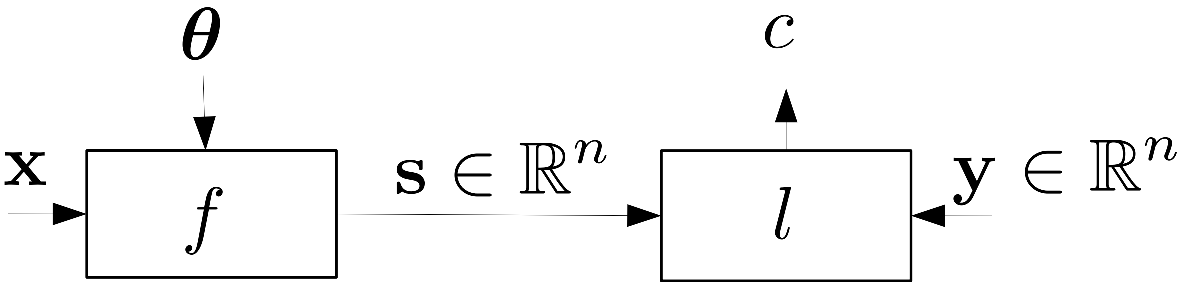
Introduction aux réseaux de neurones pour l'apprentissage supervisé (suite)

Guillaume Bourmaud

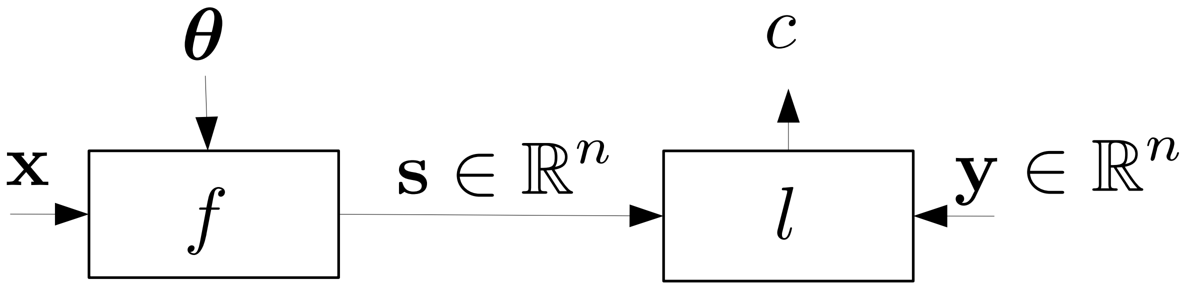
PLAN

VI) Apprentissage des paramètres d’un réseau de neurones

Choix du coût $l(\mathbf{y}, \mathbf{s})$: Régression



Choix du coût $l(\mathbf{y}, \mathbf{s})$: Régression

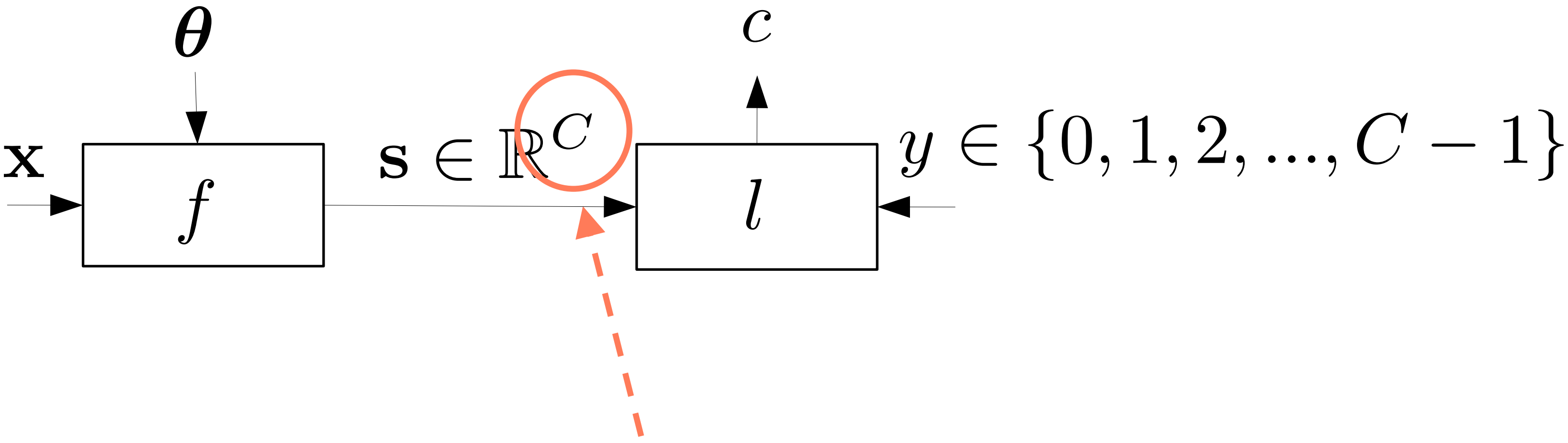


- Erreur quadratique $\| \mathbf{y} - \mathbf{s} \|_2^2 = \sum_{i=0}^{n-1} (\mathbf{y}_i - \mathbf{s}_i)^2$

- Somme des valeurs absolues $\| \mathbf{y} - \mathbf{s} \|_1 = \sum_{i=0}^{n-1} |\mathbf{y}_i - \mathbf{s}_i|$

VI)

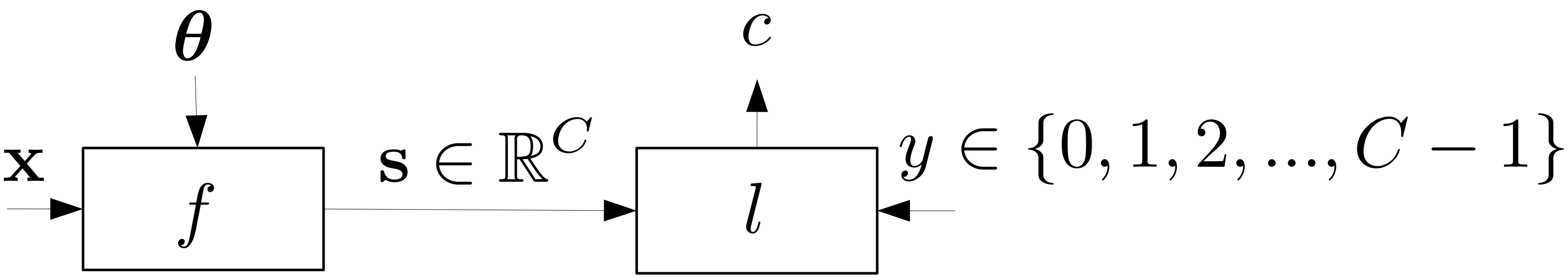
Choix du coût $l(y, \mathbf{s})$: Classification



On prédit un score par classe !

VI)

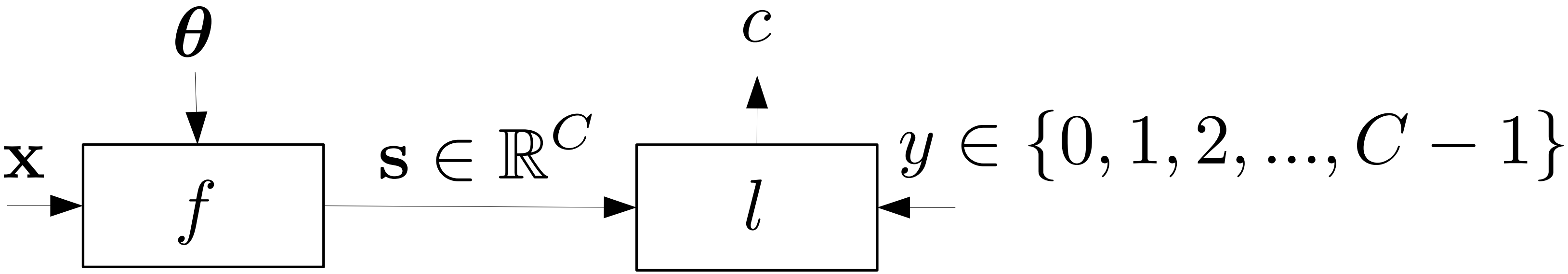
Choix du coût $l(y, \mathbf{s})$: Classification



« Cross-entropy »

VI)

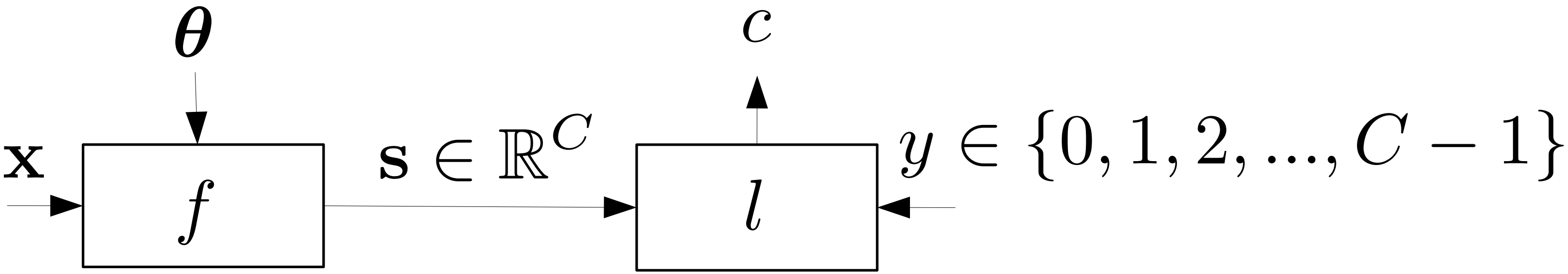
Choix du coût $l(y, \mathbf{s})$: Classification



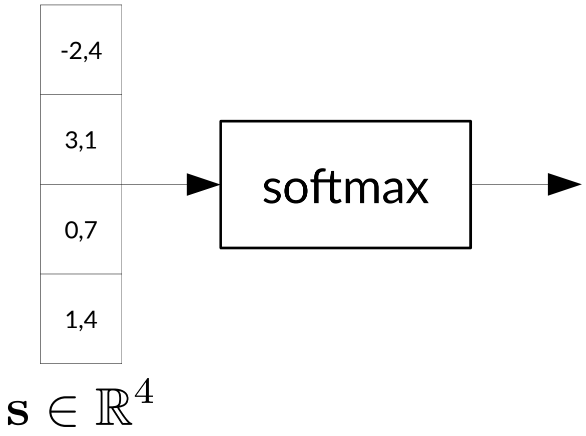
« Cross-entropy » $\mathrm{CE}(\mathbf{s}, y) = -\ln(\mathbf{p}_y)$ où $\underbrace{\mathbf{p}_i = \frac{\exp(\mathbf{s}_i)}{\sum_{c=0}^{C-1} \exp(\mathbf{s}_c)}}_{\text{« softmax »}}$

VI)

Choix du coût $l(y, \mathbf{s})$: Classification

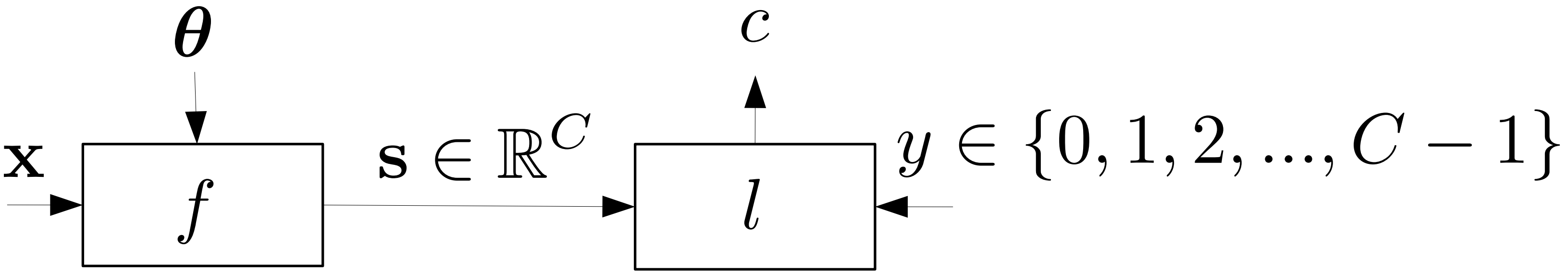


« Cross-entropy » $\mathrm{CE}(\mathbf{s}, y) = -\ln(\mathbf{p}_y)$ où $\underbrace{\mathbf{p}_i = \frac{\exp(\mathbf{s}_i)}{\sum_{c=0}^{C-1} \exp(\mathbf{s}_c)}}_{\text{« softmax »}}$

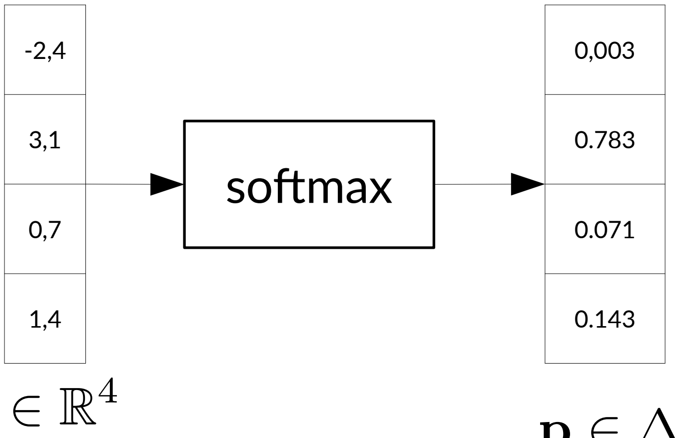


VI)

Choix du coût $l(y, \mathbf{s})$: Classification



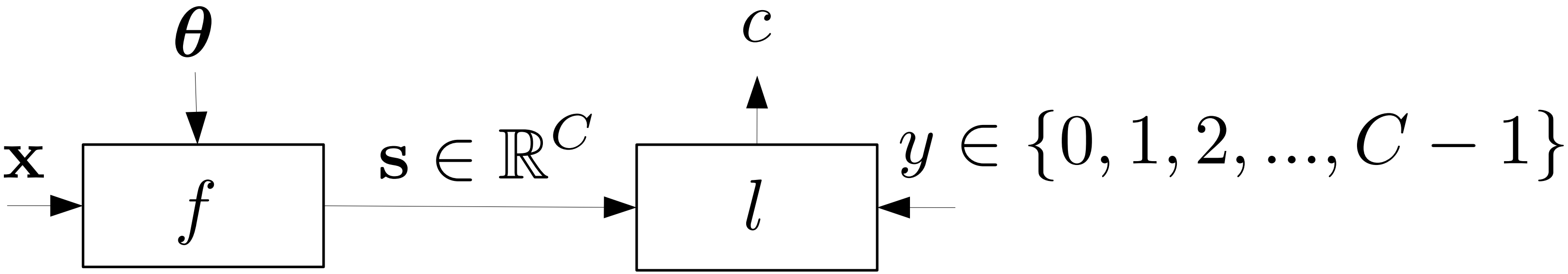
« Cross-entropy » $\mathrm{CE}(\mathbf{s}, y) = -\ln(\mathbf{p}_y)$ où $\underbrace{\mathbf{p}_i = \frac{\exp(\mathbf{s}_i)}{\sum_{c=0}^{C-1} \exp(\mathbf{s}_c)}}_{\text{« softmax »}}$



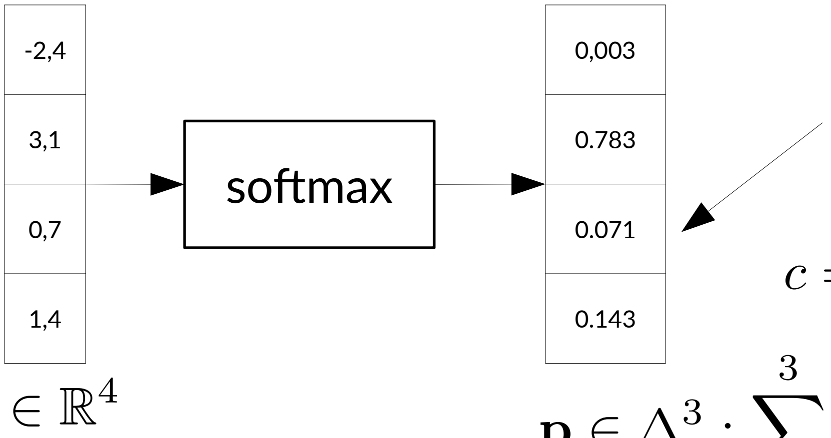
$$\mathbf{p} \in \Delta^3 : \sum_{i=0}^{3} \mathbf{p}_i = 1 \text{ et } \mathbf{p}_i \geq 0 \text{ pour } i = 0, ..., 3$$

VI)

Choix du coût $l(y, \mathbf{s})$: Classification



« Cross-entropy » $\mathrm{CE}(\mathbf{s}, y) = -\ln(\mathbf{p}_y)$ où $\underbrace{\mathbf{p}_i = \frac{\exp(\mathbf{s}_i)}{\sum_{c=0}^{C-1} \exp(\mathbf{s}_c)}}_{\text{« softmax »}}$

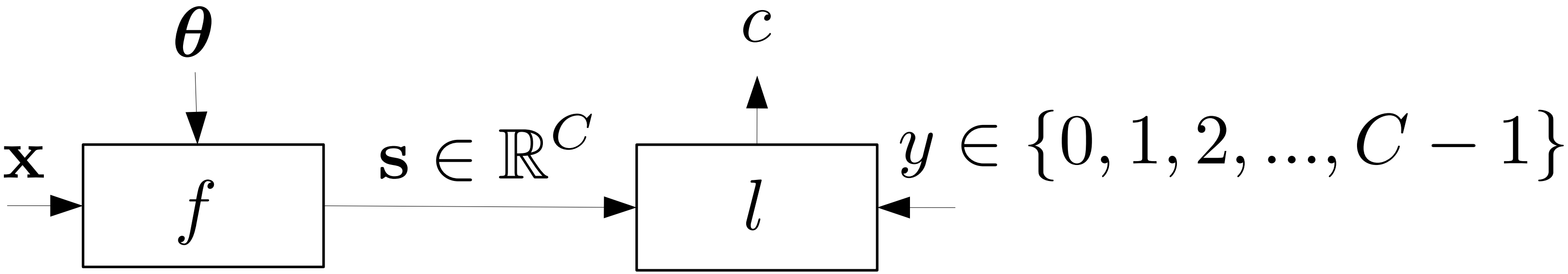


$$c = -\ln(0.071) = 2,647$$

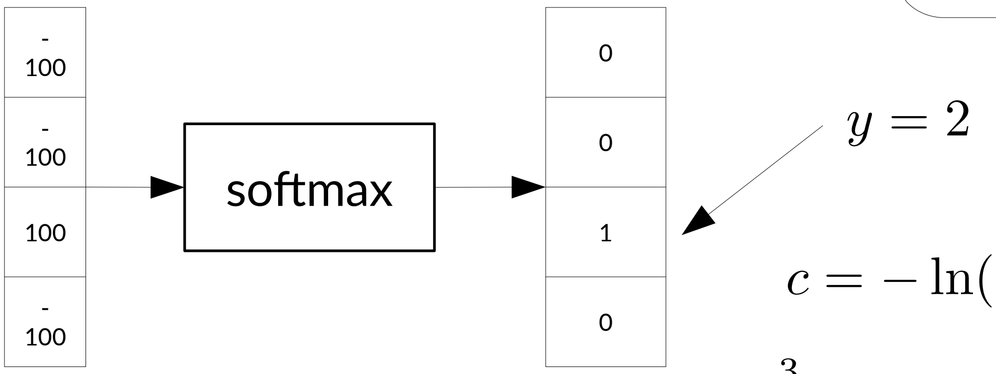
$$\mathbf{p} \in \Delta^3 : \sum_{i=0}^{3} \mathbf{p}_i = 1 \text{ et } \mathbf{p}_i \geq 0 \text{ pour } i = 0, ..., 3$$

VI)

Choix du coût $l(y, \mathbf{s})$: Classification



« Cross-entropy » $\mathrm{CE}(\mathbf{s}, y) = -\ln(\mathbf{p}_y)$ où $\underbrace{\mathbf{p}_i = \frac{\exp(\mathbf{s}_i)}{\sum_{c=0}^{C-1} \exp(\mathbf{s}_c)}}_{\text{« softmax »}}$



$\mathbf{s} \in \mathbb{R}^4$

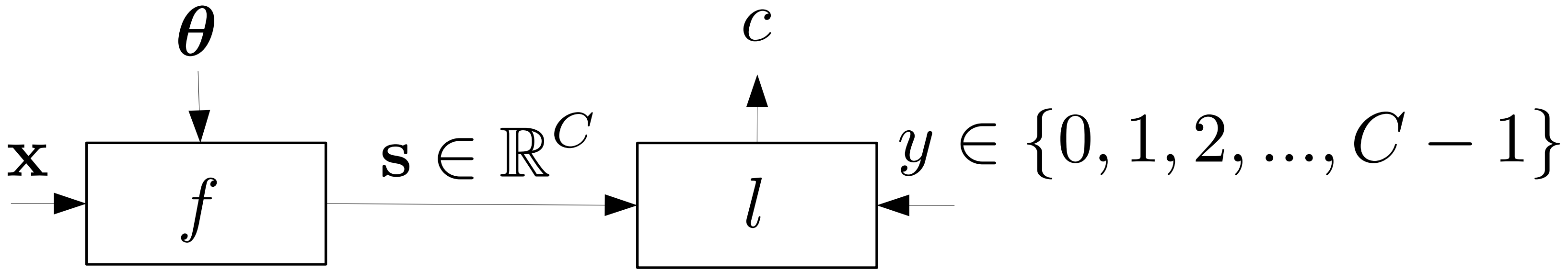
$c = -\ln(1) = 0$

$$\mathbf{p} \in \Delta^3 : \sum_{i=0}^{3} \mathbf{p}_i = 1 \text{ et } \mathbf{p}_i \geq 0 \text{ pour } i = 0, ..., 3$$

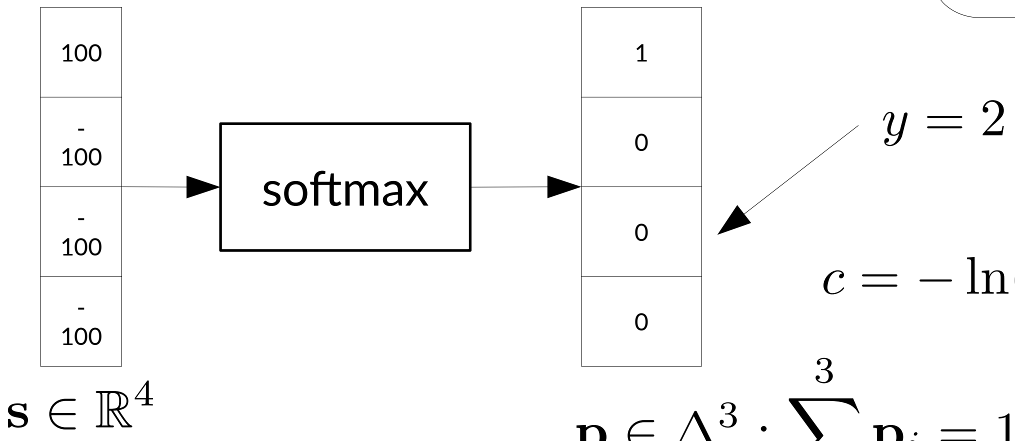
Exemple : prédiction « parfaite »

VI)

Choix du coût $l(y, \mathbf{s})$: Classification



« Cross-entropy » $\mathrm{CE}(\mathbf{s}, y) = -\ln(\mathbf{p}_y)$ où $\mathbf{p}_i = \underbrace{\frac{\exp(\mathbf{s}_i)}{\sum_{c=0}^{C-1} \exp(\mathbf{s}_c)}}_{\text{« softmax »}}$



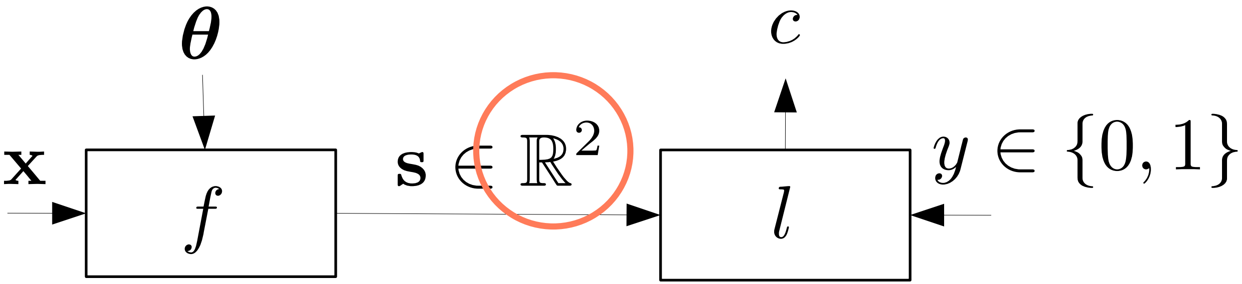
$$c = -\ln(0) = +\infty$$

$$\mathbf{p} \in \Delta^3 : \sum_{i=0}^{3} \mathbf{p}_i = 1 \text{ et } \mathbf{p}_i \geq 0 \text{ pour } i = 0, ..., 3$$

Exemple : prédiction «très mauvaise»

VI)

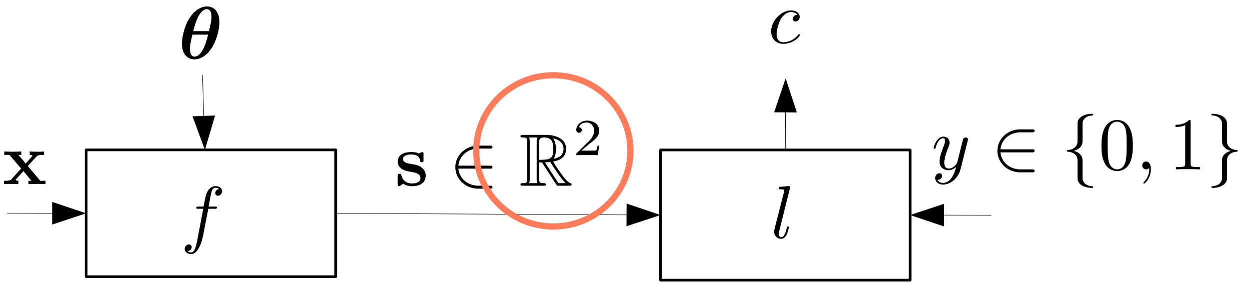
« Cross-entropy » à deux classes vs « Binary cross-entropy »



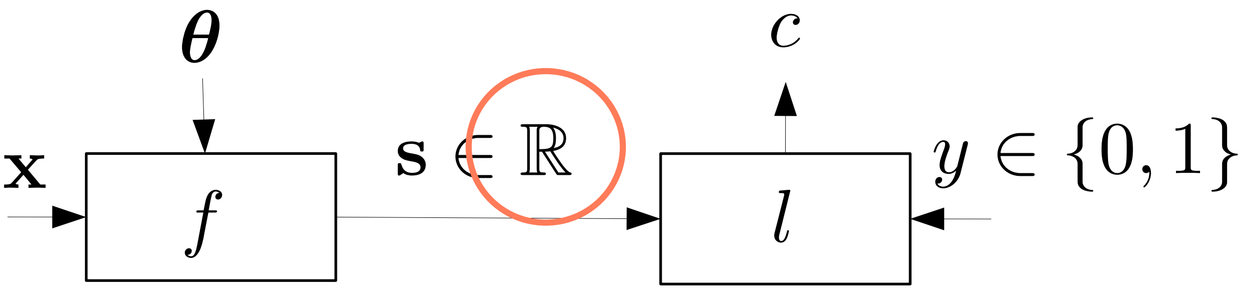
« Cross-entropy » $\mathrm{CE}(\mathbf{s}, y) = -\ln(\mathbf{p}_y)$ où $\mathbf{p}_i = \dfrac{\exp(\mathbf{s}_i)}{\sum_{c=0}^{C-1} \exp(\mathbf{s}_c)}$

VI)

« Cross-entropy » à deux classes vs « Binary cross-entropy »



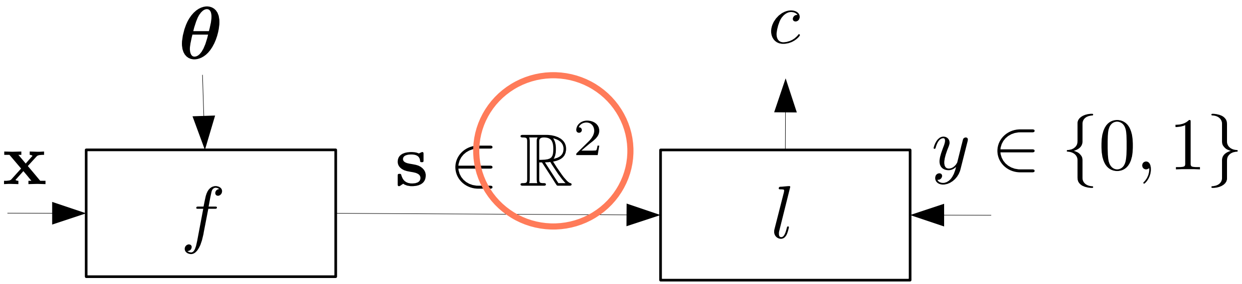
« Cross-entropy » $\mathrm{CE}(\mathbf{s}, y) = -\ln(\mathbf{p}_y)$ où $\mathbf{p}_i = \dfrac{\exp(\mathbf{s}_i)}{\sum_{c=0}^{C-1} \exp(\mathbf{s}_c)}$



« Binary cross-entropy » $-y\ln(\mathbf{p}) - (1 - y)\ln(1 - \mathbf{p})$ où $\overbrace{\mathbf{p} = \dfrac{1}{1 + \exp(-\mathbf{s})}}^{\text{« sigmoïde »}}$

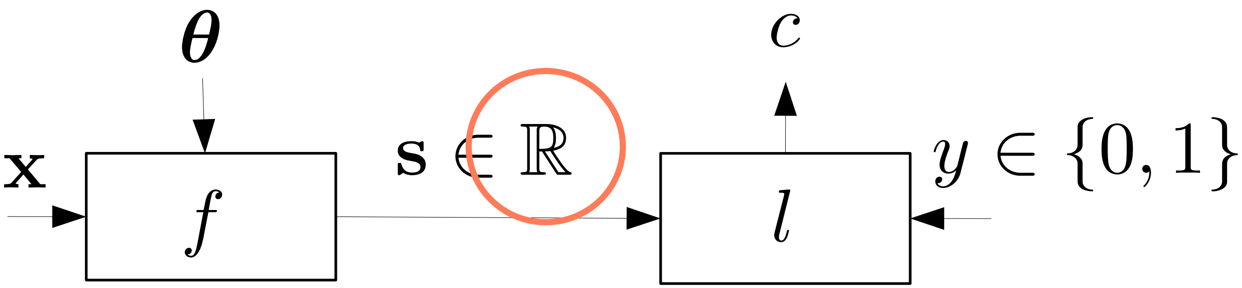
VI)

« Cross-entropy » à deux classes vs « Binary cross-entropy »



« Cross-entropy » $\quad \mathrm{CE}(\mathbf{s}, y) = -\ln(\mathbf{p}_y)$ où $\mathbf{p}_i = \dfrac{\exp(\mathbf{s}_i)}{\sum_{c=0}^{C-1} \exp(\mathbf{s}_c)}$

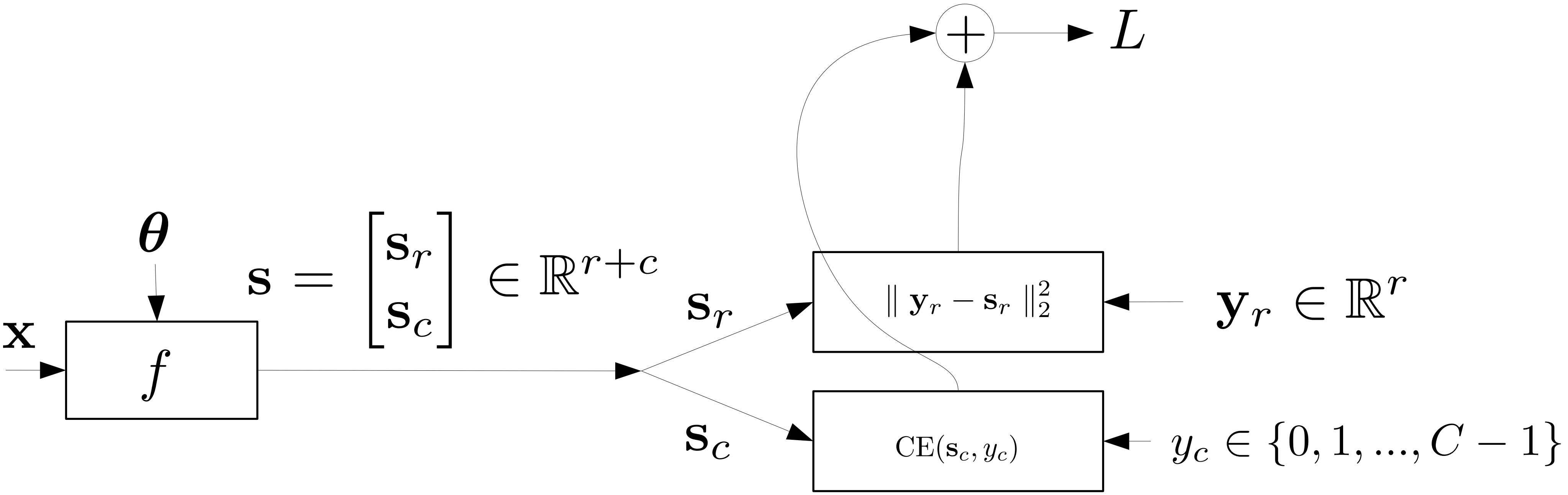
Strictement équivalent ! Juste une question d'implémentation.



« Binary cross-entropy » $\quad -y\ln(\mathbf{p}) - (1-y)\ln(1-\mathbf{p})$ où $\mathbf{p} = \overbrace{\dfrac{1}{1+\exp(-\mathbf{s})}}^{\text{« sigmoïde »}}$

VI)

Choix du coût $l(\mathbf{y}, \mathbf{s})$: Combinaison de coûts

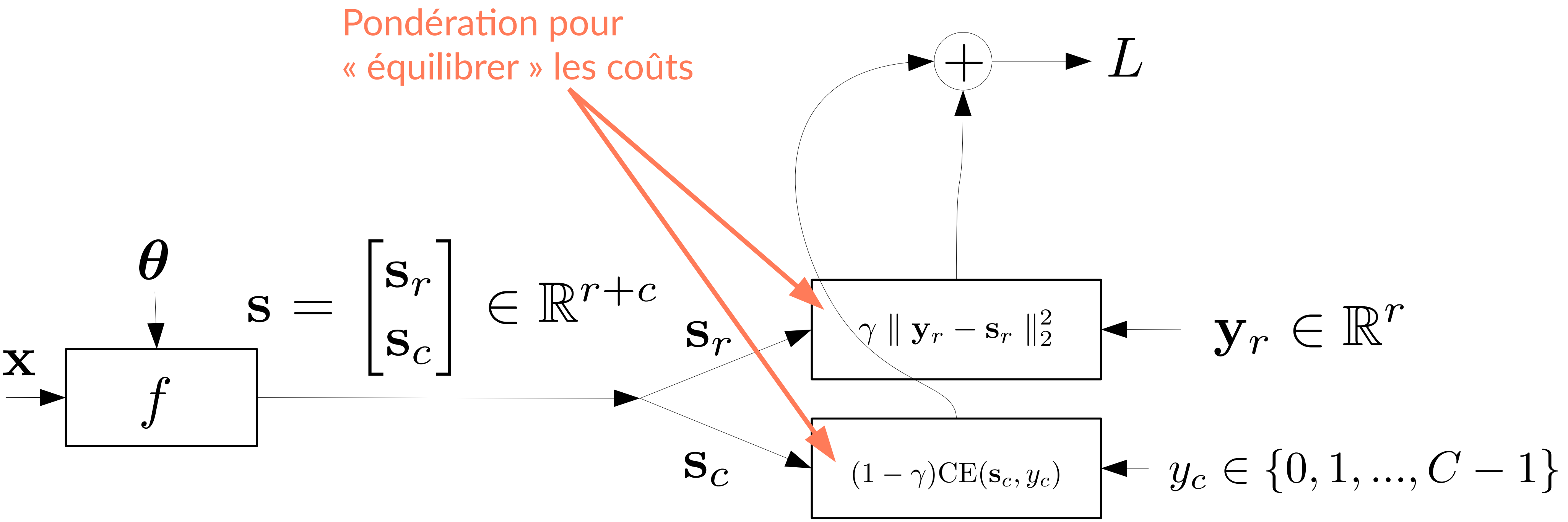


Exemple : classification et régression conjointe

Permet à un réseau d'apprendre à réaliser plusieurs tâches.

VI)

Choix du coût $l(\mathbf{y}, \mathbf{s})$: Combinaison de coûts



Exemple : classification et régression conjointe

Permet à un réseau d'apprendre à réaliser plusieurs tâches.

Optimisation des paramètres

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} l\left(\mathtt{Y}_{\mathrm{train},i}, f\left(\mathtt{X}_{\mathrm{train},i}; \boldsymbol{\theta}\right)\right)$$

où $$f(\mathbf{x}; \boldsymbol{\theta}) = f_L\left(f_{L-1}\left(...f_2\left(f_1\left(\mathbf{x}; \boldsymbol{\theta}^{(1)}\right); \boldsymbol{\theta}^{(2)}\right)...; \boldsymbol{\theta}^{(L-1)}\right); \boldsymbol{\theta}^{(L)}\right)$$

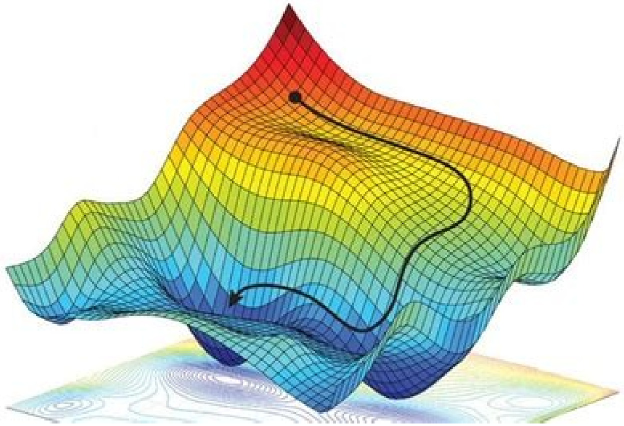
VI)

Optimisation des paramètres

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} l\left(\mathtt{Y}_{\text{train},i}, f\left(\mathtt{X}_{\text{train},i}; \boldsymbol{\theta}\right)\right)$$

où $$f(\mathbf{x};\boldsymbol{\theta}) = f_L\left(f_{L-1}\left(\ldots f_2\left(f_1\left(\mathbf{x};\boldsymbol{\theta}^{(1)}\right);\boldsymbol{\theta}^{(2)}\right)\ldots;\boldsymbol{\theta}^{(L-1)}\right);\boldsymbol{\theta}^{(L)}\right)$$

Descente de gradient $$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \alpha \frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k}$$



Pas d’apprentissage (« learning rate » en anglais)

Calcul du gradient

Objectif

Calculer le gradient de la fonction de coût par rapport aux paramètres du réseau de neurones.

VI)

Calcul du gradient

Objectif

Calculer le gradient de la fonction de coût par rapport aux paramètres du réseau de neurones.

Comment faire ?

Un réseau de neurones est une composition de fonctions.

↓

Application du théorème de dérivation d'une fonction composée (« chain rule »).

↓

L'implémentation qui en résulte s'appelle la rétropropagation du gradient (« backpropagation »)

Calcul du gradient

Rappel : Théorème de dérivation des fonctions composées

$$s = g(f(\mathbf{x})) \quad \text{avec} \quad s = g(\mathbf{y}) \quad \text{et} \quad \mathbf{y} = f(\mathbf{x})$$

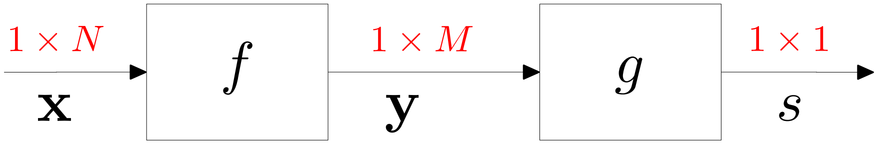
$$\xrightarrow[\mathbf{x}]{1 \times N} \boxed{f} \xrightarrow[\mathbf{y}]{1 \times M} \boxed{g} \xrightarrow[s]{1 \times 1}$$

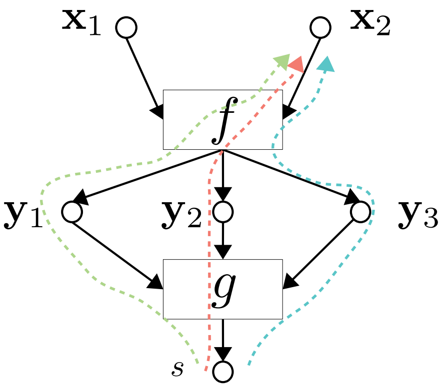
Remarque importante : l'objectif est de dériver le coût, c'est-à-dire un scalaire.

Calcul du gradient

Rappel : Théorème de dérivation des fonctions composées

$$s = g(f(\mathbf{x})) \quad \text{avec} \quad s = g(\mathbf{y}) \quad \text{et} \quad \mathbf{y} = f(\mathbf{x})$$





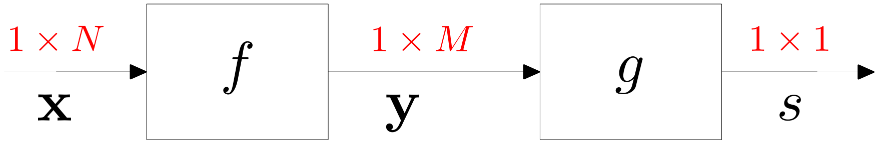
$$\frac{\partial s}{\partial \mathbf{x}_2} = \boxed{\left.\frac{\partial s}{\partial \mathbf{y}_1}\right|_{\mathbf{y}=f(\mathbf{x})} \frac{\partial \mathbf{y}_1}{\partial \mathbf{x}_2}} + \boxed{\left.\frac{\partial s}{\partial \mathbf{y}_2}\right|_{\mathbf{y}=f(\mathbf{x})} \frac{\partial \mathbf{y}_2}{\partial \mathbf{x}_2}} + \boxed{\left.\frac{\partial s}{\partial \mathbf{y}_3}\right|_{\mathbf{y}=f(\mathbf{x})} \frac{\partial \mathbf{y}_3}{\partial \mathbf{x}_2}}$$

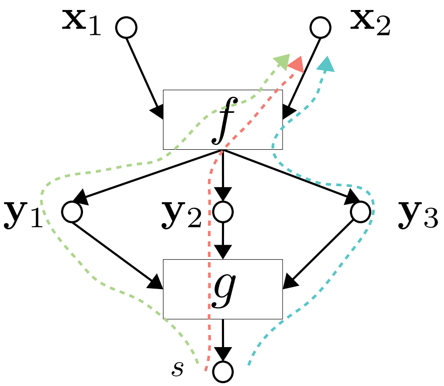
Exemple

Calcul du gradient

Rappel : Théorème de dérivation des fonctions composées

$s = g(f(\mathbf{x}))$ avec $s = g(\mathbf{y})$ et $\mathbf{y} = f(\mathbf{x})$





$$\frac{\partial s}{\partial \mathbf{x}_2} = \boxed{\frac{\partial s}{\partial \mathbf{y}_1}\bigg|_{\mathbf{y}=f(\mathbf{x})} \frac{\partial \mathbf{y}_1}{\partial \mathbf{x}_2}} + \boxed{\frac{\partial s}{\partial \mathbf{y}_2}\bigg|_{\mathbf{y}=f(\mathbf{x})} \frac{\partial \mathbf{y}_2}{\partial \mathbf{x}_2}} + \boxed{\frac{\partial s}{\partial \mathbf{y}_3}\bigg|_{\mathbf{y}=f(\mathbf{x})} \frac{\partial \mathbf{y}_3}{\partial \mathbf{x}_2}}$$

Exemple

$$\frac{\partial s}{\partial \mathbf{x}_i} = \sum_{j=1}^{M} \frac{\partial s}{\partial \mathbf{y}_j}\bigg|_{\mathbf{y}=f(\mathbf{x})} \frac{\partial \mathbf{y}_j}{\partial \mathbf{x}_i}$$

Formule : dérivation élément par élément

Calcul du gradient (suite)

Rappel : Théorème de dérivation des fonctions composées

$$s = g(f(\mathbf{x})) \quad \text{avec} \quad s = g(\mathbf{y}) \quad \text{et} \quad \mathbf{y} = f(\mathbf{x})$$

$$\frac{\partial s}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial s}{\partial \mathbf{x}_1} \\ \frac{\partial s}{\partial \mathbf{x}_2} \\ \frac{\partial s}{\partial \mathbf{x}_3} \\ \vdots \end{bmatrix}^\top = \begin{bmatrix} \sum_{j=1}^{M} \frac{\partial s}{\partial \mathbf{y}_j}\Big|_{\mathbf{y}=f(\mathbf{x})} \frac{\partial \mathbf{y}_j}{\partial \mathbf{x}_1} \\ \sum_{j=1}^{M} \frac{\partial s}{\partial \mathbf{y}_j}\Big|_{\mathbf{y}=f(\mathbf{x})} \frac{\partial \mathbf{y}_j}{\partial \mathbf{x}_2} \\ \sum_{j=1}^{M} \frac{\partial s}{\partial \mathbf{y}_j}\Big|_{\mathbf{y}=f(\mathbf{x})} \frac{\partial \mathbf{y}_j}{\partial \mathbf{x}_3} \\ \vdots \end{bmatrix}^\top = \begin{bmatrix} \frac{\partial s}{\partial \mathbf{y}_1}\Big|_{\mathbf{y}=f(\mathbf{x})} \\ \frac{\partial s}{\partial \mathbf{y}_2}\Big|_{\mathbf{y}=f(\mathbf{x})} \\ \vdots \end{bmatrix}^\top \begin{bmatrix} \frac{\partial \mathbf{y}_1}{\partial \mathbf{x}_1} & \frac{\partial \mathbf{y}_1}{\partial \mathbf{x}_2} & \frac{\partial \mathbf{y}_1}{\partial \mathbf{x}_3} & \cdots \\ \frac{\partial \mathbf{y}_2}{\partial \mathbf{x}_1} & \frac{\partial \mathbf{y}_2}{\partial \mathbf{x}_2} & \frac{\partial \mathbf{y}_2}{\partial \mathbf{x}_3} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} = \underbrace{\frac{\partial s}{\partial \mathbf{y}}\Big|_{\mathbf{y}=f(\mathbf{x})} \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}}_{\doteq \tilde{f}\left(\mathbf{x}, \frac{\partial s}{\partial \mathbf{y}}|_{\mathbf{y}=f(\mathbf{x})}\right)}$$

Calcul du gradient (suite)

Rappel : Théorème de dérivation des fonctions composées

$$s = g(f(\mathbf{x})) \quad \text{avec} \quad s = g(\mathbf{y}) \quad \text{et} \quad \mathbf{y} = f(\mathbf{x})$$

$$\frac{\partial s}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial s}{\partial \mathbf{x}_1} \\ \frac{\partial s}{\partial \mathbf{x}_2} \\ \frac{\partial s}{\partial \mathbf{x}_3} \\ \vdots \end{bmatrix}^\top = \begin{bmatrix} \sum_{j=1}^{M} \frac{\partial s}{\partial \mathbf{y}_j}\Big|_{\mathbf{y}=f(\mathbf{x})} \frac{\partial \mathbf{y}_j}{\partial \mathbf{x}_1} \\ \sum_{j=1}^{M} \frac{\partial s}{\partial \mathbf{y}_j}\Big|_{\mathbf{y}=f(\mathbf{x})} \frac{\partial \mathbf{y}_j}{\partial \mathbf{x}_2} \\ \sum_{j=1}^{M} \frac{\partial s}{\partial \mathbf{y}_j}\Big|_{\mathbf{y}=f(\mathbf{x})} \frac{\partial \mathbf{y}_j}{\partial \mathbf{x}_3} \\ \vdots \end{bmatrix}^\top = \begin{bmatrix} \frac{\partial s}{\partial \mathbf{y}_1}\Big|_{\mathbf{y}=f(\mathbf{x})} \\ \frac{\partial s}{\partial \mathbf{y}_2}\Big|_{\mathbf{y}=f(\mathbf{x})} \\ \vdots \end{bmatrix}^\top \begin{bmatrix} \frac{\partial \mathbf{y}_1}{\partial \mathbf{x}_1} & \frac{\partial \mathbf{y}_1}{\partial \mathbf{x}_2} & \frac{\partial \mathbf{y}_1}{\partial \mathbf{x}_3} & \cdots \\ \frac{\partial \mathbf{y}_2}{\partial \mathbf{x}_1} & \frac{\partial \mathbf{y}_2}{\partial \mathbf{x}_2} & \frac{\partial \mathbf{y}_2}{\partial \mathbf{x}_3} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} = \underbrace{\frac{\partial s}{\partial \mathbf{y}}\Big|_{\mathbf{y}=f(\mathbf{x})} \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}}_{\doteq \tilde{f}(\mathbf{x}, \frac{\partial s}{\partial \mathbf{y}}|_{\mathbf{y}=f(\mathbf{x})})}$$

$$\frac{\partial s}{\partial \mathbf{y}} = \begin{bmatrix} \frac{\partial s}{\partial \mathbf{y}_1} \\ \frac{\partial s}{\partial \mathbf{y}_2} \\ \vdots \end{bmatrix}^\top = \underbrace{1 \cdot \frac{\partial g(\mathbf{y})}{\partial \mathbf{y}}}_{\doteq \tilde{g}(\mathbf{y}, 1)}$$

Calcul du gradient (suite)

Rappel : Théorème de dérivation des fonctions composées

$$s = g(f(\mathbf{x})) \quad \text{avec} \quad s = g(\mathbf{y}) \quad \text{et} \quad \mathbf{y} = f(\mathbf{x})$$

$$\frac{\partial s}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial s}{\partial \mathbf{x}_1} \\ \frac{\partial s}{\partial \mathbf{x}_2} \\ \frac{\partial s}{\partial \mathbf{x}_3} \\ \vdots \end{bmatrix}^\top = \begin{bmatrix} \sum_{j=1}^{M} \frac{\partial s}{\partial \mathbf{y}_j}\Big|_{\mathbf{y}=f(\mathbf{x})} \frac{\partial \mathbf{y}_j}{\partial \mathbf{x}_1} \\ \sum_{j=1}^{M} \frac{\partial s}{\partial \mathbf{y}_j}\Big|_{\mathbf{y}=f(\mathbf{x})} \frac{\partial \mathbf{y}_j}{\partial \mathbf{x}_2} \\ \sum_{j=1}^{M} \frac{\partial s}{\partial \mathbf{y}_j}\Big|_{\mathbf{y}=f(\mathbf{x})} \frac{\partial \mathbf{y}_j}{\partial \mathbf{x}_3} \\ \vdots \end{bmatrix}^\top = \begin{bmatrix} \frac{\partial s}{\partial \mathbf{y}_1}\Big|_{\mathbf{y}=f(\mathbf{x})} \\ \frac{\partial s}{\partial \mathbf{y}_2}\Big|_{\mathbf{y}=f(\mathbf{x})} \\ \vdots \end{bmatrix}^\top \begin{bmatrix} \frac{\partial \mathbf{y}_1}{\partial \mathbf{x}_1} & \frac{\partial \mathbf{y}_1}{\partial \mathbf{x}_2} & \frac{\partial \mathbf{y}_1}{\partial \mathbf{x}_3} & \cdots \\ \frac{\partial \mathbf{y}_2}{\partial \mathbf{x}_1} & \frac{\partial \mathbf{y}_2}{\partial \mathbf{x}_2} & \frac{\partial \mathbf{y}_2}{\partial \mathbf{x}_3} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} = \underbrace{\frac{\partial s}{\partial \mathbf{y}}\Big|_{\mathbf{y}=f(\mathbf{x})} \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}}_{\doteq \tilde{f}\left(\mathbf{x}, \frac{\partial s}{\partial \mathbf{y}}\big|_{\mathbf{y}=f(\mathbf{x})}\right)}$$

$$\frac{\partial s}{\partial \mathbf{y}} = \begin{bmatrix} \frac{\partial s}{\partial \mathbf{y}_1} \\ \frac{\partial s}{\partial \mathbf{y}_2} \\ \vdots \end{bmatrix}^\top = \underbrace{1 \cdot \frac{\partial g(\mathbf{y})}{\partial \mathbf{y}}}_{\doteq \tilde{g}(\mathbf{y},1)}$$

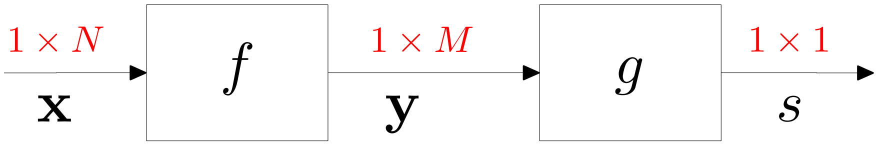
$$\frac{\partial s}{\partial \mathbf{x}} = \tilde{f}\left(\mathbf{x}, \tilde{g}\left(\mathbf{y}, 1\right)\right)$$

Composition de fonctions

Calcul du gradient (suite)

Rappel : Théorème de dérivation des fonctions composées

$s = g(f(\mathbf{x}))$ avec $s = g(\mathbf{y})$ et $\mathbf{y} = f(\mathbf{x})$



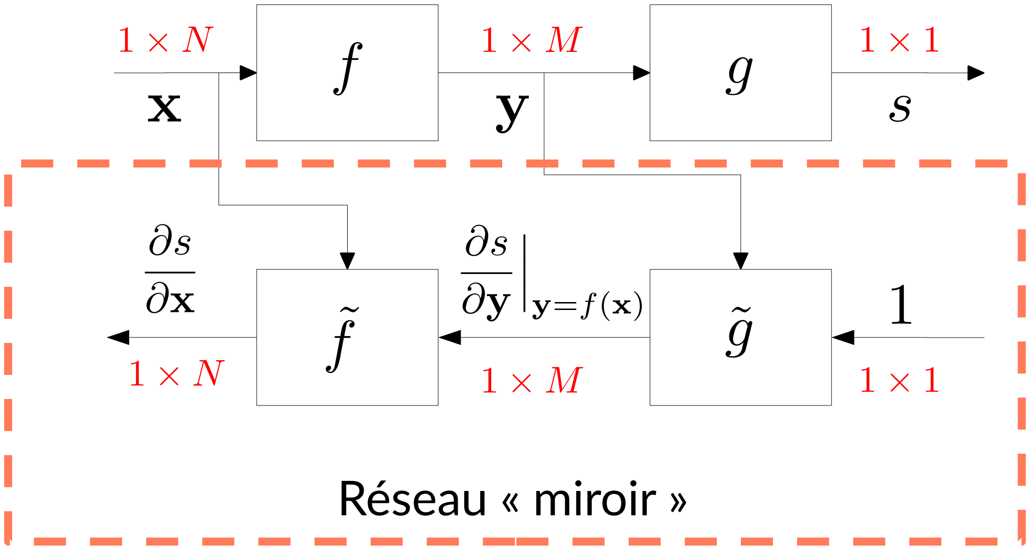
$$\frac{\partial s}{\partial \mathbf{x}} = \tilde{f}\left(\mathbf{x}, \tilde{g}\left(\mathbf{y}, 1\right)\right)$$

Calcul du gradient (suite)

Rappel : Théorème de dérivation des fonctions composées

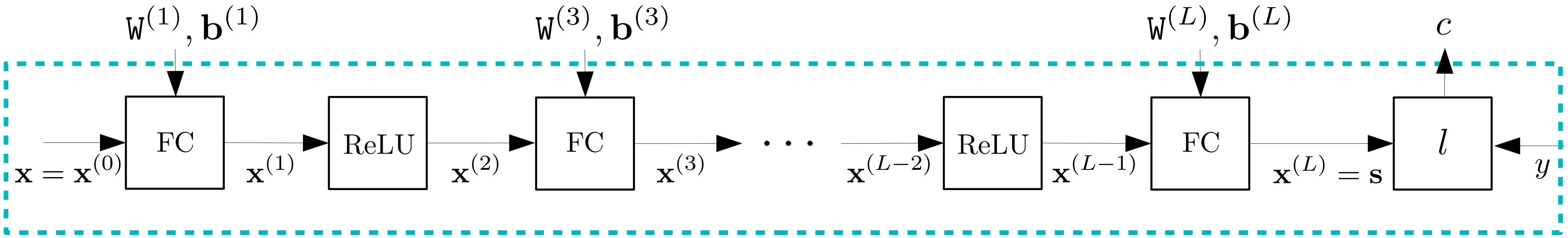
$$s = g(f(\mathbf{x})) \quad \text{avec} \quad s = g(\mathbf{y}) \quad \text{et} \quad \mathbf{y} = f(\mathbf{x})$$

$$\frac{\partial s}{\partial \mathbf{x}} = \tilde{f}\left(\mathbf{x}, \tilde{g}\left(\mathbf{y}, 1\right)\right)$$



VI)

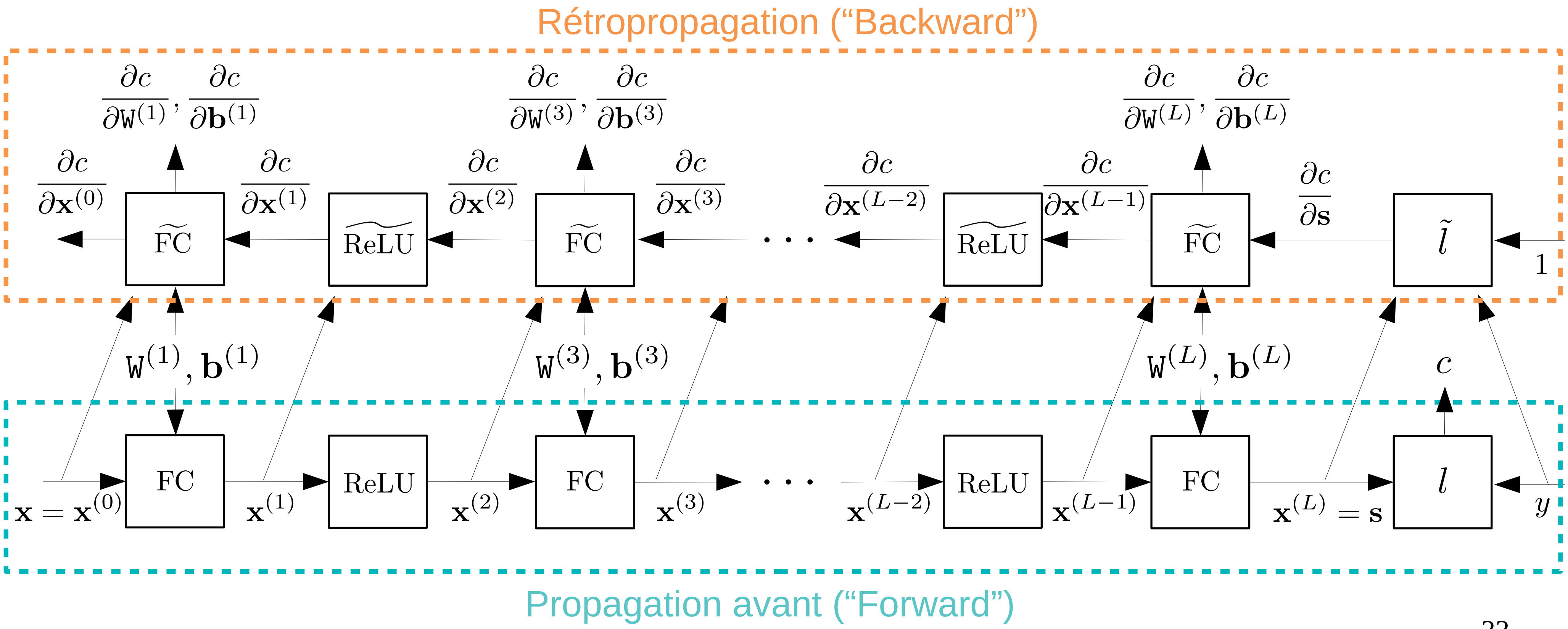
Calcul automatique du gradient du MLP



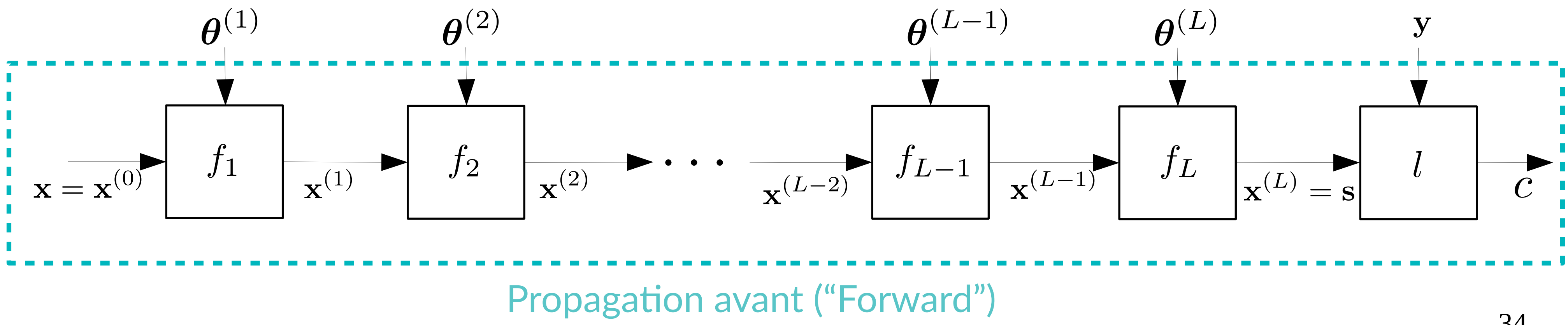
Propagation avant (“Forward”)

VI)

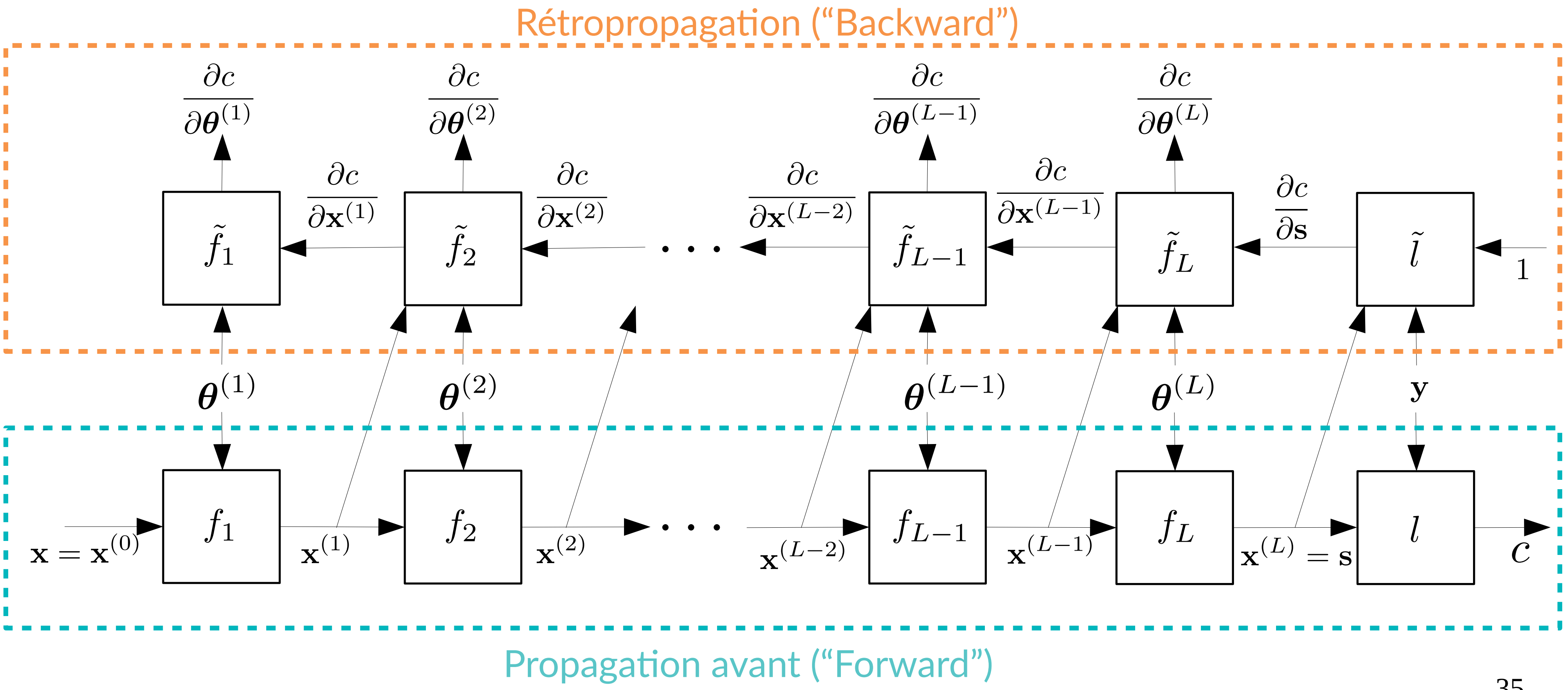
Calcul automatique du gradient du MLP



Calcul automatique du gradient



Calcul automatique du gradient



"Differentiable Programming"

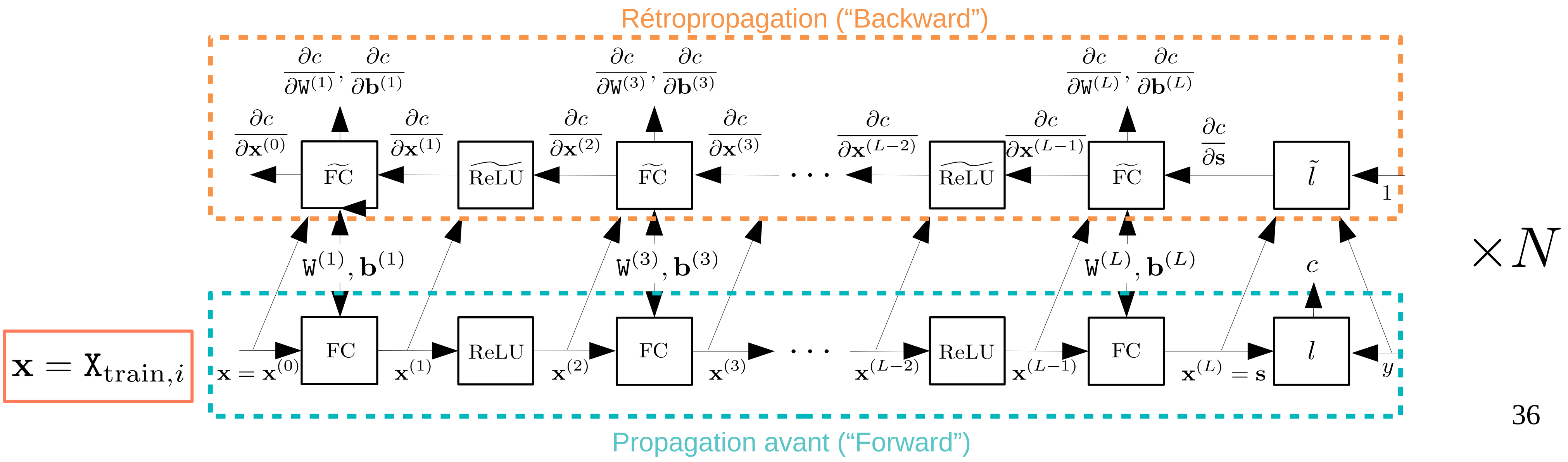
VI)

Calcul automatique du gradient (suite)

Implémentation

$$\left.\frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k} = \sum_{i=1}^{N} \left.\frac{\partial l\left(\mathtt{Y}_{\mathrm{train},i}, f\left(\mathtt{X}_{\mathrm{train},i}; \boldsymbol{\theta}\right)\right)}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k}$$

Le plus économe en mémoire : on calcule les gradients l'un après l'autre en les accumulant.



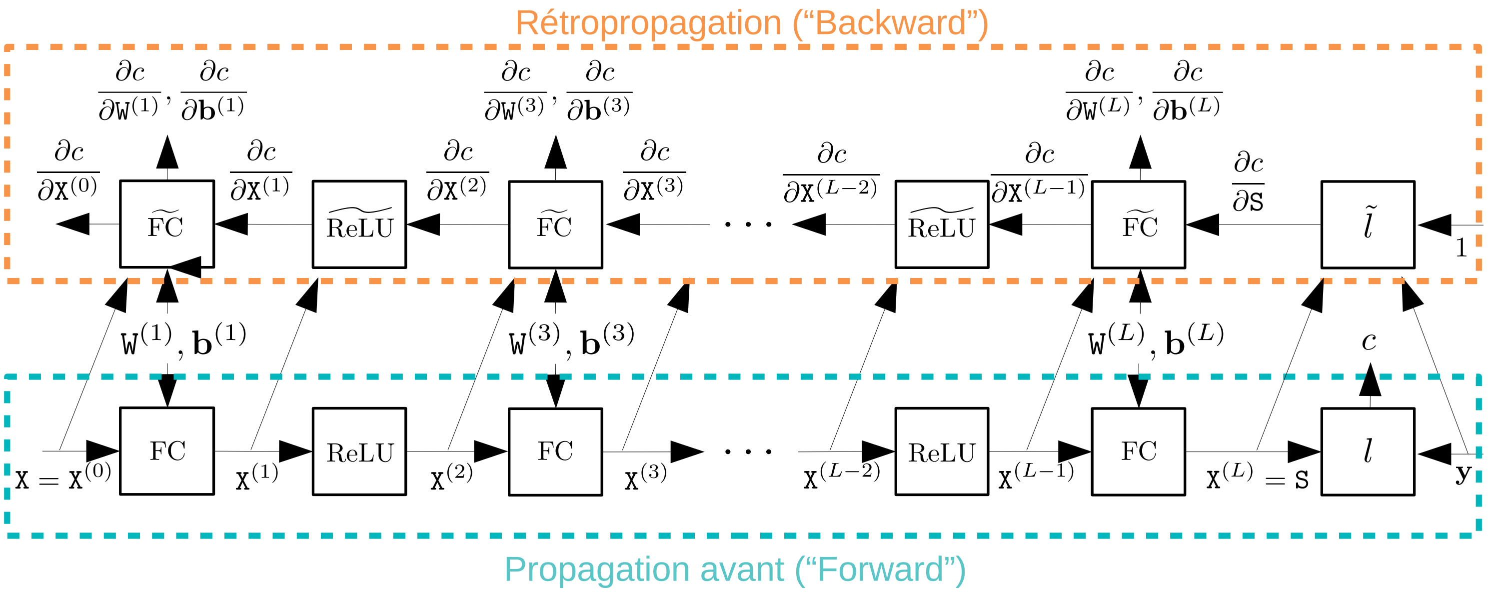
Calcul automatique du gradient (suite)

Implémentation

$$\frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k} = \sum_{i=1}^{N} \frac{\partial l\left(\mathtt{Y}_{\mathrm{train},i}, f\left(\mathtt{X}_{\mathrm{train},i};\boldsymbol{\theta}\right)\right)}{\partial \boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k}$$

Le plus rapide : on calcule tous les gradients en parallèle, puis on les somme.

$\mathtt{X} = \mathtt{X}_{\mathrm{train}}$



$\times 1$

Initialisation des paramètres

- La méthode la plus utilisée consiste à initialiser les paramètres des FC aléatoirement (distribution normale ou uniforme).

Kaiming init.

$$\mathtt{W}_0 = \sqrt{\frac{6}{n_{\text{in}}}}\mathcal{U}_{[-1,1]}(n_{\text{out}}, n_{\text{in}}) \quad \mathbf{b}_0 = \frac{1}{\sqrt{n_{\text{in}}}}\mathcal{U}_{[-1,1]}(n_{\text{out}})$$

He, K., et al. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification." ICCV 2015.

- D'autres méthodes existent mais sont moins utilisées (car la précédente fonctionne bien en pratique).

Mishkin, D., & Matas, J. All you need is a good init. 2015

Apprendre sur une grande base de données annotées

Descente de gradient (GD) :

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \alpha \sum_{i=1}^{N_{\mathrm{train}}} \left.\frac{\partial l\left(\mathtt{Y}_{\mathrm{train},i}, f\left(\mathtt{X}_{\mathrm{train},i}; \boldsymbol{\theta}\right)\right)}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k}$$

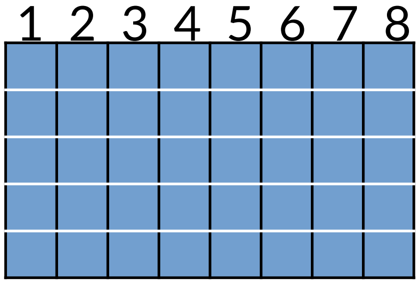
Apprendre sur une grande base de données annotées

Descente de gradient (GD) :

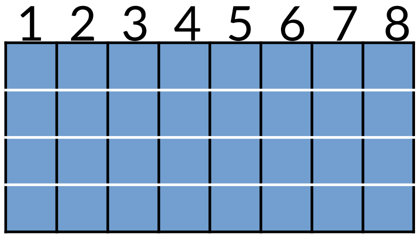
$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \alpha \sum_{i=1}^{N_{\text{train}}} \left. \frac{\partial l\left(\mathtt{Y}_{\text{train},i}, f\left(\mathtt{X}_{\text{train},i}; \boldsymbol{\theta}\right)\right)}{\partial \boldsymbol{\theta}} \right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k}$$

Descente de gradient stochastique (SGD) :

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \alpha \sum_{i \in \Omega_k} \left. \frac{\partial l\left(\mathtt{Y}_{\text{train},i}, f\left(\mathtt{X}_{\text{train},i}; \boldsymbol{\theta}\right)\right)}{\partial \boldsymbol{\theta}} \right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k}$$



$\mathtt{X}_{\text{train}}$



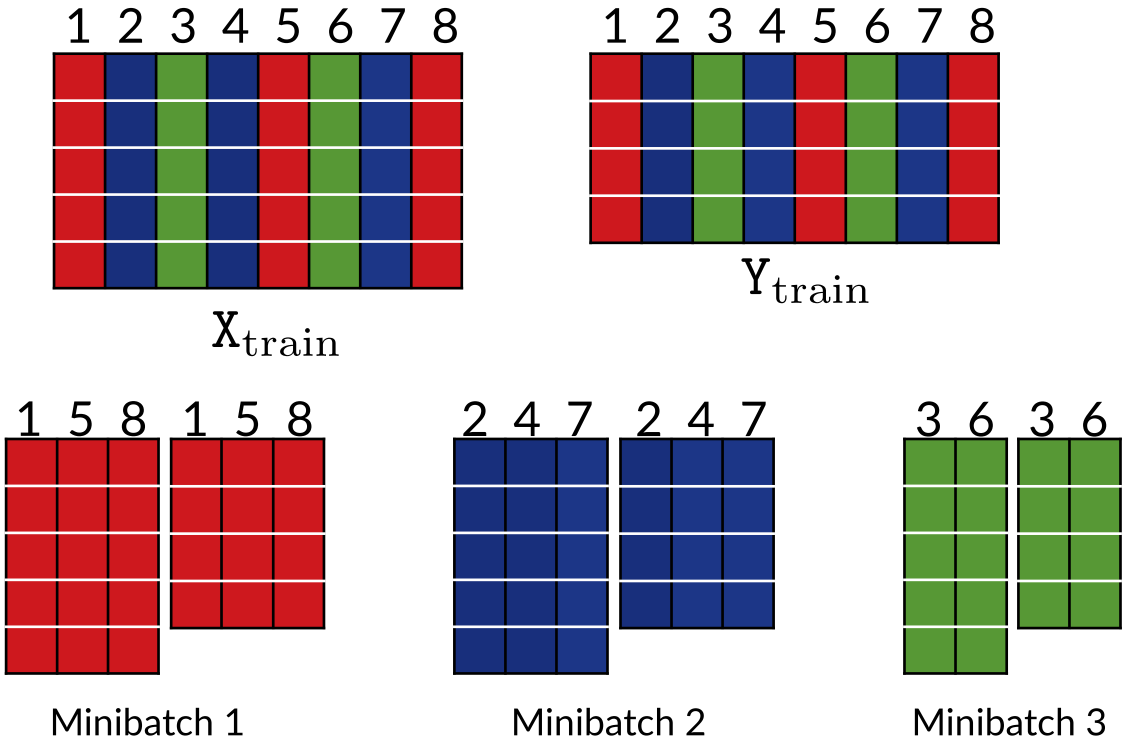
$\mathtt{Y}_{\text{train}}$

Tirage aléatoire, à chaque itération, de $|\Omega_k|$ éléments dans la base de données

Apprendre sur une grande base de données annotées (suite)

Descente de gradient stochastique (SGD) :

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \alpha \sum_{i \in \Omega_k} \left. \frac{\partial l\left(\mathtt{Y}_{\mathrm{train},i}, f\left(\mathtt{X}_{\mathrm{train},i}; \boldsymbol{\theta}\right)\right)}{\partial \boldsymbol{\theta}} \right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k}$$



Une « epoch » :

1) Découper aléatoirement la base de données en « minibatches » de taille $|\Omega_k|$

2) Faire une itération de SGD sur chaque « minibatch »

3) Fin de l' « epoch », aller à 1)

Avantages et inconvénients de la SGD

$$\sum_{i\in\Omega_k} \frac{\partial l\left(\mathtt{Y}_{\mathrm{train},i}, f\left(\mathtt{X}_{\mathrm{train},i};\boldsymbol{\theta}\right)\right)}{\partial\boldsymbol{\theta}}\bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k} = \sum_{i=1}^{N_{\mathrm{train}}} \frac{\partial l\left(\mathtt{Y}_{\mathrm{train},i}, f\left(\mathtt{X}_{\mathrm{train},i};\boldsymbol{\theta}\right)\right)}{\partial\boldsymbol{\theta}}\bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k} - \sum_{i\notin\Omega_k} \frac{\partial l\left(\mathtt{Y}_{\mathrm{train},i}, f\left(\mathtt{X}_{\mathrm{train},i};\boldsymbol{\theta}\right)\right)}{\partial\boldsymbol{\theta}}\bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k}$$

Gradient SGD
(« minibatch »)

Gradient GD
(« base de données »)

« bruit »

Avantages et inconvénients de la SGD

$$\sum_{i\in\Omega_k}\frac{\partial l\left(\mathtt{Y}_{\mathrm{train},i}, f\left(\mathtt{X}_{\mathrm{train},i};\boldsymbol{\theta}\right)\right)}{\partial\boldsymbol{\theta}}\bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k} = \sum_{i=1}^{N_{\mathrm{train}}}\frac{\partial l\left(\mathtt{Y}_{\mathrm{train},i}, f\left(\mathtt{X}_{\mathrm{train},i};\boldsymbol{\theta}\right)\right)}{\partial\boldsymbol{\theta}}\bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k} - \sum_{i\notin\Omega_k}\frac{\partial l\left(\mathtt{Y}_{\mathrm{train},i}, f\left(\mathtt{X}_{\mathrm{train},i};\boldsymbol{\theta}\right)\right)}{\partial\boldsymbol{\theta}}\bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k}$$

Gradient SGD
(« minibatch »)

Gradient GD
(« base de données »)

« bruit »

Inconvénient : l’apprentissage peut être compliqué/lent si le « bruit » est trop important (exemple : taille du « minibatch » trop faible)

VI)

Avantages et inconvénients de la SGD

$$\sum_{i\in\Omega_k} \frac{\partial l\left(\mathtt{Y}_{\mathrm{train},i}, f\left(\mathtt{X}_{\mathrm{train},i};\boldsymbol{\theta}\right)\right)}{\partial\boldsymbol{\theta}}\bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k} = \sum_{i=1}^{N_{\mathrm{train}}} \frac{\partial l\left(\mathtt{Y}_{\mathrm{train},i}, f\left(\mathtt{X}_{\mathrm{train},i};\boldsymbol{\theta}\right)\right)}{\partial\boldsymbol{\theta}}\bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k} - \sum_{i\notin\Omega_k} \frac{\partial l\left(\mathtt{Y}_{\mathrm{train},i}, f\left(\mathtt{X}_{\mathrm{train},i};\boldsymbol{\theta}\right)\right)}{\partial\boldsymbol{\theta}}\bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k}$$

Gradient SGD
(« minibatch »)

Gradient GD
(« base de données »)

« bruit »

Inconvénient : l'apprentissage peut être compliqué/lent si le « bruit » est trop important (exemple : taille du « minibatch » trop faible)

Avantage 1 : ce « bruit » peut permettre de sortir ou d'éviter de mauvais minima locaux

Avantages et inconvénients de la SGD

$$\sum_{i \in \Omega_k} \frac{\partial l\left(\mathtt{Y}_{\text{train},i}, f\left(\mathtt{X}_{\text{train},i}; \boldsymbol{\theta}\right)\right)}{\partial \boldsymbol{\theta}}\bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k} = \sum_{i=1}^{N_{\text{train}}} \frac{\partial l\left(\mathtt{Y}_{\text{train},i}, f\left(\mathtt{X}_{\text{train},i}; \boldsymbol{\theta}\right)\right)}{\partial \boldsymbol{\theta}}\bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k} - \sum_{i \notin \Omega_k} \frac{\partial l\left(\mathtt{Y}_{\text{train},i}, f\left(\mathtt{X}_{\text{train},i}; \boldsymbol{\theta}\right)\right)}{\partial \boldsymbol{\theta}}\bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k}$$

Gradient SGD
(« minibatch »)

Gradient GD
(« base de données »)

« bruit »

Inconvénient : l'apprentissage peut être compliqué/lent si le « bruit » est trop important (exemple : taille du « minibatch » trop faible)

Avantage 1 : ce « bruit » peut permettre de sortir ou d'éviter de mauvais minima locaux

Avantage 2 : le gradient est très rapide à calculer

VI)

Avantages et inconvénients de la SGD

$$\sum_{i\in\Omega_k} \frac{\partial l\left(\mathtt{Y}_{\mathrm{train},i}, f\left(\mathtt{X}_{\mathrm{train},i};\boldsymbol{\theta}\right)\right)}{\partial\boldsymbol{\theta}}\Bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k} = \sum_{i=1}^{N_{\mathrm{train}}} \frac{\partial l\left(\mathtt{Y}_{\mathrm{train},i}, f\left(\mathtt{X}_{\mathrm{train},i};\boldsymbol{\theta}\right)\right)}{\partial\boldsymbol{\theta}}\Bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k} - \sum_{i\notin\Omega_k} \frac{\partial l\left(\mathtt{Y}_{\mathrm{train},i}, f\left(\mathtt{X}_{\mathrm{train},i};\boldsymbol{\theta}\right)\right)}{\partial\boldsymbol{\theta}}\Bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k}$$

Gradient SGD (« minibatch ») | Gradient GD (« base de données ») | « bruit »

Inconvénient : l’apprentissage peut être compliqué/lent si le « bruit » est trop important (exemple : taille du « minibatch » trop faible)

Avantage 1 : ce « bruit » peut permettre de sortir ou d’éviter de mauvais minima locaux

Avantage 2 : le gradient est très rapide à calculer

Avantage 3 (empirique) : l’utilisation de petits « minibatches » (32-512) conduit à une bien meilleure généralisation que l’utilisation de grands « minibatches »

Keskar, N., et al. "On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima.", ICLR 2017

Avantages et inconvénients de la SGD

$$\sum_{i\in\Omega_k} \frac{\partial l\left(\mathtt{Y}_{\mathrm{train},i}, f\left(\mathtt{X}_{\mathrm{train},i};\boldsymbol{\theta}\right)\right)}{\partial\boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k} = \sum_{i=1}^{N_{\mathrm{train}}} \frac{\partial l\left(\mathtt{Y}_{\mathrm{train},i}, f\left(\mathtt{X}_{\mathrm{train},i};\boldsymbol{\theta}\right)\right)}{\partial\boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k} - \sum_{i\notin\Omega_k} \frac{\partial l\left(\mathtt{Y}_{\mathrm{train},i}, f\left(\mathtt{X}_{\mathrm{train},i};\boldsymbol{\theta}\right)\right)}{\partial\boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k}$$

Gradient SGD (« minibatch ») — Gradient GD (« base de données ») — « bruit »

En pratique, on choisit une taille de « minibatch » de manière à occuper entièrement le GPU.

Avantages et inconvénients de la SGD

$$\sum_{i\in\Omega_k} \frac{\partial l\left(\mathtt{Y}_{\mathrm{train},i}, f\left(\mathtt{X}_{\mathrm{train},i};\boldsymbol{\theta}\right)\right)}{\partial \boldsymbol{\theta}}\Bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k} = \sum_{i=1}^{N_{\mathrm{train}}} \frac{\partial l\left(\mathtt{Y}_{\mathrm{train},i}, f\left(\mathtt{X}_{\mathrm{train},i};\boldsymbol{\theta}\right)\right)}{\partial \boldsymbol{\theta}}\Bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k} - \sum_{i\notin\Omega_k} \frac{\partial l\left(\mathtt{Y}_{\mathrm{train},i}, f\left(\mathtt{X}_{\mathrm{train},i};\boldsymbol{\theta}\right)\right)}{\partial \boldsymbol{\theta}}\Bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k}$$

Gradient SGD
(« minibatch »)

Gradient GD
(« base de données »)

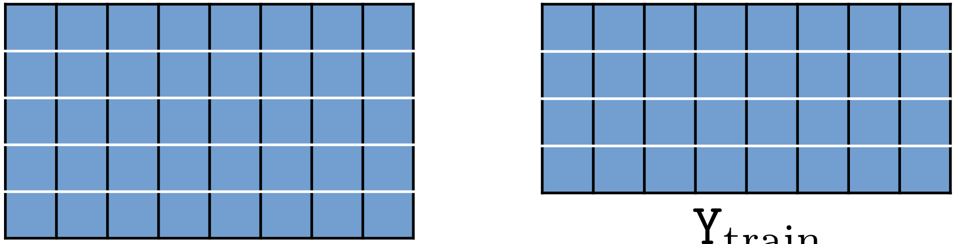
« bruit »

En pratique, on choisit une taille de « minibatch » de manière à occuper entièrement le GPU.

Si on a besoin d’un « minibatch » plus grand, on peut faire de l’accumulation de gradient, c’est-à-dire faire plusieurs « forward » + « backward » avec un « minibatch » plus petit.

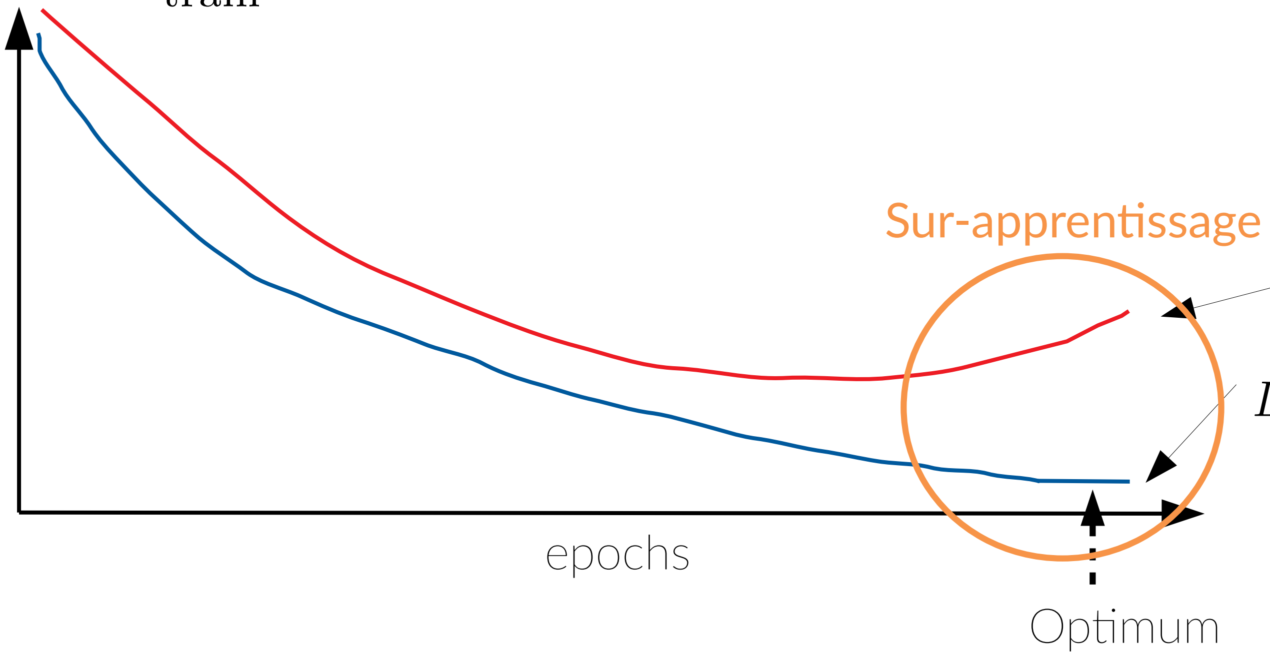
VI)

Apprendre à « bien » prédire… sur de nouvelles données



$\mathtt{X}_{\text{train}}$ $\mathtt{Y}_{\text{train}}$

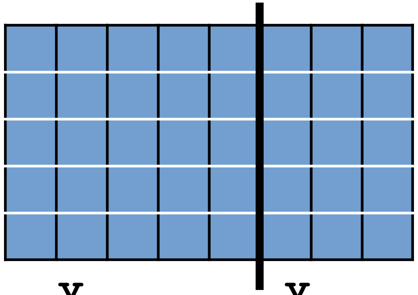
$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} L_{\text{train}}\left(\boldsymbol{\theta}\right)$$

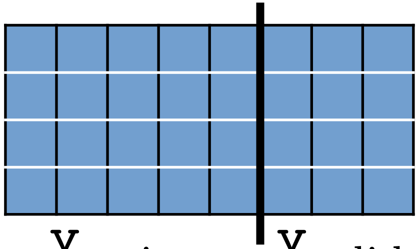


$$l\left(y_{\text{new}}, f\left(\mathbf{x}_{\text{new}}; \boldsymbol{\theta}\right)\right)$$

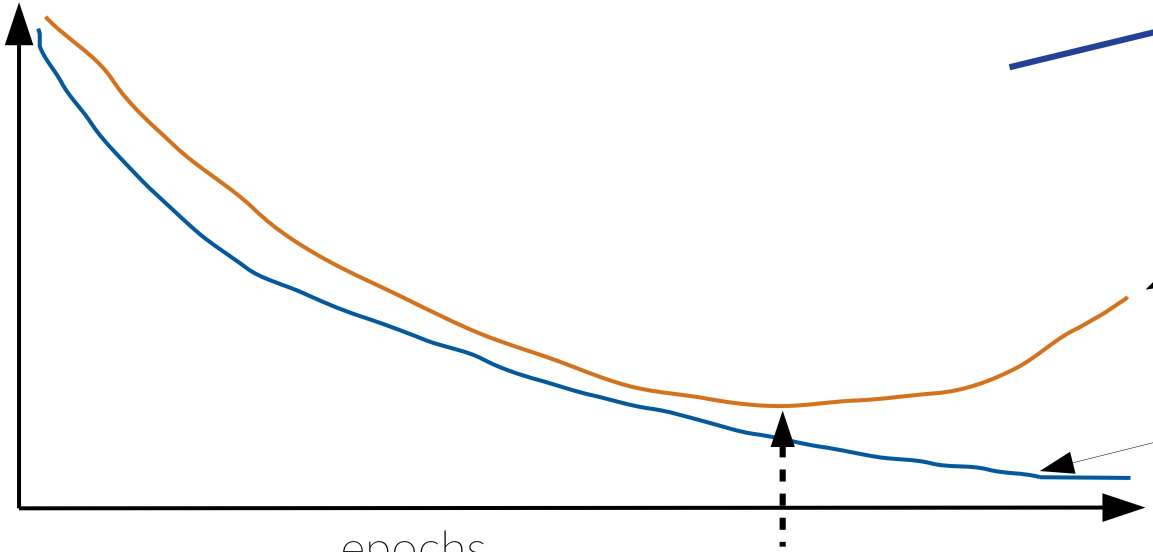
$$L_{\text{train}}\left(\boldsymbol{\theta}\right) = \sum_{i=1}^{N_{\text{train}}} l\left(\mathtt{Y}_{\text{train},i}, f\left(\mathtt{X}_{\text{train},i}; \boldsymbol{\theta}\right)\right)$$

Apprendre à « bien » prédire… sur de nouvelles données (suite)





$$\cancel{\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} L_{\text{train}}\left(\boldsymbol{\theta}\right)}$$



$$L_{\text{valid}}\left(\boldsymbol{\theta}\right) = \sum_{i=1}^{N_{\text{valid}}} l\left(\mathtt{Y}_{\text{valid},i}, f\left(\mathtt{X}_{\text{valid},i}; \boldsymbol{\theta}\right)\right)$$

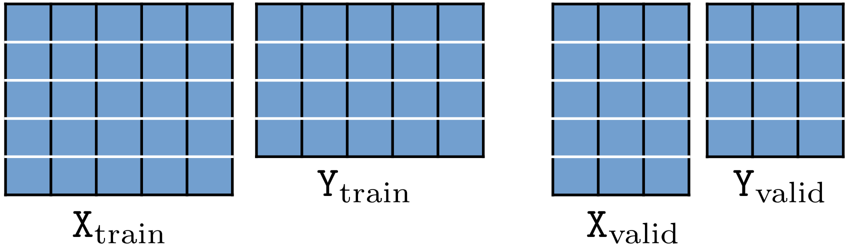
$$L_{\text{train}}\left(\boldsymbol{\theta}\right) = \sum_{i=1}^{N_{\text{train}}} l\left(\mathtt{Y}_{\text{train},i}, f\left(\mathtt{X}_{\text{train},i}; \boldsymbol{\theta}\right)\right)$$

Résumé de l'étape d'apprentissage

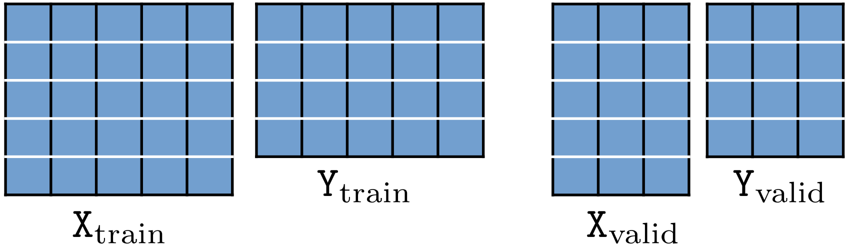
1) Découper une fois pour toutes la base de données en

une base d'apprentissage (« training set »)

une base de validation (« validation set »)



Résumé de l’étape d’apprentissage



1) Découper une fois pour toutes la base de données en

   une base d’apprentissage (« training set »)

   une base de validation (« validation set »)

2) Lancer une descente de gradient stochastique (SGD) avec arrêt prématuré

   Au début d’une « epoch », découper la base d’apprentissage aléatoirement en « minibatches »

   Faire une itération de SGD sur chaque « minibatch » : $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \alpha \sum_{i \in \Omega_k} \left. \frac{\partial l\left(\mathtt{Y}_{\mathrm{train},i}, f\left(\mathtt{X}_{\mathrm{train},i}; \boldsymbol{\theta}\right)\right)}{\partial \boldsymbol{\theta}} \right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k}$

   A la fin d’une « epoch », calculer $L_{\mathrm{valid}}\left(\boldsymbol{\theta}\right) = \sum_{i=1}^{N_{\mathrm{valid}}} l\left(\mathtt{Y}_{\mathrm{valid},i}, f\left(\mathtt{X}_{\mathrm{valid},i}; \boldsymbol{\theta}\right)\right)$

   Stocker la valeur actuelle de $\boldsymbol{\theta}$ si le coût de validation est plus faible que le précédent meilleur coût

   Stopper l’entraînement lorsqu’on est en régime de « sur-apprentissage »

Bonnes pratiques

- Lancer un apprentissage sur un seul « minibatch » jusqu'à obtention d'un coût d'apprentissage de zéro

Bonnes pratiques

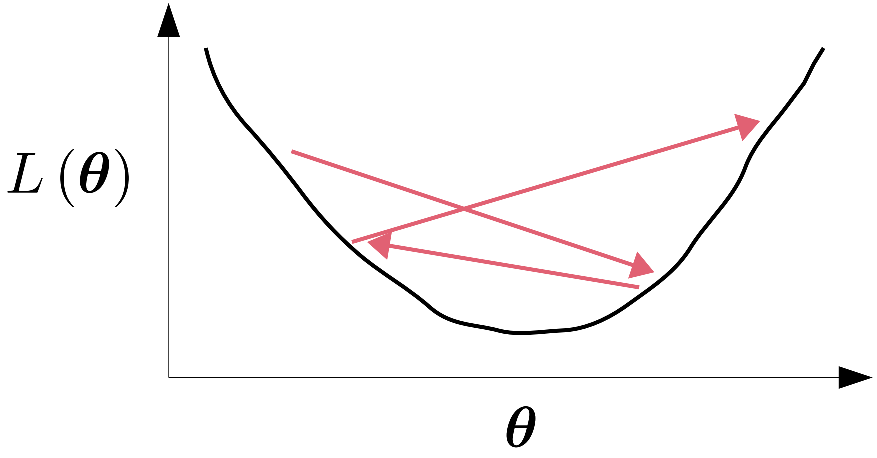
- Lancer un apprentissage sur un seul « minibatch » jusqu'à obtention d'un coût d'apprentissage de zéro

- Visualiser tout ce qu'il est possible de visualiser
  - Entrées → plage de valeurs (erreur classique : les données ne sont pas normalisées)
  - Sorties
  - Valeurs des paramètres
  - Valeur du pas d'apprentissage
  - Coûts (d'apprentissage, de validation, …)
  - Gradients
  - …

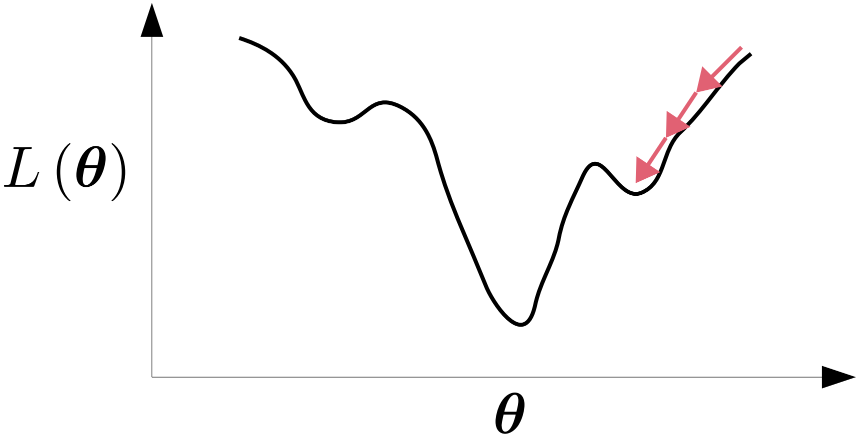
Définir la valeur du pas d'apprentissage

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \alpha \sum_{i \in \Omega_k} \frac{\partial l\left(\mathtt{Y}_{\text{train},i}, f\left(\mathtt{X}_{\text{train},i}; \boldsymbol{\theta}\right)\right)}{\partial \boldsymbol{\theta}}\bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k}$$

Pas d'apprentissage



Pas d'apprentissage trop grand



Pas d'apprentissage trop petit

Définir la valeur du pas d'apprentissage (suite)

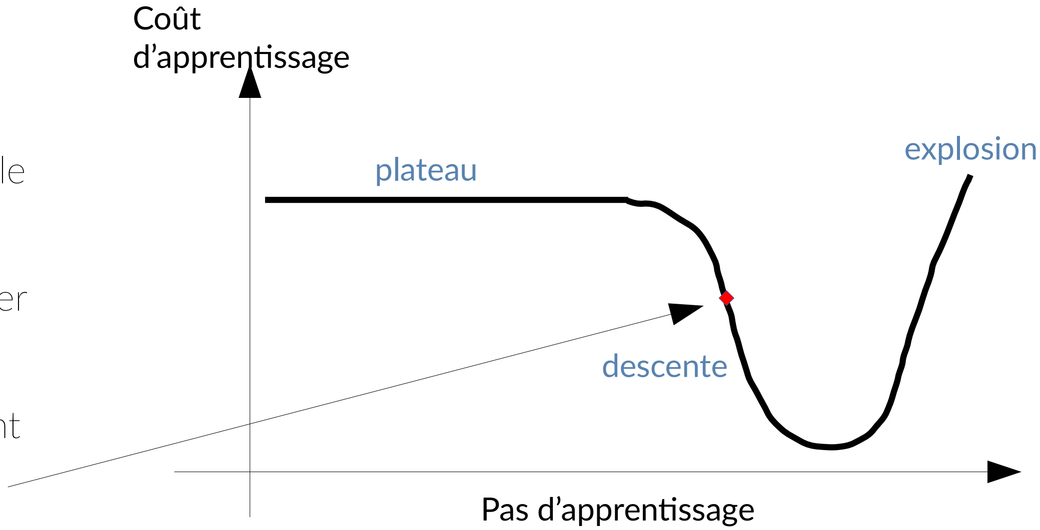
Solution 1 (la plus utilisée) : Tester différentes valeurs du pas d'apprentissage (« grid search ») en visualisant à chaque fois l'évolution du coût d'apprentissage (et du coût de validation)

Définir la valeur du pas d'apprentissage (suite)

Solution 1 (la plus utilisée) : Tester différentes valeurs du pas d'apprentissage (« grid search ») en visualisant à chaque fois l'évolution du coût d'apprentissage (et du coût de validation)

Solution 2 (rarement utilisée) :

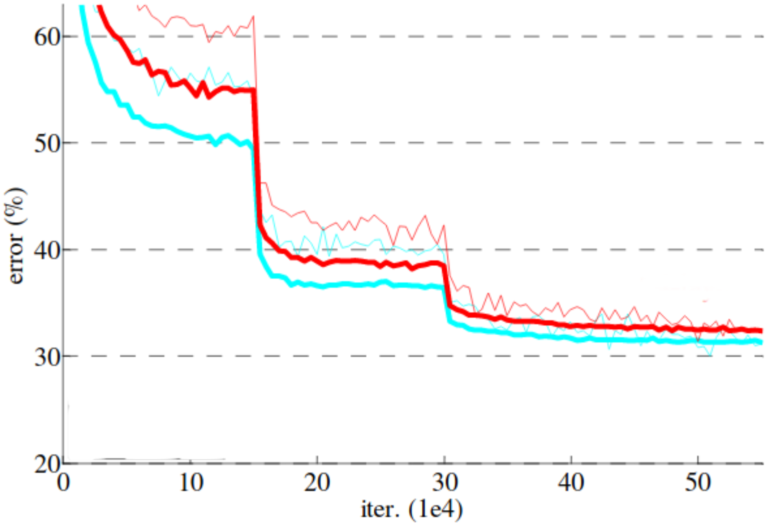
- Lancer un entraînement en partant d'un pas très faible (e.g. 1e-7).
- A chaque itération (i.e à chaque minibatch), augmenter le pas.
- Récupérer la valeur du pas correspondant au gradient le plus négatif.

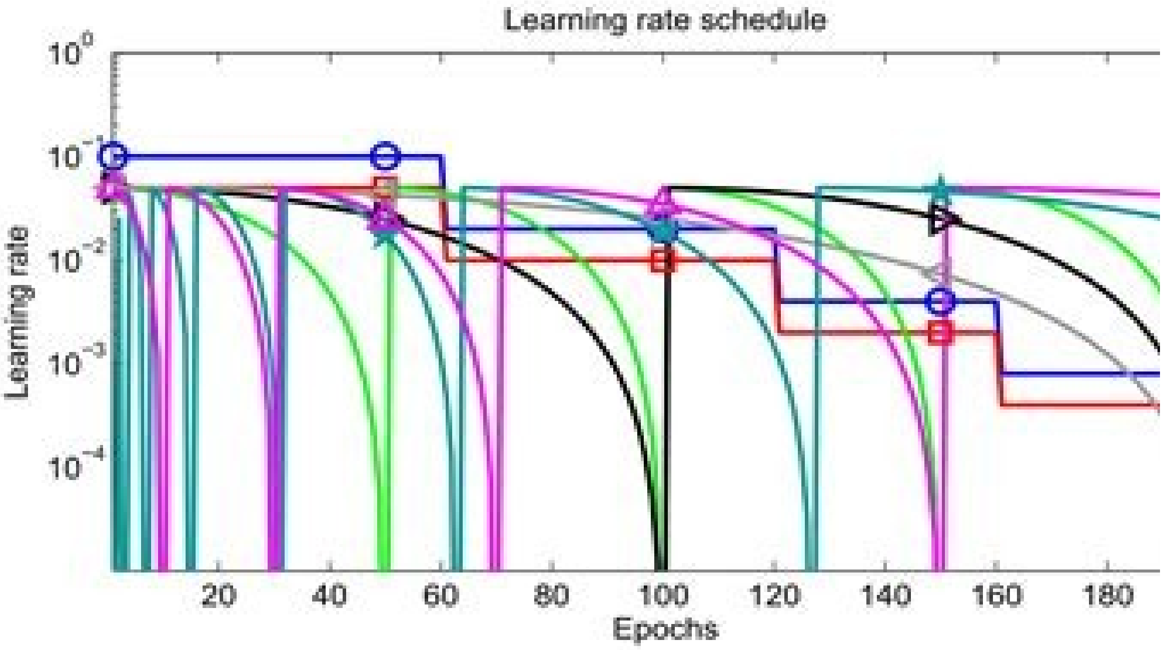


Smith, L. N. "Cyclical learning rates for training neural networks.", WACV 2017

Evolution du pas d’apprentissage durant l’optimisation

- Constant
- Décroissant
- Cyclique
- Réduction sur plateau





Loshchilov, I., & Hutter, F. "SGDR: Stochastic gradient descent with warm restarts." 2016

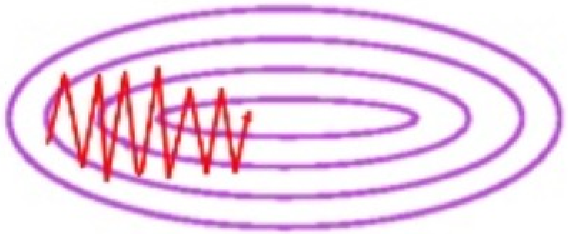
VI)

SGD avec moment (« SGD with momentum »)

SGD

$$\mathbf{g}_{k+1} = \sum_{i \in \Omega_k} \frac{\partial l\left(\mathtt{Y}_{\mathrm{train},i}, f\left(\mathtt{X}_{\mathrm{train},i}; \boldsymbol{\theta}\right)\right)}{\partial \boldsymbol{\theta}}\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k}$$

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \alpha \mathbf{g}_{k+1}$$



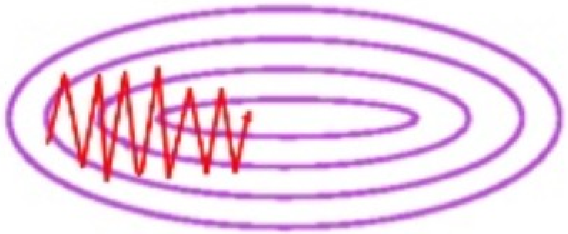
Source : https://image.slidesharecdn.com/

SGD avec moment (« SGD with momentum »)

SGD

$$\mathbf{g}_{k+1} = \sum_{i \in \Omega_k} \frac{\partial l\left(\mathtt{Y}_{\mathrm{train},i}, f\left(\mathtt{X}_{\mathrm{train},i}; \boldsymbol{\theta}\right)\right)}{\partial \boldsymbol{\theta}}\Bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k}$$

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \alpha \mathbf{g}_{k+1}$$

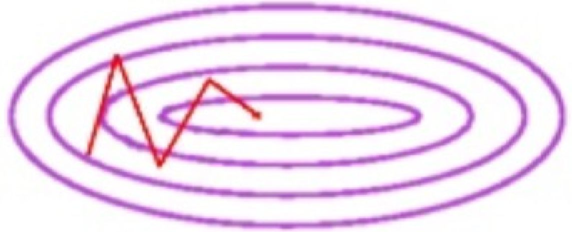


SGD avec moment

$$\mathbf{g}_{k+1} = \sum_{i \in \Omega_k} \frac{\partial l\left(\mathtt{Y}_{\mathrm{train},i}, f\left(\mathtt{X}_{\mathrm{train},i}; \boldsymbol{\theta}\right)\right)}{\partial \boldsymbol{\theta}}\Bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k}$$

$$\mathbf{m}_{k+1} = \beta \mathbf{m}_k + (1-\beta)\,\mathbf{g}_{k+1}$$

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \alpha \mathbf{m}_{k+1}$$



Source : https://image.slidesharecdn.com/

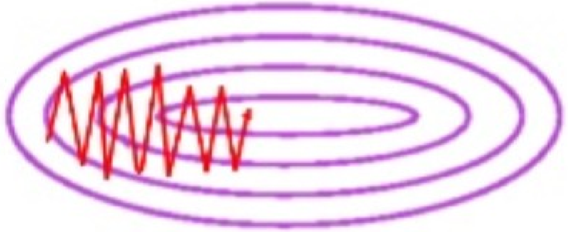
VI)

SGD avec moment (« SGD with momentum »)

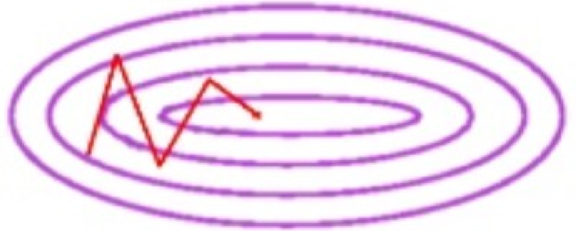
SGD

$$\mathbf{g}_{k+1} = \sum_{i \in \Omega_k} \frac{\partial l\left(\mathtt{Y}_{\mathrm{train},i}, f\left(\mathtt{X}_{\mathrm{train},i}; \boldsymbol{\theta}\right)\right)}{\partial \boldsymbol{\theta}}\Bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k}$$

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \alpha \mathbf{g}_{k+1}$$



SGD avec moment



$$\mathbf{g}_{k+1} = \sum_{i \in \Omega_k} \frac{\partial l\left(\mathtt{Y}_{\mathrm{train},i}, f\left(\mathtt{X}_{\mathrm{train},i}; \boldsymbol{\theta}\right)\right)}{\partial \boldsymbol{\theta}}\Bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k}$$

$$\mathbf{m}_{k+1} = \beta \mathbf{m}_k + (1-\beta)\, \mathbf{g}_{k+1}$$

Moyenne mobile exponentielle du gradient (permet de « lisser » le gradient)

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \alpha \mathbf{m}_{k+1}$$

Source : https://image.slidesharecdn.com/

VI)

« Adam: A Method for Stochastic Optimization »

$$\mathbf{g}_{k+1} = \sum_{i \in \Omega_k} \frac{\partial l \left(\mathtt{Y}_{\text{train},i}, f\left(\mathtt{X}_{\text{train},i}; \boldsymbol{\theta} \right) \right)}{\partial \boldsymbol{\theta}} \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k}$$

$$\mathbf{m}_{k+1} = \frac{1}{1-\beta_1^k} \left(\beta_1 \mathbf{m}_k + (1-\beta_1)\, \mathbf{g}_{k+1} \right)$$

$$\mathbf{v}_{k+1} = \frac{1}{1-\beta_2^k} \left(\beta_2 \mathbf{v}_k + (1-\beta_2)\, \mathbf{g}_{k+1}^2 \right)$$

Carré de chaque élément de $\mathbf{g}_k$

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \alpha \frac{\mathbf{m}_{k+1}}{\sqrt{\mathbf{v}_{k+1}} + \epsilon}$$

Racine carrée de chaque élément de $\mathbf{v}_{k+1}$

Kingma, D. P., & Ba, J. L. (2015). Adam: A method for stochastic gradient descent. In ICLR: International Conference on Learning Representations.

Autres techniques de régularisation

Régularisation : technique permettant de réduire le sur-apprentissage

Exemple déjà vu → « Early Stopping »

Autres techniques de régularisation

Régularisation : technique permettant de réduire le sur-apprentissage

Exemple déjà vu → « Early Stopping »

« Dropout » d'une couche FC

Lors de l'entraînement, mettre aléatoirement p % des colonnes de W à zéro
(équivaut à mettre à zéro aléatoirement p % des « neurones » d'entrée)

Autres techniques de régularisation

Régularisation : technique permettant de réduire le sur-apprentissage

Exemple déjà vu $\rightarrow$ « Early Stopping »

« Dropout » d'une couche FC

Lors de l'entraînement, mettre aléatoirement p % des colonnes de W à zéro
(équivaut à mettre à zéro aléatoirement p % des « neurones » d'entrée)

« Weight decay »

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \alpha \frac{\partial L(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}_k} - \lambda \boldsymbol{\theta}_k$$

Tire les paramètres vers zéro

$\rightarrow$ AdamW

Loshchilov, I., Hutter, F. (2019). Decoupled Weight Decay Regularization. In ICLR: International Conference on Learning Representations.

A priori un tel apprentissage ne devrait **PAS** fonctionner

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} L_{\text{train}}(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{N_{\text{train}}} l\left(\mathtt{Y}_{\text{train},i}, f\left(\mathtt{X}_{\text{train},i}; \boldsymbol{\theta}\right)\right)$$

où $$f(\mathbf{x};\boldsymbol{\theta}) = f_L\left(f_{L-1}\left(\ldots f_2\left(f_1\left(\mathbf{x};\boldsymbol{\theta}^{(1)}\right);\boldsymbol{\theta}^{(2)}\right)\ldots;\boldsymbol{\theta}^{(L-1)}\right);\boldsymbol{\theta}^{(L)}\right)$$

Raisonnement a priori

descente de gradient où
f est non-convexe

↓

mauvais minimum local

A priori un tel apprentissage ne devrait **PAS** fonctionner

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} L_{\text{train}}(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{N_{\text{train}}} l\left(\mathtt{Y}_{\text{train},i}, f\left(\mathtt{X}_{\text{train},i}; \boldsymbol{\theta}\right)\right)$$

où $$f(\mathbf{x}; \boldsymbol{\theta}) = f_L\left(f_{L-1}\left(\ldots f_2\left(f_1\left(\mathbf{x}; \boldsymbol{\theta}^{(1)}\right); \boldsymbol{\theta}^{(2)}\right) \ldots; \boldsymbol{\theta}^{(L-1)}\right); \boldsymbol{\theta}^{(L)}\right)$$

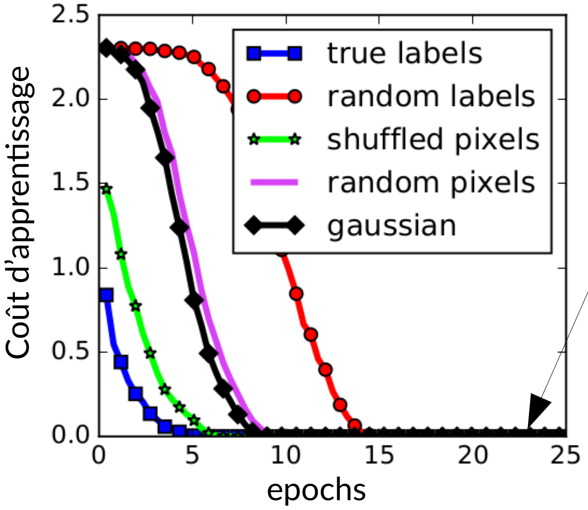
Raisonnement a priori

descente de gradient où f est non-convexe

↓

mauvais minimum local

Résultats empiriques sur CIFAR10



Zéro !

Un des minima globaux a été atteint.

Zhang. C et al. (2017). Understanding Deep Learning requires rethinking generalization. ICLR

A priori un tel apprentissage ne devrait **PAS** fonctionner (suite)

Raisonnement a priori

Coût d'apprentissage atteint zéro

↓

Le réseau a appris « par cœur » à associer la bonne étiquette pour chaque exemple de la base d'apprentissage

↓

Les performances de généralisation seront très mauvaises

VI)

A priori un tel apprentissage ne devrait **PAS** fonctionner (suite)

Raisonnement a priori

Coût d'apprentissage atteint zéro

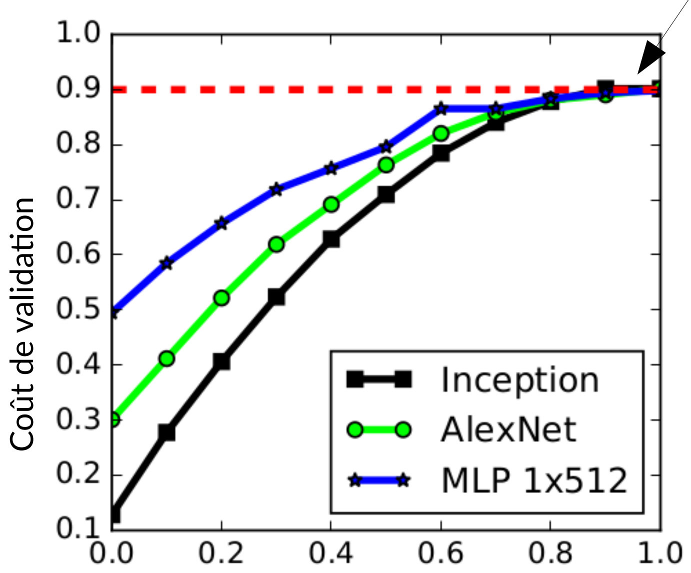
↓

Le réseau a appris « par cœur » à associer la bonne étiquette pour chaque exemple de la base d'apprentissage

↓

Les performances de généralisation seront très mauvaises

Résultats empiriques sur CIFAR10



Taux de fausses étiquettes dans la base d'apprentissage

Apprentissage avec de fausses étiquettes

CIFAR10 → 10 classes

Résultat attendu : Le réseau a appris à partir de fausses étiquettes donc il obtient de mauvaises performances quand on lui présente des exemples avec les vraies étiquettes.

Zhang. C et al. (2017). Understanding Deep Learning requires rethinking generalization. ICLR

A priori un tel apprentissage ne devrait **PAS** fonctionner (suite)

Raisonnement a priori

Coût d'apprentissage atteint zéro

↓

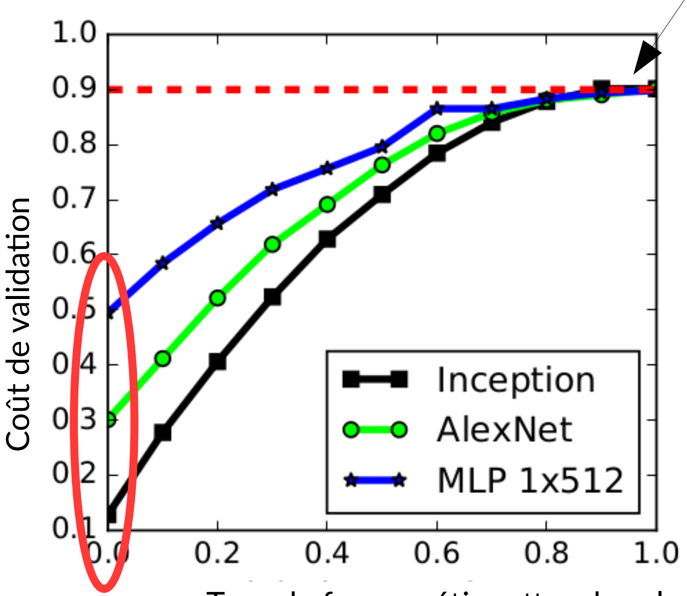
Le réseau a appris « par cœur » à associer la bonne étiquette pour chaque exemple de la base d'apprentissage

↓

Les performances de généralisation seront très mauvaises

Zhang. C et al. (2017). Understanding Deep Learning requires rethinking generalization. ICLR

Résultats empiriques sur CIFAR10



Taux de fausses étiquettes dans la base d'apprentissage

Apprentissage avec de fausses étiquettes

CIFAR10 → 10 classes

Résultat attendu : Le réseau a appris à partir de fausses étiquettes donc il obtient de mauvaises performances quand on lui présente des exemples avec les vraies étiquettes.

Apprentissage avec les vraies étiquettes

→ Résultat inattendu : Quand le réseau apprend à partir de vraies étiquettes, il a une bonne capacité de généralisation.

A priori un tel apprentissage ne devrait **PAS** fonctionner (suite)

Comment se fait-il que l'étape d'apprentissage ait permis de trouver un minimum global qui généralise bien (sachant qu'il existe des minima globaux qui généralisent mal) ?

1) Biais introduit par l'architecture (« inductive bias »)

2) Biais introduit par la descente de gradient stochastique

https://guillefix.me/nnbias/

https://hackmd.io/75gt3X6WQbu1_A3pF8svWg

Valle-Pérez. G et al. (2019). Deep learning generalizes because the parameter-function map is biased towards simple functions. ICLR

Smith, S., et al. (2021). On the origin of implicit regularization on stochastic gradient descent. ICLR

Résumé des ingrédients du « Deep Learning »

1) Grande base de données étiquetées

2) « Bonne » architecture de réseau de neurones profond

- ► « Perceptron » multicouche, Réseau de neurones à convolution, Transformer
- ► **Et optimisation par descente de gradient stochastique (AdamW, etc.)**

3) Grande capacité de calculs en parallèle (GPU)