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

$$f': \mathbb{R} \longrightarrow \mathbb{R}$$
$$x \longmapsto f'(x)$$

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

$$f': \mathbb{R} \longrightarrow \mathbb{R}$$
 $x \longmapsto f'(x)$ 

est la *dérivée* de f.



•0000000

# 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

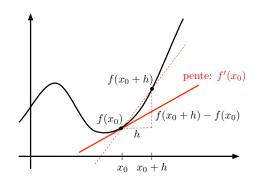
$$f': \mathbb{R} \longrightarrow \mathbb{R}$$
  
 $x \longmapsto f'(x)$ 

est la *dérivée* de f.



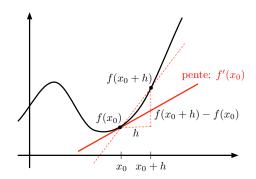
# DÉRIVÉE

- $f'(x_0)$  est la pente de la tangeante au graphe de f au point  $(x_0, f(x_0))$ .
- $ightharpoonup f'(x_0)$  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  $x_0$ .



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

$$abla f(oldsymbol{x}) := egin{pmatrix} rac{\partial f}{\partial x_1}(oldsymbol{x}) \\ dots \\ rac{\partial f}{\partial x_N}(oldsymbol{x}) \end{pmatrix}$$

La fonction

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

est le *gradient* de *f* .

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

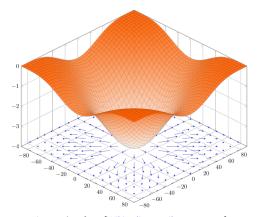
$$\nabla f(\boldsymbol{x}) := \begin{pmatrix} \frac{\partial f}{\partial x_1}(\boldsymbol{x}) \\ \vdots \\ \frac{\partial f}{\partial x_N}(\boldsymbol{x}) \end{pmatrix}$$

► La fonction

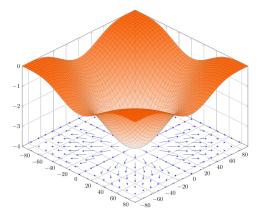
$$abla f: \mathbb{R}^N \longrightarrow \mathbb{R}^N$$
 $abla \longmapsto \nabla f(x)$ 

est le gradient de f.

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



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



- ightharpoonup 
  abla f(x) représente la direction de la plus grande pente ascendante de f au point x.
- ightharpoonup 
  abla 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.
- $ightharpoonup -\nabla f(x)$  is the direction of steepest descent at x

- ightharpoonup 
  abla 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.
- $ightharpoonup -\nabla f(x)$  is the direction of steepest descent at x

- ightharpoonup 
  abla f(x) représente la direction de la plus grande pente ascendante de f au point x.
- ightharpoonup 
  abla 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.
- $ightharpoonup -\nabla f(x)$  is the direction of steepest descent at x.

- ightharpoonup 
  abla f(x) représente la direction de la plus grande pente ascendante de f au point x.
- ightharpoonup 
  abla 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.

La dérivée directionnelle de f au point  $\boldsymbol{x}=(x_1,\dots,x_N)$  selon la direction  $\boldsymbol{v}=(v_1,\dots,v_N)$  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  $\boldsymbol{v}$ .

▶ 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  $x \in \mathbb{R}^N$ , on a:

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

Preuve: On peut montrer que

$$abla_{oldsymbol{v}}f(oldsymbol{x}) = 
abla f(oldsymbol{x}) \cdot rac{oldsymbol{v}}{\|oldsymbol{v}\|} = \|
abla f(oldsymbol{x})\| \cdot rac{\|oldsymbol{v}\|}{\|oldsymbol{v}\|} \cdot \cos heta = \|
abla f(oldsymbol{x})\| \cdot \cos( heta)$$

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,

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

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,

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

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,

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

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

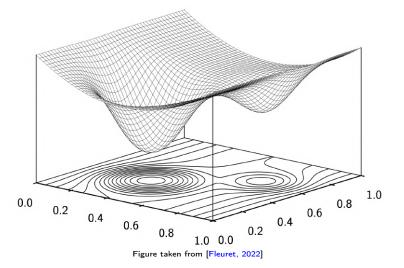
$$\nabla f(\boldsymbol{x}) \in \arg\max_{\boldsymbol{v} \in \mathbb{R}^N} \nabla_{\boldsymbol{v}} f(\boldsymbol{x}).$$

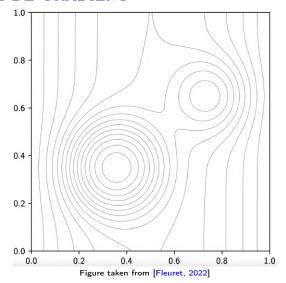
- ▶ Soit  $f: \mathbb{R}^N \longrightarrow \mathbb{R}$  une fonction différentiable.
- ightharpoonup On cherche à minimiser f: minimum local, global.
- ▶ Descente de gradient: on part d'un point au hasard; puis pas à pas, on descend le long de la surface en direction de la plus grande pente  $-\nabla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

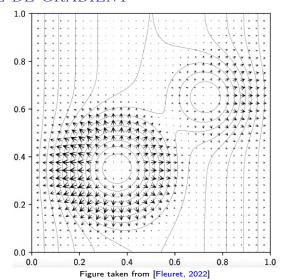
- ▶ Soit  $f : \mathbb{R}^N \longrightarrow \mathbb{R}$  une fonction différentiable.
- ightharpoonup On cherche à minimiser f: minimum local, global.
- ▶ Descente de gradient: on part d'un point au hasard; puis pas à pas, on descend le long de la surface en direction de la plus grande pente  $-\nabla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

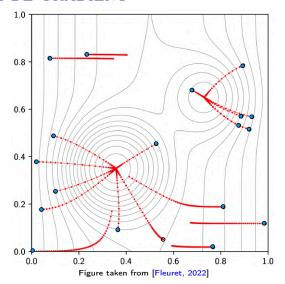
- ▶ Soit  $f: \mathbb{R}^N \longrightarrow \mathbb{R}$  une fonction différentiable.
- ightharpoonup On cherche à minimiser f: minimum local, global.
- ▶ Descente de gradient: on part d'un point au hasard; puis pas à pas, on descend le long de la surface en direction de la plus grande pente  $-\nabla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

- ▶ Soit  $f : \mathbb{R}^N \longrightarrow \mathbb{R}$  une fonction différentiable.
- ightharpoonup 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...









Movie 1

Movie 2

Movie 3

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

#### 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  $\epsilon > 0$ .

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

#### Algorithm 1: Gradient descent

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

end

return x

#### Algorithm 1: Gradient descent

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

return x

ightharpoonup 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.

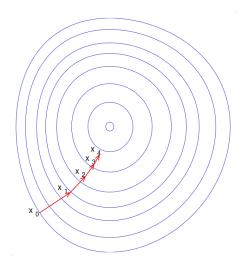


Figure taken from [Wikipedia contributors, 2022b]

### DESCENTE DE GRADIENT EN ML

- ► En machine learning, on utilise maintenant des descentes de gradient (GD) pour entraîner les modèles.
- ▶ Dans ce contexte, la descente de gradient sert à minimiser la fonction de coût (loss or cost function).

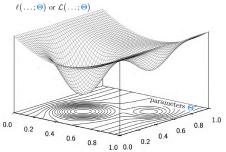


Figure adapted from [Fleuret, 2022]

# DESCENTE DE GRADIENT EN ML

- ► En machine learning, on utilise maintenant des descentes de gradient (GD) pour entraîner les modèles.
- ▶ Dans ce contexte, la descente de gradient sert à minimiser la fonction de coût (loss or cost function).

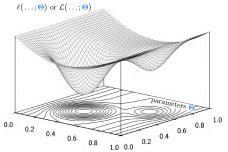


Figure adapted from [Fleuret, 2022]

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

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

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

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

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

$$egin{array}{lll} \ell: \mathbb{R}^{d_2} imes \mathbb{R}^{d_2} & \longrightarrow & \mathbb{R} \ (\hat{m{y}}_{m{i}}, m{y}_{m{i}}) & \longmapsto & \ell\left(\hat{m{y}}_{m{i}}, m{y}_{m{i}}
ight) \end{array}$$

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

$$egin{array}{lll} \mathcal{L}: \mathbb{R}^{d_2} imes \cdots & \mathbb{R}^{d_2} & \longrightarrow & \mathbb{R} \ (\hat{m{y}}_1, \ldots, \hat{m{y}}_N, m{y}_1 \ldots, m{y}_N) & \longmapsto & \mathcal{L}\left(\hat{m{y}}_1, \ldots, \hat{m{y}}_N, m{y}_1 \ldots, m{y}_N
ight) \end{array}$$

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

$$egin{array}{lll} \ell: \mathbb{R}^{d_2} imes \mathbb{R}^{d_2} & \longrightarrow & \mathbb{R} \ (\hat{m{y}}_{m{i}}, m{y}_{m{i}}) & \longmapsto & \ell\left(\hat{m{y}}_{m{i}}, m{y}_{m{i}}
ight) \end{array}$$

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

$$egin{array}{lll} \mathcal{L}: \mathbb{R}^{d_2} imes \cdots & \mathbb{R}^{d_2} & \longrightarrow & \mathbb{R} \ (\hat{m{y_1}}, \ldots, \hat{m{y_N}}, m{y_1}, \ldots, m{y_N}) & \longmapsto & \mathcal{L}\left(\hat{m{y_1}}, \ldots, \hat{m{y_N}}, m{y_1}, \ldots, m{y_N}
ight) \end{array}$$

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

$$egin{array}{lll} \ell: \mathbb{R}^{|\Theta|} & \longrightarrow & \mathbb{R} \ & oldsymbol{\Theta} & \longmapsto & \ellig(\hat{y}_i, y_i; oldsymbol{\Theta}ig) \ \mathcal{L}: \mathbb{R}^{|\Theta|} & \longrightarrow & \mathbb{R} \ & oldsymbol{\Theta} & \longmapsto & \mathcal{L}ig(\hat{y}_1, \dots, \hat{y}_N, y_1, \dots, y_N; oldsymbol{\Theta}ig) \end{array}$$

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

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

$$egin{array}{lll} \ell: \mathbb{R}^{|oldsymbol{\Theta}|} & \longrightarrow & \mathbb{R} \ oldsymbol{\Theta} & \longmapsto & \ellig(\hat{y}_i, y_i; oldsymbol{\Theta}ig) \ \mathcal{L}: \mathbb{R}^{|oldsymbol{\Theta}|} & \longrightarrow & \mathbb{R} \ oldsymbol{\Theta} & \longmapsto & \mathcal{L}(\hat{y}_1, \ldots, \hat{y}_N, y_1, \ldots, y_N; oldsymbol{\Theta}ig) \end{array}$$

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

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

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

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

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

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

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

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

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

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

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

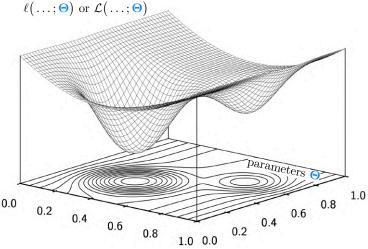
$$\ell(\ldots; \boldsymbol{\Theta})$$
 ou  $\mathcal{L}(\ldots; \boldsymbol{\Theta})$ .

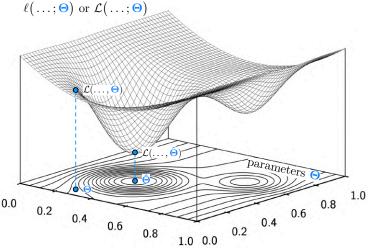
- Pour minimiser la fonction de coût, on utilise des descentes de gradient:
- gradient descent (DG)
- stochastic gradient descent (SGD)
- mini-batch stochastic gradient descent (mini batch SGD)

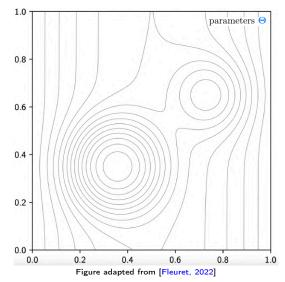
 $lackbox{L'entraînement}$  du modèle  $\hat{f}(\cdots;\Theta)$  consiste à déterminer des paramètres  $\Theta$  qui minimisent les fonctions de coût

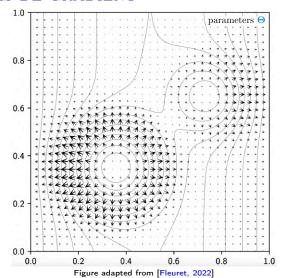
$$\ell(\ldots; \boldsymbol{\Theta})$$
 ou  $\mathcal{L}(\ldots; \boldsymbol{\Theta})$ .

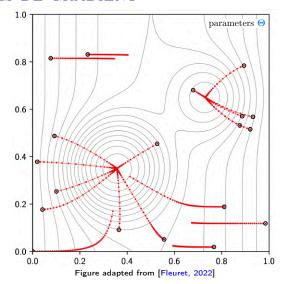
- Pour minimiser la fonction de coût, on utilise des descentes de gradient:
- gradient descent (DG)
- stochastic gradient descent (SGD)
- mini-batch stochastic gradient descent (mini batch SGD)











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

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

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

```
\begin{array}{lll} & \text{for } e=1,\ldots,E \text{ do} \\ & \text{for } i=1,\ldots,N \text{ do} & // \text{ compute predictions (dataset)} \\ & | \hat{y}_i=\hat{f}(x_i;\Theta) \\ & \text{end} \\ & \mathcal{L}(\Theta) \coloneqq \mathcal{L}(\hat{y}_1,\ldots,\hat{y}_N,y_1,\ldots,y_N;\Theta) & // \text{ compute loss (dataset)} \\ & \Theta \coloneqq \Theta - \lambda \nabla \mathcal{L}(\Theta) & // \text{ update gradient (dataset)} \\ & \text{end} \\ & \text{return } \hat{f}(\boldsymbol{\psi};\Theta) &  \end{array}
```

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

```
\begin{array}{l} \text{for } e=1,\ldots,E \text{ do} \\ & \text{for } i=1,\ldots,N \text{ do} \\ & \mid \hat{y}_i=\hat{f}\left(x_i;\Theta\right) \\ & \text{end} \\ & \mathcal{L}\left(\Theta\right):=\mathcal{L}\left(\hat{y}_1,\ldots,\hat{y}_N,y_1,\ldots,y_N;\Theta\right) \\ & \Theta:=\Theta-\lambda\nabla\mathcal{L}\left(\Theta\right) \end{array} & \text{// compute loss (dataset)} \\ & \Theta:=\Theta-\lambda\nabla\mathcal{L}\left(\Theta\right) \\ & \text{end} \end{array}
```

```
Inputs: dataset S = \{(\boldsymbol{x_i}, \boldsymbol{y_i}) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \dots, N\};
                  model \hat{f}(\cdot; \boldsymbol{\Theta}) : \mathbb{R}^{d_1} \longrightarrow \mathbb{R}^{d_2};
                  differentiable loss function \mathcal{L}: \mathbb{R}^{|\Theta|} \longrightarrow \mathbb{R}:
                  random initial parameters \Theta; learning rate \lambda > 0; nb of epochs E.
for e = 1, \ldots, E do
         for i = 1, \ldots, N do
                                                                                                // compute predictions (dataset)
        \hat{\boldsymbol{y}}_{i} = \hat{f}(\boldsymbol{x}_{i}; \boldsymbol{\Theta})
        \begin{array}{ll} \mathcal{L}\left(\Theta\right) := \mathcal{L}\left(\hat{y}_{1}, \ldots, \hat{y}_{N}, y_{1}, \ldots, y_{N}; \Theta\right) & \text{// compute loss (dataset)} \\ \Theta := \Theta - \lambda \nabla \mathcal{L}\left(\Theta\right) & \text{// update gradient (dataset)} \end{array}
```

```
Inputs: dataset S = \{(\boldsymbol{x_i}, \boldsymbol{y_i}) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \dots, N\};
                   model \hat{f}(\cdot; \boldsymbol{\Theta}) : \mathbb{R}^{d_1} \longrightarrow \mathbb{R}^{d_2};
                   differentiable loss function \mathcal{L}: \mathbb{R}^{|\Theta|} \longrightarrow \mathbb{R}:
                   random initial parameters \Theta; learning rate \lambda > 0; nb of epochs E.
for e = 1, \ldots, E do
          for i = 1, \ldots, N do
                                                                                                     // compute predictions (dataset)
         \hat{\boldsymbol{y}_i} = \hat{f}(\boldsymbol{x_i}; \boldsymbol{\Theta})
          \begin{array}{ll} \mathcal{L}\left(\Theta\right) := \mathcal{L}\left(\hat{\boldsymbol{y}}_{1}, \ldots, \hat{\boldsymbol{y}}_{N}, \boldsymbol{y}_{1}, \ldots, \boldsymbol{y}_{N}; \Theta\right) & \text{// compute loss (dataset)} \\ \Theta := \Theta - \lambda \vee \mathcal{L}\left(\Theta\right) & \text{// update gradient (dataset)} \end{array}
```

```
Inputs: dataset S = \{(\boldsymbol{x_i}, \boldsymbol{y_i}) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \dots, N\};
               model \hat{f}(\cdot; \boldsymbol{\Theta}) : \mathbb{R}^{d_1} \longrightarrow \mathbb{R}^{d_2};
               differentiable loss function \mathcal{L}: \mathbb{R}^{|\Theta|} \longrightarrow \mathbb{R}:
               random initial parameters \Theta; learning rate \lambda > 0; nb of epochs E.
for e = 1, \ldots, E do
        for i = 1, \ldots, N do
                                                                                 // compute predictions (dataset)
       \hat{\boldsymbol{y}}_{i} = \hat{f}(\boldsymbol{x}_{i}; \boldsymbol{\Theta})
       \mathcal{L}\left(\Theta
ight) := \mathcal{L}\left(\hat{y}_1, \dots, \hat{y}_N, y_1, \dots, y_N; \Theta
ight)
\Theta := \Theta - \lambda \nabla \mathcal{L}\left(\Theta
ight)
                                                                                         // compute loss (dataset)
                                                                                          // update gradient (dataset)
```

```
Inputs: dataset S = \{(\boldsymbol{x_i}, \boldsymbol{y_i}) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \dots, N\};
                  model \hat{f}(\cdot; \boldsymbol{\Theta}) : \mathbb{R}^{d_1} \longrightarrow \mathbb{R}^{d_2};
                  differentiable loss function \mathcal{L}: \mathbb{R}^{|\Theta|} \longrightarrow \mathbb{R}:
                  random initial parameters \Theta; learning rate \lambda > 0; nb of epochs E.
for e = 1, \ldots, E do
         for i=1,\ldots,N do
                                                                                                 // compute predictions (dataset)
         \hat{\boldsymbol{y}}_{i} = \hat{f}(\boldsymbol{x}_{i}; \boldsymbol{\Theta})
         \begin{aligned} \mathcal{L}\left(\boldsymbol{\Theta}\right) &:= \mathcal{L}\left(\hat{\boldsymbol{y}}_{1}, \dots, \hat{\boldsymbol{y}}_{N}, \boldsymbol{y}_{1}, \dots, \boldsymbol{y}_{N}; \boldsymbol{\Theta}\right) \\ \boldsymbol{\Theta} &:= \boldsymbol{\Theta} - \lambda \nabla \mathcal{L}\left(\boldsymbol{\Theta}\right) \end{aligned}
                                                                                                           // compute loss (dataset)
                                                                                                            // update gradient (dataset)
end
```

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

```
Inputs: dataset S = \{(\boldsymbol{x_i}, \boldsymbol{y_i}) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \dots, N\};
                  model \hat{f}(\cdot; \boldsymbol{\Theta}) : \mathbb{R}^{d_1} \longrightarrow \mathbb{R}^{d_2};
                  differentiable loss function \mathcal{L}: \mathbb{R}^{|\Theta|} \longrightarrow \mathbb{R}:
                  random initial parameters \Theta; learning rate \lambda > 0; nb of epochs E.
for e = 1, \ldots, E do
          for i=1,\ldots,N do
                                                                                                  // compute predictions (dataset)
         \hat{\boldsymbol{y}}_{i} = \hat{f}(\boldsymbol{x}_{i}; \boldsymbol{\Theta})
         \begin{aligned} \mathcal{L}\left(\boldsymbol{\Theta}\right) &:= \mathcal{L}\left(\hat{\boldsymbol{y}}_{1}, \dots, \hat{\boldsymbol{y}}_{N}, \boldsymbol{y}_{1}, \dots, \boldsymbol{y}_{N}; \boldsymbol{\Theta}\right) \\ \boldsymbol{\Theta} &:= \boldsymbol{\Theta} - \lambda \nabla \mathcal{L}\left(\boldsymbol{\Theta}\right) \end{aligned}
                                                                                                            // compute loss (dataset)
                                                                                                             // update gradient (dataset)
end
return \hat{f}(\cdot; \boldsymbol{\Theta})
```

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

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

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

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

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

```
\begin{array}{c|c} \text{for } e = 1, \dots, E \text{ do} \\ & \text{for } i = 1, \dots, N \text{ do} \\ & \hat{y}_i = \hat{f}(x_i; \Theta) \\ & \ell(\Theta) := \ell(\hat{y}_i, y_i; \Theta) \\ & \Theta := \Theta - \lambda \nabla \ell(\Theta) \end{array} \qquad \begin{array}{c} // \text{ compute prediction (sample)} \\ // \text{ compute loss (sample)} \\ // \text{ update gradient (sample)} \\ \text{end} \end{array}
```

nd

return  $\hat{f}(\cdot; \boldsymbol{\Theta})$ 

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

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

```
\begin{array}{c|c} \text{for } e = 1, \dots, E \text{ do} \\ & \text{for } i = 1, \dots, N \text{ do} \\ & \hat{y_i} = \hat{f}(x_i; \Theta) \\ & \ell(\Theta) \coloneqq \ell(\hat{y}_i, y_i; \Theta) \\ & \Theta \coloneqq \Theta - \lambda \nabla \ell(\Theta) \end{array} \qquad \begin{array}{c} \text{// compute prediction (sample)} \\ \text{// compute loss (sample)} \\ \text{// update gradient (sample)} \\ \text{end} \end{array}
```

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

```
\begin{split} & \text{Inputs: dataset } S = \left\{ (\boldsymbol{x_i}, \boldsymbol{y_i}) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \dots, N \right\}; \\ & \text{model } \hat{f}(\cdot; \boldsymbol{\Theta}) : \mathbb{R}^{d_1} \longrightarrow \mathbb{R}^{d_2}; \\ & \text{differentiable loss function } \ell : \mathbb{R}^{|\boldsymbol{\Theta}|} \longrightarrow \mathbb{R}; \\ & \text{random initial parameters } \boldsymbol{\Theta}; \text{ learning rate } \lambda > 0; \text{ nb of epochs } E. \end{split} \begin{aligned} & \text{for } e = 1, \dots, E \text{ do} \\ & \text{for } i = 1, \dots, N \text{ do} \\ & & \hat{\boldsymbol{y_i}} = \hat{f}(\boldsymbol{x_i}; \boldsymbol{\Theta}) \\ & & \ell(\boldsymbol{\Theta}) := \ell(\hat{\boldsymbol{y_i}}, \boldsymbol{y_i}; \boldsymbol{\Theta}) \\ & & \boldsymbol{\Theta} := \boldsymbol{\Theta} - \lambda \vee \ell(\boldsymbol{\Theta}) \end{aligned} \qquad // \text{ compute prediction (sample)}
```

and

return  $\hat{f}(\cdot; \boldsymbol{\Theta})$ 

return  $\hat{f}(\cdot; \boldsymbol{\Theta})$ 

- 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  $\mathcal{L}(\ldots; \Theta)$ .
- Cette méthode est la plus efficace.

- 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  $\mathcal{L}(\ldots; \Theta)$ .
- Cette méthode est la plus efficace.

- ► 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  $\mathcal{L}(\ldots; \Theta)$ .
- Cette méthode est la plus efficace.

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

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

```
\begin{aligned} & \text{Inputs: dataset } S = \left\{ (\boldsymbol{x}_i, \boldsymbol{y}_i) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \dots, N \right\}; \\ & \text{model } \hat{f}(\cdot; \Theta) : \mathbb{R}^{d_1} \longrightarrow \mathbb{R}^{d_2}; \\ & \text{differentiable loss function } \mathcal{L} : \mathbb{R}^{|\Theta|} \longrightarrow \mathbb{R}; \\ & \text{random initial parameters } \Theta; \text{ learning rate } \lambda > 0; \text{ nb of epochs } E; \\ & \text{nb of batches } B. \end{aligned} \begin{aligned} & \text{for } e = 1, \dots, E \text{ do} \\ & \hat{Y}_b = \hat{f}\left(X_b; \Theta\right); \\ & \mathcal{L}_b\left(\Theta\right) := \mathcal{L}\left(\hat{Y}_b, Y_b; \Theta\right); \end{aligned} \qquad // \text{ compute prediction (batch)} \end{aligned}
```

```
Algorithm 4: Mini-Batch Stochastic Gradient descent (SGD)
Inputs: dataset S = \{(\boldsymbol{x_i}, \boldsymbol{y_i}) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \dots, N\};
               model \hat{f}(\cdot; \boldsymbol{\Theta}) : \mathbb{R}^{d_1} \longrightarrow \mathbb{R}^{d_2};
               differentiable loss function \mathcal{L}: \mathbb{R}^{|\Theta|} \longrightarrow \mathbb{R};
               random initial parameters \Theta; learning rate \lambda > 0; nb of epochs E;
               nb of batches B.
for e = 1, \ldots, E do
        for b = 1, \ldots, B do
        \begin{split} \hat{Y}_{b} &= \hat{f}\left(X_{b}; \Theta\right); \\ \mathcal{L}_{b}\left(\Theta\right) &:= \mathcal{L}\left(\hat{Y}_{b}, Y_{b}; \Theta\right); \end{split}
```

```
Algorithm 4: Mini-Batch Stochastic Gradient descent (SGD)
Inputs: dataset S = \{(\boldsymbol{x_i}, \boldsymbol{y_i}) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \dots, N\};
                model \hat{f}(\cdot; \boldsymbol{\Theta}) : \mathbb{R}^{d_1} \longrightarrow \mathbb{R}^{d_2};
                differentiable loss function \mathcal{L}: \mathbb{R}^{|\Theta|} \longrightarrow \mathbb{R};
                random initial parameters \Theta; learning rate \lambda > 0; nb of epochs E;
                nb of batches B.
for e = 1, \ldots, E do
         for b = 1, \ldots, B do
           \hat{\boldsymbol{Y}}_{b} = \hat{\boldsymbol{f}} \left( \boldsymbol{X}_{b}; \boldsymbol{\Theta} \right); 
 \mathcal{L}_{b} \left( \boldsymbol{\Theta} \right) := \mathcal{L} \left( \hat{\boldsymbol{Y}}_{b}, \boldsymbol{Y}_{b}; \boldsymbol{\Theta} \right); 
                                                                                                // compute prediction (batch)
```

```
Algorithm 4: Mini-Batch Stochastic Gradient descent (SGD)
Inputs: dataset S = \{(\boldsymbol{x_i}, \boldsymbol{y_i}) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \dots, N\};
              model \hat{f}(\cdot; \boldsymbol{\Theta}) : \mathbb{R}^{d_1} \longrightarrow \mathbb{R}^{d_2};
              differentiable loss function \mathcal{L}: \mathbb{R}^{|\Theta|} \longrightarrow \mathbb{R}:
              random initial parameters \Theta; learning rate \lambda > 0; nb of epochs E;
              nb of batches B.
for e = 1, \ldots, E do
        for b = 1, \ldots, B do
             \hat{\mathbf{Y}}_{b} = \hat{f}(\mathbf{X}_{b}; \mathbf{\Theta});
\mathcal{L}_{b}(\mathbf{\Theta}) := \mathcal{L}(\hat{\mathbf{Y}}_{b}, \mathbf{Y}_{b}; \mathbf{\Theta});
                                                                                   // compute prediction (batch)
                                                                                                // compute loss (batch)
        end
```

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

```
Inputs: dataset S = \{(\boldsymbol{x_i}, \boldsymbol{y_i}) \in \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} : i = 1, \dots, N\};
               model \hat{f}(\cdot; \boldsymbol{\Theta}) : \mathbb{R}^{d_1} \longrightarrow \mathbb{R}^{d_2};
               differentiable loss function \mathcal{L}: \mathbb{R}^{|\Theta|} \longrightarrow \mathbb{R}:
               random initial parameters \Theta; learning rate \lambda > 0; nb of epochs E;
               nb of batches B.
for e = 1, \ldots, E do
        for b = 1, \ldots, B do
                \hat{Y}_{b} = \hat{f}(X_{b}; \Theta);
                                                                                          // compute prediction (batch)
               \mathcal{L}_{b}\left(\mathbf{\Theta}
ight):=\mathcal{L}\left(\hat{\mathbf{Y}}_{m{b}},\mathbf{Y}_{m{b}};\mathbf{\Theta}
ight);
                                                                                                        // compute loss (batch)
        end
        \mathbf{\Theta} := \mathbf{\Theta} - \lambda \sum_{i=1}^{B} \nabla \mathcal{L}_b \left( \mathbf{\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).

 $\label{eq:Gradient} \mbox{Gradient descent} \ -- \ \mbox{Wikipedia, the free encyclopedia}.$