

# BACKPROPAGATION

Jérémie Cabessa

Laboratoire DAVID, UVSQ

# CHAIN RULE

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

Soient les fonctions suivantes:

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

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:

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

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:

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

alors on a:

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

## CHAIN RULE

**Exemple:**

$$\begin{array}{ccc} \mathbb{R} & \xrightarrow{f} & \mathbb{R} & \xrightarrow{g} & \mathbb{R} \\ x & \xmapsto{f} & y = x^2 + 1 & \xmapsto{g} & \begin{aligned} z &= 5y \\ &= 5(x^2 + 1) \end{aligned} \end{array}$$

On a:

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

## CHAIN RULE

**Exemple:**

$$\begin{array}{ccc} \mathbb{R} & \xrightarrow{f} & \mathbb{R} & \xrightarrow{g} & \mathbb{R} \\ x & \xmapsto{f} & y = x^2 + 1 & \xmapsto{g} & \begin{aligned} z &= 5y \\ &= 5(x^2 + 1) \end{aligned} \end{array}$$

On a:

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

# CHAIN RULE

## Généralisation multidimensionnelle

Soient les fonctions suivantes:

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

alors on a:

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

# CHAIN RULE

## Généralisation multidimensionnelle

Soient les fonctions suivantes:

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

alors on a:

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



# CHAIN RULE

## Généralisation multidimensionnelle

Soient les fonctions suivantes:

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

alors on a:

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

## CHAIN RULE

## Formulation vectorielle

$$\begin{array}{ccccc}
 \mathbb{R}^m & \xrightarrow{f} & \mathbb{R}^n & \xrightarrow{g} & \mathbb{R} \\
 \mathbf{x} & \mapsto^f & \mathbf{y} = f(\mathbf{x}) & \mapsto^g & z = g(\mathbf{y}) \\
 & & & & = g(f(\mathbf{x}))
 \end{array}$$

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

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

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

## CHAIN RULE

## Formulation vectorielle

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

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

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

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

## CHAIN RULE

## Formulation vectorielle

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

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

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

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

## CHAIN RULE

## Formulation vectorielle

$$\begin{array}{ccccc}
 \mathbb{R}^m & \xrightarrow{f} & \mathbb{R}^n & \xrightarrow{g} & \mathbb{R} \\
 \mathbf{x} & \mapsto^f & \mathbf{y} = f(\mathbf{x}) & \mapsto^g & z = g(\mathbf{y}) \\
 & & & & = g(f(\mathbf{x}))
 \end{array}$$

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

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

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

## CHAIN RULE

**Exemple:**

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

On a:

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

## CHAIN RULE

**Exemple:**

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

On a:

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

# CHAIN RULE

**Exemple:**

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

On a:

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



## CHAIN RULE

**Exemple (suite):**

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

On a:

$$\nabla_{\mathbf{x}} z = \begin{pmatrix} \frac{\partial z}{\partial x_1} \\ \frac{\partial z}{\partial x_2} \end{pmatrix} = \begin{pmatrix} 3x_2 x_1^2 \\ x_1^2 + 3x_2^3 \end{pmatrix} = (y_2 \quad y_1 \quad 1) \begin{pmatrix} x_2 & x_1 \\ 2x_1 & 0 \\ 0 & 3x_2^2 \end{pmatrix} = [\nabla_{\mathbf{y}} z]^T \begin{bmatrix} \partial y \\ \partial x \end{bmatrix}$$

# CHAIN RULE

Exemple (suite):

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

On a:

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

# NEURAL NETWORK AS A FUNCTION

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

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

et par la dynamique

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

# NEURAL NETWORK AS A FUNCTION

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

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

et par la dynamique

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

# NEURAL NETWORK AS A FUNCTION

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

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

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

# NEURAL NETWORK AS A FUNCTION

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

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

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

# NEURAL NETWORK AS A FUNCTION

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

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

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

# LOSS FUNCTION

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

$$\begin{aligned}\ell : \mathbb{R}^{d_2} \times \mathbb{R}^{d_2} &\longrightarrow \mathbb{R} \\ (\hat{y}_i, y_i) &\longmapsto \mathcal{L}(\hat{y}_i, y_i)\end{aligned}$$

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

$$\ell(\hat{y}_i, y_i) = \frac{1}{2} \|\hat{y}_i - y_i\|_2^2$$



# LOSS FUNCTION

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

$$\begin{aligned}\ell : \mathbb{R}^{d_2} \times \mathbb{R}^{d_2} &\longrightarrow \mathbb{R} \\ (\hat{\mathbf{y}}_i, \mathbf{y}_i) &\longmapsto \mathcal{L}(\hat{\mathbf{y}}_i, \mathbf{y}_i)\end{aligned}$$

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

$$\ell(\hat{\mathbf{y}}_i, \mathbf{y}_i) = \frac{1}{2} \|\hat{\mathbf{y}}_i - \mathbf{y}_i\|_2^2$$

# LOSS FUNCTION

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

$$\begin{aligned}\mathcal{L} : \mathbb{R}^{d_2} \times \dots \times \mathbb{R}^{d_2} &\longrightarrow \mathbb{R} \\ (\hat{\mathbf{y}}_1, \dots, \hat{\mathbf{y}}_N, \mathbf{y}_1, \dots, \mathbf{y}_N) &\longmapsto \mathcal{L}(\hat{\mathbf{y}}_1, \dots, \hat{\mathbf{y}}_N, \mathbf{y}_1, \dots, \mathbf{y}_N)\end{aligned}$$

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

$$\begin{aligned}\mathcal{L}(\hat{\mathbf{y}}_1, \dots, \hat{\mathbf{y}}_N, \mathbf{y}_1, \dots, \mathbf{y}_N) &= \frac{1}{N} \sum_{i=1}^N \ell(\hat{\mathbf{y}}_i, \mathbf{y}_i) \\ &= \frac{1}{2N} \sum_{i=1}^N \|\hat{\mathbf{y}}_i - \mathbf{y}_i\|_2^2\end{aligned}$$

# LOSS FUNCTION

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

$$\begin{aligned}\mathcal{L} : \mathbb{R}^{d_2} \times \dots \times \mathbb{R}^{d_2} &\longrightarrow \mathbb{R} \\ (\hat{\mathbf{y}}_1, \dots, \hat{\mathbf{y}}_N, \mathbf{y}_1, \dots, \mathbf{y}_N) &\longmapsto \mathcal{L}(\hat{\mathbf{y}}_1, \dots, \hat{\mathbf{y}}_N, \mathbf{y}_1, \dots, \mathbf{y}_N)\end{aligned}$$

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

$$\begin{aligned}\mathcal{L}(\hat{\mathbf{y}}_1, \dots, \hat{\mathbf{y}}_N, \mathbf{y}_1, \dots, \mathbf{y}_N) &= \frac{1}{N} \sum_{i=1}^N \ell(\hat{\mathbf{y}}_i, \mathbf{y}_i) \\ &= \frac{1}{2N} \sum_{i=1}^N \|\hat{\mathbf{y}}_i - \mathbf{y}_i\|_2^2\end{aligned}$$

# LOSS FUNCTION

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

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

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

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

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

# LOSS FUNCTION

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

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

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

$$\mathcal{L} : \mathbb{R}^{|\Theta|} \longrightarrow \mathbb{R}$$

$$\Theta \longmapsto \mathcal{L}(f_{\Theta}(x_1), \dots, f_{\Theta}(x_N), y_1, \dots, y_N).$$

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

# LOSS FUNCTION

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

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

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

$$\mathcal{L} : \mathbb{R}^{|\Theta|} \longrightarrow \mathbb{R}$$

$$\Theta \longmapsto \mathcal{L}(f_{\Theta}(x_1), \dots, f_{\Theta}(x_N), y_1, \dots, y_N).$$

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

## TRAINING

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

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

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

## TRAINING

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

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

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



## TRAINING

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

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

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

## TRAINING

$$\mathcal{L}(f_{\Theta}(x_1), \dots, f_{\Theta}(x_N), y_1, \dots, y_N) = \mathcal{L}(\Theta, \dots)$$

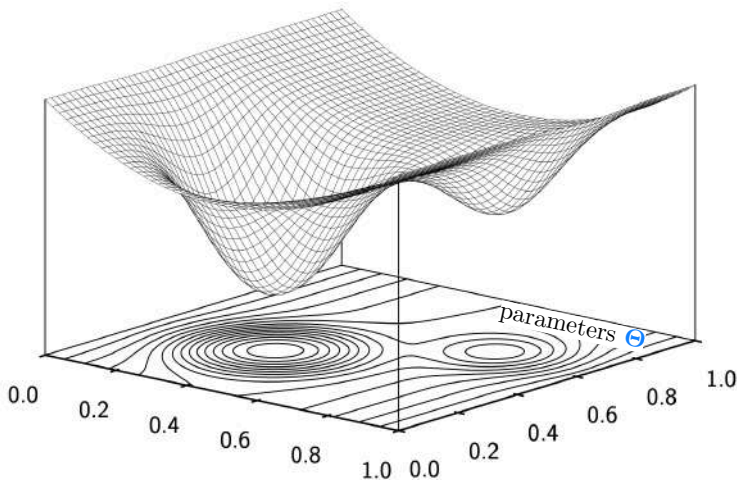


Figure adapted from [Fleuret, 2022]

## TRAINING

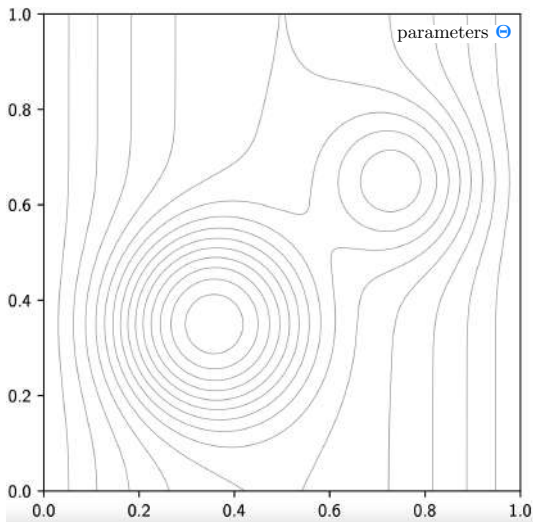


Figure adapted from [Fleuret, 2022]

## TRAINING

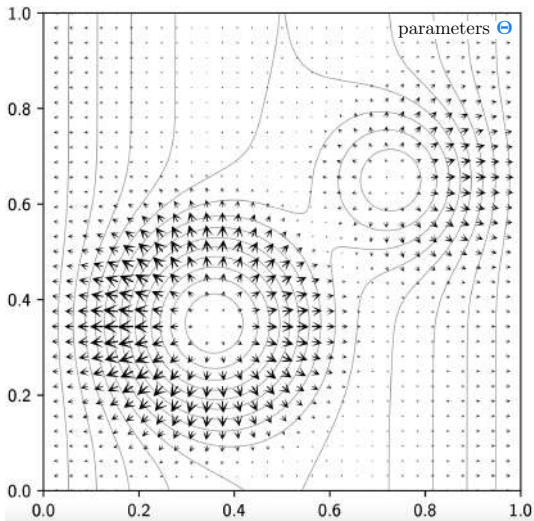


Figure adapted from [Fleuret, 2022]

# TRAINING

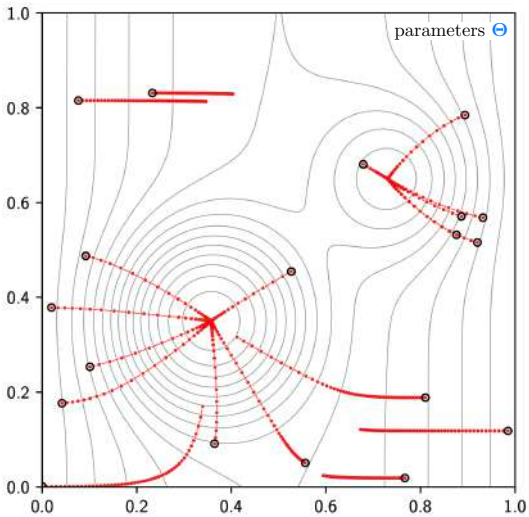


Figure adapted from [Fleuret, 2022]

# TRAINING

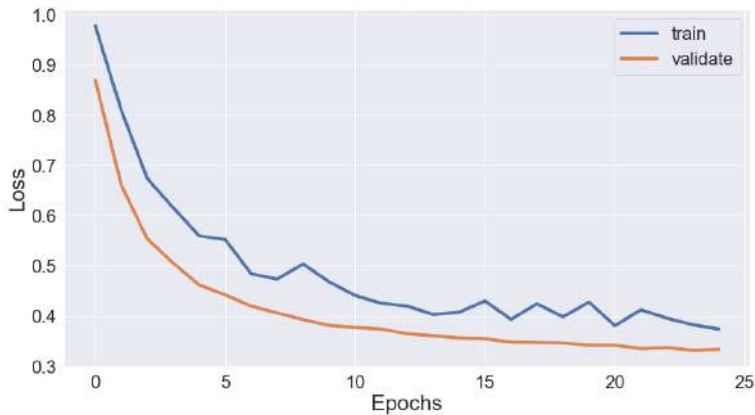
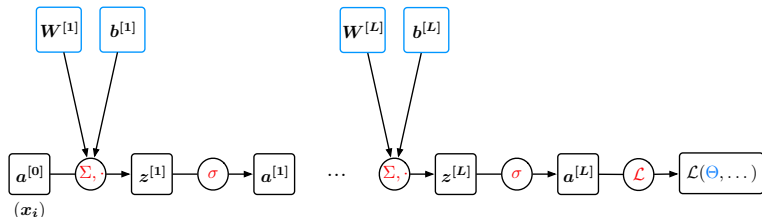


Figure taken from [towardsdatascience.com](https://towardsdatascience.com)

# GRAPHE COMPUTATIONNEL D'UN RÉSEAU DE NEURONES (FORWARD PASS)

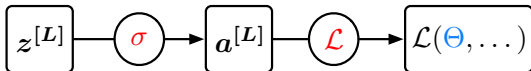


- On veut calculer les gradients:

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

pour  $l = 1, \dots, M$

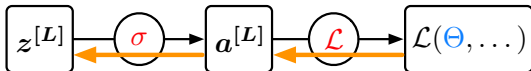
# CALCUL DES GRADIENTS: ÉQUATION 1



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

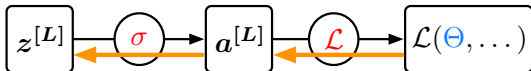


# CALCUL DES GRADIENTS: ÉQUATION 1



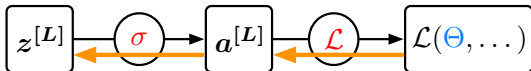
$$\begin{aligned}
 \delta_j^{[L]} &:= \frac{\partial \mathcal{L}(\Theta)}{\partial z_j^{[L]}} = \sum_{k=1}^{|a^{[L]}|} \frac{\partial \mathcal{L}(\Theta)}{\partial a_k^{[L]}} \cdot \frac{\partial a_k^{[L]}}{\partial z_j^{[L]}} \\
 &= \sum_{k=1}^{|a^{[L]}|} \frac{\partial \mathcal{L}(\Theta)}{\partial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

# CALCUL DES GRADIENTS: ÉQUATION 1



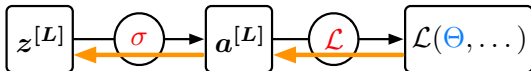
$$\begin{aligned}
 \delta_j^{[L]} &:= \frac{\partial \mathcal{L}(\Theta)}{\partial z_j^{[L]}} = \sum_{k=1}^{|a^{[L]}|} \frac{\partial \mathcal{L}(\Theta)}{\partial a_k^{[L]}} \cdot \frac{\partial a_k^{[L]}}{\partial z_j^{[L]}} \\
 &= \sum_{k=1}^{|a^{[L]}|} \frac{\partial \mathcal{L}(\Theta)}{\partial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

# CALCUL DES GRADIENTS: ÉQUATION 1



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

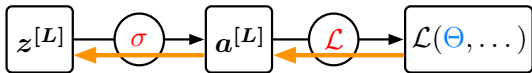
# CALCUL DES GRADIENTS: ÉQUATION 1



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

# CALCUL DES GRADIENTS: ÉQUATION 1

Formulation vectorielle:

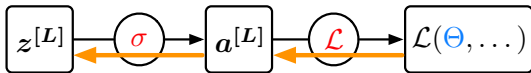


$$\delta^{[L]} := \nabla_{z^{[L]}} \mathcal{L}(\Theta) = \nabla_{a^{[L]}} \mathcal{L}(\Theta) \odot \sigma'(z^{[L]})$$

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

# CALCUL DES GRADIENTS: ÉQUATION 1

Formulation vectorielle:

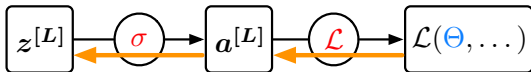


$$\delta^{[L]} := \nabla_{z^{[L]}} \mathcal{L}(\Theta) = \nabla_{a^{[L]}} \mathcal{L}(\Theta) \odot \sigma'(z^{[L]})$$

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

## CALCUL DES GRADIENTS: ÉQUATION 1

Formulation vectorielle:

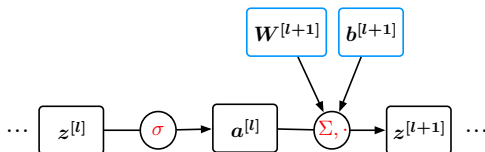


$$\delta^{[L]} := \nabla_{z^{[L]}} \mathcal{L}(\Theta) = \nabla_{a^{[L]}} \mathcal{L}(\Theta) \odot \sigma'(z^{[L]})$$

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

# CALCUL DES GRADIENTS: ÉQUATION 2

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

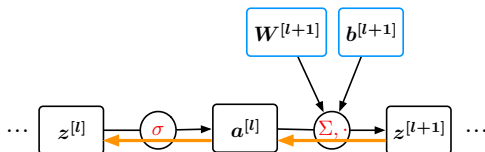


$$\begin{aligned}
 \delta_j^{[l]} &:= \frac{\partial \mathcal{L}(\Theta)}{\partial z_j^{[l]}} = \sum_{k=1}^{|a^{[l]}|} \frac{\partial \mathcal{L}(\Theta)}{\partial a_k^{[l]}} \cdot \frac{\partial a_k^{[l]}}{\partial z_j^{[l]}} = \sum_{k=1}^{|a^{[l]}|} \frac{\partial \mathcal{L}(\Theta)}{\partial a_k^{[l]}}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



# CALCUL DES GRADIENTS: ÉQUATION 2

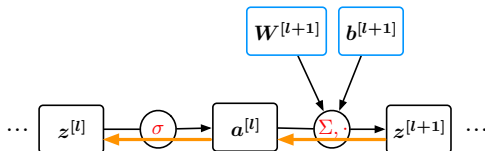
Supposons que les  $\delta_k^{[l+1]}$  ont été calculés pour tous  $k$ :



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

# CALCUL DES GRADIENTS: ÉQUATION 2

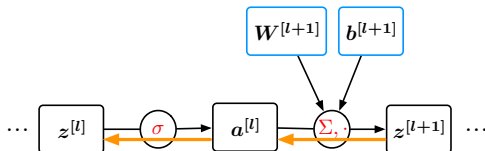
Supposons que les  $\delta_k^{[l+1]}$  ont été calculés pour tous  $k$ :



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

## CALCUL DES GRADIENTS: ÉQUATION 2

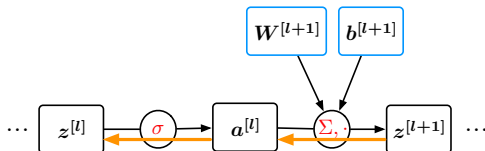
Supposons que les  $\delta_k^{[l+1]}$  ont été calculés pour tous  $k$ :



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

# CALCUL DES GRADIENTS: ÉQUATION 2

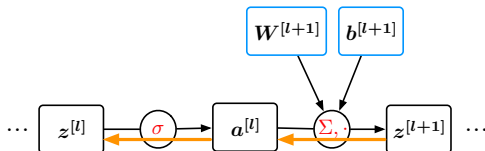
Supposons que les  $\delta_k^{[l+1]}$  ont été calculés pour tous  $k$ :



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

## CALCUL DES GRADIENTS: ÉQUATION 2

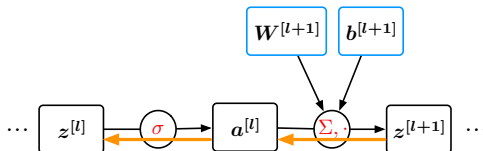
Supposons que les  $\delta_k^{[l+1]}$  ont été calculés pour tous  $k$ :



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

## CALCUL DES GRADIENTS: ÉQUATION 2

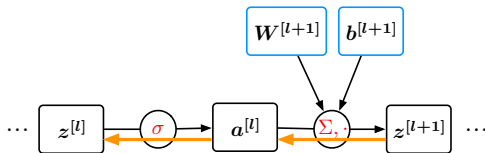
Supposons que les  $\delta_k^{[l+1]}$  ont été calculés pour tous  $k$ :



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

## CALCUL DES GRADIENTS: ÉQUATION 2

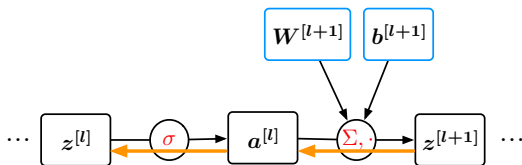
Supposons que les  $\delta_k^{[l+1]}$  ont été calculés pour tous  $k$ :



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

# CALCUL DES GRADIENTS: ÉQUATION 2

Formulation vectorielle:



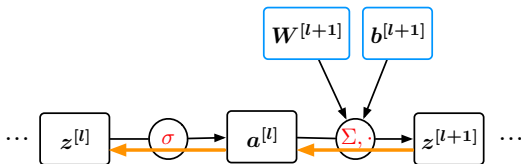
$$\begin{aligned}\delta^{[l]} &:= \nabla_{z^{[l]}} \mathcal{L}(\Theta) \\ &= \left[ \delta^{[l+1]} \right]^T W^{[l+1]} \odot \sigma'(z^{[l]}) \\ &= \left[ W^{[l+1]} \right]^T \delta^{[l+1]} \odot \sigma'(z^{[l]})\end{aligned}$$

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



# CALCUL DES GRADIENTS: ÉQUATION 2

Formulation vectorielle:

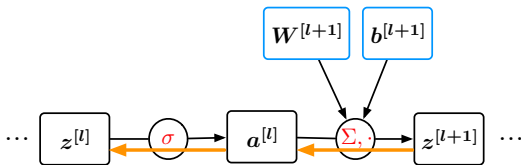


$$\begin{aligned}\delta^{[l]} &:= \nabla_{z^{[l]}} \mathcal{L}(\Theta) \\ &= \left[ \delta^{[l+1]} \right]^T W^{[l+1]} \odot \sigma'(z^{[l]}) \\ &= \left[ W^{[l+1]} \right]^T \delta^{[l+1]} \odot \sigma'(z^{[l]})\end{aligned}$$

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

# CALCUL DES GRADIENTS: ÉQUATION 2

Formulation vectorielle:

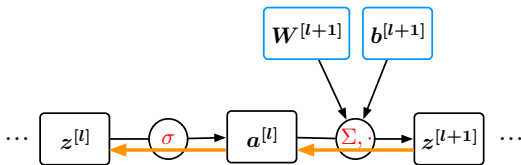


$$\begin{aligned}\delta^{[l]} &:= \nabla_{z^{[l]}} \mathcal{L}(\Theta) \\ &= \left[ \delta^{[l+1]} \right]^T W^{[l+1]} \odot \sigma'(z^{[l]}) \\ &= \left[ W^{[l+1]} \right]^T \delta^{[l+1]} \odot \sigma'(z^{[l]})\end{aligned}$$

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

## CALCUL DES GRADIENTS: ÉQUATION 2

Formulation vectorielle:

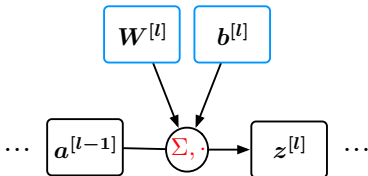


$$\begin{aligned}\delta^{[l]} &:= \nabla_{z^{[l]}} \mathcal{L}(\Theta) \\ &= \left[ \delta^{[l+1]} \right]^T W^{[l+1]} \odot \sigma'(z^{[l]}) \\ &= \left[ W^{[l+1]} \right]^T \delta^{[l+1]} \odot \sigma'(z^{[l]})\end{aligned}$$

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

## CALCUL DES GRADIENTS: ÉQUATION 3

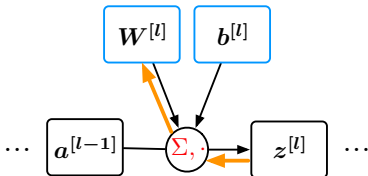
Calcul des gradients proprement dits en utilisant les erreurs  $\delta_j^l$ :



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

## CALCUL DES GRADIENTS: ÉQUATION 3

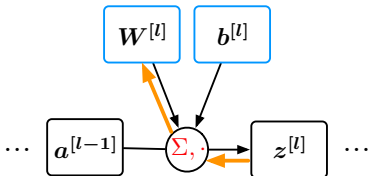
Calcul des gradients proprement dits en utilisant les erreurs  $\delta_j^l$ :



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

# CALCUL DES GRADIENTS: ÉQUATION 3

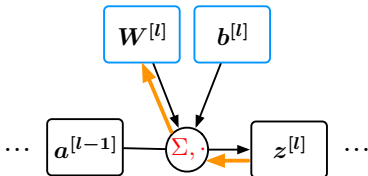
Calcul des gradients proprement dits en utilisant les erreurs  $\delta_j^l$ :



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

# CALCUL DES GRADIENTS: ÉQUATION 3

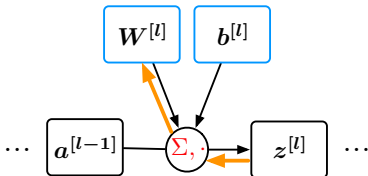
Calcul des gradients proprement dits en utilisant les erreurs  $\delta_j^l$ :



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

# CALCUL DES GRADIENTS: ÉQUATION 3

Calcul des gradients proprement dits en utilisant les erreurs  $\delta_j^l$ :

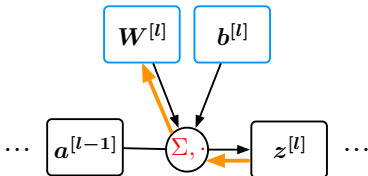


$$\begin{aligned} \frac{\partial \mathcal{L}(\Theta)}{\partial w_{jk}^{[l]}} &= \sum_{k'=1}^{|z^{[l]}|} \frac{\partial \mathcal{L}(\Theta)}{\partial z_{k'}^{[l]}} \cdot \frac{\partial z_{k'}^{[l]}}{\partial w_{jk}^{[l]}} = \frac{\partial \mathcal{L}(\Theta)}{\partial z_j^{[l]}} \cdot \frac{\partial z_j^{[l]}}{\partial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



# CALCUL DES GRADIENTS: ÉQUATION 3

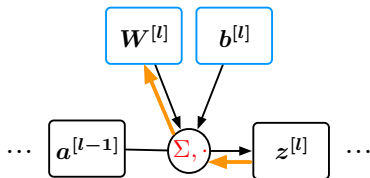
Calcul des gradients proprement dits en utilisant les erreurs  $\delta_j^l$ :



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

# CALCUL DES GRADIENTS: ÉQUATION 3

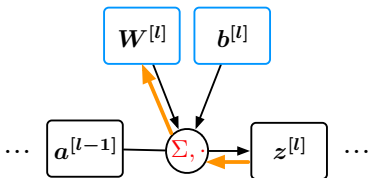
Formulation vectorielle:



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

# CALCUL DES GRADIENTS: ÉQUATION 3

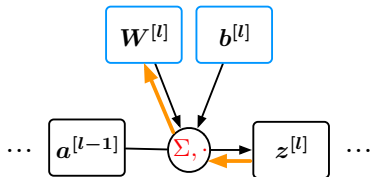
Formulation vectorielle:



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

# CALCUL DES GRADIENTS: ÉQUATION 3

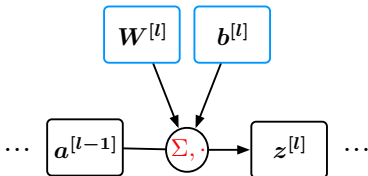
Formulation vectorielle:



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

## CALCUL DES GRADIENTS: ÉQUATION 4

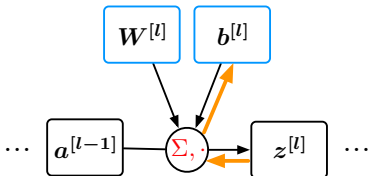
Calcul des gradients proprement dits en utilisant les erreurs  $\delta_j^l$ :



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

# CALCUL DES GRADIENTS: ÉQUATION 4

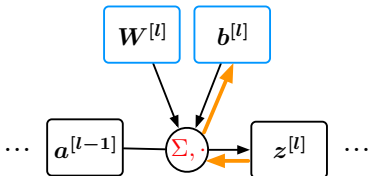
Calcul des gradients proprement dits en utilisant les erreurs  $\delta_j^l$ :



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

# CALCUL DES GRADIENTS: ÉQUATION 4

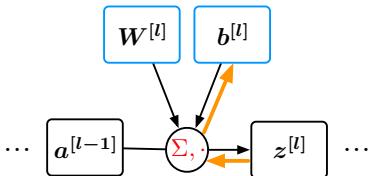
Calcul des gradients proprement dits en utilisant les erreurs  $\delta_j^l$ :



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

# CALCUL DES GRADIENTS: ÉQUATION 4

Calcul des gradients proprement dits en utilisant les erreurs  $\delta_j^l$ :

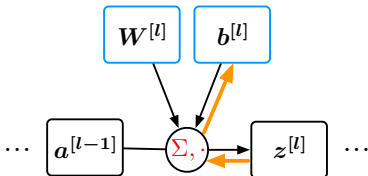


$$\begin{aligned}
 \frac{\partial \mathcal{L}(\Theta)}{\partial b_j^{[l]}} &= \sum_{k'=1}^{|z^{[l]}|} \frac{\partial \mathcal{L}(\Theta)}{\partial z_{k'}^{[l]}} \cdot \frac{\partial z_{k'}^{[l]}}{\partial b_j^{[l]}} = \frac{\partial \mathcal{L}(\Theta)}{\partial z_j^{[l]}} \cdot \frac{\partial z_j^{[l]}}{\partial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



# CALCUL DES GRADIENTS: ÉQUATION 4

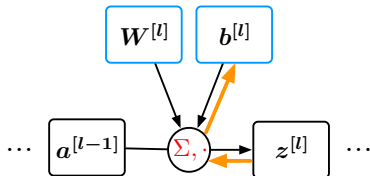
Calcul des gradients proprement dits en utilisant les erreurs  $\delta_j^l$ :



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

# CALCUL DES GRADIENTS: ÉQUATION 4

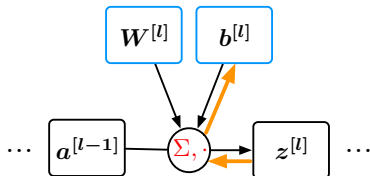
Calcul des gradients proprement dits en utilisant les erreurs  $\delta_j^l$ :



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

## CALCUL DES GRADIENTS: ÉQUATION 4

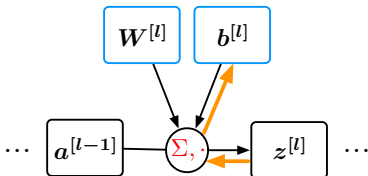
Formulation vectorielle:



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

## CALCUL DES GRADIENTS: ÉQUATION 4

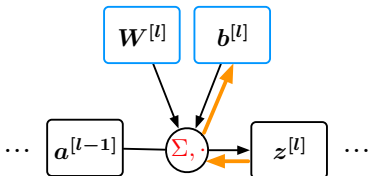
Formulation vectorielle:



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

## CALCUL DES GRADIENTS: ÉQUATION 4

Formulation vectorielle:



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

## ÉQUATIONS

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

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

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

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

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

## ÉQUATIONS

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

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

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

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

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

## ÉQUATIONS

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

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

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

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

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



## ÉQUATIONS

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

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

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

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

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

## ÉQUATIONS

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

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

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

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

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

## ÉQUATIONS

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

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

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

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

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

## ÉQUATIONS

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

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

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

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

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

## ÉQUATIONS

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

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

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

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

où  $\eta$  est le *learning rate*.

## ÉQUATIONS

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

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

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

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

où  $\eta$  est le *learning rate*.

## ÉQUATIONS

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

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

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

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

où  $\eta$  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:

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

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

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

- Soit  $\mathcal{L}(a) = \mathcal{L}(a, a^{[L]}, y_k)$  la loss pour l'exemple  $k$  (avec  $a = x_k$ )



# 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  $k = 1, \dots, B$ .

## 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  $k = 1, \dots, B$ .

## 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:

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

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

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

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

## ÉQUATIONS: BATCHED

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

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

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

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

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

$$\nabla_{b^{[l]}} \mathcal{L}_k(\Theta) = \delta_k^{[l]} \quad (8)$$

## ÉQUATIONS: BATCHED

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

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

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

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

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

$$\nabla_{b^{[l]}} \mathcal{L}_k(\Theta) = \delta_k^{[l]} \quad (8)$$

## ÉQUATIONS: BATCHED

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

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

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

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

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

$$\nabla_{b^{[l]}} \mathcal{L}_k(\Theta) = \delta_k^{[l]} \quad (8)$$

## ÉQUATIONS: BATCHED

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

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

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

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

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

$$\nabla_{b^{[l]}} \mathcal{L}_k(\Theta) = \delta_k^{[l]} \quad (8)$$

## ÉQUATIONS: BATCHED

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

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

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

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

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

$$\nabla_{b^{[l]}} \mathcal{L}_k(\Theta) = \delta_k^{[l]} \quad (8)$$



## ÉQUATIONS: BATCHED

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

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

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

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

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

$$\nabla_{b^{[l]}} \mathcal{L}_k(\Theta) = \delta_k^{[l]} \quad (8)$$

# ÉQUATIONS: BATCHED

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

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

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

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

où  $\eta$  est le *learning rate*.

## ÉQUATIONS: BATCHED

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

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

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

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

où  $\eta$  est le *learning rate*.

# ÉQUATIONS: BATCHED

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

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

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

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

où  $\eta$  est le *learning rate*.

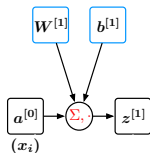
# FORWARD PASS AND BACKWARD PASS

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

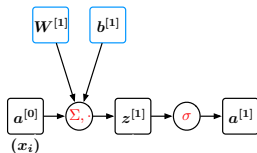
# FORWARD PASS AND BACKWARD PASS

 $\mathbf{W}^{[1]}$  $\mathbf{b}^{[1]}$  $\mathbf{a}^{[0]}$   
 $(\mathbf{x}_i)$

# FORWARD PASS AND BACKWARD PASS

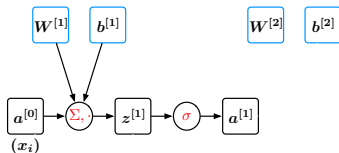


# FORWARD PASS AND BACKWARD PASS

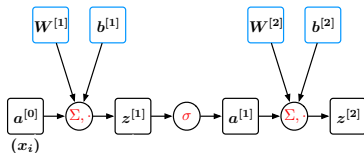




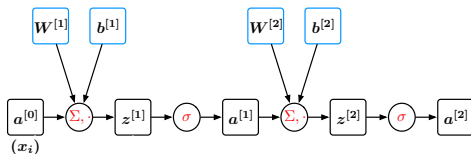
# FORWARD PASS AND BACKWARD PASS



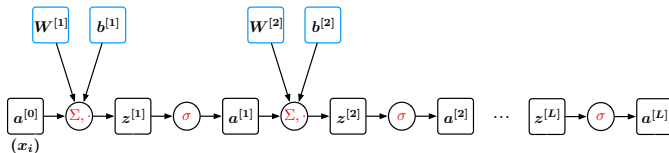
# FORWARD PASS AND BACKWARD PASS



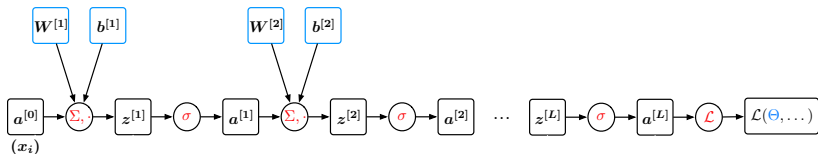
# FORWARD PASS AND BACKWARD PASS



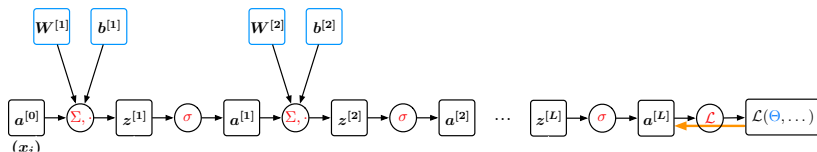
# FORWARD PASS AND BACKWARD PASS



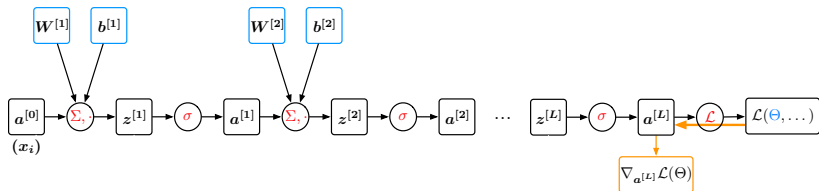
# FORWARD PASS AND BACKWARD PASS



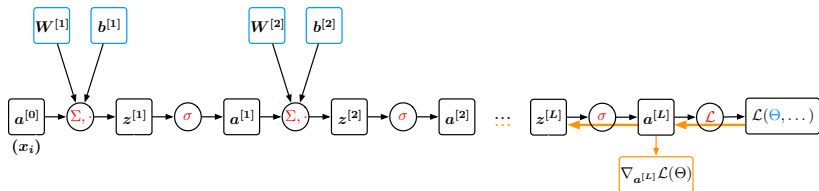
# FORWARD PASS AND BACKWARD PASS



# FORWARD PASS AND BACKWARD PASS

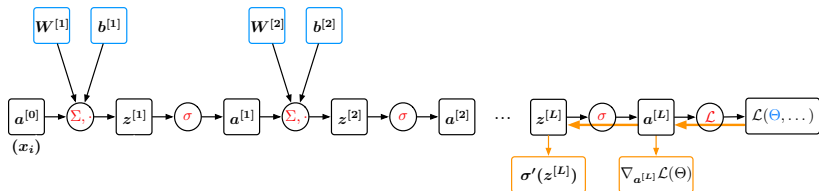


# FORWARD PASS AND BACKWARD PASS

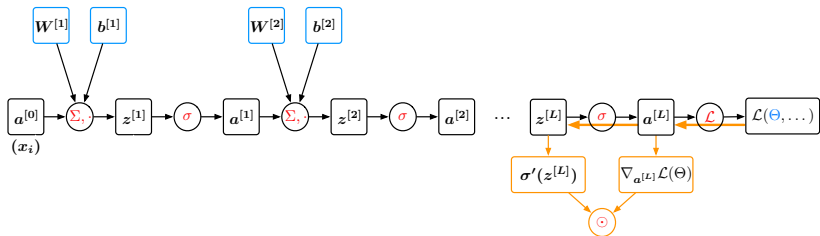




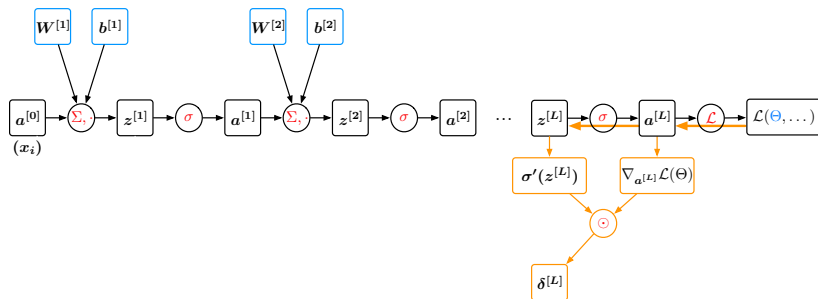
# FORWARD PASS AND BACKWARD PASS



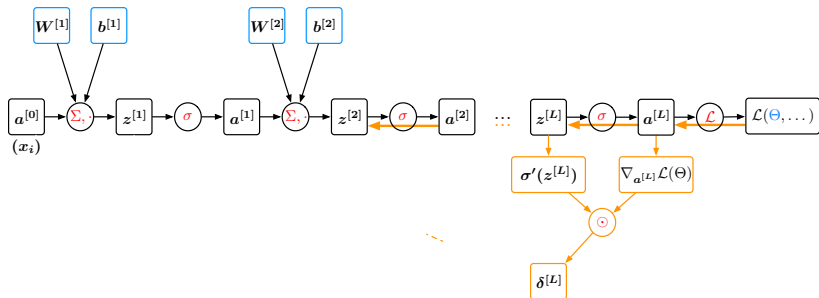
# FORWARD PASS AND BACKWARD PASS



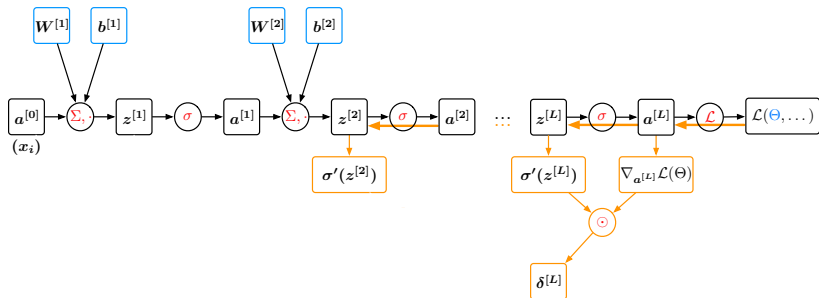
# FORWARD PASS AND BACKWARD PASS



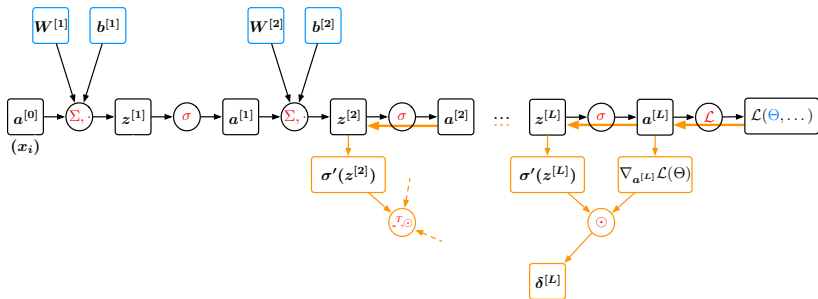
# FORWARD PASS AND BACKWARD PASS



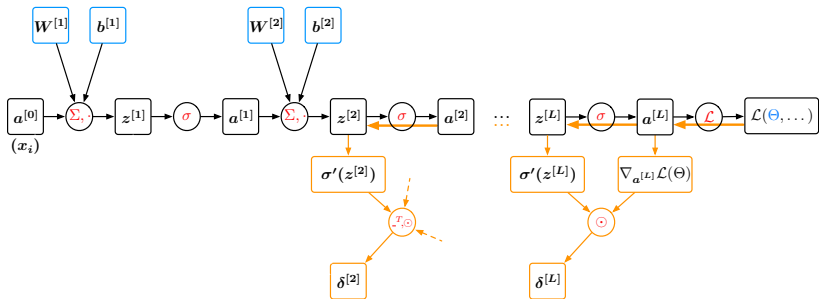
# FORWARD PASS AND BACKWARD PASS



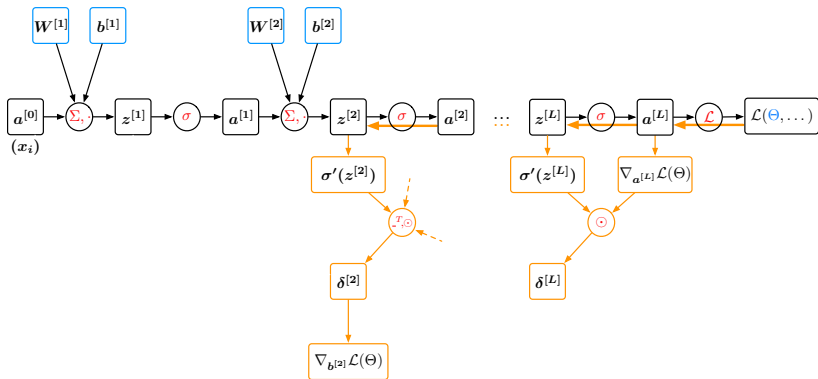
# FORWARD PASS AND BACKWARD PASS



# FORWARD PASS AND BACKWARD PASS

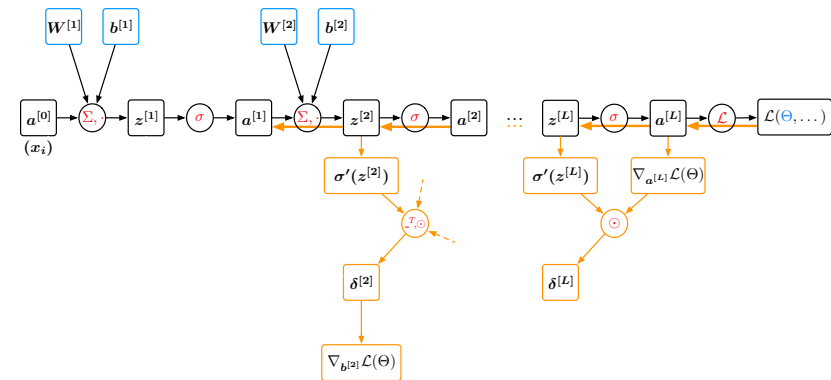


# FORWARD PASS AND BACKWARD PASS

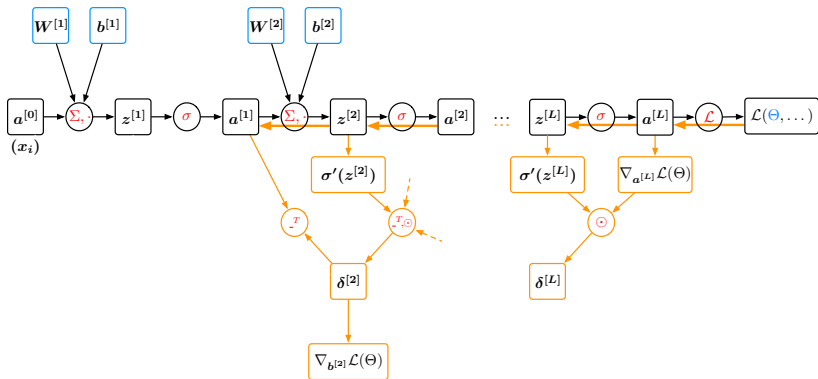




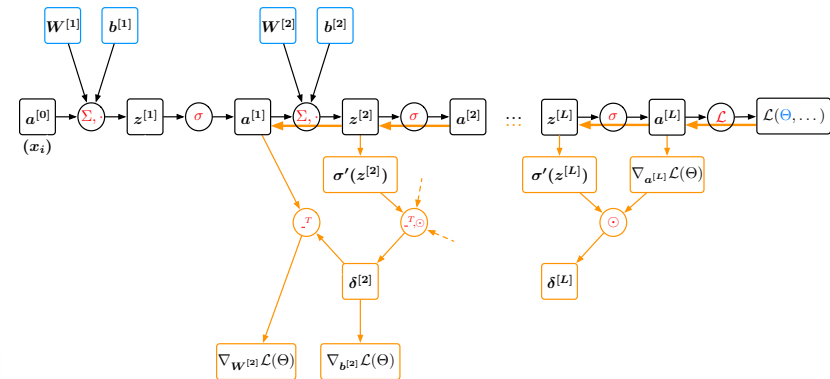
# FORWARD PASS AND BACKWARD PASS



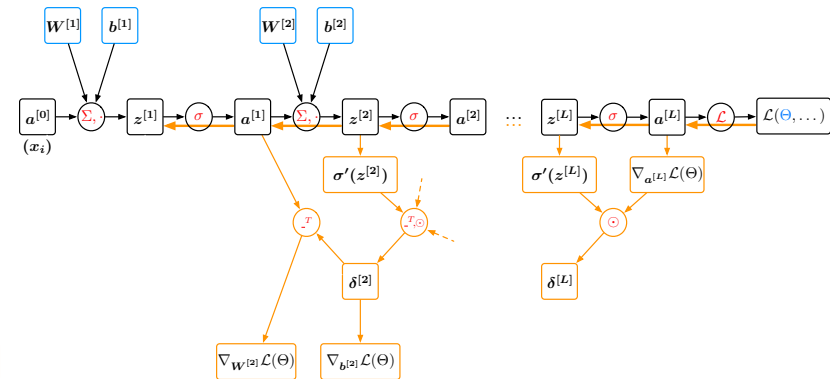
# FORWARD PASS AND BACKWARD PASS



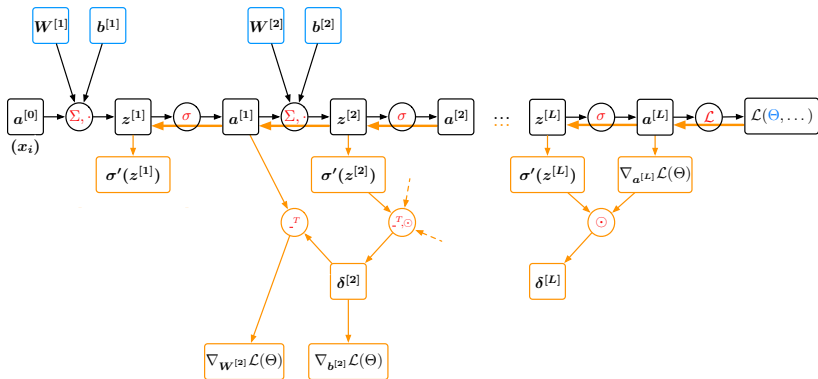
# FORWARD PASS AND BACKWARD PASS



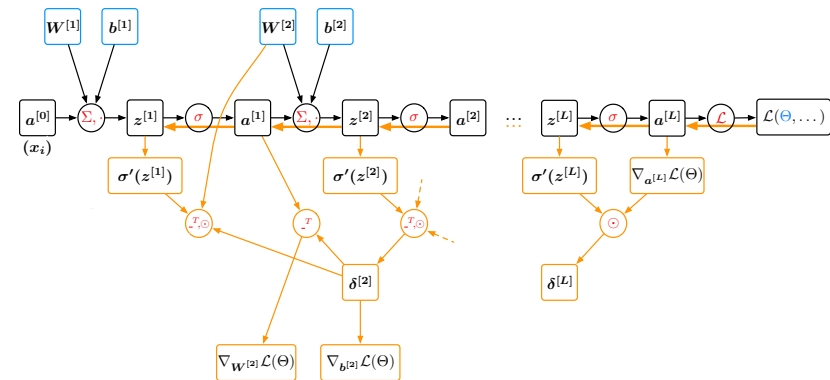
# FORWARD PASS AND BACKWARD PASS



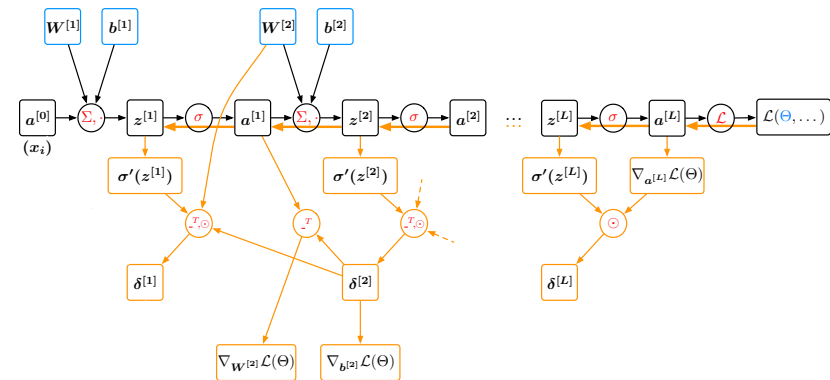
# FORWARD PASS AND BACKWARD PASS



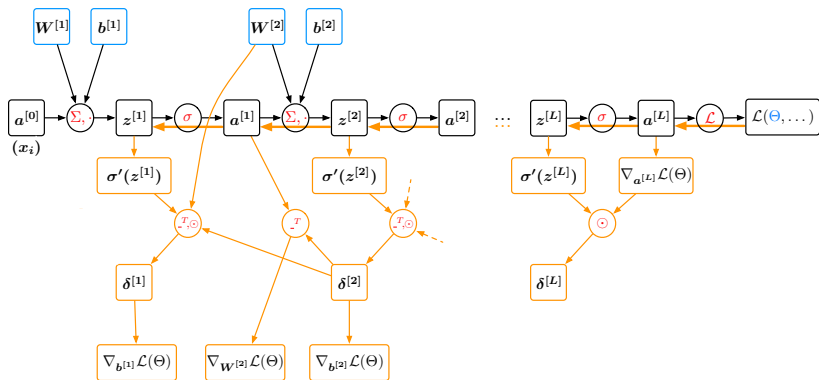
## FORWARD PASS AND BACKWARD PASS



## FORWARD PASS AND BACKWARD PASS

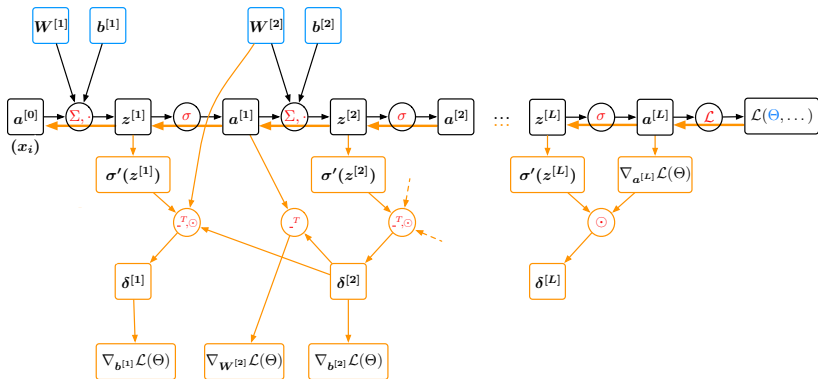


## FORWARD PASS AND BACKWARD PASS

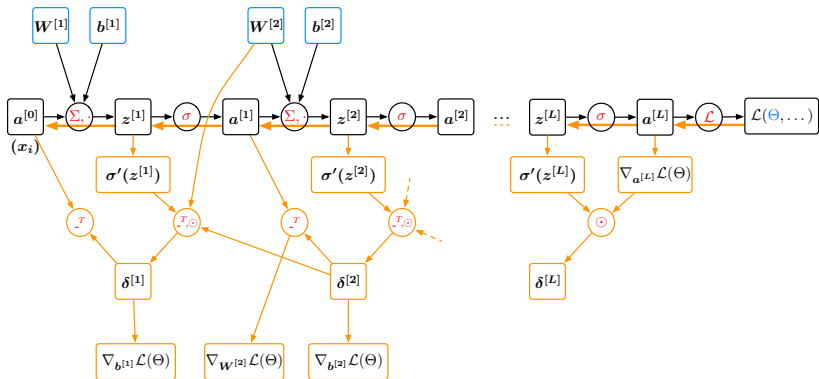




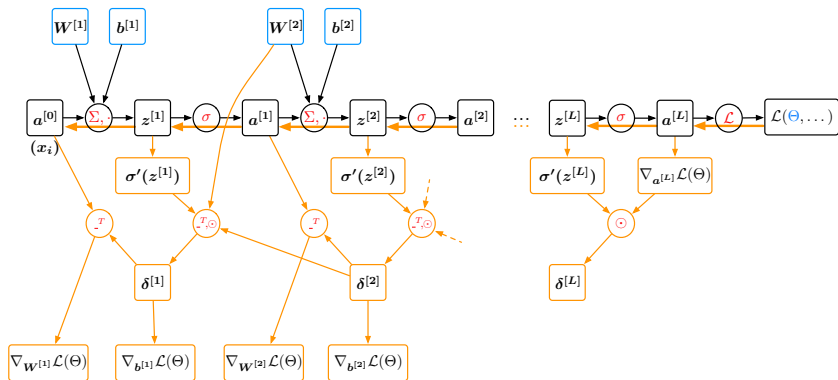
# FORWARD PASS AND BACKWARD PASS



## FORWARD PASS AND BACKWARD PASS



## FORWARD PASS AND BACKWARD PASS



# BACKPROPAGATION (BP)

---

## Algorithm 1: Backpropagation (stochastic gradient descent)

---

**Data:**  $\text{dataloader} = \{B_i = (X_i, Y_i) : i = 1, \dots, nb\_batches\}$

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

```
for epochs = 1 to nb_epochs do                                // loop over epochs
  for i = 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
    end                                                       // update gradient
  end
end
end
```

---

# BACKPROPAGATION (BP)

---

**Algorithm 1:** Backpropagation (stochastic gradient descent)

**Data:**  $\text{dataloader} = \{B_i = (X_i, Y_i) : i = 1, \dots, nb\_batches\}$

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

```
for epochs = 1 to nb_epochs do                                // loop over epochs
|   for i = 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
|   |   |   ...
|   |   |   ...
|   |   |   ...
|   |   end
|   end
end
```

---

# BACKPROPAGATION (BP)

---

**Algorithm 1:** Backpropagation (stochastic gradient descent)

**Data:**  $\text{dataloader} = \{B_i = (X_i, Y_i) : i = 1, \dots, nb\_batches\}$

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

```
for epochs = 1 to nb_epochs do                                // loop over epochs
  for i = 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:**  $\text{dataloader} = \{B_i = (X_i, Y_i) : i = 1, \dots, nb\_batches\}$

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

```
for epochs = 1 to nb_epochs do                                // loop over epochs
  for i = 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:**  $\text{dataloader} = \{B_i = (X_i, Y_i) : i = 1, \dots, nb\_batches\}$

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

```
for epochs = 1 to nb_epochs do                                // loop over epochs
  for i = 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
    end                                                         // update gradient
  end
end
```

---



# BACKPROPAGATION (BP)

---

**Algorithm 1:** Backpropagation (stochastic gradient descent)

**Data:**  $\text{dataloader} = \{B_i = (X_i, Y_i) : i = 1, \dots, nb\_batches\}$

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

```
for epochs = 1 to nb_epochs do                                     // loop over epochs
  for i = 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:**  $\text{dataloader} = \{B_i = (X_i, Y_i) : i = 1, \dots, nb\_batches\}$

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

```
for epochs = 1 to nb_epochs do                                // loop over epochs
  for i = 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
    end                                                         // update gradient
  end
end
```

---

# BACKPROPAGATION (BP)

---

**Algorithm 1:** Backpropagation (stochastic gradient descent)

---

**Data:**  $\text{dataloader} = \{B_i = (X_i, Y_i) : i = 1, \dots, nb\_batches\}$

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

```
for epochs = 1 to nb_epochs do                                // loop over epochs
  for i = 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
    end                                                         // update gradient
  end
end
```

---

# BACKPROPAGATION (BP)

---

**Algorithm 1:** Backpropagation (stochastic gradient descent)

---

**Data:**  $\text{dataloader} = \{B_i = (X_i, Y_i) : i = 1, \dots, nb\_batches\}$

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

```
for epochs = 1 to nb_epochs do                                // loop over epochs
  for i = 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:**  $\text{dataloader} = \{B_i = (X_i, Y_i) : i = 1, \dots, nb\_batches\}$

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

```
for epochs = 1 to nb_epochs do                                // loop over epochs
  for i = 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
    end                                                         // update gradient
  end
end
```

---

# BACKPROPAGATION (BP)

## Algorithm 2: Backpropagation (stochastic gradient descent)

**Data:**  $\text{dataloader} = \{B_i = (X_i, Y_i) : i = 1, \dots, nb\_batches\}$

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

```

for epochs = 1 to nb_epochs do                                // loop over epochs
  for i = 1 to nb_batches do                                   // loop over batches
     $A^{[0]} = \bar{X}_i$ 
    for l = 1 to L do                                          // forward pass
       $Z^{[l]} = W^{[l]}A^{[l-1]} + b^{[l]}$ 
       $A^{[l]} = \sigma(Z^{[l]})$ 
    end
    // backward pass
    for l = L to 1 do
      if l = L then
         $\delta_k^{[L]} = \nabla_{a_k^{[L]}} \mathcal{L}_k(\Theta) \odot \sigma'(z_k^{[L]})$     for  $k = 1, \dots, B$ 
      else if  $L > l \geq 1$  then
         $\delta_k^{[l]} = [W^{[l+1]}]^T \delta_k^{[l+1]} \odot \sigma'(z_k^{[l]})$     for  $k = 1, \dots, B$ 
      end
       $\nabla_{W^{[l]}} \mathcal{L}_k(\Theta) = \delta_k^{[l]} [a_k^{[l-1]}]^T$     for  $k = 1, \dots, B$ 
       $\nabla_{b^{[l]}} \mathcal{L}_k(\Theta) = \delta_k^{[l]}$     for  $k = 1, \dots, B$ 
       $W^{[l]} := W^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^B \nabla_{W^{[l]}} \mathcal{L}_k(\Theta)$     // update gradient
       $b^{[l]} := b^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^B \nabla_{b^{[l]}} \mathcal{L}_k(\Theta)$     // update gradient
    end
  end
end

```

# BACKPROPAGATION (BP)

## Algorithm 2: Backpropagation (stochastic gradient descent)

**Data:**  $\text{dataloader} = \{B_i = (X_i, Y_i) : i = 1, \dots, nb\_batches\}$

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

```

for  $epochs = 1$  to  $nb\_epochs$  do                                // loop over epochs
    for  $i = 1$  to  $nb\_batches$  do                                // loop over batches
         $A^{[0]} = \bar{X}_i$ 
        for  $l = 1$  to  $L$  do                                        // forward pass
             $Z^{[l]} = W^{[l]}A^{[l-1]} + b^{[l]}$ 
             $A^{[l]} = \sigma(Z^{[l]})$ 
        end
        for  $l = L$  to  $1$  do                                        // backward pass
            if  $l = L$  then                                        // compute error
                 $\delta_k^{[L]} = \nabla_{\alpha_k^{[L]}} \mathcal{L}_k(\Theta) \odot \sigma'(z_k^{[L]})$     for  $k = 1, \dots, B$ 
            else if  $L > l \geq 1$  then
                 $\delta_k^{[l]} = [W^{[l+1]}]^T \delta_k^{[l+1]} \odot \sigma'(z_k^{[l]})$     for  $k = 1, \dots, B$ 
            end
             $\nabla_{W^{[l]}} \mathcal{L}_k(\Theta) = \delta_k^{[l]} [\alpha_k^{[l-1]}]^T$     for  $k = 1, \dots, B$ 
             $\nabla_{b^{[l]}} \mathcal{L}_k(\Theta) = \delta_k^{[l]}$     for  $k = 1, \dots, B$ 
             $W^{[l]} := W^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^B \nabla_{W^{[l]}} \mathcal{L}_k(\Theta)$     // update gradient
             $b^{[l]} := b^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^B \nabla_{b^{[l]}} \mathcal{L}_k(\Theta)$     // update gradient
        end
    end
end

```

# BACKPROPAGATION (BP)

## Algorithm 2: Backpropagation (stochastic gradient descent)

**Data:**  $\text{dataloader} = \{B_i = (X_i, Y_i) : i = 1, \dots, nb\_batches\}$

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

```

for  $epochs = 1$  to  $nb\_epochs$  do                                // loop over epochs
  for  $i = 1$  to  $nb\_batches$  do                                    // loop over batches
     $A^{[0]} = X_i$ 
    for  $l = 1$  to  $L$  do                                            // forward pass
       $Z^{[l]} = W^{[l]} A^{[l-1]} + b^{[l]}$ 
       $A^{[l]} = \sigma(Z^{[l]})$ 
    end
    for  $l = L$  to  $1$  do                                            // backward pass
      if  $l = L$  then                                              // compute error
         $\delta_k^{[L]} = \nabla_{a_k^{[L]}} \mathcal{L}_k(\Theta) \odot \sigma'(z_k^{[L]})$     for  $k = 1, \dots, B$ 
      else if  $L > l \geq 1$  then
         $\delta_k^{[l]} = [W^{[l+1]}]^T \delta_k^{[l+1]} \odot \sigma'(z_k^{[l]})$     for  $k = 1, \dots, B$ 
      end
       $\nabla_{W^{[l]}} \mathcal{L}_k(\Theta) = \delta_k^{[l]} [a_k^{[l-1]}]^T$                 for  $k = 1, \dots, B$ 
       $\nabla_{b^{[l]}} \mathcal{L}_k(\Theta) = \delta_k^{[l]}$                             for  $k = 1, \dots, B$ 
       $W^{[l]} := W^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^B \nabla_{W^{[l]}} \mathcal{L}_k(\Theta)$         // update gradient
       $b^{[l]} := b^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^B \nabla_{b^{[l]}} \mathcal{L}_k(\Theta)$     // update gradient
    end
  end
end

```



# BACKPROPAGATION (BP)

## Algorithm 2: Backpropagation (stochastic gradient descent)

**Data:**  $\text{dataloader} = \{B_i = (X_i, Y_i) : i = 1, \dots, nb\_batches\}$

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

```

for  $epochs = 1$  to  $nb\_epochs$  do                                // loop over epochs
  for  $i = 1$  to  $nb\_batches$  do                                    // loop over batches
     $A^{[0]} = X_i$ 
    for  $l = 1$  to  $L$  do                                            // forward pass
       $Z^{[l]} = W^{[l]} A^{[l-1]} + b^{[l]}$ 
       $A^{[l]} = \sigma(Z^{[l]})$ 
    end
    for  $l = L$  to  $1$  do                                            // backward pass
      if  $l = L$  then                                              // compute error
         $\delta_k^{[L]} = \nabla_{a_k^{[L]}} \mathcal{L}_k(\Theta) \odot \sigma'(z_k^{[L]})$     for  $k = 1, \dots, B$ 
      else if  $L > l \geq 1$  then
         $\delta_k^{[l]} = [W^{[l+1]}]^T \delta_k^{[l+1]} \odot \sigma'(z_k^{[l]})$     for  $k = 1, \dots, B$ 
      end
       $\nabla_{W^{[l]}} \mathcal{L}_k(\Theta) = \delta_k^{[l]} [a_k^{[l-1]}]^T$     for  $k = 1, \dots, B$ 
       $\nabla_{b^{[l]}} \mathcal{L}_k(\Theta) = \delta_k^{[l]}$     for  $k = 1, \dots, B$ 
       $W^{[l]} := W^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^B \nabla_{W^{[l]}} \mathcal{L}_k(\Theta)$     // update gradient
       $b^{[l]} := b^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^B \nabla_{b^{[l]}} \mathcal{L}_k(\Theta)$     // update gradient
    end
  end
end

```

# BACKPROPAGATION (BP)

## Algorithm 2: Backpropagation (stochastic gradient descent)

**Data:**  $\text{dataloader} = \{B_i = (X_i, Y_i) : i = 1, \dots, nb\_batches\}$

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

```

for  $epochs = 1$  to  $nb\_epochs$  do                                // loop over epochs
  for  $i = 1$  to  $nb\_batches$  do                                    // loop over batches
     $A^{[0]} = X_i$ 
    for  $l = 1$  to  $L$  do                                            // forward pass
       $Z^{[l]} = W^{[l]} A^{[l-1]} + b^{[l]}$ 
       $A^{[l]} = \sigma(Z^{[l]})$ 
    end
    for  $l = L$  to  $1$  do                                            // backward pass
      if  $l = L$  then                                              // compute error
         $\delta_k^{[L]} = \nabla_{a_k^{[L]}} \mathcal{L}_k(\Theta) \odot \sigma'(z_k^{[L]})$     for  $k = 1, \dots, B$ 
      else if  $L > l \geq 1$  then
         $\delta_k^{[l]} = [W^{[l+1]}]^T \delta_k^{[l+1]} \odot \sigma'(z_k^{[l]})$     for  $k = 1, \dots, B$ 
      end
       $\nabla_{W^{[l]}} \mathcal{L}_k(\Theta) = \delta_k^{[l]} [a_k^{[l-1]}]^T$     for  $k = 1, \dots, B$ 
       $\nabla_{b^{[l]}} \mathcal{L}_k(\Theta) = \delta_k^{[l]}$     for  $k = 1, \dots, B$ 
       $W^{[l]} := W^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^B \nabla_{W^{[l]}} \mathcal{L}_k(\Theta)$     // update gradient
       $b^{[l]} := b^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^B \nabla_{b^{[l]}} \mathcal{L}_k(\Theta)$     // update gradient
    end
  end
end

```

# BACKPROPAGATION (BP)

## Algorithm 2: Backpropagation (stochastic gradient descent)

**Data:**  $\text{dataloader} = \{B_i = (X_i, Y_i) : i = 1, \dots, nb\_batches\}$

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

```

for  $epochs = 1$  to  $nb\_epochs$  do                                // loop over epochs
  for  $i = 1$  to  $nb\_batches$  do                                    // loop over batches
     $A^{[0]} = X_i$ 
    for  $l = 1$  to  $L$  do                                            // forward pass
       $Z^{[l]} = W^{[l]} A^{[l-1]} + b^{[l]}$ 
       $A^{[l]} = \sigma(Z^{[l]})$ 
    end
    for  $l = L$  to  $1$  do                                            // backward pass
      if  $l = L$  then                                              // compute error
         $\delta_k^{[L]} = \nabla_{a_k^{[L]}} \mathcal{L}_k(\Theta) \odot \sigma'(z_k^{[L]})$     for  $k = 1, \dots, B$ 
      else if  $L > l \geq 1$  then
         $\delta_k^{[l]} = [W^{[l+1]}]^T \delta_k^{[l+1]} \odot \sigma'(z_k^{[l]})$     for  $k = 1, \dots, B$ 
      end
       $\nabla_{W^{[l]}} \mathcal{L}_k(\Theta) = \delta_k^{[l]} [a_k^{[l-1]}]^T$     for  $k = 1, \dots, B$ 
       $\nabla_{b^{[l]}} \mathcal{L}_k(\Theta) = \delta_k^{[l]}$     for  $k = 1, \dots, B$ 
       $W^{[l]} := W^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^B \nabla_{W^{[l]}} \mathcal{L}_k(\Theta)$     // update gradient
       $b^{[l]} := b^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^B \nabla_{b^{[l]}} \mathcal{L}_k(\Theta)$     // update gradient
    end
  end
end

```

# BACKPROPAGATION (BP)

## Algorithm 2: Backpropagation (stochastic gradient descent)

**Data:**  $\text{dataloader} = \{B_i = (X_i, Y_i) : i = 1, \dots, nb\_batches\}$

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

```

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

```

# BACKPROPAGATION (BP)

## Algorithm 2: Backpropagation (stochastic gradient descent)

**Data:**  $\text{dataloader} = \{B_i = (X_i, Y_i) : i = 1, \dots, nb\_batches\}$

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

```

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

```

# BACKPROPAGATION (BP)

## Algorithm 2: Backpropagation (stochastic gradient descent)

**Data:**  $\text{dataloader} = \{B_i = (X_i, Y_i) : i = 1, \dots, nb\_batches\}$

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

```

for  $epochs = 1$  to  $nb\_epochs$  do                                // loop over epochs
  for  $i = 1$  to  $nb\_batches$  do                                    // loop over batches
     $A^{[0]} = X_i$ 
    for  $l = 1$  to  $L$  do                                           // forward pass
       $Z^{[l]} = W^{[l]} A^{[l-1]} + b^{[l]}$ 
       $A^{[l]} = \sigma(Z^{[l]})$ 
    end
    for  $l = L$  to  $1$  do                                           // backward pass
      if  $l = L$  then                                             // compute error
         $\delta_k^{[L]} = \nabla_{a_k^{[L]}} \mathcal{L}_k(\Theta) \odot \sigma'(z_k^{[L]})$     for  $k = 1, \dots, B$ 
      else if  $L > l \geq 1$  then
         $\delta_k^{[l]} = [W^{[l+1]}]^T \delta_k^{[l+1]} \odot \sigma'(z_k^{[l]})$     for  $k = 1, \dots, B$ 
      end
       $\nabla_{W^{[l]}} \mathcal{L}_k(\Theta) = \delta_k^{[l]} [a_k^{[l-1]}]^T$             for  $k = 1, \dots, B$ 
       $\nabla_{b^{[l]}} \mathcal{L}_k(\Theta) = \delta_k^{[l]}$                         for  $k = 1, \dots, B$ 
       $W^{[l]} := W^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^B \nabla_{W^{[l]}} \mathcal{L}_k(\Theta)$     // update gradient
       $b^{[l]} := b^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^B \nabla_{b^{[l]}} \mathcal{L}_k(\Theta)$     // update gradient
    end
  end
end

```

# BACKPROPAGATION (BP)

## Algorithm 2: Backpropagation (stochastic gradient descent)

**Data:**  $\text{dataloader} = \{B_i = (X_i, Y_i) : i = 1, \dots, nb\_batches\}$

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

```

for epochs = 1 to nb_epochs do                                // loop over epochs
  for i = 1 to nb_batches do                                   // loop over batches
     $A^{[0]} = X_i$ 
    for l = 1 to L do                                          // forward pass
       $Z^{[l]} = W^{[l]} A^{[l-1]} + b^{[l]}$ 
       $A^{[l]} = \sigma(Z^{[l]})$ 
    end
    for l = L to 1 do                                          // backward pass
      if l = L then                                           // compute error
         $\delta_k^{[L]} = \nabla_{a^{[L]}} \mathcal{L}_k(\Theta) \odot \sigma'(z_k^{[L]})$  for  $k = 1, \dots, B$ 
      else if  $L > l \geq 1$  then
         $\delta_k^{[l]} = [W^{[l+1]}]^T \delta_k^{[l+1]} \odot \sigma'(z_k^{[l]})$  for  $k = 1, \dots, B$ 
      end
       $\nabla_{W^{[l]}} \mathcal{L}_k(\Theta) = \delta_k^{[l]} [a_k^{[l-1]}]^T$  for  $k = 1, \dots, B$ 
       $\nabla_{b^{[l]}} \mathcal{L}_k(\Theta) = \delta_k^{[l]}$  for  $k = 1, \dots, B$ 
       $W^{[l]} := W^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^B \nabla_{W^{[l]}} \mathcal{L}_k(\Theta)$  // update gradient
       $b^{[l]} := b^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^B \nabla_{b^{[l]}} \mathcal{L}_k(\Theta)$  // update gradient
    end
  end
end

```

# BACKPROPAGATION (BP)

## Algorithm 2: Backpropagation (stochastic gradient descent)

**Data:**  $\text{dataloader} = \{ \mathbf{B}_i = (\mathbf{X}_i, \mathbf{Y}_i) : i = 1, \dots, nb\_batches \}$

**Inputs:**  $\text{MLP} = \{ (\mathbf{W}^{[l]}, \mathbf{b}^{[l]}) : l = 1, \dots, L \}$  initialized randomly

```

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

```



# BACKPROPAGATION (BP)

## Algorithm 2: Backpropagation (stochastic gradient descent)

**Data:**  $\text{dataloader} = \{B_i = (X_i, Y_i) : i = 1, \dots, nb\_batches\}$

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

```

for epochs = 1 to nb_epochs do                                // loop over epochs
  for i = 1 to nb_batches do                                    // loop over batches
     $A^{[0]} = X_i$ 
    for l = 1 to L do                                           // forward pass
       $Z^{[l]} = W^{[l]} A^{[l-1]} + b^{[l]}$ 
       $A^{[l]} = \sigma(Z^{[l]})$ 
    end
    // backward pass
    for l = L to 1 do                                           // compute error
      if l = L then
         $\delta_k^{[L]} = \nabla_{a_k^{[L]}} \mathcal{L}_k(\Theta) \odot \sigma'(z_k^{[L]})$     for  $k = 1, \dots, B$ 
      else if  $L > l \geq 1$  then
         $\delta_k^{[l]} = [W^{[l+1]}]^T \delta_k^{[l+1]} \odot \sigma'(z_k^{[l]})$     for  $k = 1, \dots, B$ 
      end
       $\nabla_{W^{[l]}} \mathcal{L}_k(\Theta) = \delta_k^{[l]} [a_k^{[l-1]}]^T$     for  $k = 1, \dots, B$ 
       $\nabla_{b^{[l]}} \mathcal{L}_k(\Theta) = \delta_k^{[l]}$     for  $k = 1, \dots, B$ 
       $W^{[l]} := W^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^B \nabla_{W^{[l]}} \mathcal{L}_k(\Theta)$     // update gradient
       $b^{[l]} := b^{[l]} - \frac{\eta}{B} \cdot \sum_{k=1}^B \nabla_{b^{[l]}} \mathcal{L}_k(\Theta)$     // update gradient
    end
  end
end

```

# EXEMPLE DE TRAINING VIA BACKROP

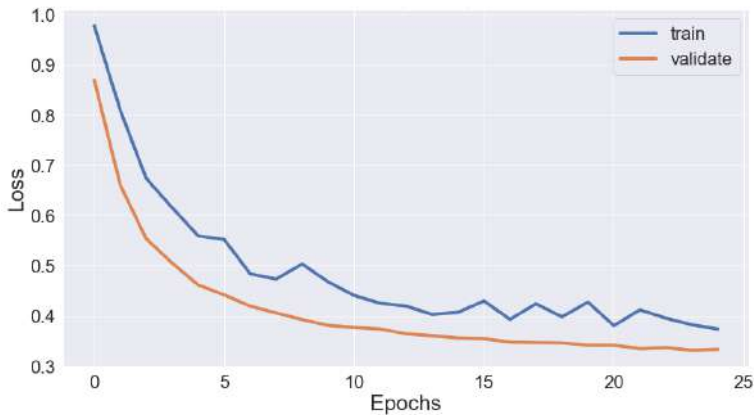


Figure taken from [towardsdatascience.com](https://towardsdatascience.com)

## 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.