[l]}
\end{aligned}$$

Calcul des gradients: équation 4

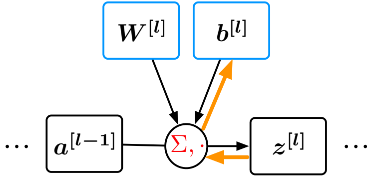
Formulation vectorielle:



$$\nabla_{b^{[l]}} \mathcal{L}(\Theta) = \delta^{[l]}$$

Calcul des gradients: équation 4

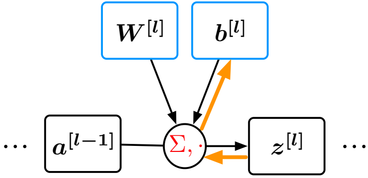
Formulation vectorielle:



$$\nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}(\boldsymbol{\Theta}) = \delta^{[l]}$$

Calcul des gradients: équation 4

Formulation vectorielle:



$$\nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}(\boldsymbol{\Theta}) \; = \; \boldsymbol{\delta}^{[l]}$$

Équations

Calcul des *erreurs* $\boldsymbol{\delta}^{[l]} := \nabla_{\boldsymbol{z}^{[l]}} \mathcal{L}(\boldsymbol{\Theta})$ et des *gradients* $\nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}(\boldsymbol{\Theta})$ et $\nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}(\boldsymbol{\Theta})$, pour toute couche $l = L, \ldots, 1$:

$$\boldsymbol{\delta}^{[l]} = \begin{cases} \nabla_{\boldsymbol{a}^{[l]}} \mathcal{L}(\boldsymbol{\Theta}) \odot \boldsymbol{\sigma}'(\boldsymbol{z}^{[l]}), & \text{si } l = L \\ \left[\boldsymbol{W}^{[l+1]}\right]^T \boldsymbol{\delta}^{[l+1]} \odot \boldsymbol{\sigma}'(\boldsymbol{z}^{[l]}), & \text{si } L > l \geq 1 \end{cases} \tag{1}$$

Pour $l = L, \ldots, 1$:

$$\nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}(\boldsymbol{\Theta}) = \boldsymbol{\delta}^{[l]} \left[\boldsymbol{a}^{[l-1]}\right]^T \tag{2}$$

$$\nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}(\boldsymbol{\Theta}) = \boldsymbol{\delta}^{[l]} \tag{3}$$

Équations

Calcul des *erreurs* $\boldsymbol{\delta}^{[l]} := \nabla_{\boldsymbol{z}^{[l]}} \mathcal{L}(\boldsymbol{\Theta})$ et des *gradients* $\nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}(\boldsymbol{\Theta})$ et $\nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}(\boldsymbol{\Theta})$, pour toute couche $l = L, \ldots, 1$:

$$\boldsymbol{\delta}^{[l]} = \begin{cases} \nabla_{\boldsymbol{a}^{[l]}} \mathcal{L}(\boldsymbol{\Theta}) \odot \boldsymbol{\sigma}'(\boldsymbol{z}^{[l]}), & \text{si } l = L \\ \left[\boldsymbol{W}^{[l+1]}\right]^T \boldsymbol{\delta}^{[l+1]} \odot \boldsymbol{\sigma}'(\boldsymbol{z}^{[l]}), & \text{si } L > l \geq 1 \end{cases} \tag{1}$$

Pour $l = L, \ldots, 1$:

$$\nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}(\boldsymbol{\Theta}) = \boldsymbol{\delta}^{[l]} \left[\boldsymbol{a}^{[l-1]}\right]^T \tag{2}$$

$$\nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}(\boldsymbol{\Theta}) = \boldsymbol{\delta}^{[l]} \tag{3}$$

Équations

Calcul des *erreurs* $\boldsymbol{\delta}^{[l]} := \nabla_{\boldsymbol{z}^{[l]}} \mathcal{L}(\boldsymbol{\Theta})$ et des *gradients* $\nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}(\boldsymbol{\Theta})$ et $\nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}(\boldsymbol{\Theta})$, pour toute couche $l = L, \ldots, 1$:

$$\boldsymbol{\delta}^{[l]} = \begin{cases} \nabla_{\boldsymbol{a}^{[l]}} \mathcal{L}(\boldsymbol{\Theta}) \odot \boldsymbol{\sigma}'(\boldsymbol{z}^{[l]}), & \text{si } l = L \\ \left[\boldsymbol{W}^{[l+1]}\right]^T \boldsymbol{\delta}^{[l+1]} \odot \boldsymbol{\sigma}'(\boldsymbol{z}^{[l]}), & \text{si } L > l \geq 1 \end{cases} \quad (1)$$

Pour $l = L, \ldots, 1$:

$$\nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}(\boldsymbol{\Theta}) = \boldsymbol{\delta}^{[l]} \left[\boldsymbol{a}^{[l-1]}\right]^T \quad (2)$$

$$\nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}(\boldsymbol{\Theta}) = \boldsymbol{\delta}^{[l]} \quad (3)$$

Équations

Calcul des *erreurs* $\boldsymbol{\delta}^{[l]} := \nabla_{\boldsymbol{z}^{[l]}} \mathcal{L}(\boldsymbol{\Theta})$ et des *gradients* $\nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}(\boldsymbol{\Theta})$ et $\nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}(\boldsymbol{\Theta})$, pour toute couche $l = L, \ldots, 1$:

$$\boldsymbol{\delta}^{[l]} = \begin{cases} \nabla_{\boldsymbol{a}^{[l]}} \mathcal{L}(\boldsymbol{\Theta}) \odot \boldsymbol{\sigma}'(\boldsymbol{z}^{[l]}), & \text{si } l = L \\ \left[\boldsymbol{W}^{[l+1]}\right]^T \boldsymbol{\delta}^{[l+1]} \odot \boldsymbol{\sigma}'(\boldsymbol{z}^{[l]}), & \text{si } L > l \geq 1 \end{cases} \tag{1}$$

Pour $l = L, \ldots, 1$:

$$\nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}(\boldsymbol{\Theta}) = \boldsymbol{\delta}^{[l]} \left[\boldsymbol{a}^{[l-1]}\right]^T \tag{2}$$

$$\nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}(\boldsymbol{\Theta}) = \boldsymbol{\delta}^{[l]} \tag{3}$$

Équations

Calcul des *erreurs* $\boldsymbol{\delta}^{[l]} := \nabla_{\boldsymbol{z}^{[l]}} \mathcal{L}(\boldsymbol{\Theta})$ et des *gradients* $\nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}(\boldsymbol{\Theta})$ et $\nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}(\boldsymbol{\Theta})$, pour toute couche $l = L, \ldots, 1$:

$$\boldsymbol{\delta}^{[l]} = \begin{cases} \nabla_{\boldsymbol{a}^{[l]}} \mathcal{L}(\boldsymbol{\Theta}) \odot \boldsymbol{\sigma}'(\boldsymbol{z}^{[l]}), & \text{si } l = L \\ \left[\boldsymbol{W}^{[l+1]}\right]^T \boldsymbol{\delta}^{[l+1]} \odot \boldsymbol{\sigma}'(\boldsymbol{z}^{[l]}), & \text{si } L > l \geq 1 \end{cases} \quad (1)$$

Pour $l = L, \ldots, 1$:

$$\nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}(\boldsymbol{\Theta}) = \boldsymbol{\delta}^{[l]} \left[\boldsymbol{a}^{[l-1]}\right]^T \quad (2)$$

$$\nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}(\boldsymbol{\Theta}) = \boldsymbol{\delta}^{[l]} \quad (3)$$

Équations

Calcul des *erreurs* $\boldsymbol{\delta}^{[l]} := \nabla_{\boldsymbol{z}^{[l]}} \mathcal{L}(\boldsymbol{\Theta})$ et des *gradients* $\nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}(\boldsymbol{\Theta})$ et $\nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}(\boldsymbol{\Theta})$, pour toute couche $l = L, \ldots, 1$:

$$\boldsymbol{\delta}^{[l]} = \begin{cases} \nabla_{\boldsymbol{a}^{[l]}} \mathcal{L}(\boldsymbol{\Theta}) \odot \boldsymbol{\sigma}'(\boldsymbol{z}^{[l]}), & \text{si } l = L \\ \left[\boldsymbol{W}^{[l+1]}\right]^T \boldsymbol{\delta}^{[l+1]} \odot \boldsymbol{\sigma}'(\boldsymbol{z}^{[l]}), & \text{si } L > l \geq 1 \end{cases} \tag{1}$$

Pour $l = L, \ldots, 1$:

$$\nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}(\boldsymbol{\Theta}) = \boldsymbol{\delta}^{[l]} \left[\boldsymbol{a}^{[l-1]}\right]^T \tag{2}$$

$$\nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}(\boldsymbol{\Theta}) = \boldsymbol{\delta}^{[l]} \tag{3}$$

Équations

Calcul des *erreurs* $\boldsymbol{\delta}^{[l]} := \nabla_{\boldsymbol{z}^{[l]}} \mathcal{L}(\boldsymbol{\Theta})$ et des *gradients* $\nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}(\boldsymbol{\Theta})$ et $\nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}(\boldsymbol{\Theta})$, pour toute couche $l = L, \dots, 1$:

$$\boldsymbol{\delta}^{[l]} = \begin{cases} \nabla_{\boldsymbol{a}^{[l]}} \mathcal{L}(\boldsymbol{\Theta}) \odot \boldsymbol{\sigma}'(\boldsymbol{z}^{[l]}), & \text{si } l = L \\ \left[\boldsymbol{W}^{[l+1]}\right]^T \boldsymbol{\delta}^{[l+1]} \odot \boldsymbol{\sigma}'(\boldsymbol{z}^{[l]}), & \text{si } L > l \geq 1 \end{cases} \tag{1}$$

Pour $l = L, \dots, 1$:

$$\nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}(\boldsymbol{\Theta}) = \boldsymbol{\delta}^{[l]} \left[\boldsymbol{a}^{[l-1]}\right]^T \tag{2}$$

$$\nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}(\boldsymbol{\Theta}) = \boldsymbol{\delta}^{[l]} \tag{3}$$

Équations

Une fois gradients calculés, on effectue l'update des poids et biais (cf. gradient descent algo):

$$\text{Pour } l = L, \ldots, 1 :$$

$$\boldsymbol{W}^{[l]} := \boldsymbol{W}^{[l]} - \eta \cdot \nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}(\boldsymbol{\Theta}) \tag{4}$$

$$\boldsymbol{b}^{[l]} := \boldsymbol{b}^{[l]} - \eta \cdot \nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}(\boldsymbol{\Theta}) \tag{5}$$

où η est le *learning rate*.

Équations

Une fois gradients calculés, on effectue l'update des poids et biais (cf. gradient descent algo):

$$
\begin{aligned}
\text{Pour } l &= L, \ldots, 1 : \\
\boldsymbol{W}^{[l]} &:= \boldsymbol{W}^{[l]} - \eta \cdot \nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}(\boldsymbol{\Theta}) \qquad (4) \\
\boldsymbol{b}^{[l]} &:= \boldsymbol{b}^{[l]} - \eta \cdot \nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}(\boldsymbol{\Theta}) \qquad (5)
\end{aligned}
$$

où η est le *learning rate*.

Équations

Une fois gradients calculés, on effectue l'update des poids et biais (cf. gradient descent algo):

$$\text{Pour } l = L, \ldots, 1 :$$

$$\boldsymbol{W}^{[l]} := \boldsymbol{W}^{[l]} - \eta \cdot \nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}(\boldsymbol{\Theta}) \tag{4}$$

$$\boldsymbol{b}^{[l]} := \boldsymbol{b}^{[l]} - \eta \cdot \nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}(\boldsymbol{\Theta}) \tag{5}$$

où η est le *learning rate*.

Équations: batched

Remarque: on peut déduire une version "batched" des équations.

- Soit $B = (X, Y)$ un batch composé de B inputs et outputs x_k et y_k alignés en deux matrices:

$$X = \begin{pmatrix} \vdots & \vdots & \cdots & \vdots \\ x_1 & x_2 & \cdots & x_B \\ \vdots & \vdots & \cdots & \vdots \end{pmatrix} \text{ et } Y = \begin{pmatrix} \vdots & \vdots & \cdots & \vdots \\ y_1 & y_2 & \cdots & y_B \\ \vdots & \vdots & \cdots & \vdots \end{pmatrix}$$

- Soit $A^{[L]}$ les outputs du réseau associés aux inputs X:

$$A^{[L]} = \begin{pmatrix} \vdots & \vdots & \cdots & \vdots \\ a_1^{[L]} & a_2^{[L]} & \cdots & a_B^{[L]} \\ \vdots & \vdots & \cdots & \vdots \end{pmatrix}$$

- Soit $\mathcal{L}_k(\Theta) = \mathcal{L}(\Theta, a_k^{[L]}, y_k)$ la loss associée à l'exemple k, pour $k = 1, \ldots, B$.

Équations: batched

Remarque: on peut déduire une version "batched" des équations.

- Soit $\boldsymbol{B} = (\boldsymbol{X}, \boldsymbol{Y})$ un batch composé de B inputs et outputs $\boldsymbol{x_k}$ et $\boldsymbol{y_k}$ alignés en deux matrices:

$$\boldsymbol{X} = \begin{pmatrix} \vdots & \vdots & \cdots & \vdots \\ \boldsymbol{x_1} & \boldsymbol{x_2} & \cdots & \boldsymbol{x_B} \\ \vdots & \vdots & \cdots & \vdots \end{pmatrix} \text{ et } \boldsymbol{Y} = \begin{pmatrix} \vdots & \vdots & \cdots & \vdots \\ \boldsymbol{y_1} & \boldsymbol{y_2} & \cdots & \boldsymbol{y_B} \\ \vdots & \vdots & \cdots & \vdots \end{pmatrix}$$

- Soit $\boldsymbol{A}^{[L]}$ les outputs du réseau associés aux inputs $\boldsymbol{X}$:

$$\boldsymbol{A}^{[L]} = \begin{pmatrix} \vdots & \vdots & \cdots & \vdots \\ \boldsymbol{a}_1^{[L]} & \boldsymbol{a}_2^{[L]} & \cdots & \boldsymbol{a}_B^{[L]} \\ \vdots & \vdots & \cdots & \vdots \end{pmatrix}$$

- Soit $\mathcal{L}_k(\boldsymbol{\Theta}) := \mathcal{L}(\boldsymbol{\Theta}, \boldsymbol{a}_k^{[L]}, \boldsymbol{y_k}))$ la loss associée à l'exemple k, pour $k = 1, \ldots, B$.

Équations: batched

Remarque: on peut déduire une version "batched" des équations.

- Soit $\boldsymbol{B} = (\boldsymbol{X}, \boldsymbol{Y})$ un batch composé de B inputs et outputs $\boldsymbol{x_k}$ et $\boldsymbol{y_k}$ alignés en deux matrices:

$$\boldsymbol{X} = \begin{pmatrix} \vdots & \vdots & \cdots & \vdots \\ \boldsymbol{x_1} & \boldsymbol{x_2} & \cdots & \boldsymbol{x_B} \\ \vdots & \vdots & \cdots & \vdots \end{pmatrix} \text{ et } \boldsymbol{Y} = \begin{pmatrix} \vdots & \vdots & \cdots & \vdots \\ \boldsymbol{y_1} & \boldsymbol{y_2} & \cdots & \boldsymbol{y_B} \\ \vdots & \vdots & \cdots & \vdots \end{pmatrix}$$

- Soit $\boldsymbol{A}^{[L]}$ les outputs du réseau associés aux inputs $\boldsymbol{X}$:

$$\boldsymbol{A}^{[L]} = \begin{pmatrix} \vdots & \vdots & \cdots & \vdots \\ \boldsymbol{a}_{\boldsymbol{1}}^{[L]} & \boldsymbol{a}_{\boldsymbol{2}}^{[L]} & \cdots & \boldsymbol{a}_{\boldsymbol{B}}^{[L]} \\ \vdots & \vdots & \cdots & \vdots \end{pmatrix}$$

- Soit $\mathcal{L}_k(\boldsymbol{\Theta}) := \mathcal{L}(\boldsymbol{\Theta}, \boldsymbol{a}_{\boldsymbol{k}}^{[L]}, \boldsymbol{y_k}))$ la loss associée à l'exemple k, pour $k = 1, \ldots, B$.

Équations: batched

Remarque: on peut déduire une version "batched" des équations.

- Soit $\boldsymbol{B} = (\boldsymbol{X}, \boldsymbol{Y})$ un batch composé de B inputs et outputs $\boldsymbol{x_k}$ et $\boldsymbol{y_k}$ alignés en deux matrices:

$$\boldsymbol{X} = \begin{pmatrix} \vdots & \vdots & \cdots & \vdots \\ \boldsymbol{x_1} & \boldsymbol{x_2} & \cdots & \boldsymbol{x_B} \\ \vdots & \vdots & \cdots & \vdots \end{pmatrix} \text{ et } \boldsymbol{Y} = \begin{pmatrix} \vdots & \vdots & \cdots & \vdots \\ \boldsymbol{y_1} & \boldsymbol{y_2} & \cdots & \boldsymbol{y_B} \\ \vdots & \vdots & \cdots & \vdots \end{pmatrix}$$

- Soit $\boldsymbol{A}^{[L]}$ les outputs du réseau associés aux inputs $\boldsymbol{X}$:

$$\boldsymbol{A}^{[L]} = \begin{pmatrix} \vdots & \vdots & \cdots & \vdots \\ \boldsymbol{a}_1^{[L]} & \boldsymbol{a}_2^{[L]} & \cdots & \boldsymbol{a}_B^{[L]} \\ \vdots & \vdots & \cdots & \vdots \end{pmatrix}$$

- Soit $\mathcal{L}_k(\boldsymbol{\Theta}) := \mathcal{L}(\boldsymbol{\Theta}, \boldsymbol{a}_k^{[L]}, \boldsymbol{y_k}))$ la loss associée à l'exemple k, pour $k = 1, \ldots, B$.

Équations: batched

Calcul des *erreurs* $\boldsymbol{\delta}_{\boldsymbol{k}}^{[l]} := \nabla_{\boldsymbol{z}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta})$ et des *gradients* $\nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta})$ et $\nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta})$, pour tout exemple $k = 1, \ldots, B$ pour toute couche $l = L, \ldots, 1$:

Pour $k = 1, \ldots, B$:

$$\boldsymbol{\delta}_{\boldsymbol{k}}^{[l]} = \begin{cases} \nabla_{\boldsymbol{a}^{[L]}} \mathcal{L}_k(\boldsymbol{\Theta}) \odot \boldsymbol{\sigma}'(\boldsymbol{z}_{\boldsymbol{k}}^{[L]}), & \text{si } l = L \\ \left[\boldsymbol{W}^{[l+1]}\right]^T \boldsymbol{\delta}_{\boldsymbol{k}}^{[l+1]} \odot \boldsymbol{\sigma}'(\boldsymbol{z}_{\boldsymbol{k}}^{[l]}), & \text{si } L > l \geq 1 \end{cases} \tag{6}$$

Pour $l = L, \ldots, 1$ et pour $k = 1, \ldots, B$:

$$\nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta}) = \boldsymbol{\delta}_{\boldsymbol{k}}^{[l]} \left[\boldsymbol{a}_{\boldsymbol{k}}^{[l-1]}\right]^T \tag{7}$$

$$\nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta}) = \boldsymbol{\delta}^{[l]} \tag{8}$$

Équations: batched

Calcul des *erreurs* $\boldsymbol{\delta}_{\boldsymbol{k}}^{[l]} := \nabla_{\boldsymbol{z}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta})$ et des *gradients* $\nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta})$ et $\nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta})$, pour tout exemple $k = 1, \ldots, B$ pour toute couche $l = L, \ldots, 1$:

$$\text{Pour } k = 1, \ldots, B :$$

$$\boldsymbol{\delta}_{\boldsymbol{k}}^{[l]} = \begin{cases} \nabla_{\boldsymbol{a}^{[L]}} \mathcal{L}_k(\boldsymbol{\Theta}) \odot \boldsymbol{\sigma}'(\boldsymbol{z}_{\boldsymbol{k}}^{[L]}), & \text{si } l = L \\ \left[\boldsymbol{W}^{[l+1]}\right]^T \boldsymbol{\delta}_{\boldsymbol{k}}^{[l+1]} \odot \boldsymbol{\sigma}'(\boldsymbol{z}_{\boldsymbol{k}}^{[l]}), & \text{si } L > l \geq 1 \end{cases} \tag{6}$$

$$\text{Pour } l = L, \ldots, 1 \text{ et pour } k = 1, \ldots, B :$$

$$\nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta}) = \boldsymbol{\delta}_{\boldsymbol{k}}^{[l]} \left[\boldsymbol{a}_{\boldsymbol{k}}^{[l-1]}\right]^T \tag{7}$$

$$\nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta}) = \boldsymbol{\delta}^{[l]} \tag{8}$$

Équations: batched

Calcul des *erreurs* $\boldsymbol{\delta_k^{[l]}} := \nabla_{\boldsymbol{z}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta})$ et des *gradients* $\nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta})$ et $\nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta})$, pour tout exemple $k = 1, \ldots, B$ pour toute couche $l = L, \ldots, 1$:

$$\text{Pour } k \;=\; 1, \ldots, B :$$

$$\boldsymbol{\delta_k^{[l]}} = \begin{cases} \nabla_{\boldsymbol{a}^{[L]}} \mathcal{L}_k(\boldsymbol{\Theta}) \odot \boldsymbol{\sigma'}(\boldsymbol{z_k^{[L]}}), & \text{si } l = L \\ \left[\boldsymbol{W}^{[l+1]}\right]^T \boldsymbol{\delta_k^{[l+1]}} \odot \boldsymbol{\sigma'}(\boldsymbol{z_k^{[l]}}), & \text{si } L > l \geq 1 \end{cases} \tag{6}$$

$$\text{Pour } l \;=\; L, \ldots, 1 \text{ et pour } k \;=\; 1, \ldots, B :$$

$$\nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta}) = \boldsymbol{\delta_k^{[l]}} \left[\boldsymbol{a_k^{[l-1]}}\right]^T \tag{7}$$

$$\nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta}) = \boldsymbol{\delta}^{[l]} \tag{8}$$

Équations: batched

Calcul des *erreurs* $\boldsymbol{\delta}_{\boldsymbol{k}}^{[l]} := \nabla_{\boldsymbol{z}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta})$ et des *gradients* $\nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta})$ et $\nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta})$, pour tout exemple $k = 1, \ldots, B$ pour toute couche $l = L, \ldots, 1$:

$$
\begin{aligned}
\text{Pour } k &= 1, \ldots, B : \\
\boldsymbol{\delta}_{\boldsymbol{k}}^{[l]} &= \begin{cases} \nabla_{\boldsymbol{a}^{[L]}} \mathcal{L}_k(\boldsymbol{\Theta}) \odot \boldsymbol{\sigma'}(\boldsymbol{z}_{\boldsymbol{k}}^{[L]}), & \text{si } l = L \\ \left[\boldsymbol{W}^{[l+1]}\right]^T \boldsymbol{\delta}_{\boldsymbol{k}}^{[l+1]} \odot \boldsymbol{\sigma'}(\boldsymbol{z}_{\boldsymbol{k}}^{[l]}), & \text{si } L > l \geq 1 \end{cases} \qquad (6) \\
\text{Pour } l &= L, \ldots, 1 \text{ et pour } k = 1, \ldots, B : \\
\nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta}) &= \boldsymbol{\delta}_{\boldsymbol{k}}^{[l]} \left[\boldsymbol{a}_{\boldsymbol{k}}^{[l-1]}\right]^T \qquad (7) \\
\nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta}) &= \boldsymbol{\delta}^{[l]} \qquad (8)
\end{aligned}
$$

Équations: batched

Calcul des *erreurs* $\boldsymbol{\delta}_{\boldsymbol{k}}^{[l]} := \nabla_{\boldsymbol{z}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta})$ et des *gradients* $\nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta})$ et $\nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta})$, pour tout exemple $k = 1, \dots, B$ pour toute couche $l = L, \dots, 1$:

$$\text{Pour } k = 1, \dots, B :$$

$$\boldsymbol{\delta}_{\boldsymbol{k}}^{[l]} = \begin{cases} \nabla_{\boldsymbol{a}^{[L]}} \mathcal{L}_k(\boldsymbol{\Theta}) \odot \boldsymbol{\sigma}'(\boldsymbol{z}_{\boldsymbol{k}}^{[L]}), & \text{si } l = L \\ \left[\boldsymbol{W}^{[l+1]}\right]^T \boldsymbol{\delta}_{\boldsymbol{k}}^{[l+1]} \odot \boldsymbol{\sigma}'(\boldsymbol{z}_{\boldsymbol{k}}^{[l]}), & \text{si } L > l \geq 1 \end{cases} \tag{6}$$

$$\text{Pour } l = L, \dots, 1 \text{ et pour } k = 1, \dots, B :$$

$$\nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta}) = \boldsymbol{\delta}_{\boldsymbol{k}}^{[l]} \left[\boldsymbol{a}_{\boldsymbol{k}}^{[l-1]}\right]^T \tag{7}$$

$$\nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta}) = \boldsymbol{\delta}^{[l]} \tag{8}$$

Équations: batched

Calcul des *erreurs* $\boldsymbol{\delta}_{\boldsymbol{k}}^{[l]} := \nabla_{\boldsymbol{z}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta})$ et des *gradients* $\nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta})$ et $\nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta})$, pour tout exemple $k = 1, \ldots, B$ pour toute couche $l = L, \ldots, 1$:

$$\text{Pour } k = 1, \ldots, B :$$

$$\boldsymbol{\delta}_{\boldsymbol{k}}^{[l]} = \begin{cases} \nabla_{\boldsymbol{a}^{[L]}} \mathcal{L}_k(\boldsymbol{\Theta}) \odot \boldsymbol{\sigma'}(\boldsymbol{z}_{\boldsymbol{k}}^{[L]}), & \text{si } l = L \\ \left[\boldsymbol{W}^{[l+1]}\right]^T \boldsymbol{\delta}_{\boldsymbol{k}}^{[l+1]} \odot \boldsymbol{\sigma'}(\boldsymbol{z}_{\boldsymbol{k}}^{[l]}), & \text{si } L > l \geq 1 \end{cases} \tag{6}$$

$$\text{Pour } l = L, \ldots, 1 \text{ et pour } k = 1, \ldots, B :$$

$$\nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta}) = \boldsymbol{\delta}_{\boldsymbol{k}}^{[l]} \left[\boldsymbol{a}_{\boldsymbol{k}}^{[l-1]}\right]^T \tag{7}$$

$$\nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta}) = \boldsymbol{\delta}^{[l]} \tag{8}$$

Équations: batched

Une fois gradients calculés pour tout $k = 1, \ldots, B$, on effectue l'update des poids et biais (cf. stochastic gradient descent algo):

$$\text{Pour } l = L, \ldots, 1 :$$

$$\boldsymbol{W}^{[l]} := \boldsymbol{W}^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^{B} \nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta}) \tag{9}$$

$$\boldsymbol{b}^{[l]} := \boldsymbol{b}^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^{B} \nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta}) \tag{10}$$

où η est le *learning rate*.

Équations: batched

Une fois gradients calculés pour tout $k = 1, \ldots, B$, on effectue l'update des poids et biais (cf. stochastic gradient descent algo):

$$\text{Pour } l = L, \ldots, 1 :$$

$$\boldsymbol{W}^{[l]} := \boldsymbol{W}^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^{B} \nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta}) \tag{9}$$

$$\boldsymbol{b}^{[l]} := \boldsymbol{b}^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^{B} \nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta}) \tag{10}$$

où η est le *learning rate*.

Équations: batched

Une fois gradients calculés pour tout $k = 1, \ldots, B$, on effectue l'update des poids et biais (cf. stochastic gradient descent algo):

$$\text{Pour } l = L, \ldots, 1:$$

$$\boldsymbol{W}^{[l]} := \boldsymbol{W}^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^{B} \nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta}) \tag{9}$$

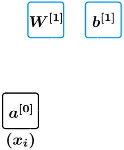
$$\boldsymbol{b}^{[l]} := \boldsymbol{b}^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^{B} \nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta}) \tag{10}$$

où η est le *learning rate*.

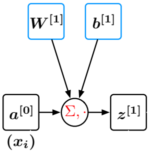
Forward Pass and Backward Pass

$$\boxed{\boldsymbol{a}^{[0]}}$$
$(\boldsymbol{x}_i)$

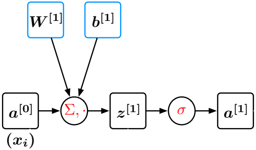
Forward Pass and Backward Pass



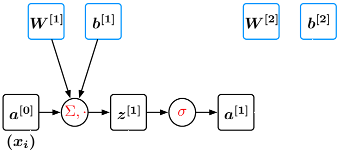
Forward Pass and Backward Pass



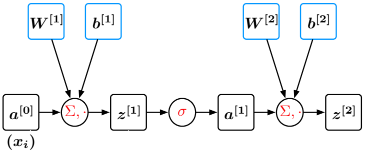
Forward Pass and Backward Pass



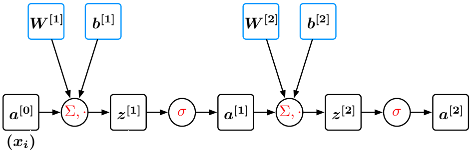
Forward Pass and Backward Pass



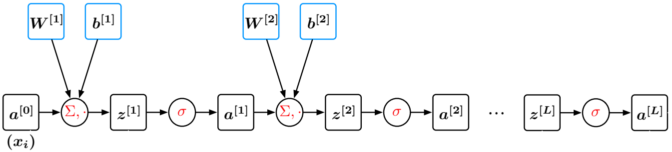
Forward Pass and Backward Pass



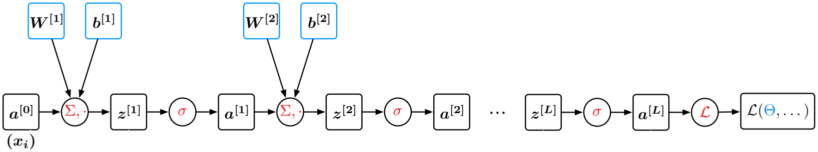
Forward Pass and Backward Pass



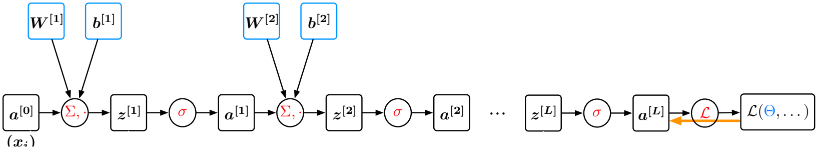
Forward Pass and Backward Pass



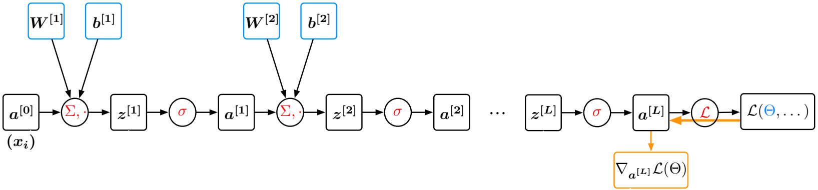
Forward Pass and Backward Pass



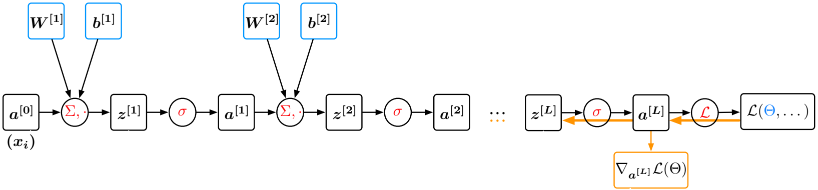
Forward Pass and Backward Pass



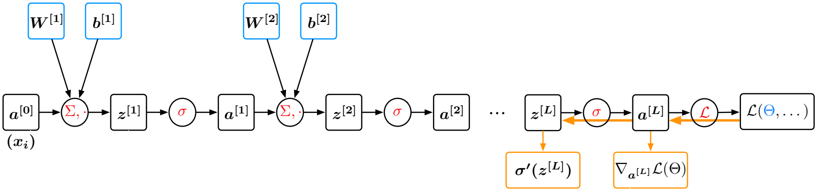
Forward Pass and Backward Pass



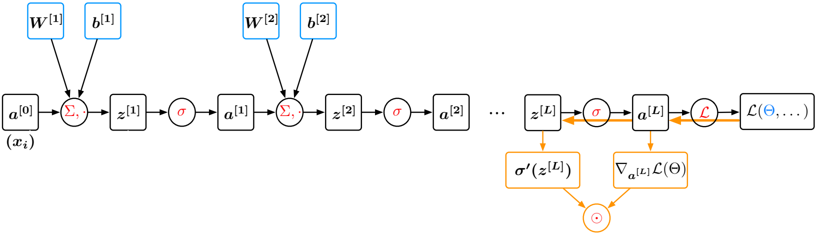
Forward Pass and Backward Pass



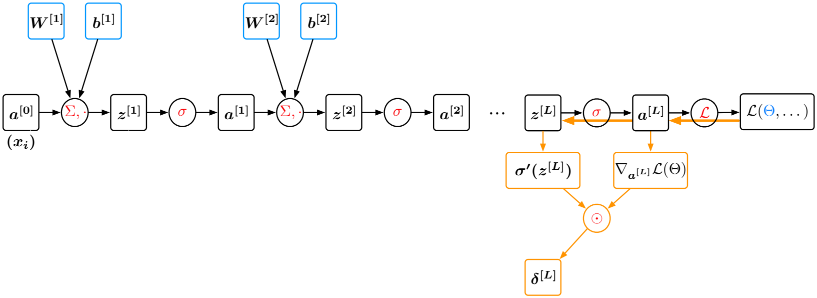
Forward Pass and Backward Pass



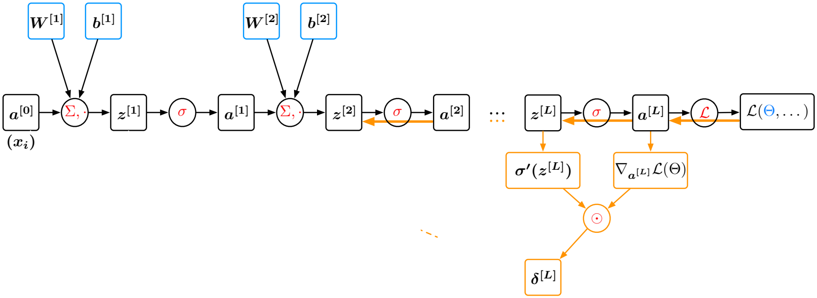
Forward Pass and Backward Pass



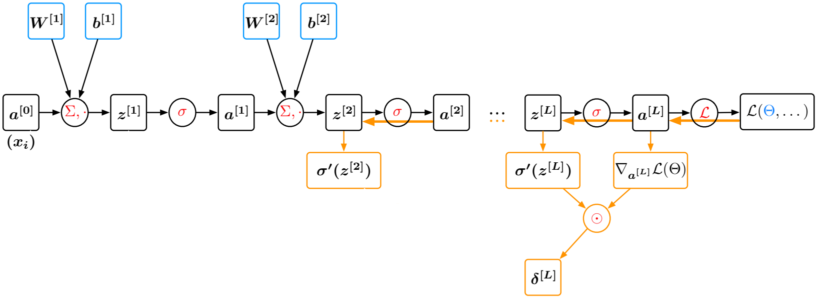
Forward Pass and Backward Pass



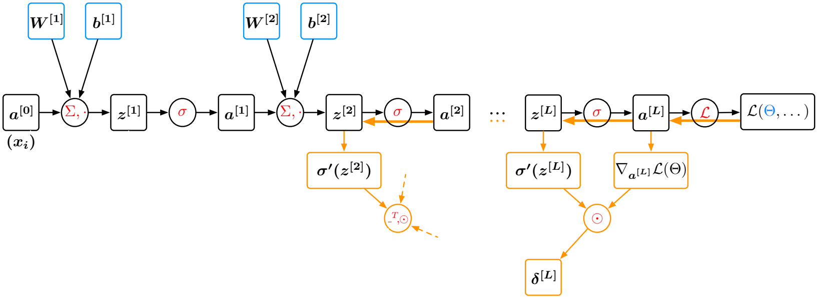
Forward Pass and Backward Pass



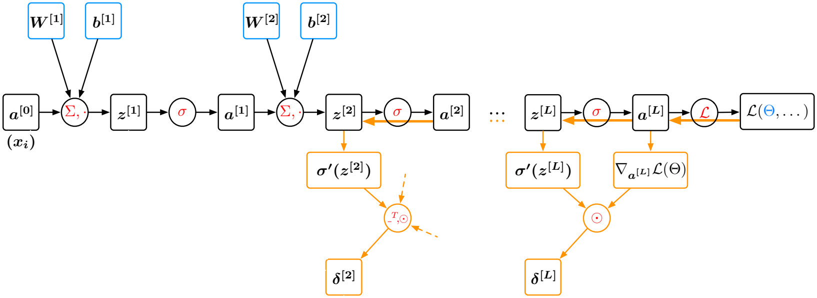
Forward Pass and Backward Pass



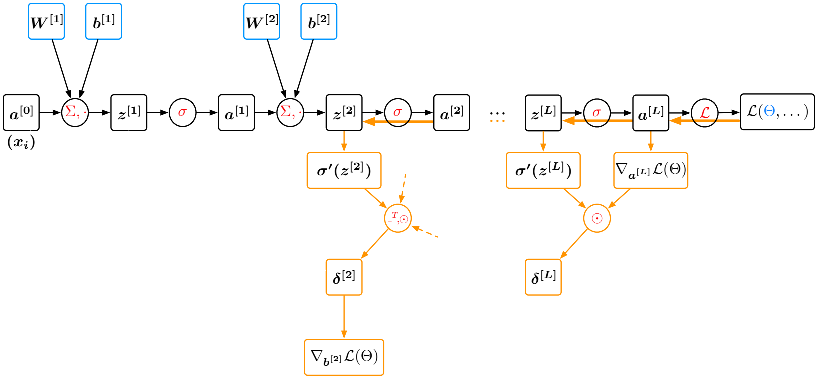
Forward Pass and Backward Pass



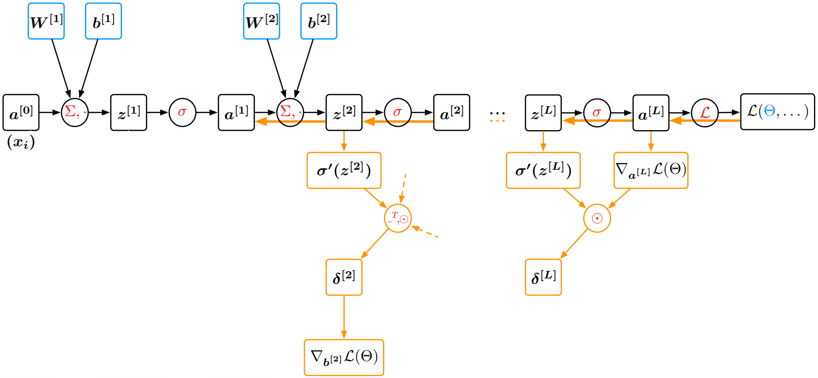
Forward Pass and Backward Pass



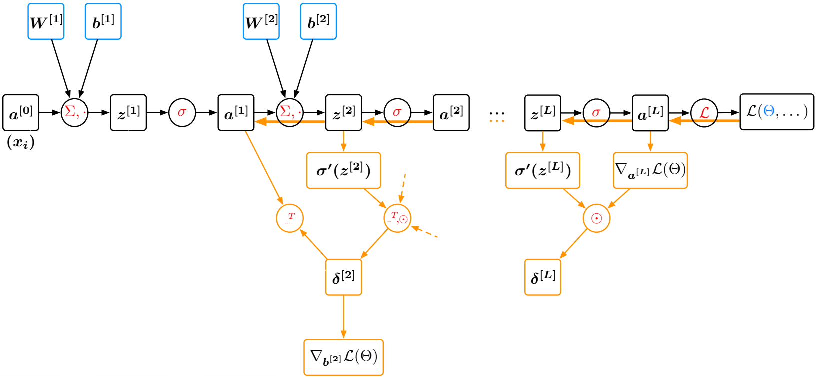
Forward Pass and Backward Pass



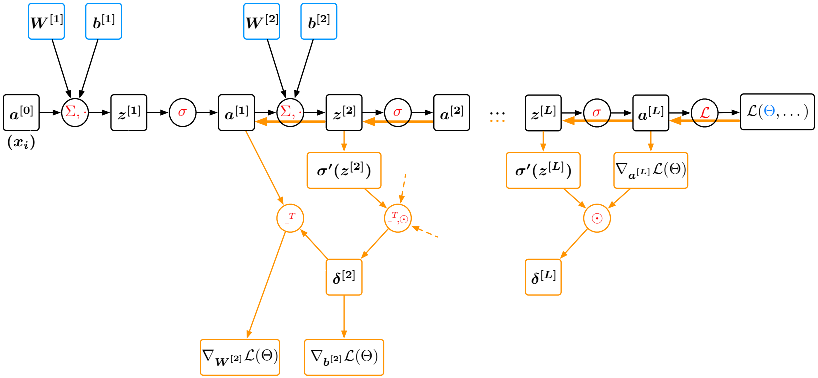
Forward Pass and Backward Pass



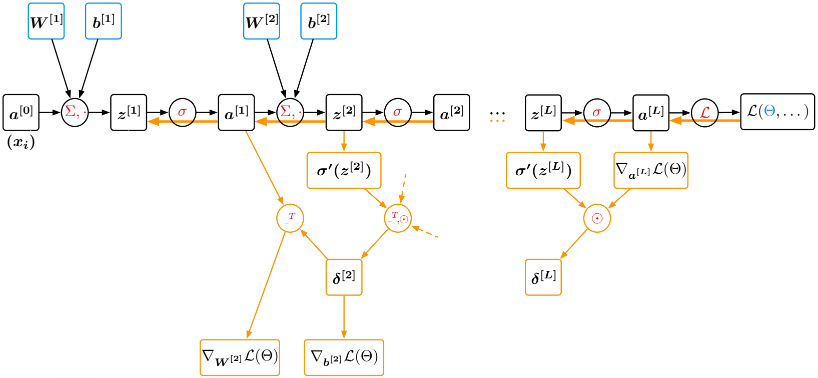
Forward Pass and Backward Pass



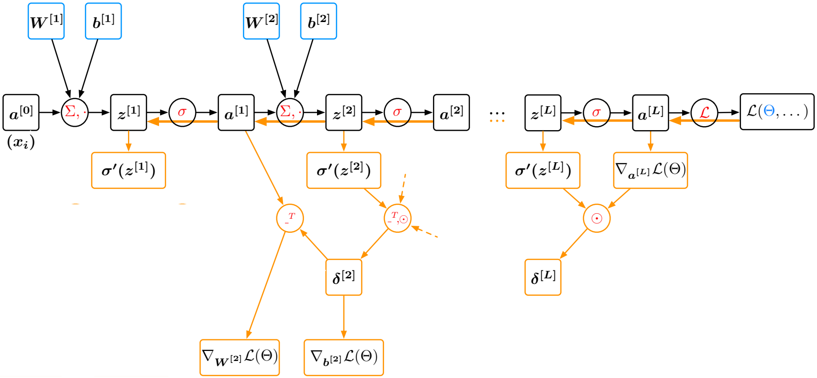
Forward Pass and Backward Pass



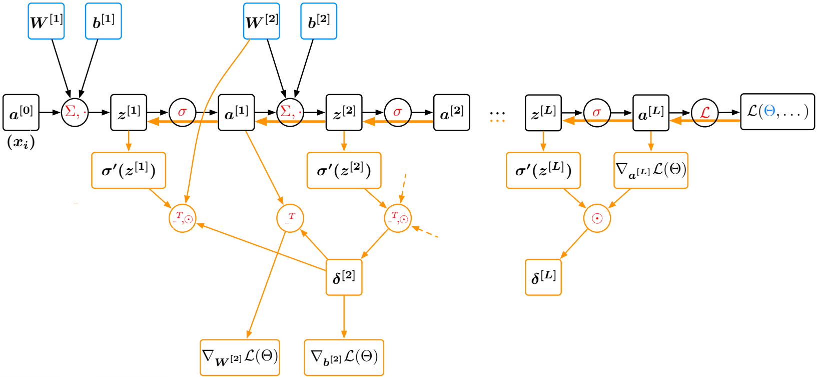
Forward Pass and Backward Pass



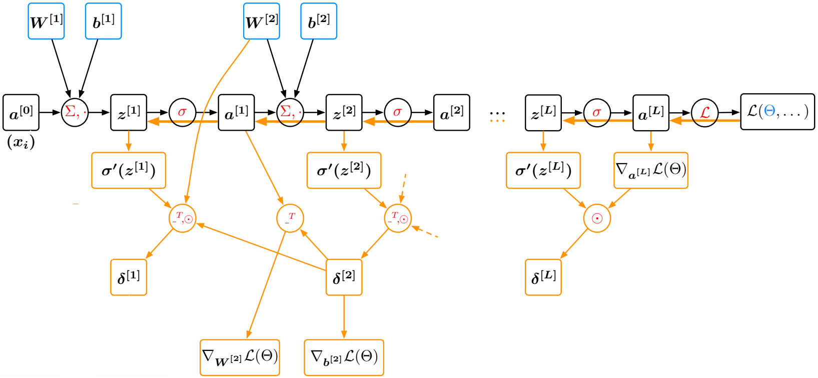
Forward Pass and Backward Pass



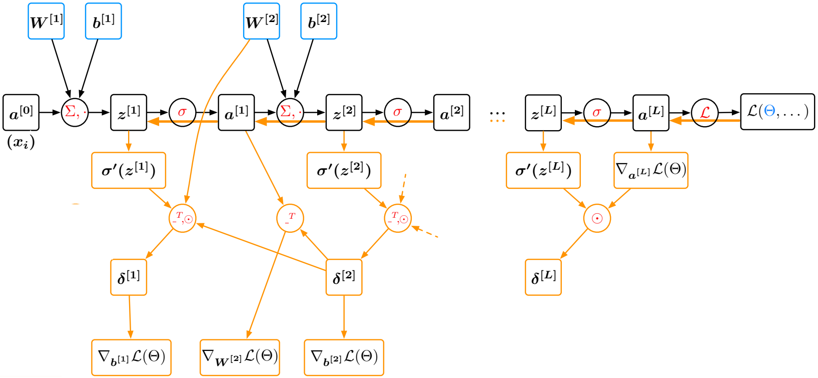
Forward Pass and Backward Pass



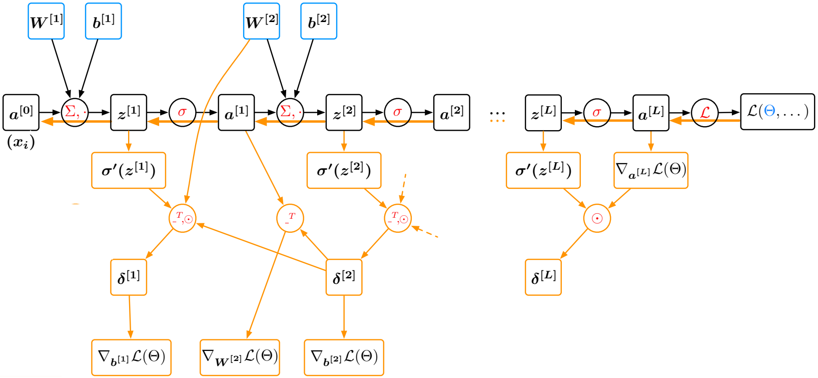
Forward Pass and Backward Pass



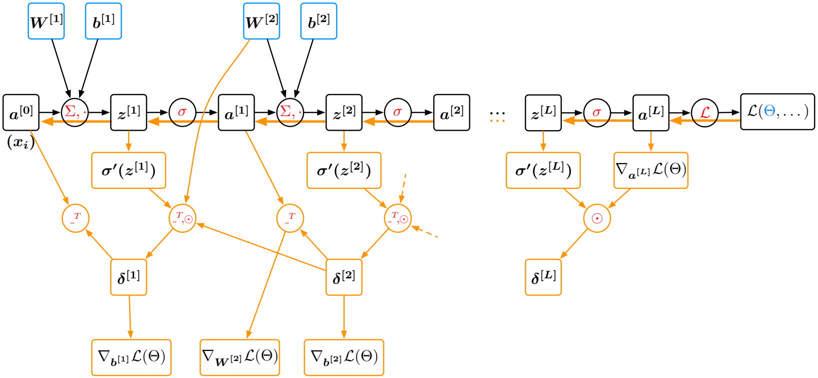
Forward Pass and Backward Pass



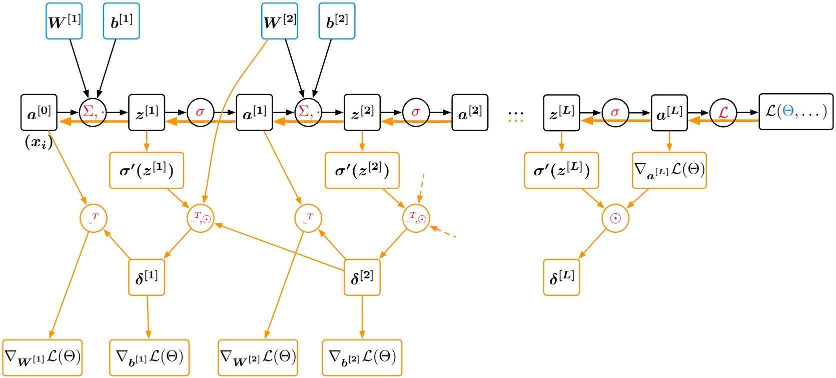
Forward Pass and Backward Pass



Forward Pass and Backward Pass



Forward Pass and Backward Pass



Backpropagation (BP)

Algorithm 1: Backpropagation (stochastic gradient descent)

Data: dataloader $= \{\boldsymbol{B_b} = (\boldsymbol{X_b}, \boldsymbol{Y_b}) : b = 1, \ldots, nb\_batches\}$

Inputs: MLP $= \left\{\left(\boldsymbol{W}^{[l]}, \boldsymbol{b}^{[l]}\right) : l = 1, \ldots, L\right\}$ initialized randomly

```
for epochs = 1 to nb_epochs do                          // loop over epochs
    for k = 1 to nb_batches do                          // loop over batches
        for l = 1 to L do                               // forward pass
            ...
            ...
        end
        for l = L to 1 do                               // backward pass
            ...                                         // compute error
            ...                                         // update gradient
            ...                                         // update gradient
        end
    end
end
```

Backpropagation (BP)

Algorithm 1: Backpropagation (stochastic gradient descent)

Data: dataloader $= \{\boldsymbol{B_b} = (\boldsymbol{X_b}, \boldsymbol{Y_b}) : b = 1, \dots, nb\_batches\}$
Inputs: MLP $= \left\{\left(\boldsymbol{W}^{[l]}, \boldsymbol{b}^{[l]}\right) : l = 1, \dots, L\right\}$ initialized randomly

```
for epochs = 1 to nb_epochs do                          // loop over epochs
    for k = 1 to nb_batches do                          // loop over batches
        for l = 1 to L do                               // forward pass
            ...
            ...
        end
        for l = L to 1 do                               // backward pass
            ...                                         // compute error
            ...                                         // update gradient
            ...                                         // update gradient
        end
    end
end
```

Backpropagation (BP)

Algorithm 1: Backpropagation (stochastic gradient descent)

Data: dataloader $= \{\boldsymbol{B_b} = (\boldsymbol{X_b}, \boldsymbol{Y_b}) : b = 1, \ldots, nb\_batches\}$

Inputs: MLP $= \left\{ \left(\boldsymbol{W}^{[l]}, \boldsymbol{b}^{[l]} \right) : l = 1, \ldots, L \right\}$ initialized randomly

```
for epochs = 1 to nb_epochs do                         // loop over epochs
    for k = 1 to nb_batches do                         // loop over batches
        for l = 1 to L do                              // forward pass
            ...
            ...
        end
        for l = L to 1 do                              // backward pass
            ...                                        // compute error
            ...                                        // update gradient
            ...                                        // update gradient
        end
    end
end
```

Backpropagation (BP)

Algorithm 1: Backpropagation (stochastic gradient descent)

Data: dataloader $= \{\boldsymbol{B_b} = (\boldsymbol{X_b}, \boldsymbol{Y_b}) : b = 1, \ldots, nb\_batches\}$

Inputs: MLP $= \left\{\left(\boldsymbol{W}^{[l]}, \boldsymbol{b}^{[l]}\right) : l = 1, \ldots, L\right\}$ initialized randomly

```
for epochs = 1 to nb_epochs do                        // loop over epochs
    for k = 1 to nb_batches do                        // loop over batches
        for l = 1 to L do                             // forward pass
            ...
            ...
        end
        for l = L to 1 do                             // backward pass
            ...                                       // compute error
            ...                                       // update gradient
            ...                                       // update gradient
        end
    end
end
```

Backpropagation (BP)

Algorithm 1: Backpropagation (stochastic gradient descent)

Data: dataloader $= \{\boldsymbol{B_b} = (\boldsymbol{X_b}, \boldsymbol{Y_b}) : b = 1, \ldots, nb\_batches\}$

Inputs: MLP $= \left\{\left(\boldsymbol{W}^{[l]}, \boldsymbol{b}^{[l]}\right) : l = 1, \ldots, L\right\}$ initialized randomly

for $epochs = 1$ ***to*** nb_epochs **do** // loop over epochs

  for $k = 1$ ***to*** $nb\_batches$ **do** // loop over batches

    for $l = 1$ ***to*** L **do** // forward pass

      ...

      ...

    end

    for $l = L$ ***to*** 1 **do** // backward pass

      ... // compute error

      ... // update gradient

      ... // update gradient

    end

  end

end

Backpropagation (BP)

Algorithm 1: Backpropagation (stochastic gradient descent)

Data: dataloader $= \{\boldsymbol{B_b} = (\boldsymbol{X_b}, \boldsymbol{Y_b}) : b = 1, \ldots, nb\_batches\}$

Inputs: MLP $= \left\{\left(\boldsymbol{W}^{[l]}, \boldsymbol{b}^{[l]}\right) : l = 1, \ldots, L\right\}$ initialized randomly

```
for epochs = 1 to nb_epochs do                    // loop over epochs
    for k = 1 to nb_batches do                    // loop over batches
        for l = 1 to L do                         // forward pass
            ...
            ...
        end
        for l = L to 1 do                         // backward pass
            ...                                   // compute error
            ...                                   // update gradient
            ...                                   // update gradient
        end
    end
end
```

Backpropagation (BP)

Algorithm 1: Backpropagation (stochastic gradient descent)

Data: dataloader $= \{\boldsymbol{B_b} = (\boldsymbol{X_b}, \boldsymbol{Y_b}) : b = 1, \ldots, nb\_batches\}$

Inputs: MLP $= \left\{ \left(\boldsymbol{W}^{[l]}, \boldsymbol{b}^{[l]} \right) : l = 1, \ldots, L \right\}$ initialized randomly

```
for epochs = 1 to nb_epochs do                       // loop over epochs
    for k = 1 to nb_batches do                       // loop over batches
        for l = 1 to L do                            // forward pass
            ...
            ...
        end
        for l = L to 1 do                            // backward pass
            ...                                      // compute error
            ...                                      // update gradient
            ...                                      // update gradient
        end
    end
end
```

Backpropagation (BP)

Algorithm 1: Backpropagation (stochastic gradient descent)

Data: dataloader $= \{\boldsymbol{B_b} = (\boldsymbol{X_b}, \boldsymbol{Y_b}) : b = 1, \ldots, nb\_batches\}$

Inputs: MLP $= \left\{ \left(\boldsymbol{W}^{[l]}, \boldsymbol{b}^{[l]} \right) : l = 1, \ldots, L \right\}$ initialized randomly

for $epochs = 1$ ***to*** nb_epochs **do** `// loop over epochs`
    for $k = 1$ ***to*** $nb\_batches$ **do** `// loop over batches`
        for $l = 1$ ***to*** L **do** `// forward pass`
            $\ldots$
            $\ldots$
        end
        for $l = L$ ***to*** 1 **do** `// backward pass`
            $\ldots$ `// compute error`
            $\ldots$ `// update gradient`
            $\ldots$ `// update gradient`
        end
    end
end

Backpropagation (BP)

Algorithm 1: Backpropagation (stochastic gradient descent)

Data: dataloader = $\{\boldsymbol{B_b} = (\boldsymbol{X_b}, \boldsymbol{Y_b}) : b = 1, \ldots, nb\_batches\}$

Inputs: MLP = $\left\{\left(\boldsymbol{W}^{[l]}, \boldsymbol{b}^{[l]}\right) : l = 1, \ldots, L\right\}$ initialized randomly

```
for epochs = 1 to nb_epochs do                          // loop over epochs
    for k = 1 to nb_batches do                          // loop over batches
        for l = 1 to L do                               // forward pass
            ...
            ...
        end
        for l = L to 1 do                               // backward pass
            ...                                         // compute error
            ...                                         // update gradient
            ...                                         // update gradient
        end
    end
end
```

Backpropagation (BP)

Algorithm 1: Backpropagation (stochastic gradient descent)

Data: dataloader $= \{\boldsymbol{B_b} = (\boldsymbol{X_b}, \boldsymbol{Y_b}) : b = 1, \dots, nb\_batches\}$

Inputs: MLP $= \left\{\left(\boldsymbol{W}^{[l]}, \boldsymbol{b}^{[l]}\right) : l = 1, \dots, L\right\}$ initialized randomly

```
for epochs = 1 to nb_epochs do                    // loop over epochs
    for k = 1 to nb_batches do                    // loop over batches
        for l = 1 to L do                         // forward pass
            ...
            ...
        end
        for l = L to 1 do                         // backward pass
            ...                                   // compute error
            ...                                   // update gradient
            ...                                   // update gradient
        end
    end
end
```

Backpropagation (BP)

Algorithm 2: Backpropagation (stochastic gradient descent)

Data: dataloader $= \{\boldsymbol{B_b} = (\boldsymbol{X_b}, \boldsymbol{Y_b}) : b = 1, \ldots, nb\_batches\}$

Inputs: MLP $= \left\{\left(\boldsymbol{W}^{[l]}, \boldsymbol{b}^{[l]}\right) : l = 1, \ldots, L\right\}$ initialized randomly

for $epochs = 1$ *to* nb_epochs **do** // loop over epochs
  for $b = 1$ *to* $nb\_batches$ **do** // loop over batches
    $\boldsymbol{A}^{[0]} = \boldsymbol{X_b}$
    for $l = 1$ *to* L **do** // forward pass
      $\boldsymbol{Z}^{[l]} = \boldsymbol{W}^{[l]}\boldsymbol{A}^{[l-1]} + \boldsymbol{b}^{[l]}$
      $\boldsymbol{A}^{[l]} = \boldsymbol{\sigma}\left(\boldsymbol{Z}^{[l]}\right)$
    end
    for $l = L$ *to* 1 **do** // backward pass
      if $l = L$ **then** // compute error
        $\boldsymbol{\delta}_k^{[l]} = \nabla_{\boldsymbol{a}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta}) \odot \boldsymbol{\sigma}'(\boldsymbol{z}_k^{[l]})$ for $k = 1, \ldots, B$
      else if $L > l \geq 1$ **then**
        $\boldsymbol{\delta}_k^{[l]} = \left[\boldsymbol{W}^{[l+1]}\right]^T \boldsymbol{\delta}_k^{[l+1]} \odot \boldsymbol{\sigma}'(\boldsymbol{z}_k^{[l]})$ for $k = 1, \ldots, B$
      end
      $\nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta}) = \boldsymbol{\delta}_k^{[l]} \left[\boldsymbol{a}_k^{[l-1]}\right]^T$ for $k = 1, \ldots, B$
      $\nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta}) = \boldsymbol{\delta}_k^{[l]}$ for $k = 1, \ldots, B$
      $\boldsymbol{W}^{[l]} := \boldsymbol{W}^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^{B} \nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta})$ // update gradient
      $\boldsymbol{b}^{[l]} := \boldsymbol{b}^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^{B} \nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta})$ // update gradient
    end
  end
end

Backpropagation (BP)

Algorithm 2: Backpropagation (stochastic gradient descent)

Data: dataloader $= \{\boldsymbol{B_b} = (\boldsymbol{X_b}, \boldsymbol{Y_b}) : b = 1, \dots, nb\_batches\}$

Inputs: MLP $= \left\{\left(\boldsymbol{W}^{[l]}, \boldsymbol{b}^{[l]}\right) : l = 1, \dots, L\right\}$ initialized randomly

for $epochs = 1$ ***to*** nb_epochs **do** // loop over epochs

  for $b = 1$ ***to*** $nb\_batches$ **do** // loop over batches

    $\boldsymbol{A}^{[0]} = \boldsymbol{X_b}$

    for $l = 1$ ***to*** L **do** // forward pass

      $\boldsymbol{Z}^{[l]} = \boldsymbol{W}^{[l]}\boldsymbol{A}^{[l-1]} + \boldsymbol{b}^{[l]}$

      $\boldsymbol{A}^{[l]} = \boldsymbol{\sigma}\left(\boldsymbol{Z}^{[l]}\right)$

    end

    for $l = L$ ***to*** 1 **do** // backward pass

      if $l = L$ **then** // compute error

        $\delta_k^{[l]} = \nabla_{\boldsymbol{a}^{[l]}} \mathcal{L}_k(\Theta) \odot \boldsymbol{\sigma}'(\boldsymbol{z}_k^{[l]})$ for $k = 1, \dots, B$

      else if $L > l \geq 1$ **then**

        $\delta_k^{[l]} = \left[\boldsymbol{W}^{[l+1]}\right]^T \delta_k^{[l+1]} \odot \boldsymbol{\sigma}'(\boldsymbol{z}_k^{[l]})$ for $k = 1, \dots, B$

      end

      $\nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}_k(\Theta) = \delta_k^{[l]} \left[\boldsymbol{a}_k^{[l-1]}\right]^T$ for $k = 1, \dots, B$

      $\nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}_k(\Theta) = \delta_k^{[l]}$ for $k = 1, \dots, B$

      $\boldsymbol{W}^{[l]} := \boldsymbol{W}^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^{B} \nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}_k(\Theta)$ // update gradient

      $\boldsymbol{b}^{[l]} := \boldsymbol{b}^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^{B} \nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}_k(\Theta)$ // update gradient

    end

  end

end

Backpropagation (BP)

Algorithm 2: Backpropagation (stochastic gradient descent)

Data: dataloader $= \{\boldsymbol{B_b} = (\boldsymbol{X_b}, \boldsymbol{Y_b}) : b = 1, \dots, nb\_batches\}$

Inputs: MLP $= \left\{ \left(\boldsymbol{W}^{[l]}, \boldsymbol{b}^{[l]} \right) : l = 1, \dots, L \right\}$ initialized randomly

for $epochs = 1$ ***to*** nb_epochs **do** // loop over epochs

  for $b = 1$ ***to*** $nb\_batches$ **do** // loop over batches

    $\boldsymbol{A}^{[0]} = \boldsymbol{X_b}$

    for $l = 1$ ***to*** L **do** // forward pass

      $\boldsymbol{Z}^{[l]} = \boldsymbol{W}^{[l]} \boldsymbol{A}^{[l-1]} + \boldsymbol{b}^{[l]}$

      $\boldsymbol{A}^{[l]} = \boldsymbol{\sigma}\left(\boldsymbol{Z}^{[l]}\right)$

    end

    for $l = L$ ***to*** 1 **do** // backward pass

      if $l = L$ **then** // compute error

        $\delta_k^{[l]} = \nabla_{\boldsymbol{a}^{[l]}} \mathcal{L}_k(\Theta) \odot \boldsymbol{\sigma}'(\boldsymbol{z}_k^{[l]})$ for $k = 1, \dots, B$

      else if $L > l \geq 1$ **then**

        $\delta_k^{[l]} = \left[\boldsymbol{W}^{[l+1]}\right]^T \delta_k^{[l+1]} \odot \boldsymbol{\sigma}'(\boldsymbol{z}_k^{[l]})$ for $k = 1, \dots, B$

      end

      $\nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}_k(\Theta) = \delta_k^{[l]} \left[\boldsymbol{a}_k^{[l-1]}\right]^T$ for $k = 1, \dots, B$

      $\nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}_k(\Theta) = \delta_k^{[l]}$ for $k = 1, \dots, B$

      $\boldsymbol{W}^{[l]} := \boldsymbol{W}^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^{B} \nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}_k(\Theta)$ // update gradient

      $\boldsymbol{b}^{[l]} := \boldsymbol{b}^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^{B} \nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}_k(\Theta)$ // update gradient

    end

  end

end

Backpropagation (BP)

Algorithm 2: Backpropagation (stochastic gradient descent)

Data: dataloader $= \{\boldsymbol{B_b} = (\boldsymbol{X_b}, \boldsymbol{Y_b}) : b = 1, \ldots, nb\_batches\}$

Inputs: MLP $= \left\{ \left(\boldsymbol{W}^{[l]}, \boldsymbol{b}^{[l]} \right) : l = 1, \ldots, L \right\}$ initialized randomly

for $epochs = 1$ ***to*** nb_epochs **do** // loop over epochs

  for $b = 1$ ***to*** $nb\_batches$ **do** // loop over batches

    $\boldsymbol{A}^{[0]} = \boldsymbol{X_b}$

    for $l = 1$ ***to*** L **do** // forward pass

      $\boldsymbol{Z}^{[l]} = \boldsymbol{W}^{[l]} \boldsymbol{A}^{[l-1]} + \boldsymbol{b}^{[l]}$

      $\boldsymbol{A}^{[l]} = \boldsymbol{\sigma}\left(\boldsymbol{Z}^{[l]}\right)$

    end

    for $l = L$ ***to*** 1 **do** // backward pass

      if $l = L$ **then** // compute error

        $\delta_k^{[l]} = \nabla_{a^{[l]}} \mathcal{L}_k(\Theta) \odot \boldsymbol{\sigma}'(z_k^{[l]})$ for $k = 1, \ldots, B$

      else if $L > l \geq 1$ **then**

        $\delta_k^{[l]} = \left[\boldsymbol{W}^{[l+1]} \right]^T \delta_k^{[l+1]} \odot \boldsymbol{\sigma}'(z_k^{[l]})$ for $k = 1, \ldots, B$

      end

      $\nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}_k(\Theta) = \delta_k^{[l]} \left[a_k^{[l-1]} \right]^T$ for $k = 1, \ldots, B$

      $\nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}_k(\Theta) = \delta_k^{[l]}$ for $k = 1, \ldots, B$

      $\boldsymbol{W}^{[l]} := \boldsymbol{W}^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^{B} \nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}_k(\Theta)$ // update gradient

      $\boldsymbol{b}^{[l]} := \boldsymbol{b}^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^{B} \nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}_k(\Theta)$ // update gradient

    end

  end

end

Backpropagation (BP)

Algorithm 2: Backpropagation (stochastic gradient descent)

Data: dataloader $= \{\boldsymbol{B_b} = (\boldsymbol{X_b}, \boldsymbol{Y_b}) : b = 1, \dots, nb\_batches\}$

Inputs: MLP $= \left\{\left(\boldsymbol{W}^{[l]}, \boldsymbol{b}^{[l]}\right) : l = 1, \dots, L\right\}$ initialized randomly

for $epochs = 1$ ***to*** nb_epochs **do** `// loop over epochs`

  for $b = 1$ ***to*** $nb\_batches$ **do** `// loop over batches`

    $\boldsymbol{A}^{[0]} = \boldsymbol{X_b}$

    for $l = 1$ ***to*** L **do** `// forward pass`

      $\boldsymbol{Z}^{[l]} = \boldsymbol{W}^{[l]} \boldsymbol{A}^{[l-1]} + \boldsymbol{b}^{[l]}$

      $\boldsymbol{A}^{[l]} = \boldsymbol{\sigma}\left(\boldsymbol{Z}^{[l]}\right)$

    end

    for $l = L$ ***to*** 1 **do** `// backward pass`

      if $l = L$ **then** `// compute error`

        $\delta_k^{[l]} = \nabla_{a^{[l]}} \mathcal{L}_k(\Theta) \odot \boldsymbol{\sigma}'(z_k^{[l]})$ for $k = 1, \dots, B$

      else if $L > l \geq 1$ **then**

        $\delta_k^{[l]} = \left[\boldsymbol{W}^{[l+1]}\right]^T \delta_k^{[l+1]} \odot \boldsymbol{\sigma}'(z_k^{[l]})$ for $k = 1, \dots, B$

      end

      $\nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}_k(\Theta) = \delta_k^{[l]} \left[a_k^{[l-1]}\right]^T$ for $k = 1, \dots, B$

      $\nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}_k(\Theta) = \delta_k^{[l]}$ for $k = 1, \dots, B$

      $\boldsymbol{W}^{[l]} := \boldsymbol{W}^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^{B} \nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}_k(\Theta)$ `// update gradient`

      $\boldsymbol{b}^{[l]} := \boldsymbol{b}^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^{B} \nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}_k(\Theta)$ `// update gradient`

    end

  end

end

Backpropagation (BP)

Algorithm 2: Backpropagation (stochastic gradient descent)

Data: dataloader $= \{\boldsymbol{B_b} = (\boldsymbol{X_b}, \boldsymbol{Y_b}) : b = 1, \dots, nb\_batches\}$

Inputs: MLP $= \left\{ \left(\boldsymbol{W}^{[l]}, \boldsymbol{b}^{[l]} \right) : l = 1, \dots, L \right\}$ initialized randomly

for $epochs = 1$ *to* nb_epochs **do** `// loop over epochs`

  for $b = 1$ *to* $nb\_batches$ **do** `// loop over batches`

    $\boldsymbol{A}^{[0]} = \boldsymbol{X_b}$

    for $l = 1$ *to* L **do** `// forward pass`

      $\boldsymbol{Z}^{[l]} = \boldsymbol{W}^{[l]} \boldsymbol{A}^{[l-1]} + \boldsymbol{b}^{[l]}$

      $\boldsymbol{A}^{[l]} = \boldsymbol{\sigma}\left(\boldsymbol{Z}^{[l]}\right)$

    end

    for $l = L$ *to* 1 **do** `// backward pass`

      if $l = L$ **then** `// compute error`

        $\delta_k^{[l]} = \nabla_{a^{[l]}} \mathcal{L}_k(\Theta) \odot \boldsymbol{\sigma}'(z_k^{[l]})$ for $k = 1, \dots, B$

      else if $L > l \geq 1$ **then**

        $\delta_k^{[l]} = \left[\boldsymbol{W}^{[l+1]}\right]^T \delta_k^{[l+1]} \odot \boldsymbol{\sigma}'(z_k^{[l]})$ for $k = 1, \dots, B$

      end

      $\nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}_k(\Theta) = \delta_k^{[l]} \left[a_k^{[l-1]}\right]^T$ for $k = 1, \dots, B$

      $\nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}_k(\Theta) = \delta_k^{[l]}$ for $k = 1, \dots, B$

      $\boldsymbol{W}^{[l]} := \boldsymbol{W}^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^{B} \nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}_k(\Theta)$ `// update gradient`

      $\boldsymbol{b}^{[l]} := \boldsymbol{b}^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^{B} \nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}_k(\Theta)$ `// update gradient`

    end

  end

end

Backpropagation (BP)

Algorithm 2: Backpropagation (stochastic gradient descent)

Data: dataloader $= \{\boldsymbol{B_b} = (\boldsymbol{X_b}, \boldsymbol{Y_b}) : b = 1, \dots, nb\_batches\}$

Inputs: MLP $= \left\{ \left(\boldsymbol{W}^{[l]}, \boldsymbol{b}^{[l]} \right) : l = 1, \dots, L \right\}$ initialized randomly

for $epochs = 1$ ***to*** nb_epochs **do** `// loop over epochs`

  for $b = 1$ ***to*** $nb\_batches$ **do** `// loop over batches`

    $\boldsymbol{A}^{[0]} = \boldsymbol{X_b}$

    for $l = 1$ ***to*** L **do** `// forward pass`

      $\boldsymbol{Z}^{[l]} = \boldsymbol{W}^{[l]} \boldsymbol{A}^{[l-1]} + \boldsymbol{b}^{[l]}$

      $\boldsymbol{A}^{[l]} = \boldsymbol{\sigma} \left(\boldsymbol{Z}^{[l]} \right)$

    end

    for $l = L$ ***to*** 1 **do** `// backward pass`

      if $l = L$ **then** `// compute error`

        $\boldsymbol{\delta}_{\boldsymbol{k}}^{[l]} = \nabla_{\boldsymbol{a}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta}) \odot \boldsymbol{\sigma}'(\boldsymbol{z}_{\boldsymbol{k}}^{[l]})$ **for** $k = 1, \dots, B$

      else if $L > l \geq 1$ **then**

        $\boldsymbol{\delta}_{\boldsymbol{k}}^{[l]} = \left[\boldsymbol{W}^{[l+1]} \right]^T \boldsymbol{\delta}_{\boldsymbol{k}}^{[l+1]} \odot \boldsymbol{\sigma}'(\boldsymbol{z}_{\boldsymbol{k}}^{[l]})$ **for** $k = 1, \dots, B$

      end

      $\nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta}) = \boldsymbol{\delta}_{\boldsymbol{k}}^{[l]} \left[\boldsymbol{a}_{\boldsymbol{k}}^{[l-1]} \right]^T$ **for** $k = 1, \dots, B$

      $\nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta}) = \boldsymbol{\delta}_{\boldsymbol{k}}^{[l]}$ **for** $k = 1, \dots, B$

      $\boldsymbol{W}^{[l]} := \boldsymbol{W}^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^{B} \nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta})$ `// update gradient`

      $\boldsymbol{b}^{[l]} := \boldsymbol{b}^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^{B} \nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta})$ `// update gradient`

    end

  end

end

Backpropagation (BP)

Algorithm 2: Backpropagation (stochastic gradient descent)

Data: dataloader $= \{\boldsymbol{B_b} = (\boldsymbol{X_b}, \boldsymbol{Y_b}) : b = 1, \dots, nb\_batches\}$

Inputs: MLP $= \left\{\left(\boldsymbol{W}^{[l]}, \boldsymbol{b}^{[l]}\right) : l = 1, \dots, L\right\}$ initialized randomly

for $epochs = 1$ ***to*** nb_epochs **do** // loop over epochs

  for $b = 1$ ***to*** $nb\_batches$ **do** // loop over batches

    $\boldsymbol{A}^{[0]} = \boldsymbol{X_b}$

    for $l = 1$ ***to*** L **do** // forward pass

      $\boldsymbol{Z}^{[l]} = \boldsymbol{W}^{[l]} \boldsymbol{A}^{[l-1]} + \boldsymbol{b}^{[l]}$

      $\boldsymbol{A}^{[l]} = \boldsymbol{\sigma}\left(\boldsymbol{Z}^{[l]}\right)$

    end

    for $l = L$ ***to*** 1 **do** // backward pass

      if $l = L$ **then** // compute error

        $\boldsymbol{\delta}_{\boldsymbol{k}}^{[l]} = \nabla_{\boldsymbol{a}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta}) \odot \boldsymbol{\sigma}'(\boldsymbol{z}_{\boldsymbol{k}}^{[l]})$ for $k = 1, \dots, B$

      else if $L > l \geq 1$ **then**

        $\boldsymbol{\delta}_{\boldsymbol{k}}^{[l]} = \left[\boldsymbol{W}^{[l+1]}\right]^T \boldsymbol{\delta}_{\boldsymbol{k}}^{[l+1]} \odot \boldsymbol{\sigma}'(\boldsymbol{z}_{\boldsymbol{k}}^{[l]})$ for $k = 1, \dots, B$

      end

      $\nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta}) = \boldsymbol{\delta}_{\boldsymbol{k}}^{[l]} \left[\boldsymbol{a}_{\boldsymbol{k}}^{[l-1]}\right]^T$ for $k = 1, \dots, B$

      $\nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta}) = \boldsymbol{\delta}_{\boldsymbol{k}}^{[l]}$ for $k = 1, \dots, B$

      $\boldsymbol{W}^{[l]} := \boldsymbol{W}^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^{B} \nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta})$ // update gradient

      $\boldsymbol{b}^{[l]} := \boldsymbol{b}^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^{B} \nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta})$ // update gradient

    end

  end

end

Backpropagation (BP)

Algorithm 2: Backpropagation (stochastic gradient descent)

Data: dataloader $= \{\boldsymbol{B_b} = (\boldsymbol{X_b}, \boldsymbol{Y_b}) : b = 1, \dots, nb\_batches\}$

Inputs: MLP $= \left\{\left(\boldsymbol{W}^{[l]}, \boldsymbol{b}^{[l]}\right) : l = 1, \dots, L\right\}$ initialized randomly

for $epochs = 1$ ***to*** nb_epochs **do** // loop over epochs

  for $b = 1$ ***to*** $nb\_batches$ **do** // loop over batches

    $\boldsymbol{A}^{[0]} = \boldsymbol{X_b}$

    for $l = 1$ ***to*** L **do** // forward pass

      $\boldsymbol{Z}^{[l]} = \boldsymbol{W}^{[l]} \boldsymbol{A}^{[l-1]} + \boldsymbol{b}^{[l]}$

      $\boldsymbol{A}^{[l]} = \boldsymbol{\sigma}\left(\boldsymbol{Z}^{[l]}\right)$

    end

    for $l = L$ ***to*** 1 **do** // backward pass

      if $l = L$ **then** // compute error

        $\boldsymbol{\delta}_{\boldsymbol{k}}^{[l]} = \nabla_{\boldsymbol{a}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta}) \odot \boldsymbol{\sigma'}(\boldsymbol{z}_{\boldsymbol{k}}^{[l]})$ for $k = 1, \dots, B$

      else if $L > l \geq 1$ **then**

        $\boldsymbol{\delta}_{\boldsymbol{k}}^{[l]} = \left[\boldsymbol{W}^{[l+1]}\right]^T \boldsymbol{\delta}_{\boldsymbol{k}}^{[l+1]} \odot \boldsymbol{\sigma'}(\boldsymbol{z}_{\boldsymbol{k}}^{[l]})$ for $k = 1, \dots, B$

      end

      $\nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta}) = \boldsymbol{\delta}_{\boldsymbol{k}}^{[l]} \left[\boldsymbol{a}_{\boldsymbol{k}}^{[l-1]}\right]^T$ for $k = 1, \dots, B$

      $\nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta}) = \boldsymbol{\delta}_{\boldsymbol{k}}^{[l]}$ for $k = 1, \dots, B$

      $\boldsymbol{W}^{[l]} := \boldsymbol{W}^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^{B} \nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta})$ // update gradient

      $\boldsymbol{b}^{[l]} := \boldsymbol{b}^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^{B} \nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta})$ // update gradient

    end

  end

end

Backpropagation (BP)

Algorithm 2: Backpropagation (stochastic gradient descent)

Data: dataloader $= \{\boldsymbol{B_b} = (\boldsymbol{X_b}, \boldsymbol{Y_b}) : b = 1, \ldots, nb\_batches\}$

Inputs: MLP $= \left\{\left(\boldsymbol{W}^{[l]}, \boldsymbol{b}^{[l]}\right) : l = 1, \ldots, L\right\}$ initialized randomly

for $epochs = 1$ ***to*** nb_epochs **do** // loop over epochs

  for $b = 1$ ***to*** $nb\_batches$ **do** // loop over batches

    $\boldsymbol{A}^{[0]} = \boldsymbol{X_b}$

    for $l = 1$ ***to*** L **do** // forward pass

      $\boldsymbol{Z}^{[l]} = \boldsymbol{W}^{[l]} \boldsymbol{A}^{[l-1]} + \boldsymbol{b}^{[l]}$

      $\boldsymbol{A}^{[l]} = \boldsymbol{\sigma}\left(\boldsymbol{Z}^{[l]}\right)$

    end

    for $l = L$ ***to*** 1 **do** // backward pass

      if $l = L$ **then** // compute error

        $\boldsymbol{\delta}_{\boldsymbol{k}}^{[l]} = \nabla_{\boldsymbol{a}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta}) \odot \boldsymbol{\sigma}'(\boldsymbol{z}_{\boldsymbol{k}}^{[l]})$ for $k = 1, \ldots, B$

      else if $L > l \geq 1$ **then**

        $\boldsymbol{\delta}_{\boldsymbol{k}}^{[l]} = \left[\boldsymbol{W}^{[l+1]}\right]^T \boldsymbol{\delta}_{\boldsymbol{k}}^{[l+1]} \odot \boldsymbol{\sigma}'(\boldsymbol{z}_{\boldsymbol{k}}^{[l]})$ for $k = 1, \ldots, B$

      end

      $\nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta}) = \boldsymbol{\delta}_{\boldsymbol{k}}^{[l]} \left[\boldsymbol{a}_{\boldsymbol{k}}^{[l-1]}\right]^T$ for $k = 1, \ldots, B$

      $\nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta}) = \boldsymbol{\delta}_{\boldsymbol{k}}^{[l]}$ for $k = 1, \ldots, B$

      $\boldsymbol{W}^{[l]} := \boldsymbol{W}^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^{B} \nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta})$ // update gradient

      $\boldsymbol{b}^{[l]} := \boldsymbol{b}^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^{B} \nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta})$ // update gradient

    end

  end

end

Backpropagation (BP)

Algorithm 2: Backpropagation (stochastic gradient descent)

Data: dataloader $= \{\boldsymbol{B_b} = (\boldsymbol{X_b}, \boldsymbol{Y_b}) : b = 1, \ldots, nb\_batches\}$

Inputs: MLP $= \left\{\left(\boldsymbol{W}^{[l]}, \boldsymbol{b}^{[l]}\right) : l = 1, \ldots, L\right\}$ initialized randomly

```
for epochs = 1 to nb_epochs do                                   // loop over epochs
    for b = 1 to nb_batches do                                   // loop over batches
        A^[0] = X_b
        for l = 1 to L do                                        // forward pass
            Z^[l] = W^[l] A^[l-1] + b^[l]
            A^[l] = σ(Z^[l])
        end
        for l = L to 1 do                                        // backward pass
            if l = L then                                        // compute error
                δ_k^[l] = ∇_{a^[l]} L_k(Θ) ⊙ σ'(z_k^[l])            for k = 1, ..., B
            else if L > l ≥ 1 then
                δ_k^[l] = [W^[l+1]]^T δ_k^[l+1] ⊙ σ'(z_k^[l])        for k = 1, ..., B
            end
            ∇_{W^[l]} L_k(Θ) = δ_k^[l] [a_k^[l-1]]^T                 for k = 1, ..., B
            ∇_{b^[l]} L_k(Θ) = δ_k^[l]                               for k = 1, ..., B
            W^[l] := W^[l] - η/B · Σ_{k=1}^B ∇_{W^[l]} L_k(Θ)       // update gradient
            b^[l] := b^[l] - η/B · Σ_{k=1}^B ∇_{b^[l]} L_k(Θ)       // update gradient
        end
    end
end
```

Where, in mathematical notation:

$\boldsymbol{A}^{[0]} = \boldsymbol{X_b}$

$\boldsymbol{Z}^{[l]} = \boldsymbol{W}^{[l]} \boldsymbol{A}^{[l-1]} + \boldsymbol{b}^{[l]}$

$\boldsymbol{A}^{[l]} = \boldsymbol{\sigma}\left(\boldsymbol{Z}^{[l]}\right)$

$\boldsymbol{\delta}_{\boldsymbol{k}}^{[l]} = \nabla_{\boldsymbol{a}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta}) \odot \boldsymbol{\sigma}'(\boldsymbol{z}_{\boldsymbol{k}}^{[l]})$ for $k = 1, \ldots, B$

$\boldsymbol{\delta}_{\boldsymbol{k}}^{[l]} = \left[\boldsymbol{W}^{[l+1]}\right]^T \boldsymbol{\delta}_{\boldsymbol{k}}^{[l+1]} \odot \boldsymbol{\sigma}'(\boldsymbol{z}_{\boldsymbol{k}}^{[l]})$ for $k = 1, \ldots, B$

$\nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta}) = \boldsymbol{\delta}_{\boldsymbol{k}}^{[l]} \left[\boldsymbol{a}_{\boldsymbol{k}}^{[l-1]}\right]^T$ for $k = 1, \ldots, B$

$\nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta}) = \boldsymbol{\delta}_{\boldsymbol{k}}^{[l]}$ for $k = 1, \ldots, B$

$\boldsymbol{W}^{[l]} := \boldsymbol{W}^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^{B} \nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta})$

$\boldsymbol{b}^{[l]} := \boldsymbol{b}^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^{B} \nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta})$

Backpropagation (BP)

Algorithm 2: Backpropagation (stochastic gradient descent)

Data: dataloader $= \{\boldsymbol{B_b} = (\boldsymbol{X_b}, \boldsymbol{Y_b}) : b = 1, \dots, nb\_batches\}$

Inputs: MLP $= \left\{ \left(\boldsymbol{W}^{[l]}, \boldsymbol{b}^{[l]} \right) : l = 1, \dots, L \right\}$ initialized randomly

for $epochs = 1$ ***to*** nb_epochs **do** // loop over epochs
- **for** $b = 1$ ***to*** $nb\_batches$ **do** // loop over batches
  - $\boldsymbol{A}^{[0]} = \boldsymbol{X_b}$
  - **for** $l = 1$ ***to*** L **do** // forward pass
    - $\boldsymbol{Z}^{[l]} = \boldsymbol{W}^{[l]} \boldsymbol{A}^{[l-1]} + \boldsymbol{b}^{[l]}$
    - $\boldsymbol{A}^{[l]} = \boldsymbol{\sigma} \left(\boldsymbol{Z}^{[l]} \right)$
  - **end**
  - **for** $l = L$ ***to*** 1 **do** // backward pass
    - **if** $l = L$ **then** // compute error
      - $\boldsymbol{\delta}_{\boldsymbol{k}}^{[l]} = \nabla_{\boldsymbol{a}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta}) \odot \boldsymbol{\sigma}'(\boldsymbol{z}_{\boldsymbol{k}}^{[l]})$ for $k = 1, \dots, B$
    - **else if** $L > l \geq 1$ **then**
      - $\boldsymbol{\delta}_{\boldsymbol{k}}^{[l]} = \left[\boldsymbol{W}^{[l+1]} \right]^T \boldsymbol{\delta}_{\boldsymbol{k}}^{[l+1]} \odot \boldsymbol{\sigma}'(\boldsymbol{z}_{\boldsymbol{k}}^{[l]})$ for $k = 1, \dots, B$
    - **end**
    - $\nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta}) = \boldsymbol{\delta}_{\boldsymbol{k}}^{[l]} \left[\boldsymbol{a}_{\boldsymbol{k}}^{[l-1]} \right]^T$ for $k = 1, \dots, B$
    - $\nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta}) = \boldsymbol{\delta}_{\boldsymbol{k}}^{[l]}$ for $k = 1, \dots, B$
    - $\boldsymbol{W}^{[l]} := \boldsymbol{W}^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^{B} \nabla_{\boldsymbol{W}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta})$ // update gradient
    - $\boldsymbol{b}^{[l]} := \boldsymbol{b}^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^{B} \nabla_{\boldsymbol{b}^{[l]}} \mathcal{L}_k(\boldsymbol{\Theta})$ // update gradient
  - **end**
- **end**

end

Exemple de training via Backrop

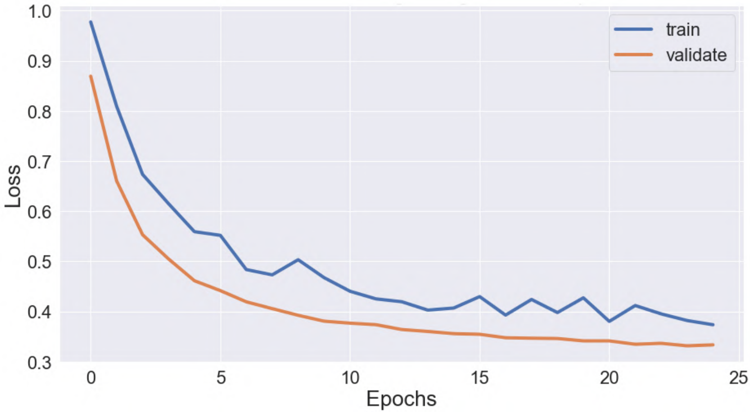


Figure taken from towardsdatascience.com

Bibliographie

Fleuret, F. (2022).
Deep Learning Course.

Goodfellow, I. J., Bengio, Y., and Courville, A. C. (2016).
Deep Learning.
Adaptive computation and machine learning. MIT Press.

Nielsen, M. A. (2018).
Neural networks and deep learning.

Wikipedia contributors (2022).
Backpropagation — Wikipedia, the free encyclopedia.