

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:

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

- 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:

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

- 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:

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

- 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* .

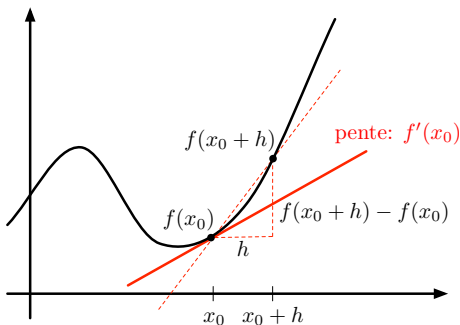
- La fonction

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

est la *dérivée de f* .

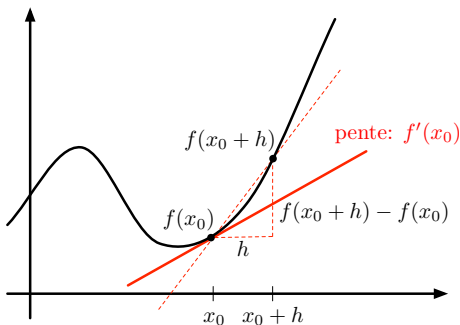
DÉRIVÉE

- ▶ $f'(x_0)$ est la pente de la tangente 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 tangente 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 $f : \mathbb{R}^N \rightarrow \mathbb{R}$ une fonction différentiable (définition un peu différente). Le *gradient de f au point $\mathbf{x} = (x_1, \dots, x_N)$* est le vecteur défini par

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

- La fonction

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

est le *gradient* de f .

GRADIENT

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

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

- La fonction

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

est le *gradient* de f .

GRADIENT

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

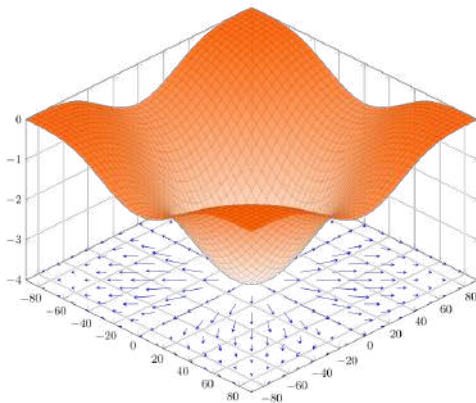


Figure taken from [Wikipedia contributors, 2022a]

GRADIENT

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

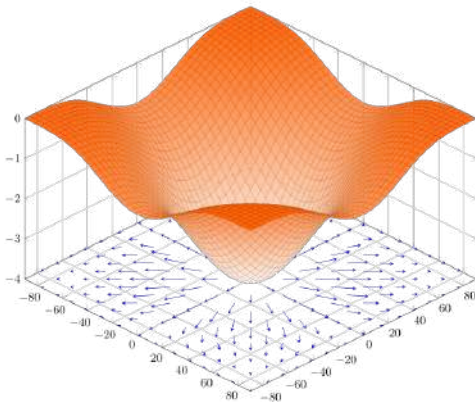


Figure taken from [Wikipedia contributors, 2022a]

GRADIENT

- ▶ $\nabla f(x)$ représente la direction de la plus grande pente ascendante de f au point 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

- ▶ $\nabla f(x)$ représente la direction de la plus grande pente ascendante de f au point 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

- ▶ $\nabla f(x)$ représente la direction de la plus grande pente ascendante de f au point 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

- ▶ $\nabla f(x)$ représente la direction de la plus grande pente ascendante de f au point 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

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

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

- Intuitivement, $\nabla_{\mathbf{v}} f(\mathbf{x})$ représente le taux d'accroissement de f en \mathbf{x} dans la direction \mathbf{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 $f : \mathbb{R}^N \rightarrow \mathbb{R}$ une fonction différentiable. Pour tout $x \in \mathbb{R}^N$, on a:

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

GRADIENT

Preuve: On peut montrer que

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

où $\theta := \theta(\nabla f(x), v)$.

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

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



GRADIENT

Preuve: On peut montrer que

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

où $\theta := \theta(\nabla f(x), v)$.

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

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



GRADIENT

Preuve: On peut montrer que

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

où $\theta := \theta(\nabla f(x), v)$.

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

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



GRADIENT

Preuve: On peut montrer que

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

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

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

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



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 $-\nabla f(\mathbf{x})$.
- ▶ 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 $-\nabla f(\mathbf{x})$.
- ▶ **Remarque:** on ne descend pas vraiment le long de la surface, on bouge dans le sol...

DESCENTE DE GRADIENT

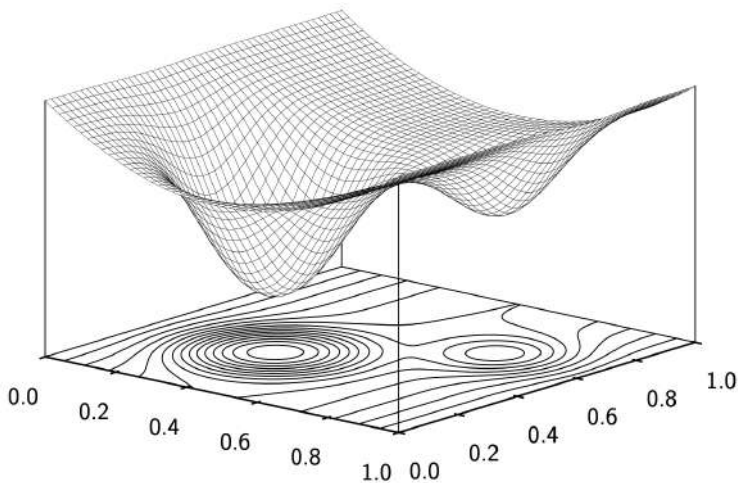


Figure taken from [Fleuret, 2022]

DESCENTE DE GRADIENT

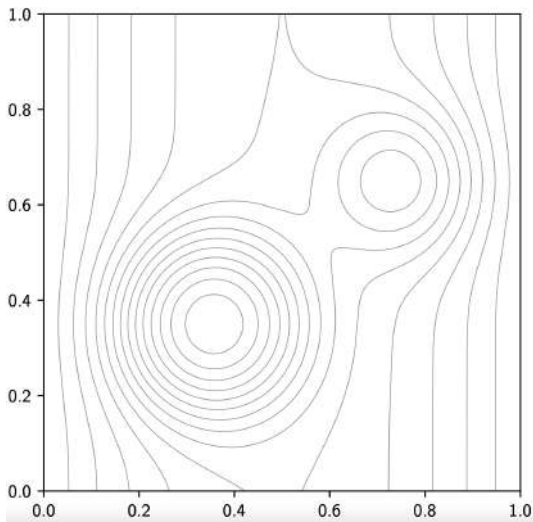


Figure taken from [Fleuret, 2022]

DESCENTE DE GRADIENT

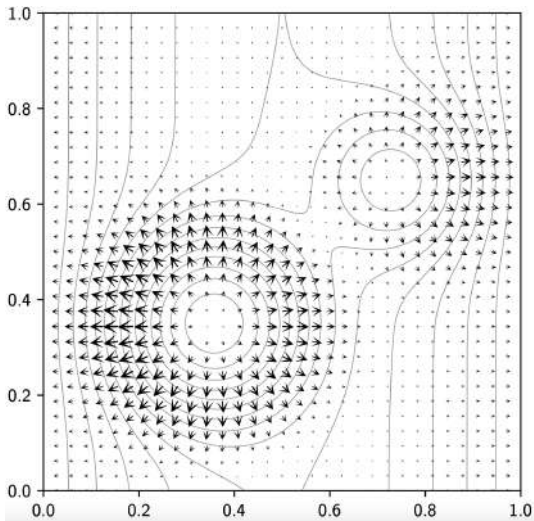


Figure taken from [Fleuret, 2022]

DESCENTE DE GRADIENT

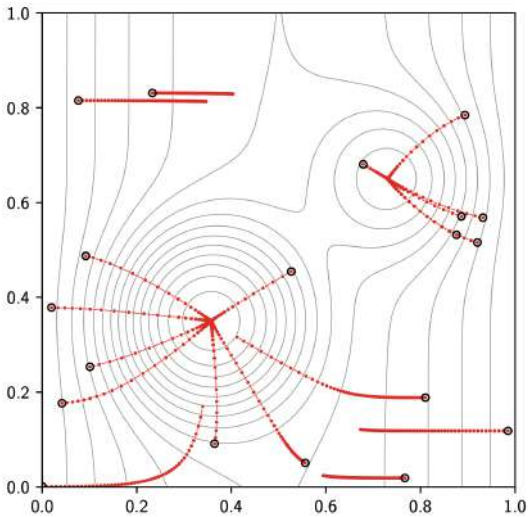


Figure taken from [Fleuret, 2022]

DESCENTE DE GRADIENT

Algorithm 1: Gradient descent

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

```
while  $\|\nabla f(x)\| > \epsilon$  do  
   $x := x - \lambda \nabla f(x)$   
end  
return  $x$ 
```

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

DESCENTE DE GRADIENT

Algorithm 1: Gradient descent

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

```
while  $\|\nabla f(x)\| > \epsilon$  do  
   $x := x - \lambda \nabla f(x)$   
end  
return  $x$ 
```

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

DESCENTE DE GRADIENT

Algorithm 1: Gradient descent

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

```
while  $\|\nabla f(x)\| > \epsilon$  do  
   $x := x - \lambda \nabla f(x)$   
end  
return  $x$ 
```

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

DESCENTE DE GRADIENT

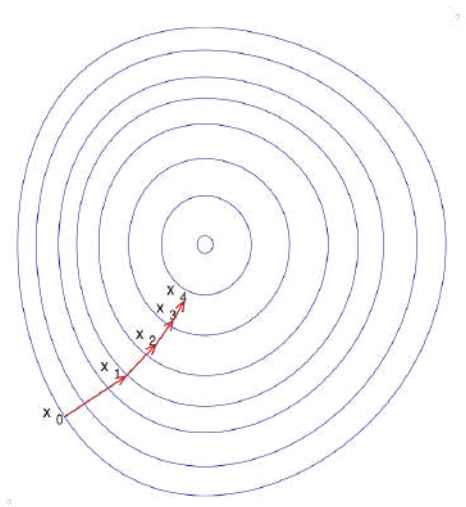


Figure taken from [Wikipedia contributors, 2022b]

LEARNING PROBLEM

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

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

LEARNING PROBLEM

- ▶ Soit $S = \{(\mathbf{x}_i, \mathbf{y}_i) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \dots, N\}$ un dataset.
- ▶ Soit $\hat{f}(\cdot; \Theta)$ un modèle qui dépend des paramètres Θ :

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

LEARNING PROBLEM

- 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 :

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

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

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

LEARNING PROBLEM

- 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 :

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

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

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

LEARNING PROBLEM

- Pour différents paramètres Θ , on aura différentes prédictions $\hat{y}_i = \hat{f}(x_i; \Theta)$, et donc différentes erreurs $\ell(\dots)$ et $\mathcal{L}(\dots)$.
- Ainsi, ℓ et \mathcal{L} sont aussi des fonctions des paramètres Θ :

$$\ell : \mathbb{R}^{|\Theta|} \longrightarrow \mathbb{R}$$

$$\Theta \longmapsto \ell(\hat{y}_i, y_i; \Theta)$$

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

$$\Theta \longmapsto \mathcal{L}(\hat{y}_1, \dots, \hat{y}_N, y_1, \dots, y_N; \Theta)$$

où $|\Theta|$ est le nombre de paramètres Θ .

LEARNING PROBLEM

- Pour différents paramètres Θ , on aura différentes prédictions $\hat{y}_i = \hat{f}(x_i; \Theta)$, et donc différentes erreurs $\ell(\dots)$ et $\mathcal{L}(\dots)$.
- Ainsi, ℓ et \mathcal{L} sont aussi des fonctions des paramètres Θ :

$$\ell : \mathbb{R}^{|\Theta|} \longrightarrow \mathbb{R}$$

$$\Theta \longmapsto \ell(\hat{y}_i, y_i; \Theta)$$

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

$$\Theta \longmapsto \mathcal{L}(\hat{y}_1, \dots, \hat{y}_N, y_1, \dots, y_N; \Theta)$$

où $|\Theta|$ est le nombre de paramètres Θ .

LEARNING PROBLEM

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

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

- ▶ 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 $\hat{f}(\dots; \Theta)$ consiste à déterminer des paramètres Θ qui minimisent la fonction de coût

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

- ▶ 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 $\hat{f}(\dots; \Theta)$ consiste à déterminer des paramètres Θ qui minimisent la fonction de coût

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

- ▶ 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 $\hat{f}(\dots; \Theta)$ consiste à déterminer des paramètres Θ qui minimisent la fonction de coût

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

- ▶ 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 $\hat{f}(\dots; \Theta)$ consiste à déterminer des paramètres Θ qui minimisent la fonction de coût

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

- 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

$$\ell(\dots; \Theta) \text{ or } \mathcal{L}(\dots; \Theta)$$

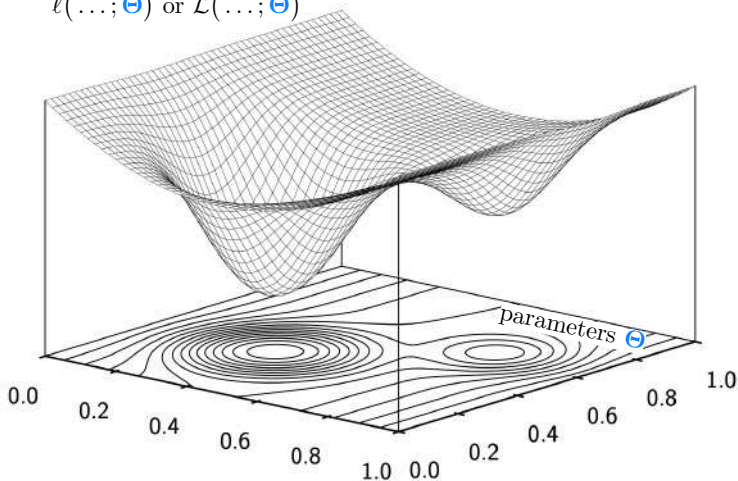


Figure adapted from [Fleuret, 2022]

DESCENTES DE GRADIENT

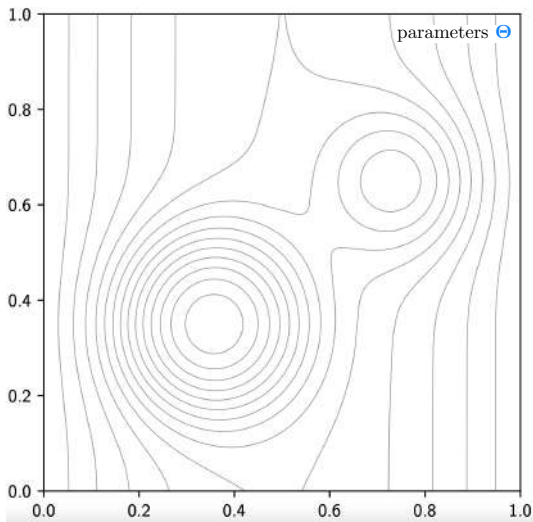


Figure adapted from [Fleuret, 2022]

DESCENTES DE GRADIENT

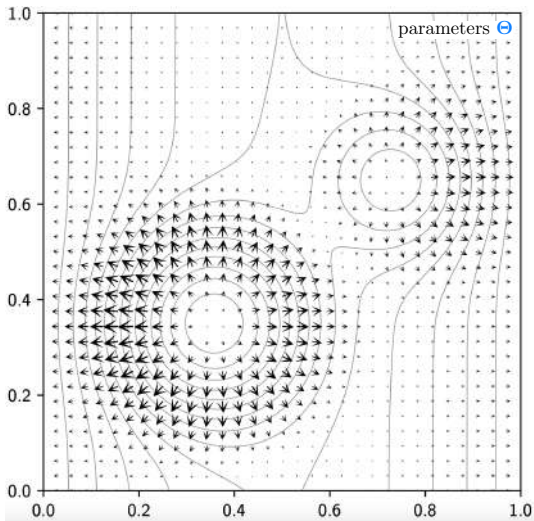


Figure adapted from [Fleuret, 2022]

DESCENTES DE GRADIENT

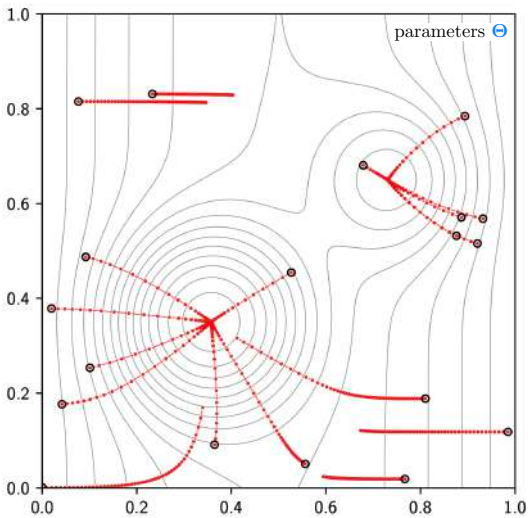


Figure adapted from [Fleuret, 2022]

GRADIENT DESCENT (GD)

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

GRADIENT DESCENT (GD)

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

GRADIENT DESCENT (GD)

Algorithm 2: Gradient descent (GD)

Inputs: model $\hat{f}(\cdot; \Theta) : \mathbb{R}^{d_1} \rightarrow \mathbb{R}^{d_2}$;
dataset $S = \{(\mathbf{x}_i, \mathbf{y}_i) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \dots, N\}$;
differentiable loss function $\mathcal{L} : |\Theta| \rightarrow \mathbb{R}$;
random initial parameters Θ ; learning rate $\lambda > 0$; nb of epochs nb_epochs .

```
for  $e = 1, \dots, nb\_epochs$  do
  for  $i = 1, \dots, N$  do                // compute predictions (dataset)
     $\hat{\mathbf{y}}_i = \hat{f}(\mathbf{x}_i; \Theta)$ 
  end
   $\mathcal{L}(\Theta) := \mathcal{L}(\hat{\mathbf{y}}_1, \dots, \hat{\mathbf{y}}_N, \mathbf{y}_1, \dots, \mathbf{y}_N; \Theta)$  // compute loss (dataset)
   $\Theta := \Theta - \lambda \nabla \mathcal{L}(\Theta)$  // update gradient (dataset)
end
return  $\hat{f}(\cdot; \Theta)$ 
```

GRADIENT DESCENT (GD)

Algorithm 2: Gradient descent (GD)

Inputs: model $\hat{f}(\cdot; \Theta) : \mathbb{R}^{d_1} \rightarrow \mathbb{R}^{d_2}$;
dataset $S = \{(\mathbf{x}_i, \mathbf{y}_i) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \dots, N\}$;
differentiable loss function $\mathcal{L} : |\Theta| \rightarrow \mathbb{R}$;
random initial parameters Θ ; learning rate $\lambda > 0$; nb of epochs nb_epochs .

```
for  $e = 1, \dots, nb\_epochs$  do  
  for  $i = 1, \dots, N$  do                                // compute predictions (dataset)  
     $\hat{\mathbf{y}}_i = \hat{f}(\mathbf{x}_i; \Theta)$   
  end  
   $\mathcal{L}(\Theta) := \mathcal{L}(\hat{\mathbf{y}}_1, \dots, \hat{\mathbf{y}}_N, \mathbf{y}_1, \dots, \mathbf{y}_N; \Theta)$  // compute loss (dataset)  
   $\Theta := \Theta - \lambda \nabla \mathcal{L}(\Theta)$                         // update gradient (dataset)  
end  
return  $\hat{f}(\cdot; \Theta)$ 
```

GRADIENT DESCENT (GD)

Algorithm 2: Gradient descent (GD)

Inputs: model $\hat{f}(\cdot; \Theta) : \mathbb{R}^{d_1} \rightarrow \mathbb{R}^{d_2}$;
dataset $S = \{(\mathbf{x}_i, \mathbf{y}_i) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \dots, N\}$;
differentiable loss function $\mathcal{L} : |\Theta| \rightarrow \mathbb{R}$;
random initial parameters Θ ; learning rate $\lambda > 0$; nb of epochs nb_epochs .

```
for  $e = 1, \dots, nb\_epochs$  do
    for  $i = 1, \dots, N$  do                // compute predictions (dataset)
         $\hat{\mathbf{y}}_i = \hat{f}(\mathbf{x}_i; \Theta)$ 
    end
     $\mathcal{L}(\Theta) := \mathcal{L}(\hat{\mathbf{y}}_1, \dots, \hat{\mathbf{y}}_N, \mathbf{y}_1, \dots, \mathbf{y}_N; \Theta)$  // compute loss (dataset)
     $\Theta := \Theta - \lambda \nabla \mathcal{L}(\Theta)$  // update gradient (dataset)
end
return  $\hat{f}(\cdot; \Theta)$ 
```

GRADIENT DESCENT (GD)

Algorithm 2: Gradient descent (GD)

Inputs: model $\hat{f}(\cdot; \Theta) : \mathbb{R}^{d_1} \rightarrow \mathbb{R}^{d_2}$;
dataset $S = \{(\mathbf{x}_i, \mathbf{y}_i) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \dots, N\}$;
differentiable loss function $\mathcal{L} : |\Theta| \rightarrow \mathbb{R}$;
random initial parameters Θ ; learning rate $\lambda > 0$; nb of epochs nb_epochs .

```
for  $e = 1, \dots, nb\_epochs$  do
    for  $i = 1, \dots, N$  do                // compute predictions (dataset)
         $\hat{\mathbf{y}}_i = \hat{f}(\mathbf{x}_i; \Theta)$ 
    end
     $\mathcal{L}(\Theta) := \mathcal{L}(\hat{\mathbf{y}}_1, \dots, \hat{\mathbf{y}}_N, \mathbf{y}_1, \dots, \mathbf{y}_N; \Theta)$  // compute loss (dataset)
     $\Theta := \Theta - \lambda \nabla \mathcal{L}(\Theta)$  // update gradient (dataset)
end
return  $\hat{f}(\cdot; \Theta)$ 
```

GRADIENT DESCENT (GD)

Algorithm 2: Gradient descent (GD)

Inputs: model $\hat{f}(\cdot; \Theta) : \mathbb{R}^{d_1} \rightarrow \mathbb{R}^{d_2}$;
dataset $S = \{(\mathbf{x}_i, \mathbf{y}_i) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \dots, N\}$;
differentiable loss function $\mathcal{L} : |\Theta| \rightarrow \mathbb{R}$;
random initial parameters Θ ; learning rate $\lambda > 0$; nb of epochs nb_epochs .

```
for  $e = 1, \dots, nb\_epochs$  do
    for  $i = 1, \dots, N$  do                // compute predictions (dataset)
         $\hat{\mathbf{y}}_i = \hat{f}(\mathbf{x}_i; \Theta)$ 
    end
     $\mathcal{L}(\Theta) := \mathcal{L}(\hat{\mathbf{y}}_1, \dots, \hat{\mathbf{y}}_N, \mathbf{y}_1, \dots, \mathbf{y}_N; \Theta)$     // compute loss (dataset)
     $\Theta := \Theta - \lambda \nabla \mathcal{L}(\Theta)$     // update gradient (dataset)
end
return  $\hat{f}(\cdot; \Theta)$ 
```

GRADIENT DESCENT (GD)

Algorithm 2: Gradient descent (GD)

Inputs: model $\hat{f}(\cdot; \Theta) : \mathbb{R}^{d_1} \rightarrow \mathbb{R}^{d_2}$;
dataset $S = \{(\mathbf{x}_i, \mathbf{y}_i) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \dots, N\}$;
differentiable loss function $\mathcal{L} : |\Theta| \rightarrow \mathbb{R}$;
random initial parameters Θ ; learning rate $\lambda > 0$; nb of epochs nb_epochs .

```
for  $e = 1, \dots, nb\_epochs$  do
    for  $i = 1, \dots, N$  do                // compute predictions (dataset)
         $\hat{\mathbf{y}}_i = \hat{f}(\mathbf{x}_i; \Theta)$ 
    end
     $\mathcal{L}(\Theta) := \mathcal{L}(\hat{\mathbf{y}}_1, \dots, \hat{\mathbf{y}}_N, \mathbf{y}_1, \dots, \mathbf{y}_N; \Theta)$     // compute loss (dataset)
     $\Theta := \Theta - \lambda \nabla \mathcal{L}(\Theta)$     // update gradient (dataset)
end
return  $\hat{f}(\cdot; \Theta)$ 
```

GRADIENT DESCENT (GD)

Algorithm 2: Gradient descent (GD)

Inputs: model $\hat{f}(\cdot; \Theta) : \mathbb{R}^{d_1} \rightarrow \mathbb{R}^{d_2}$;
dataset $S = \{(\mathbf{x}_i, \mathbf{y}_i) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \dots, N\}$;
differentiable loss function $\mathcal{L} : |\Theta| \rightarrow \mathbb{R}$;
random initial parameters Θ ; learning rate $\lambda > 0$; nb of epochs nb_epochs .

```
for  $e = 1, \dots, nb\_epochs$  do
    for  $i = 1, \dots, N$  do                // compute predictions (dataset)
         $\hat{\mathbf{y}}_i = \hat{f}(\mathbf{x}_i; \Theta)$ 
    end
     $\mathcal{L}(\Theta) := \mathcal{L}(\hat{\mathbf{y}}_1, \dots, \hat{\mathbf{y}}_N, \mathbf{y}_1, \dots, \mathbf{y}_N; \Theta)$  // compute loss (dataset)
     $\Theta := \Theta - \lambda \nabla \mathcal{L}(\Theta)$  // update gradient (dataset)
end
return  $\hat{f}(\cdot; \Theta)$ 
```

STOCHASTIC GRADIENT DESCENT (SGD)

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

STOCHASTIC GRADIENT DESCENT (SGD)

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

STOCHASTIC GRADIENT DESCENT (SGD)

Algorithm 3: Stochastic Gradient descent (SGD)

Inputs: model $\hat{f}(\cdot; \Theta) : \mathbb{R}^{d_1} \rightarrow \mathbb{R}^{d_2}$;
dataset $S = \{(\mathbf{x}_i, \mathbf{y}_i) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \dots, N\}$;
differentiable loss function $\ell : |\Theta| \rightarrow \mathbb{R}$;
random initial parameters Θ ; learning rate $\lambda > 0$; nb of epochs nb_epochs .

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

STOCHASTIC GRADIENT DESCENT (SGD)

Algorithm 3: Stochastic Gradient descent (SGD)

Inputs: model $\hat{f}(\cdot; \Theta) : \mathbb{R}^{d_1} \rightarrow \mathbb{R}^{d_2}$;
dataset $S = \{(\mathbf{x}_i, \mathbf{y}_i) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \dots, N\}$;
differentiable loss function $\ell : |\Theta| \rightarrow \mathbb{R}$;
random initial parameters Θ ; learning rate $\lambda > 0$; nb of epochs nb_epochs .

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

STOCHASTIC GRADIENT DESCENT (SGD)

Algorithm 3: Stochastic Gradient descent (SGD)

Inputs: model $\hat{f}(\cdot; \Theta) : \mathbb{R}^{d_1} \rightarrow \mathbb{R}^{d_2}$;
dataset $S = \{(\mathbf{x}_i, \mathbf{y}_i) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \dots, N\}$;
differentiable loss function $\ell : |\Theta| \rightarrow \mathbb{R}$;
random initial parameters Θ ; learning rate $\lambda > 0$; nb of epochs nb_epochs .

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

STOCHASTIC GRADIENT DESCENT (SGD)

Algorithm 3: Stochastic Gradient descent (SGD)

Inputs: model $\hat{f}(\cdot; \Theta) : \mathbb{R}^{d_1} \rightarrow \mathbb{R}^{d_2}$;
dataset $S = \{(\mathbf{x}_i, \mathbf{y}_i) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \dots, N\}$;
differentiable loss function $\ell : |\Theta| \rightarrow \mathbb{R}$;
random initial parameters Θ ; learning rate $\lambda > 0$; nb of epochs nb_epochs .

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

STOCHASTIC GRADIENT DESCENT (SGD)

Algorithm 3: Stochastic Gradient descent (SGD)

Inputs: model $\hat{f}(\cdot; \Theta) : \mathbb{R}^{d_1} \rightarrow \mathbb{R}^{d_2}$;
dataset $S = \{(\mathbf{x}_i, \mathbf{y}_i) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \dots, N\}$;
differentiable loss function $\ell : |\Theta| \rightarrow \mathbb{R}$;
random initial parameters Θ ; learning rate $\lambda > 0$; nb of epochs nb_epochs .

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

STOCHASTIC GRADIENT DESCENT (SGD)

Algorithm 3: Stochastic Gradient descent (SGD)

Inputs: model $\hat{f}(\cdot; \Theta) : \mathbb{R}^{d_1} \rightarrow \mathbb{R}^{d_2}$;
dataset $S = \{(\mathbf{x}_i, \mathbf{y}_i) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \dots, N\}$;
differentiable loss function $\ell : |\Theta| \rightarrow \mathbb{R}$;
random initial parameters Θ ; learning rate $\lambda > 0$; nb of epochs nb_epochs .

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

MINI-BATCH STOCHASTIC GRADIENT DESCENT (MB-SGD)

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

MINI-BATCH STOCHASTIC GRADIENT DESCENT (MB-SGD)

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

Inputs: model $\hat{f}(\cdot; \Theta) : \mathbb{R}^{d_1} \rightarrow \mathbb{R}^{d_2}$;
dataset $S = \{(\mathbf{x}_i, \mathbf{y}_i) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \dots, N\}$;
differentiable loss function $\mathcal{L} : |\Theta| \rightarrow \mathbb{R}$;
random initial parameters Θ ; 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
    for  $i = 1, \dots, B$  do // compute predictions and losses (batch)
       $\hat{\mathbf{y}}_{b_i} = \hat{f}(\mathbf{x}_{b_i}; \Theta)$ 
       $\ell_{b_i}(\Theta) := \ell(\hat{\mathbf{y}}_{b_i}, \mathbf{y}_{b_i}; \Theta)$ 
    end
     $\Theta := \Theta - \lambda \sum_{i=1}^B \nabla \ell_{b_i}(\Theta)$  // update gradient (batch)
  end
end
return  $\hat{f}(\cdot; \Theta)$ 
```

MINI-BATCH STOCHASTIC GRADIENT DESCENT (MB-SGD)

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

Inputs: model $\hat{f}(\cdot; \Theta) : \mathbb{R}^{d_1} \rightarrow \mathbb{R}^{d_2}$;
dataset $S = \{(\mathbf{x}_i, \mathbf{y}_i) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \dots, N\}$;
differentiable loss function $\mathcal{L} : |\Theta| \rightarrow \mathbb{R}$;
random initial parameters Θ ; 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
    for  $i = 1, \dots, B$  do // compute predictions and losses (batch)
       $\hat{\mathbf{y}}_{b_i} = \hat{f}(\mathbf{x}_{b_i}; \Theta)$ 
       $\ell_{b_i}(\Theta) := \ell(\hat{\mathbf{y}}_{b_i}, \mathbf{y}_{b_i}; \Theta)$ 
    end
     $\Theta := \Theta - \lambda \sum_{i=1}^B \nabla \ell_{b_i}(\Theta)$  // update gradient (batch)
  end
end
return  $\hat{f}(\cdot; \Theta)$ 
```

MINI-BATCH STOCHASTIC GRADIENT DESCENT (MB-SGD)

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

Inputs: model $\hat{f}(\cdot; \Theta) : \mathbb{R}^{d_1} \rightarrow \mathbb{R}^{d_2}$;
dataset $S = \{(\mathbf{x}_i, \mathbf{y}_i) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \dots, N\}$;
differentiable loss function $\mathcal{L} : |\Theta| \rightarrow \mathbb{R}$;
random initial parameters Θ ; 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
    for  $i = 1, \dots, B$  do // compute predictions and losses (batch)
       $\hat{\mathbf{y}}_{b_i} = \hat{f}(\mathbf{x}_{b_i}; \Theta)$ 
       $\ell_{b_i}(\Theta) := \ell(\hat{\mathbf{y}}_{b_i}, \mathbf{y}_{b_i}; \Theta)$ 
    end
     $\Theta := \Theta - \lambda \sum_{i=1}^B \nabla \ell_{b_i}(\Theta)$  // update gradient (batch)
  end
end
return  $\hat{f}(\cdot; \Theta)$ 
```

MINI-BATCH STOCHASTIC GRADIENT DESCENT (MB-SGD)

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

Inputs: model $\hat{f}(\cdot; \Theta) : \mathbb{R}^{d_1} \rightarrow \mathbb{R}^{d_2}$;
dataset $S = \{(\mathbf{x}_i, \mathbf{y}_i) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \dots, N\}$;
differentiable loss function $\mathcal{L} : |\Theta| \rightarrow \mathbb{R}$;
random initial parameters Θ ; 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
    for  $i = 1, \dots, B$  do // compute predictions and losses (batch)
       $\hat{\mathbf{y}}_{b_i} = \hat{f}(\mathbf{x}_{b_i}; \Theta)$ 
       $\ell_{b_i}(\Theta) := \ell(\hat{\mathbf{y}}_{b_i}, \mathbf{y}_{b_i}; \Theta)$ 
    end
     $\Theta := \Theta - \lambda \sum_{i=1}^B \nabla \ell_{b_i}(\Theta)$  // update gradient (batch)
  end
end
return  $\hat{f}(\cdot; \Theta)$ 
```

MINI-BATCH STOCHASTIC GRADIENT DESCENT (MB-SGD)

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

Inputs: model $\hat{f}(\cdot; \Theta) : \mathbb{R}^{d_1} \rightarrow \mathbb{R}^{d_2}$;
dataset $S = \{(\mathbf{x}_i, \mathbf{y}_i) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \dots, N\}$;
differentiable loss function $\mathcal{L} : |\Theta| \rightarrow \mathbb{R}$;
random initial parameters Θ ; 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
    for  $i = 1, \dots, B$  do // compute predictions and losses (batch)
       $\hat{\mathbf{y}}_{b_i} = \hat{f}(\mathbf{x}_{b_i}; \Theta)$ 
       $\ell_{b_i}(\Theta) := \ell(\hat{\mathbf{y}}_{b_i}, \mathbf{y}_{b_i}; \Theta)$ 
    end
     $\Theta := \Theta - \lambda \sum_{i=1}^B \nabla \ell_{b_i}(\Theta)$  // update gradient (batch)
  end
end
return  $\hat{f}(\cdot; \Theta)$ 
```

MINI-BATCH STOCHASTIC GRADIENT DESCENT (MB-SGD)

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

Inputs: model $\hat{f}(\cdot; \Theta) : \mathbb{R}^{d_1} \rightarrow \mathbb{R}^{d_2}$;
dataset $S = \{(\mathbf{x}_i, \mathbf{y}_i) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \dots, N\}$;
differentiable loss function $\mathcal{L} : |\Theta| \rightarrow \mathbb{R}$;
random initial parameters Θ ; 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
    for  $i = 1, \dots, B$  do // compute predictions and losses (batch)
       $\hat{\mathbf{y}}_{b_i} = \hat{f}(\mathbf{x}_{b_i}; \Theta)$ 
       $\ell_{b_i}(\Theta) := \ell(\hat{\mathbf{y}}_{b_i}, \mathbf{y}_{b_i}; \Theta)$ 
    end
     $\Theta := \Theta - \lambda \sum_{i=1}^B \nabla \ell_{b_i}(\Theta)$  // update gradient (batch)
  end
end
return  $\hat{f}(\cdot; \Theta)$ 
```

MINI-BATCH STOCHASTIC GRADIENT DESCENT (MB-SGD)

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

Inputs: model $\hat{f}(\cdot; \Theta) : \mathbb{R}^{d_1} \rightarrow \mathbb{R}^{d_2}$;
dataset $S = \{(\mathbf{x}_i, \mathbf{y}_i) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \dots, N\}$;
differentiable loss function $\mathcal{L} : |\Theta| \rightarrow \mathbb{R}$;
random initial parameters Θ ; 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
    for  $i = 1, \dots, B$  do // compute predictions and losses (batch)
       $\hat{\mathbf{y}}_{b_i} = \hat{f}(\mathbf{x}_{b_i}; \Theta)$ 
       $\ell_{b_i}(\Theta) := \ell(\hat{\mathbf{y}}_{b_i}, \mathbf{y}_{b_i}; \Theta)$ 
    end
     $\Theta := \Theta - \lambda \sum_{i=1}^B \nabla \ell_{b_i}(\Theta)$  // update gradient (batch)
  end
end
return  $\hat{f}(\cdot; \Theta)$ 
```

BIBLIOGRAPHIE



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



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



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