# Descente de gradient

Jérémie Cabessa

Laboratoire DAVID, UVSQ

# Dérivée

- Soit $f : \mathbb{R} \longrightarrow \mathbb{R}$ une fonction. On dit que $f$ est *différentiable* au point $x_0 \in \mathbb{R}$ si la limite suivante existe:

$$\frac{df}{dx}(x_0) := \lim_{h \to 0} \frac{f(x_0 + h) - f(x_0)}{h}$$

- Supposons $f$ différentiable en tous points. Alors la limite

$$f'(x_0) := \frac{df}{dx}(x_0)$$

est la *dérivée de $f$ en* $x_0$.

- La fonction

$$\begin{array}{rcl} f' : \mathbb{R} & \longrightarrow & \mathbb{R} \\ x & \longmapsto & f'(x) \end{array}$$

est la *dérivée* de $f$.

# Dérivée

- Soit $f : \mathbb{R} \longrightarrow \mathbb{R}$ une fonction. On dit que $f$ est *différentiable* au point $x_0 \in \mathbb{R}$ si la limite suivante existe:

$$\frac{df}{dx}(x_0) := \lim_{h \to 0} \frac{f(x_0 + h) - f(x_0)}{h}$$

- Supposons $f$ différentiable en tous points. Alors la limite

$$f'(x_0) := \frac{df}{dx}(x_0)$$

est la *dérivée de $f$ en* $x_0$.

- La fonction

$$\begin{aligned} f' : \mathbb{R} &\longrightarrow \mathbb{R} \\ x &\longmapsto f'(x) \end{aligned}$$

est la *dérivée* de $f$.

# Dérivée

- Soit $f : \mathbb{R} \longrightarrow \mathbb{R}$ une fonction. On dit que $f$ est *différentiable* au point $x_0 \in \mathbb{R}$ si la limite suivante existe:

$$\frac{df}{dx}(x_0) := \lim_{h \to 0} \frac{f(x_0 + h) - f(x_0)}{h}$$

- Supposons $f$ différentiable en tous points. Alors la limite

$$f'(x_0) := \frac{df}{dx}(x_0)$$

est la *dérivée de $f$ en* $x_0$.

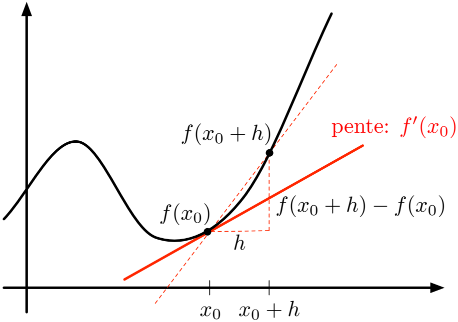
- La fonction

$$\begin{aligned} f' : \mathbb{R} &\longrightarrow \mathbb{R} \\ x &\longmapsto f'(x) \end{aligned}$$

est la *dérivée* de $f$.

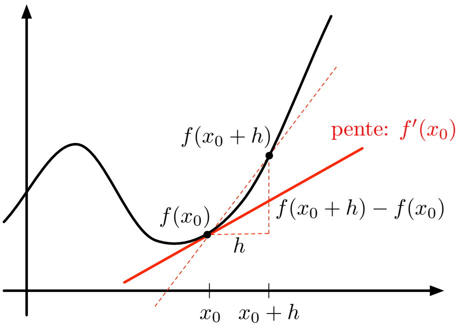
# Dérivée

- $f'(x_0)$ est la pente de la tangeante au graphe de $f$ au point $(x_0, f(x_0))$.
- $f'(x_0)$ est le taux d'accroissement de $f$ en $x_0$.



# Dérivée

- $f'(x_0)$ est la pente de la tangeante au graphe de $f$ au point $(x_0, f(x_0))$.
- $f'(x_0)$ est le taux d'accroissement de $f$ en $x_0$.



# Gradient

- Soient $f : \mathbb{R}^N \longrightarrow \mathbb{R}$ une fonction différentiable (définition un peu différente). Le *gradient de $f$ au point $\boldsymbol{x} = (x_1, \dots, x_N)$* est le vecteur défini par

$$\nabla f(\boldsymbol{x}) := \begin{pmatrix} \frac{\partial f}{\partial x_1}(\boldsymbol{x}) \\ \vdots \\ \frac{\partial f}{\partial x_N}(\boldsymbol{x}) \end{pmatrix}$$

- La fonction

$$\begin{aligned} \nabla f : \mathbb{R}^N &\longrightarrow \mathbb{R}^N \\ \boldsymbol{x} &\longmapsto \nabla f(\boldsymbol{x}) \end{aligned}$$

est le *gradient* de $f$.

# Gradient

- Soient $f : \mathbb{R}^N \longrightarrow \mathbb{R}$ une fonction différentiable (définition un peu différente). Le *gradient de $f$ au point $\boldsymbol{x} = (x_1, \ldots, x_N)$* est le vecteur défini par

$$\nabla f(\boldsymbol{x}) := \begin{pmatrix} \frac{\partial f}{\partial x_1}(\boldsymbol{x}) \\ \vdots \\ \frac{\partial f}{\partial x_N}(\boldsymbol{x}) \end{pmatrix}$$

- La fonction

$$\begin{aligned} \nabla f : \mathbb{R}^N &\longrightarrow \mathbb{R}^N \\ \boldsymbol{x} &\longmapsto \nabla f(\boldsymbol{x}) \end{aligned}$$

est le *gradient* de $f$.

# Gradient

- $\nabla f(\boldsymbol{x})$ est un vecteur de dimension $N$ (“dans le sol”).
- Chaque dimension $\frac{\partial f}{\partial x_i}(\boldsymbol{x})$ de $\nabla f(\boldsymbol{x})$ représente le taux d'accroissement de $f$ en $\boldsymbol{x}$ dans la direction $\boldsymbol{e}_i$.

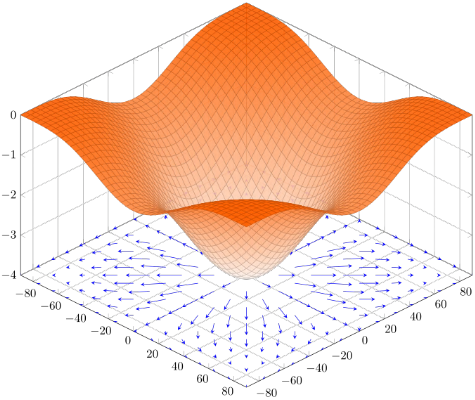



Figure taken from [Wikipedia contributors, 2022a]

# Gradient

- $\nabla f(\boldsymbol{x})$ est un vecteur de dimension $N$ ("dans le sol").
- Chaque dimension $\frac{\partial f}{\partial x_i}(\boldsymbol{x})$ de $\nabla f(\boldsymbol{x})$ représente le taux d'accroissement de $f$ en $\boldsymbol{x}$ dans la direction $\boldsymbol{e_i}$.

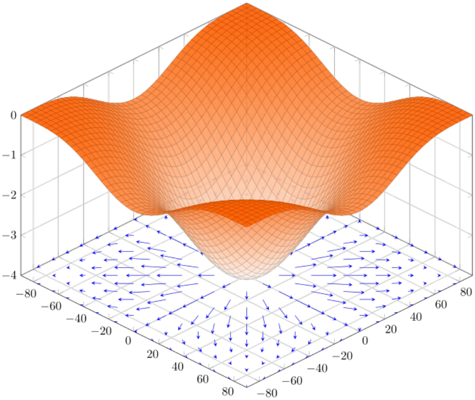



Figure taken from [Wikipedia contributors, 2022a]

# Gradient

- $\nabla f(\boldsymbol{x})$ représente la direction de la plus grande pente ascendante de $f$ au point $\boldsymbol{x}$.
- $\nabla f(x)$ is the **direction of steepest ascent** at $x$.
- De manière équivalente, $-\nabla f(x)$ représente la direction de la plus grande pente descendante de $f$ au point $x$.
- $-\nabla f(x)$ is the **direction of steepest descent** at $x$.

# Gradient

- $\nabla f(\boldsymbol{x})$ représente la direction de la plus grande pente ascendante de $f$ au point $\boldsymbol{x}$.
- $\nabla f(\boldsymbol{x})$ is the **direction of steepest ascent** at $\boldsymbol{x}$.
- De manière équivalente, $-\nabla f(\boldsymbol{x})$ représente la direction de la plus grande pente descendante de $f$ au point $\boldsymbol{x}$.
- $-\nabla f(\boldsymbol{x})$ is the **direction of steepest descent** at $\boldsymbol{x}$.

# Gradient

- $\nabla f(\boldsymbol{x})$ représente la direction de la plus grande pente ascendante de $f$ au point $\boldsymbol{x}$.
- $\nabla f(\boldsymbol{x})$ is the **direction of steepest ascent** at $\boldsymbol{x}$.
- De manière équivalente, $-\nabla f(\boldsymbol{x})$ représente la direction de la plus grande pente descendante de $f$ au point $\boldsymbol{x}$.
- $-\nabla f(\boldsymbol{x})$ is the **direction of steepest descent** at $\boldsymbol{x}$.

# Gradient

- $\nabla f(\boldsymbol{x})$ représente la direction de la plus grande pente ascendante de $f$ au point $\boldsymbol{x}$.
- $\nabla f(\boldsymbol{x})$ is the **direction of steepest ascent** at $\boldsymbol{x}$.
- De manière équivalente, $-\nabla f(\boldsymbol{x})$ représente la direction de la plus grande pente descendante de $f$ au point $\boldsymbol{x}$.
- $-\nabla f(\boldsymbol{x})$ is the **direction of steepest descent** at $\boldsymbol{x}$.

# Gradient

- La *dérivée directionnelle* de $f$ au point $\boldsymbol{x} = (x_1, \ldots, x_N)$ selon la direction $\boldsymbol{v} = (v_1, \ldots, v_N)$ est défini par

$$\nabla_{\boldsymbol{v}} f(\boldsymbol{x}) := \lim_{h \to 0} \frac{f(\boldsymbol{x} + h\boldsymbol{v}) - f(\boldsymbol{x})}{h\|\boldsymbol{v}\|}.$$

- Intuitivement, $\nabla_{\boldsymbol{v}} f(\boldsymbol{x})$ représente le taux d'accroissement de $f$ en $\boldsymbol{x}$ dans la direction $\boldsymbol{v}$.

# Gradient

- Dire que le **gradient** est la **direction de la plus grande pente ascendante** signifie formellement que:

  le **gradient** est le vecteur qui maximise la **dérivée directionnelle**.

**Theorem**

*Soit* $f : \mathbb{R}^N \longrightarrow \mathbb{R}$ *une fonction différentiable. Pour tout* $\boldsymbol{x} \in \mathbb{R}^N$, *on a:*

$$\nabla f(\boldsymbol{x}) \in \arg\max_{\boldsymbol{v} \in \mathbb{R}^N} \nabla_{\boldsymbol{v}} f(\boldsymbol{x}).$$

# Gradient

**Preuve:** On peut montrer que

$$\nabla_{\boldsymbol{v}} f(\boldsymbol{x}) = \nabla f(\boldsymbol{x}) \cdot \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|} = \|\nabla f(\boldsymbol{x})\| \cdot \frac{\|\boldsymbol{v}\|}{\|\boldsymbol{v}\|} \cdot \cos\theta = \|\nabla f(\boldsymbol{x})\| \cdot \cos(\theta)$$

où $\theta := \theta\left(\nabla f(\boldsymbol{x}), \boldsymbol{v}\right)$.

Ainsi, $\nabla_{\boldsymbol{v}} f(\boldsymbol{x})$ est maximal lorsque $\cos(\theta) = 1$, i.e., lorsque $\boldsymbol{v}$ est parallèle à $\nabla f(\boldsymbol{x})$. Donc en particulier,

$$\nabla f(\boldsymbol{x}) \in \arg\max_{\boldsymbol{v} \in \mathbb{R}^N} \nabla_{\boldsymbol{v}} f(\boldsymbol{x}).$$

$\square$

# Gradient

**Preuve:** On peut montrer que

$$\nabla_{\boldsymbol{v}} f(\boldsymbol{x}) = \nabla f(\boldsymbol{x}) \cdot \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|} = \|\nabla f(\boldsymbol{x})\| \cdot \frac{\|\boldsymbol{v}\|}{\|\boldsymbol{v}\|} \cdot \cos\theta = \|\nabla f(\boldsymbol{x})\| \cdot \cos(\theta)$$

où $\theta := \theta\left(\nabla f(\boldsymbol{x}), \boldsymbol{v}\right)$.

Ainsi, $\nabla_{\boldsymbol{v}} f(\boldsymbol{x})$ est maximal lorsque $\cos(\theta) = 1$, i.e., lorsque $\boldsymbol{v}$ est parallèle à $\nabla f(\boldsymbol{x})$. Donc en particulier,

$$\nabla f(\boldsymbol{x}) \in \arg\max_{\boldsymbol{v} \in \mathbb{R}^N} \nabla_{\boldsymbol{v}} f(\boldsymbol{x}).$$

□

# Gradient

**Preuve:** On peut montrer que

$$\nabla_{\boldsymbol{v}} f(\boldsymbol{x}) = \nabla f(\boldsymbol{x}) \cdot \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|} = \|\nabla f(\boldsymbol{x})\| \cdot \frac{\|\boldsymbol{v}\|}{\|\boldsymbol{v}\|} \cdot \cos\theta = \|\nabla f(\boldsymbol{x})\| \cdot \cos(\theta)$$

où $\theta := \theta\left(\nabla f(\boldsymbol{x}), \boldsymbol{v}\right)$.

Ainsi, $\nabla_{\boldsymbol{v}} f(\boldsymbol{x})$ est maximal lorsque $\cos(\theta) = 1$, i.e., lorsque $\boldsymbol{v}$ est parallèle à $\nabla f(\boldsymbol{x})$. Donc en particulier,

$$\nabla f(\boldsymbol{x}) \in \arg\max_{\boldsymbol{v} \in \mathbb{R}^N} \nabla_{\boldsymbol{v}} f(\boldsymbol{x}).$$

$\square$

# Gradient

**Preuve:** On peut montrer que

$$\nabla_{\boldsymbol{v}} f(\boldsymbol{x}) = \nabla f(\boldsymbol{x}) \cdot \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|} = \|\nabla f(\boldsymbol{x})\| \cdot \frac{\|\boldsymbol{v}\|}{\|\boldsymbol{v}\|} \cdot \cos\theta = \|\nabla f(\boldsymbol{x})\| \cdot \cos(\theta)$$

où $\theta := \theta\left(\nabla f(\boldsymbol{x}), \boldsymbol{v}\right)$.

Ainsi, $\nabla_{\boldsymbol{v}} f(\boldsymbol{x})$ est maximal lorsque $\cos(\theta) = 1$, i.e., lorsque $\boldsymbol{v}$ est parallèle à $\nabla f(\boldsymbol{x})$. Donc en particulier,

$$\nabla f(\boldsymbol{x}) \in \arg\max_{\boldsymbol{v} \in \mathbb{R}^N} \nabla_{\boldsymbol{v}} f(\boldsymbol{x}).$$

$\square$

# Descente de gradient

- Soit $f : \mathbb{R}^N \longrightarrow \mathbb{R}$ une fonction différentiable.
- On cherche à minimiser $f$: minimum local, global.
- **Descente de gradient:** on part d'un point au hasard; puis pas à pas, on descend le long de la surface en direction de la plus grande pente $-\nabla f(x)$.
- **Remarque:** on ne descend pas vraiment le long de la surface, on bouge dans le sol...

# Descente de gradient

- Soit $f : \mathbb{R}^N \longrightarrow \mathbb{R}$ une fonction différentiable.
- On cherche à minimiser $f$: minimum local, global.
- **Descente de gradient:** on part d'un point au hasard; puis pas à pas, on descend le long de la surface en direction de la plus grande pente $-\nabla f(x)$.
- **Remarque:** on ne descend pas vraiment le long de la surface, on bouge dans le sol...

# Descente de gradient

- Soit $f : \mathbb{R}^N \longrightarrow \mathbb{R}$ une fonction différentiable.
- On cherche à minimiser $f$: minimum local, global.
- **Descente de gradient:** on part d'un point au hasard; puis pas à pas, on descend le long de la surface en direction de la plus grande pente $-\nabla f(\boldsymbol{x})$.
- **Remarque:** on ne descend pas vraiment le long de la surface, on bouge dans le sol...

# Descente de gradient

- Soit $f : \mathbb{R}^N \longrightarrow \mathbb{R}$ une fonction différentiable.
- On cherche à minimiser $f$: minimum local, global.
- **Descente de gradient:** on part d'un point au hasard; puis pas à pas, on descend le long de la surface en direction de la plus grande pente $-\nabla f(\boldsymbol{x})$.
- **Remarque:** on ne descend pas vraiment le long de la surface, on bouge dans le sol...

# Descente de gradient

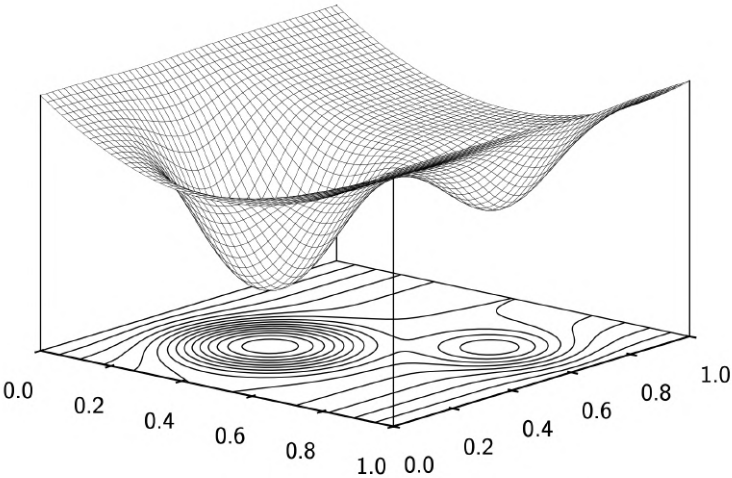



Figure taken from [Fleuret, 2022]

# Descente de gradient

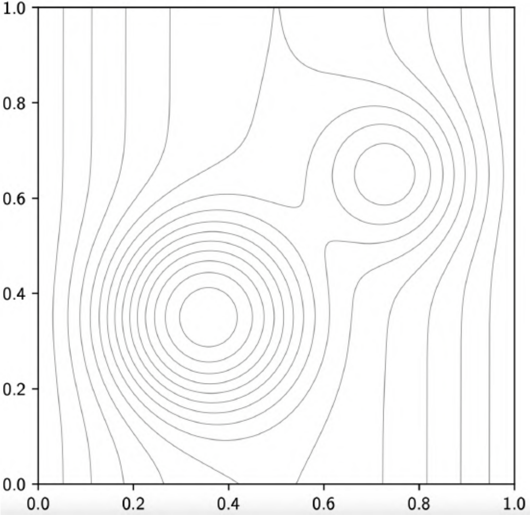



Figure taken from [Fleuret, 2022]

# Descente de gradient

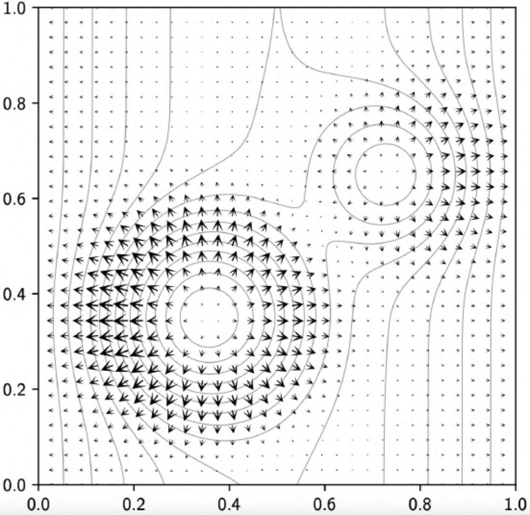



Figure taken from [Fleuret, 2022]

# Descente de gradient

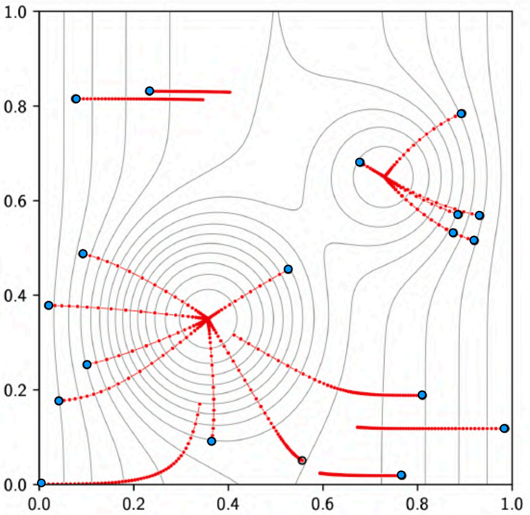



Figure taken from [Fleuret, 2022]

# Descente de gradient

**Movie 1**

**Movie 2**

**Movie 3**

Videos taken from the "3 Blue 1 Braun" YouTube channel

# Descente de gradient

**Algorithm 1:** Gradient descent

**Inputs:** differentiable function $f : \mathbb{R}^N \longrightarrow \mathbb{R}$;
initial point $\boldsymbol{x} \in \mathbb{R}^N$; learning rate $\lambda > 0$; tolerence $\epsilon > 0$.

**while** $\|\nabla f(\boldsymbol{x})\| > \epsilon$ **do**
$\quad \boldsymbol{x} := \boldsymbol{x} - \lambda \nabla f(\boldsymbol{x})$
**end**
**return** $\boldsymbol{x}$

- L'algorithme donne lieu à une suite de points $x_0, x_1, x_2, \ldots$ qui, espérons, converge vers un minimum local ou global.

# Descente de gradient

---

**Algorithm 1:** Gradient descent

---

**Inputs:** differentiable function $f : \mathbb{R}^N \longrightarrow \mathbb{R}$;
initial point $\boldsymbol{x} \in \mathbb{R}^N$; learning rate $\lambda > 0$; tolerence $\epsilon > 0$.

**while** $\|\nabla f(\boldsymbol{x})\| > \epsilon$ **do**
  $\boldsymbol{x} := \boldsymbol{x} - \lambda \nabla f(\boldsymbol{x})$
**end**
**return** $\boldsymbol{x}$

---

- L'algorithme donne lieu à une suite de points $x_0, x_1, x_2, \ldots$ qui, espérons, converge vers un minimum local ou global.

# Descente de gradient

**Algorithm 1:** Gradient descent

**Inputs:** differentiable function $f : \mathbb{R}^N \longrightarrow \mathbb{R}$;
initial point $\boldsymbol{x} \in \mathbb{R}^N$; learning rate $\lambda > 0$; tolerence $\epsilon > 0$.

**while** $\|\nabla f(\boldsymbol{x})\| > \epsilon$ **do**
  $\boldsymbol{x} := \boldsymbol{x} - \lambda \nabla f(\boldsymbol{x})$
**end**
**return** $\boldsymbol{x}$

- L'algorithme donne lieu à une suite de points $\boldsymbol{x_0}, \boldsymbol{x_1}, \boldsymbol{x_2}, \ldots$ qui, espérons, converge vers un minimum local ou global.

# Descente de gradient

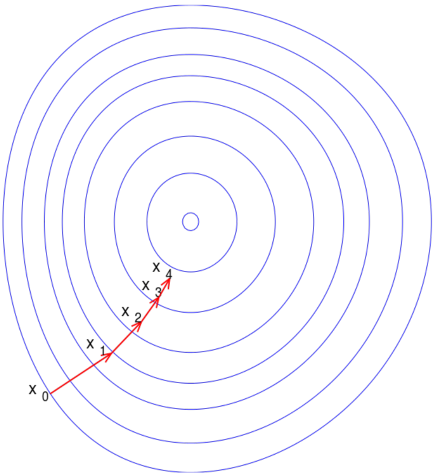



Figure taken from [Wikipedia contributors, 2022b]

# Learning Problem

- Soit $S = \left\{(\boldsymbol{x_i}, \boldsymbol{y_i}) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \ldots, N\right\}$ un dataset.
- Soit $\hat{f}(\cdot; \boldsymbol{\Theta})$ un modèle qui dépend des paramètres $\boldsymbol{\Theta}$:

$$\begin{aligned} \hat{f}(\cdot; \boldsymbol{\Theta}) : \mathbb{R}^{d_1} & \longrightarrow \mathbb{R}^{d_2} \\ \boldsymbol{x_i} & \longmapsto \hat{\boldsymbol{y}}_i := \hat{f}(\boldsymbol{x_i}; \boldsymbol{\Theta}) \end{aligned}$$

# Learning Problem

- Soit $S = \left\{(\boldsymbol{x_i}, \boldsymbol{y_i}) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \ldots, N\right\}$ un dataset.
- Soit $\hat{f}(\cdot; \boldsymbol{\Theta})$ un modèle qui dépend des paramètres $\boldsymbol{\Theta}$:

$$\begin{aligned} \hat{f}(\cdot; \boldsymbol{\Theta}) : \mathbb{R}^{d_1} & \longrightarrow \mathbb{R}^{d_2} \\ \boldsymbol{x_i} & \longmapsto \boldsymbol{\hat{y}_i} := \hat{f}(\boldsymbol{x_i}; \boldsymbol{\Theta}) \end{aligned}$$

# Learning Problem

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

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

- La *fonction de coût* peut être généralisée à un ensemble de *prédictions* et de *réalités* (e.g., moyenne des $\ell$ individuelles):

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

# Learning Problem

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

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

- La *fonction de coût* peut être généralisée à un ensemble de *prédictions* et de *réalités* (e.g., moyenne des $\ell$ individuelles):

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

# Learning Problem

- Pour différents paramètres $\boldsymbol{\Theta}$, on aura différentes prédictions $\hat{\boldsymbol{y}}_{\boldsymbol{i}} = \hat{f}\left(\boldsymbol{x}_{\boldsymbol{i}}; \boldsymbol{\Theta}\right)$, et donc différentes erreurs $\ell\left(\dots\right)$ et $\mathcal{L}\left(\dots\right)$.
- Ainsi, $\ell$ et $\mathcal{L}$ sont aussi des fonctions des paramètres $\boldsymbol{\Theta}$:

$$
\begin{aligned}
\ell : \mathbb{R}^{|\boldsymbol{\Theta}|} &\longrightarrow \mathbb{R} \\
\boldsymbol{\Theta} &\longmapsto \ell\left(\hat{\boldsymbol{y}}_{\boldsymbol{i}}, \boldsymbol{y}_{\boldsymbol{i}}; \boldsymbol{\Theta}\right) \\
\mathcal{L} : \mathbb{R}^{|\boldsymbol{\Theta}|} &\longrightarrow \mathbb{R} \\
\boldsymbol{\Theta} &\longmapsto \mathcal{L}\left(\hat{\boldsymbol{y}}_{\boldsymbol{1}}, \dots, \hat{\boldsymbol{y}}_{\boldsymbol{N}}, \boldsymbol{y}_{\boldsymbol{1}}, \dots, \boldsymbol{y}_{\boldsymbol{N}}; \boldsymbol{\Theta}\right)
\end{aligned}
$$

où $|\boldsymbol{\Theta}|$ est le nombre de paramètres $\boldsymbol{\Theta}$.

# Learning Problem

- Pour différents paramètres $\boldsymbol{\Theta}$, on aura différentes prédictions $\hat{\boldsymbol{y}}_{\boldsymbol{i}} = \hat{f}\left(\boldsymbol{x}_{\boldsymbol{i}}; \boldsymbol{\Theta}\right)$, et donc différentes erreurs $\ell\left(\dots\right)$ et $\mathcal{L}\left(\dots\right)$.
- **Ainsi, $\ell$ et $\mathcal{L}$ sont aussi des fonctions des paramètres $\boldsymbol{\Theta}$:**

$$
\begin{aligned}
\ell : \mathbb{R}^{|\boldsymbol{\Theta}|} & \longrightarrow \mathbb{R} \\
\boldsymbol{\Theta} & \longmapsto \ell\big(\hat{\boldsymbol{y}}_{\boldsymbol{i}}, \boldsymbol{y}_{\boldsymbol{i}}; \boldsymbol{\Theta}\big) \\
\mathcal{L} : \mathbb{R}^{|\boldsymbol{\Theta}|} & \longrightarrow \mathbb{R} \\
\boldsymbol{\Theta} & \longmapsto \mathcal{L}\big(\hat{\boldsymbol{y}}_{\boldsymbol{1}}, \dots, \hat{\boldsymbol{y}}_{\boldsymbol{N}}, \boldsymbol{y}_{\boldsymbol{1}}, \dots, \boldsymbol{y}_{\boldsymbol{N}}; \boldsymbol{\Theta}\big)
\end{aligned}
$$

où $|\boldsymbol{\Theta}|$ est le nombre de paramètres $\boldsymbol{\Theta}$.

# Learning Problem

- *L'entraînement* du modèle $\hat{f}(\cdots; \boldsymbol{\Theta})$ consiste à déterminer des paramètres $\boldsymbol{\Theta}$ qui minimisent les fonction de coût

$$\ell(\ldots; \boldsymbol{\Theta}) \quad \text{ou} \quad \mathcal{L}(\ldots; \boldsymbol{\Theta}).$$

- Pour minimiser la fonction de coût, on utilise des descentes de gradient:
  - **gradient descent**
  - stochastic gradient descent
  - mini-batch stochastic gradient descent

# Learning Problem

- *L'entraînement* du modèle $\hat{f}(\cdots;\boldsymbol{\Theta})$ consiste à déterminer des paramètres $\boldsymbol{\Theta}$ qui minimisent les fonction de coût

$$\ell(\ldots;\boldsymbol{\Theta}) \text{ ou } \mathcal{L}(\ldots;\boldsymbol{\Theta}).$$

- Pour minimiser la fonction de coût, on utilise des descentes de gradient:
  - gradient descent
  - stochastic gradient descent
  - mini-batch stochastic gradient descent

# Learning Problem

- *L'entraînement* du modèle $\hat{f}(\cdots;\mathbf{\Theta})$ consiste à déterminer des paramètres $\mathbf{\Theta}$ qui minimisent les fonction de coût

$$\ell(\ldots;\mathbf{\Theta}) \text{ ou } \mathcal{L}(\ldots;\mathbf{\Theta}).$$

- Pour minimiser la fonction de coût, on utilise des descentes de gradient:
  - **gradient descent**
  - stochastic gradient descent
  - mini-batch stochastic gradient descent

# Learning Problem

- *L'entraînement* du modèle $\hat{f}(\cdots; \mathbf{\Theta})$ consiste à déterminer des paramètres $\mathbf{\Theta}$ qui minimisent les fonction de coût

$$\ell(\ldots; \mathbf{\Theta}) \quad \text{ou} \quad \mathcal{L}(\ldots; \mathbf{\Theta}).$$

- Pour minimiser la fonction de coût, on utilise des descentes de gradient:

  - **gradient descent**
  - **stochastic gradient descent**
  - mini-batch stochastic gradient descent

# Learning Problem

- *L'entraînement* du modèle $\hat{f}(\cdots;\mathbf{\Theta})$ consiste à déterminer des paramètres $\mathbf{\Theta}$ qui minimisent les fonction de coût

$$\ell(\ldots;\mathbf{\Theta}) \text{ ou } \mathcal{L}(\ldots;\mathbf{\Theta}).$$

- Pour minimiser la fonction de coût, on utilise des descentes de gradient:
- **gradient descent**
- **stochastic gradient descent**
- **mini-batch stochastic gradient descent**

# Descentes de gradient

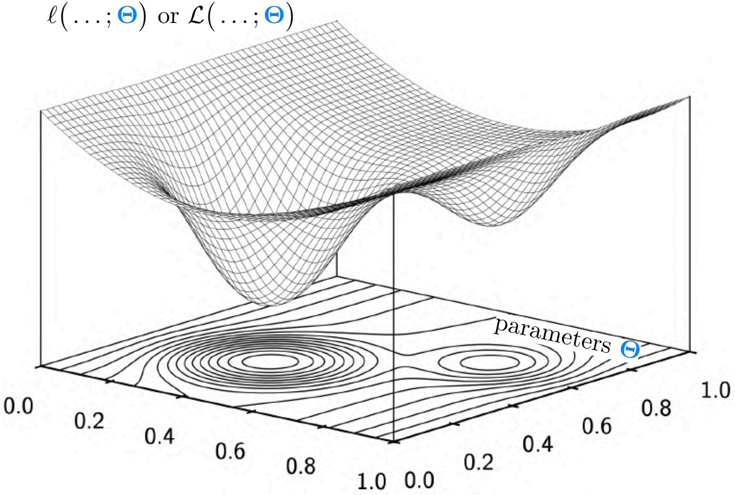



Figure adapted from [Fleuret, 2022]

# Descentes de gradient

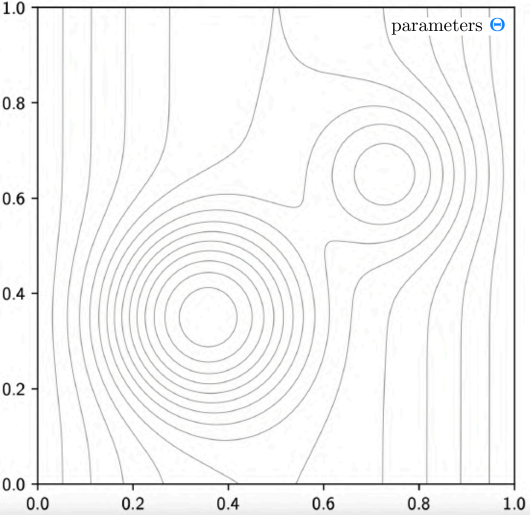



Figure adapted from [Fleuret, 2022]

# Descentes de gradient

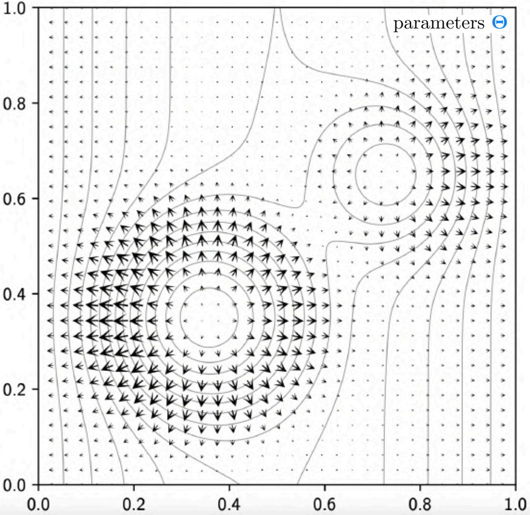



Figure adapted from [Fleuret, 2022]

# Descentes de gradient

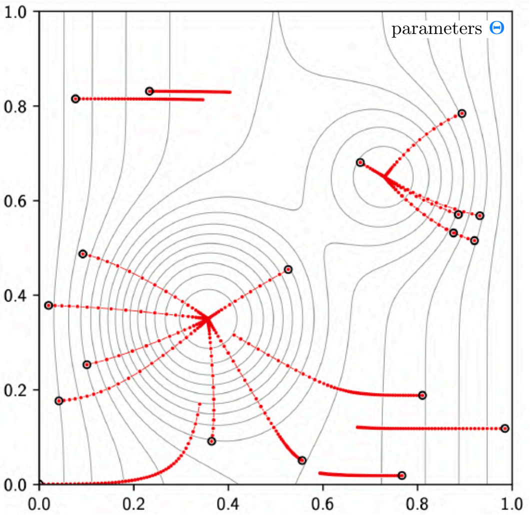



Figure adapted from [Fleuret, 2022]

# Gradient Descent (GD)

- **Gradient Descent (GD):** on update les paramètres $\boldsymbol{\Theta}$ après avoir passé le dataset en entier.
- Descente de gradient appliquée à la fonction de coût totale $\mathcal{L}(\ldots;\boldsymbol{\Theta})$.

# Gradient Descent (GD)

- **Gradient Descent (GD):** on update les paramètres $\mathbf{\Theta}$ après avoir passé le dataset en entier.
- Descente de gradient appliquée à la fonction de coût totale $\mathcal{L}(\ldots;\mathbf{\Theta})$.

# Gradient Descent (GD)

**Algorithm 2:** Gradient descent (GD)

**Inputs:** model $\hat{f}(\cdot;\boldsymbol{\Theta}) : \mathbb{R}^{d_1} \longrightarrow \mathbb{R}^{d_2}$;
dataset $S = \{(\boldsymbol{x_i}, \boldsymbol{y_i}) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \dots, N\}$;
differentiable loss function $\mathcal{L} : \mathbb{R}^{|\boldsymbol{\Theta}|} \longrightarrow \mathbb{R}$;
random initial parameters $\boldsymbol{\Theta}$; learning rate $\lambda > 0$; nb of epochs $nb\_epochs$.

```
for e = 1, ..., nb_epochs do
    for i = 1, ..., N do                          // compute predictions (dataset)
        ŷ_i = f̂(x_i; Θ)
    end
    L(Θ) := L(ŷ_1, ..., ŷ_N, y_1, ..., y_N; Θ)   // compute loss (dataset)
    Θ := Θ − λ∇L(Θ)                               // update gradient (dataset)
end
return f̂(·; Θ)
```

# Gradient Descent (GD)

**Algorithm 2:** Gradient descent (GD)

**Inputs:** model $\hat{f}(\cdot;\boldsymbol{\Theta}) : \mathbb{R}^{d_1} \longrightarrow \mathbb{R}^{d_2}$;
dataset $S = \left\{(\boldsymbol{x_i}, \boldsymbol{y_i}) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \ldots, N\right\}$;
differentiable loss function $\mathcal{L} : \mathbb{R}^{|\boldsymbol{\Theta}|} \longrightarrow \mathbb{R}$;
random initial parameters $\boldsymbol{\Theta}$; learning rate $\lambda > 0$; nb of epochs $nb\_epochs$.

```
for e = 1, ..., nb_epochs do
    for i = 1, ..., N do                          // compute predictions (dataset)
        ŷ_i = f̂(x_i; Θ)
    end
    L(Θ) := L(ŷ_1, ..., ŷ_N, y_1, ..., y_N; Θ)     // compute loss (dataset)
    Θ := Θ − λ∇L(Θ)                               // update gradient (dataset)
end
return f̂(·; Θ)
```

# Gradient Descent (GD)

**Algorithm 2:** Gradient descent (GD)

**Inputs:** model $\hat{f}(\cdot;\boldsymbol{\Theta}) : \mathbb{R}^{d_1} \longrightarrow \mathbb{R}^{d_2}$;
dataset $S = \left\{(\boldsymbol{x_i}, \boldsymbol{y_i}) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \ldots, N\right\}$;
differentiable loss function $\mathcal{L} : \mathbb{R}^{|\boldsymbol{\Theta}|} \longrightarrow \mathbb{R}$;
random initial parameters $\boldsymbol{\Theta}$; learning rate $\lambda > 0$; nb of epochs $nb\_epochs$.

**for** $e = 1, \ldots, nb\_epochs$ **do**
  **for** $i = 1, \ldots, N$ **do** `// compute predictions (dataset)`
    $\hat{\boldsymbol{y}}_{\boldsymbol{i}} = \hat{f}\left(\boldsymbol{x_i}; \boldsymbol{\Theta}\right)$
  **end**
  $\mathcal{L}\left(\boldsymbol{\Theta}\right) := \mathcal{L}\left(\hat{\boldsymbol{y}}_{\boldsymbol{1}}, \ldots, \hat{\boldsymbol{y}}_{\boldsymbol{N}}, \boldsymbol{y_1}, \ldots, \boldsymbol{y_N}; \boldsymbol{\Theta}\right)$ `// compute loss (dataset)`
  $\boldsymbol{\Theta} := \boldsymbol{\Theta} - \lambda \nabla \mathcal{L}\left(\boldsymbol{\Theta}\right)$ `// update gradient (dataset)`
**end**
**return** $\hat{f}\left(\cdot; \boldsymbol{\Theta}\right)$

# Gradient Descent (GD)

**Algorithm 2:** Gradient descent (GD)

**Inputs:** model $\hat{f}(\cdot; \mathbf{\Theta}) : \mathbb{R}^{d_1} \longrightarrow \mathbb{R}^{d_2}$;
dataset $S = \left\{ (\boldsymbol{x_i}, \boldsymbol{y_i}) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \ldots, N \right\}$;
differentiable loss function $\mathcal{L} : \mathbb{R}^{|\mathbf{\Theta}|} \longrightarrow \mathbb{R}$;
random initial parameters $\mathbf{\Theta}$; learning rate $\lambda > 0$; nb of epochs $nb\_epochs$.

**for** $e = 1, \ldots, nb\_epochs$ **do**
  **for** $i = 1, \ldots, N$ **do** `// compute predictions (dataset)`
    $\hat{\boldsymbol{y}}_{\boldsymbol{i}} = \hat{f}\left(\boldsymbol{x_i}; \mathbf{\Theta}\right)$
  **end**
  $\mathcal{L}\left(\mathbf{\Theta}\right) := \mathcal{L}\left(\hat{\boldsymbol{y}}_{\mathbf{1}}, \ldots, \hat{\boldsymbol{y}}_{\boldsymbol{N}}, \boldsymbol{y_1}, \ldots, \boldsymbol{y_N}; \mathbf{\Theta}\right)$ `// compute loss (dataset)`
  $\mathbf{\Theta} := \mathbf{\Theta} - \lambda \nabla \mathcal{L}\left(\mathbf{\Theta}\right)$ `// update gradient (dataset)`
**end**
**return** $\hat{f}\left(\cdot; \mathbf{\Theta}\right)$

# Gradient Descent (GD)

**Algorithm 2:** Gradient descent (GD)

**Inputs:** model $\hat{f}(\cdot;\boldsymbol{\Theta}) : \mathbb{R}^{d_1} \longrightarrow \mathbb{R}^{d_2}$;
dataset $S = \{(\boldsymbol{x_i}, \boldsymbol{y_i}) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \ldots, N\}$;
differentiable loss function $\mathcal{L} : \mathbb{R}^{|\boldsymbol{\Theta}|} \longrightarrow \mathbb{R}$;
random initial parameters $\boldsymbol{\Theta}$; learning rate $\lambda > 0$; nb of epochs $nb\_epochs$.

**for** $e = 1, \ldots, nb\_epochs$ **do**
  **for** $i = 1, \ldots, N$ **do** `// compute predictions (dataset)`
    $\hat{\boldsymbol{y}}_{\boldsymbol{i}} = \hat{f}(\boldsymbol{x_i}; \boldsymbol{\Theta})$
  **end**
  $\mathcal{L}(\boldsymbol{\Theta}) := \mathcal{L}(\hat{\boldsymbol{y}}_{\boldsymbol{1}}, \ldots, \hat{\boldsymbol{y}}_{\boldsymbol{N}}, \boldsymbol{y_1}, \ldots, \boldsymbol{y_N}; \boldsymbol{\Theta})$ `// compute loss (dataset)`
  $\boldsymbol{\Theta} := \boldsymbol{\Theta} - \lambda \nabla \mathcal{L}(\boldsymbol{\Theta})$ `// update gradient (dataset)`
**end**
**return** $\hat{f}(\cdot;\boldsymbol{\Theta})$

# Gradient Descent (GD)

**Algorithm 2:** Gradient descent (GD)

**Inputs:** model $\hat{f}(\cdot;\boldsymbol{\Theta}) : \mathbb{R}^{d_1} \longrightarrow \mathbb{R}^{d_2}$;
dataset $S = \left\{(\boldsymbol{x_i}, \boldsymbol{y_i}) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \ldots, N\right\}$;
differentiable loss function $\mathcal{L} : \mathbb{R}^{|\boldsymbol{\Theta}|} \longrightarrow \mathbb{R}$;
random initial parameters $\boldsymbol{\Theta}$; learning rate $\lambda > 0$; nb of epochs $nb\_epochs$.

**for** $e = 1, \ldots, nb\_epochs$ **do**
  **for** $i = 1, \ldots, N$ **do** `// compute predictions (dataset)`
    $\hat{\boldsymbol{y}}_{\boldsymbol{i}} = \hat{f}\left(\boldsymbol{x_i};\boldsymbol{\Theta}\right)$
  **end**
  $\mathcal{L}\left(\boldsymbol{\Theta}\right) := \mathcal{L}\left(\hat{\boldsymbol{y}}_{\boldsymbol{1}}, \ldots, \hat{\boldsymbol{y}}_{\boldsymbol{N}}, \boldsymbol{y_1}, \ldots, \boldsymbol{y_N};\boldsymbol{\Theta}\right)$ `// compute loss (dataset)`
  $\boldsymbol{\Theta} := \boldsymbol{\Theta} - \lambda\nabla\mathcal{L}\left(\boldsymbol{\Theta}\right)$ `// update gradient (dataset)`
**end**
**return** $\hat{f}\left(\cdot;\boldsymbol{\Theta}\right)$

# Gradient Descent (GD)

**Algorithm 2:** Gradient descent (GD)

**Inputs:** model $\hat{f}(\cdot; \boldsymbol{\Theta}) : \mathbb{R}^{d_1} \longrightarrow \mathbb{R}^{d_2}$;
dataset $S = \{(\boldsymbol{x_i}, \boldsymbol{y_i}) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \ldots, N\}$;
differentiable loss function $\mathcal{L} : \mathbb{R}^{|\boldsymbol{\Theta}|} \longrightarrow \mathbb{R}$;
random initial parameters $\boldsymbol{\Theta}$; learning rate $\lambda > 0$; nb of epochs $nb\_epochs$.

**for** $e = 1, \ldots, nb\_epochs$ **do**
  **for** $i = 1, \ldots, N$ **do** `// compute predictions (dataset)`
    $\hat{\boldsymbol{y}}_{\boldsymbol{i}} = \hat{f}(\boldsymbol{x_i}; \boldsymbol{\Theta})$
  **end**
  $\mathcal{L}(\boldsymbol{\Theta}) := \mathcal{L}(\hat{\boldsymbol{y}}_{\boldsymbol{1}}, \ldots, \hat{\boldsymbol{y}}_{\boldsymbol{N}}, \boldsymbol{y_1}, \ldots, \boldsymbol{y_N}; \boldsymbol{\Theta})$ `// compute loss (dataset)`
  $\boldsymbol{\Theta} := \boldsymbol{\Theta} - \lambda \nabla \mathcal{L}(\boldsymbol{\Theta})$ `// update gradient (dataset)`
**end**
**return** $\hat{f}(\cdot; \boldsymbol{\Theta})$

# Stochastic Gradient Descent (SGD)

- **Stochastic Gradient Descent (SGD):** on update les paramètres $\mathbf{\Theta}$ après chaque sample du dataset.
- Descente de gradient appliquée à chaque fonction de coût particulière $\ell(\ldots; \mathbf{\Theta})$.

# Stochastic Gradient Descent (SGD)

- **Stochastic Gradient Descent (SGD):** on update les paramètres $\mathbf{\Theta}$ après chaque sample du dataset.
- Descente de gradient appliquée à chaque fonction de coût particulière $\ell\big(\ldots; \mathbf{\Theta}\big)$.

# Stochastic Gradient Descent (SGD)

**Algorithm 3:** Stochastic Gradient descent (SGD)

**Inputs:** model $\hat{f}(\cdot; \boldsymbol{\Theta}) : \mathbb{R}^{d_1} \longrightarrow \mathbb{R}^{d_2}$;
dataset $S = \{(\boldsymbol{x_i}, \boldsymbol{y_i}) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \ldots, N\}$;
differentiable loss function $\ell : \mathbb{R}^{|\boldsymbol{\Theta}|} \longrightarrow \mathbb{R}$;
random initial parameters $\boldsymbol{\Theta}$; learning rate $\lambda > 0$; nb of epochs $nb\_epochs$.

**for** $e = 1, \ldots, nb\_epochs$ **do**
  **for** $i = 1, \ldots, N$ **do**
    $\hat{\boldsymbol{y}}_i = \hat{f}(\boldsymbol{x_i}; \boldsymbol{\Theta})$ // compute prediction (sample)
    $\ell(\boldsymbol{\Theta}) := \ell(\hat{\boldsymbol{y}}_i, \boldsymbol{y}_i; \boldsymbol{\Theta})$ // compute loss (sample)
    $\boldsymbol{\Theta} := \boldsymbol{\Theta} - \lambda \nabla \ell(\boldsymbol{\Theta})$ // update gradient (sample)
  **end**
**end**
**return** $\hat{f}(\cdot; \boldsymbol{\Theta})$

# Stochastic Gradient Descent (SGD)

**Algorithm 3:** Stochastic Gradient descent (SGD)

**Inputs:** model $\hat{f}(\cdot; \boldsymbol{\Theta}) : \mathbb{R}^{d_1} \longrightarrow \mathbb{R}^{d_2}$;
dataset $S = \{(\boldsymbol{x_i}, \boldsymbol{y_i}) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \ldots, N\}$;
differentiable loss function $\ell : \mathbb{R}^{|\boldsymbol{\Theta}|} \longrightarrow \mathbb{R}$;
random initial parameters $\boldsymbol{\Theta}$; learning rate $\lambda > 0$; nb of epochs $nb\_epochs$.

```
for e = 1, ..., nb_epochs do
    for i = 1, ..., N do
        ŷ_i = f̂(x_i; Θ)                // compute prediction (sample)
        ℓ(Θ) := ℓ(ŷ_i, y_i; Θ)         // compute loss (sample)
        Θ := Θ − λ∇ℓ(Θ)                // update gradient (sample)
    end
end
return f̂(·; Θ)
```

# Stochastic Gradient Descent (SGD)

**Algorithm 3:** Stochastic Gradient descent (SGD)

**Inputs:** model $\hat{f}(\cdot; \boldsymbol{\Theta}) : \mathbb{R}^{d_1} \longrightarrow \mathbb{R}^{d_2}$;
dataset $S = \{(\boldsymbol{x_i}, \boldsymbol{y_i}) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \dots, N\}$;
differentiable loss function $\ell : \mathbb{R}^{|\boldsymbol{\Theta}|} \longrightarrow \mathbb{R}$;
random initial parameters $\boldsymbol{\Theta}$; learning rate $\lambda > 0$; nb of epochs $nb\_epochs$.

```
for e = 1, ..., nb_epochs do
    for i = 1, ..., N do
        ŷ_i = f̂(x_i; Θ)                    // compute prediction (sample)
        ℓ(Θ) := ℓ(ŷ_i, y_i; Θ)             // compute loss (sample)
        Θ := Θ − λ∇ℓ(Θ)                    // update gradient (sample)
    end
end
return f̂(·; Θ)
```

# Stochastic Gradient Descent (SGD)

**Algorithm 3:** Stochastic Gradient descent (SGD)

**Inputs:** model $\hat{f}(\cdot;\boldsymbol{\Theta}) : \mathbb{R}^{d_1} \longrightarrow \mathbb{R}^{d_2}$;
dataset $S = \{(\boldsymbol{x_i}, \boldsymbol{y_i}) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \dots, N\}$;
differentiable loss function $\ell : \mathbb{R}^{|\boldsymbol{\Theta}|} \longrightarrow \mathbb{R}$;
random initial parameters $\boldsymbol{\Theta}$; learning rate $\lambda > 0$; nb of epochs $nb\_epochs$.

```
for e = 1, ..., nb_epochs do
    for i = 1, ..., N do
        ŷ_i = f̂(x_i; Θ)                       // compute prediction (sample)
        ℓ(Θ) := ℓ(ŷ_i, y_i; Θ)                 // compute loss (sample)
        Θ := Θ − λ∇ℓ(Θ)                        // update gradient (sample)
    end
end
return f̂(·; Θ)
```

# Stochastic Gradient Descent (SGD)

**Algorithm 3:** Stochastic Gradient descent (SGD)

**Inputs:** model $\hat{f}(\cdot; \boldsymbol{\Theta}) : \mathbb{R}^{d_1} \longrightarrow \mathbb{R}^{d_2}$;
dataset $S = \left\{ (\boldsymbol{x_i}, \boldsymbol{y_i}) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \dots, N \right\}$;
differentiable loss function $\ell : \mathbb{R}^{|\boldsymbol{\Theta}|} \longrightarrow \mathbb{R}$;
random initial parameters $\boldsymbol{\Theta}$; learning rate $\lambda > 0$; nb of epochs $nb\_epochs$.

**for** $e = 1, \dots, nb\_epochs$ **do**
  **for** $i = 1, \dots, N$ **do**
    $\hat{\boldsymbol{y}}_{\boldsymbol{i}} = \hat{f}\left(\boldsymbol{x_i}; \boldsymbol{\Theta}\right)$ `// compute prediction (sample)`
    $\ell\left(\boldsymbol{\Theta}\right) := \ell\left(\hat{\boldsymbol{y}}_{\boldsymbol{i}}, \boldsymbol{y_i}; \boldsymbol{\Theta}\right)$ `// compute loss (sample)`
    $\boldsymbol{\Theta} := \boldsymbol{\Theta} - \lambda \nabla \ell\left(\boldsymbol{\Theta}\right)$ `// update gradient (sample)`
  **end**
**end**
**return** $\hat{f}\left(\cdot; \boldsymbol{\Theta}\right)$

# Stochastic Gradient Descent (SGD)

**Algorithm 3:** Stochastic Gradient descent (SGD)

**Inputs:** model $\hat{f}(\cdot; \boldsymbol{\Theta}) : \mathbb{R}^{d_1} \longrightarrow \mathbb{R}^{d_2}$;
dataset $S = \{(\boldsymbol{x_i}, \boldsymbol{y_i}) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \ldots, N\}$;
differentiable loss function $\ell : \mathbb{R}^{|\boldsymbol{\Theta}|} \longrightarrow \mathbb{R}$;
random initial parameters $\boldsymbol{\Theta}$; learning rate $\lambda > 0$; nb of epochs $nb\_epochs$.

**for** $e = 1, \ldots, nb\_epochs$ **do**
  **for** $i = 1, \ldots, N$ **do**
    $\hat{\boldsymbol{y}}_{\boldsymbol{i}} = \hat{f}(\boldsymbol{x_i}; \boldsymbol{\Theta})$ `// compute prediction (sample)`
    $\ell(\boldsymbol{\Theta}) := \ell(\hat{\boldsymbol{y}}_{\boldsymbol{i}}, \boldsymbol{y_i}; \boldsymbol{\Theta})$ `// compute loss (sample)`
    $\boldsymbol{\Theta} := \boldsymbol{\Theta} - \lambda \nabla \ell(\boldsymbol{\Theta})$ `// update gradient (sample)`
  **end**
**end**
**return** $\hat{f}(\cdot; \boldsymbol{\Theta})$

# Mini-Batch Stochastic Gradient Descent (mb-SGD)

- **Mini-Batch Stochastic Gradient Descent (mb-SGD ou SGD):** on update les paramètres $\mathbf{\Theta}$ après chaque batch du dataset.
- Descente de gradient appliquée à la fonction de coût totale du batch $\mathcal{L}(\dots;\mathbf{\Theta})$.
- Cette méthode est la plus efficace.

# Mini-Batch Stochastic Gradient Descent (mb-SGD)

- **Mini-Batch Stochastic Gradient Descent (mb-SGD ou SGD):** on update les paramètres $\boldsymbol{\Theta}$ après chaque batch du dataset.
- Descente de gradient appliquée à la fonction de coût totale du batch $\mathcal{L}(\ldots;\boldsymbol{\Theta})$.
- Cette méthode est la plus efficace.

# Mini-Batch Stochastic Gradient Descent (mb-SGD)

- **Mini-Batch Stochastic Gradient Descent (mb-SGD ou SGD):** on update les paramètres $\boldsymbol{\Theta}$ après chaque batch du dataset.
- Descente de gradient appliquée à la fonction de coût totale du batch $\mathcal{L}(\ldots; \boldsymbol{\Theta})$.
- Cette méthode est la plus efficace.

# Mini-Batch Stochastic Gradient Descent (mb-SGD)

**Algorithm 4:** Mini-Batch Stochastic Gradient descent (SGD)

**Inputs:** model $\hat{f}(\cdot; \boldsymbol{\Theta}) : \mathbb{R}^{d_1} \longrightarrow \mathbb{R}^{d_2}$;
dataset $S = \left\{(\boldsymbol{x_i}, \boldsymbol{y_i}) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \ldots, N\right\}$;
differentiable loss function $\mathcal{L} : \mathbb{R}^{|\boldsymbol{\Theta}|} \longrightarrow \mathbb{R}$;
random initial parameters $\boldsymbol{\Theta}$; learning rate $\lambda > 0$; nb of epochs $nb\_epochs$;
batch size $B$.

**for** $e = 1, \ldots, nb\_epochs$ **do**
  **for** $b = 1, \ldots, nb\_batches$ **do**
    $\hat{\boldsymbol{Y}}_b = \hat{f}\left(\boldsymbol{X}_b; \boldsymbol{\Theta}\right)$ ; // compute prediction (batch)
    $\mathcal{L}_b\left(\boldsymbol{\Theta}\right) := \mathcal{L}\left(\hat{\boldsymbol{Y}}_b, \boldsymbol{Y}_b; \boldsymbol{\Theta}\right)$; // compute loss (batch)
  **end**
  $\boldsymbol{\Theta} := \boldsymbol{\Theta} - \lambda \sum_{i=1}^{B} \nabla \mathcal{L}_b\left(\boldsymbol{\Theta}\right)$ // update gradient
**end**
**return** $\hat{f}(\cdot; \boldsymbol{\Theta})$

# Mini-Batch Stochastic Gradient Descent (mb-SGD)

**Algorithm 4:** Mini-Batch Stochastic Gradient descent (SGD)

**Inputs:** model $\hat{f}(\cdot;\boldsymbol{\Theta}) : \mathbb{R}^{d_1} \longrightarrow \mathbb{R}^{d_2}$;
dataset $S = \left\{(\boldsymbol{x_i}, \boldsymbol{y_i}) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \ldots, N\right\}$;
differentiable loss function $\mathcal{L} : \mathbb{R}^{|\boldsymbol{\Theta}|} \longrightarrow \mathbb{R}$;
random initial parameters $\boldsymbol{\Theta}$; learning rate $\lambda > 0$; nb of epochs $nb\_epochs$;
batch size $B$.

**for** $e = 1, \ldots, nb\_epochs$ **do**
  **for** $b = 1, \ldots, nb\_batches$ **do**
    $\hat{\boldsymbol{Y}}_{\boldsymbol{b}} = \hat{f}\left(\boldsymbol{X}_{\boldsymbol{b}}; \boldsymbol{\Theta}\right)$ ;  `// compute prediction (batch)`
    $\mathcal{L}_b\left(\boldsymbol{\Theta}\right) := \mathcal{L}\left(\hat{\boldsymbol{Y}}_{\boldsymbol{b}}, \boldsymbol{Y}_{\boldsymbol{b}}; \boldsymbol{\Theta}\right)$;  `// compute loss (batch)`
  **end**
  $\boldsymbol{\Theta} := \boldsymbol{\Theta} - \lambda \sum_{i=1}^{B} \nabla \mathcal{L}_b\left(\boldsymbol{\Theta}\right)$  `// update gradient`
**end**
**return** $\hat{f}(\cdot;\boldsymbol{\Theta})$

# Mini-Batch Stochastic Gradient Descent (mb-SGD)

**Algorithm 4:** Mini-Batch Stochastic Gradient descent (SGD)

**Inputs:** model $\hat{f}(\cdot;\boldsymbol{\Theta}) : \mathbb{R}^{d_1} \longrightarrow \mathbb{R}^{d_2}$;
dataset $S = \left\{(\boldsymbol{x_i}, \boldsymbol{y_i}) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \ldots, N\right\}$;
differentiable loss function $\mathcal{L} : \mathbb{R}^{|\boldsymbol{\Theta}|} \longrightarrow \mathbb{R}$;
random initial parameters $\boldsymbol{\Theta}$; learning rate $\lambda > 0$; nb of epochs $nb\_epochs$;
batch size $B$.

**for** $e = 1, \ldots, nb\_epochs$ **do**
  **for** $b = 1, \ldots, nb\_batches$ **do**
    $\hat{\boldsymbol{Y}}_b = \hat{f}(\boldsymbol{X}_b;\boldsymbol{\Theta})$ ; // compute prediction (batch)
    $\mathcal{L}_b(\boldsymbol{\Theta}) := \mathcal{L}\left(\hat{\boldsymbol{Y}}_b, \boldsymbol{Y}_b;\boldsymbol{\Theta}\right)$; // compute loss (batch)
  **end**
  $\boldsymbol{\Theta} := \boldsymbol{\Theta} - \lambda \sum_{i=1}^{B} \nabla \mathcal{L}_b(\boldsymbol{\Theta})$ // update gradient
**end**
**return** $\hat{f}(\cdot;\boldsymbol{\Theta})$

# Mini-Batch Stochastic Gradient Descent (mb-SGD)

**Algorithm 4:** Mini-Batch Stochastic Gradient descent (SGD)

**Inputs:** model $\hat{f}(\cdot;\boldsymbol{\Theta}) : \mathbb{R}^{d_1} \longrightarrow \mathbb{R}^{d_2}$;
dataset $S = \left\{(\boldsymbol{x_i}, \boldsymbol{y_i}) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \dots, N\right\}$;
differentiable loss function $\mathcal{L} : \mathbb{R}^{|\boldsymbol{\Theta}|} \longrightarrow \mathbb{R}$;
random initial parameters $\boldsymbol{\Theta}$; learning rate $\lambda > 0$; nb of epochs $nb\_epochs$;
batch size $B$.

**for** $e = 1, \dots, nb\_epochs$ **do**
  **for** $b = 1, \dots, nb\_batches$ **do**
    $\hat{\boldsymbol{Y}}_{\boldsymbol{b}} = \hat{f}\left(\boldsymbol{X_b}; \boldsymbol{\Theta}\right)$ ; `// compute prediction (batch)`
    $\mathcal{L}_b\left(\boldsymbol{\Theta}\right) := \mathcal{L}\left(\hat{\boldsymbol{Y}}_b, \boldsymbol{Y_b}; \boldsymbol{\Theta}\right)$; `// compute loss (batch)`
  **end**
  $\boldsymbol{\Theta} := \boldsymbol{\Theta} - \lambda \sum_{i=1}^{B} \nabla \mathcal{L}_b\left(\boldsymbol{\Theta}\right)$ `// update gradient`
**end**
**return** $\hat{f}(\cdot;\boldsymbol{\Theta})$

# Mini-Batch Stochastic Gradient Descent (mb-SGD)

**Algorithm 4:** Mini-Batch Stochastic Gradient descent (SGD)

**Inputs:** model $\hat{f}(\cdot; \boldsymbol{\Theta}) : \mathbb{R}^{d_1} \longrightarrow \mathbb{R}^{d_2}$;
dataset $S = \{(\boldsymbol{x_i}, \boldsymbol{y_i}) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \dots, N\}$;
differentiable loss function $\mathcal{L} : \mathbb{R}^{|\boldsymbol{\Theta}|} \longrightarrow \mathbb{R}$;
random initial parameters $\boldsymbol{\Theta}$; learning rate $\lambda > 0$; nb of epochs $nb\_epochs$;
batch size $B$.

```
for e = 1, ..., nb_epochs do
    for b = 1, ..., nb_batches do
        Ŷ_b = f̂(X_b; Θ) ;                         // compute prediction (batch)
        L_b(Θ) := L(Ŷ_b, Y_b; Θ);                 // compute loss (batch)
    end
    Θ := Θ − λ Σ_{i=1}^{B} ∇L_b(Θ)                 // update gradient
end
return f̂(·; Θ)
```

$\boldsymbol{\hat{Y}_b} = \hat{f}\left(\boldsymbol{X_b}; \boldsymbol{\Theta}\right)$

$\mathcal{L}_b\left(\boldsymbol{\Theta}\right) := \mathcal{L}\left(\boldsymbol{\hat{Y}_b}, \boldsymbol{Y_b}; \boldsymbol{\Theta}\right)$

$\boldsymbol{\Theta} := \boldsymbol{\Theta} - \lambda \sum_{i=1}^{B} \nabla \mathcal{L}_b\left(\boldsymbol{\Theta}\right)$

# Mini-Batch Stochastic Gradient Descent (mb-SGD)

**Algorithm 4:** Mini-Batch Stochastic Gradient descent (SGD)

**Inputs:** model $\hat{f}(\cdot;\boldsymbol{\Theta}) : \mathbb{R}^{d_1} \longrightarrow \mathbb{R}^{d_2}$;
dataset $S = \left\{(\boldsymbol{x_i}, \boldsymbol{y_i}) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \dots, N\right\}$;
differentiable loss function $\mathcal{L} : \mathbb{R}^{|\boldsymbol{\Theta}|} \longrightarrow \mathbb{R}$;
random initial parameters $\boldsymbol{\Theta}$; learning rate $\lambda > 0$; nb of epochs $nb\_epochs$;
batch size $B$.

**for** $e = 1, \dots, nb\_epochs$ **do**
  **for** $b = 1, \dots, nb\_batches$ **do**
    $\hat{\boldsymbol{Y}}_{\boldsymbol{b}} = \hat{f}\left(\boldsymbol{X}_{\boldsymbol{b}};\boldsymbol{\Theta}\right)$ ; `// compute prediction (batch)`
    $\mathcal{L}_b\left(\boldsymbol{\Theta}\right) := \mathcal{L}\left(\hat{\boldsymbol{Y}}_{\boldsymbol{b}}, \boldsymbol{Y}_{\boldsymbol{b}};\boldsymbol{\Theta}\right)$; `// compute loss (batch)`
  **end**
  $\boldsymbol{\Theta} := \boldsymbol{\Theta} - \lambda \sum_{i=1}^{B} \nabla \mathcal{L}_b\left(\boldsymbol{\Theta}\right)$ `// update gradient`
**end**
**return** $\hat{f}(\cdot;\boldsymbol{\Theta})$

# Bibliographie


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

Wikipedia contributors (2022a).
Gradient — Wikipedia, the free encyclopedia.

Wikipedia contributors (2022b).
Gradient descent — Wikipedia, the free encyclopedia.