OPTIMISATION

TMI

Intérêt des méthodes numériques

Solutions directes rares

- Estimation = minimisation d'un critère
- Fonctions complexes, pas d'expression analytique pour le minimum
- Attention : gradient nul est nécessaire mais pas suffisant

Plusieurs difficultés

- Recherche d'optimum local
- Recherche d'optimum global
- Dimension du problème
- Propriétés du critère (dérivable, convexe,…)
- Espace de recherche du paramètre
  - discret → optimisation combinatoire
  - Contraint → optimisation sous contrainte

DIFFÉRENTS PROFILS DE CRITÈRES

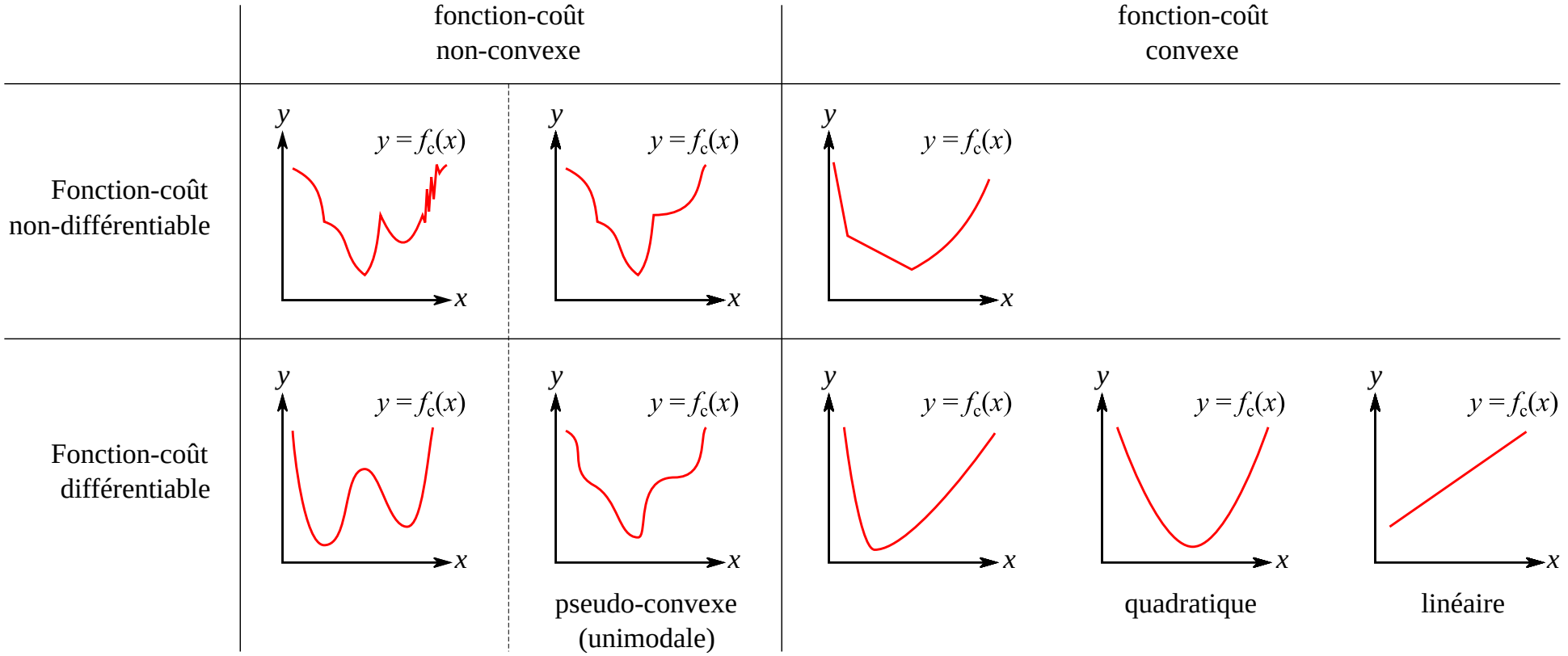


Image issue de Wikipédia, page Optimisation

Plan du cours

Quelques méthodes d'optimisation

- Méthode sans dérivée explicite
- Méthode de Newton
- Descente de gradient
- Gradient stochastique

Calcul d'une dérivée

- Méthode directe
- Approximation numérique
- Différentiation automatique
- Application à la rétropropagation de gradient

Optimisation pratique

- Paramétrisation, pondération des coûts, équilibrage des données
- Intégration de contraintes

Plan du cours

Quelques méthodes d'optimisation

- Méthode sans dérivée explicite
- Méthode de Newton
- Descente de gradient
- Gradient stochastique

Calcul d'une dérivée

- Méthode directe
- Approximation numérique
- Différentiation automatique
- Application à la rétropropagation de gradient

Optimisation pratique

- Paramétrisation, pondération des coûts, équilibrage des données
- Intégration de contraintes

QUELQUES CONSIDÉRATIONS

- Max f(x)=min -f(x) : maximiser ou minimiser, même problème → optimiser
- Si x est un optimum local de f, alors
  - f'(x) = 0 (gradient si dim > 1)
  - f''(x) > 0 si minimum, f''(x) < 0 si maximum (valeurs propres de la hessienne si dim > 1)
- Résoudre f'(x) = 0 n'est souvent pas facile, minimiser l'est car il existe des algorithmes efficaces et assez génériques
- On peut minimiser une fonction sans connaître sa dérivée.
- Plus on connaît d'ordre de dérivées, plus la minimisation peut être efficace (forme locale de la fonction plus précise)
- Dans la plupart des cas, on n'a que très peu d'idée de la forme du critère : il faut essayer plusieurs algorithmes
- Matlab : toolbox Optimisation ; python : scipy.optimize

BRACKETTING

Principe

On traque la valeur en l'encadrant par a et b et l'approchant par c

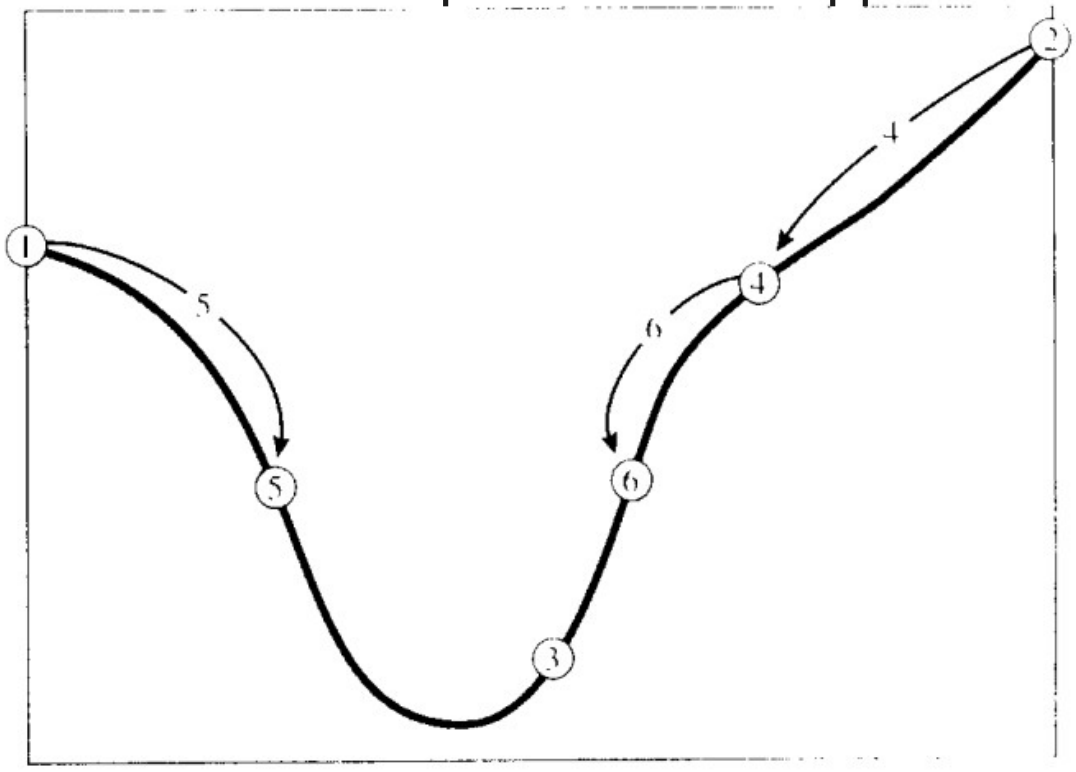


Figure 10.1.1. Successive bracketing of a minimum. The minimum is originally bracketed by points 1,3,2. The function is evaluated at 4, which replaces 2; then at 5, which replaces 1; then at 6, which replaces 4. The rule at each stage is to keep a center point that is lower than the two outside points. After the steps shown, the minimum is bracketed by points 5,3,6.

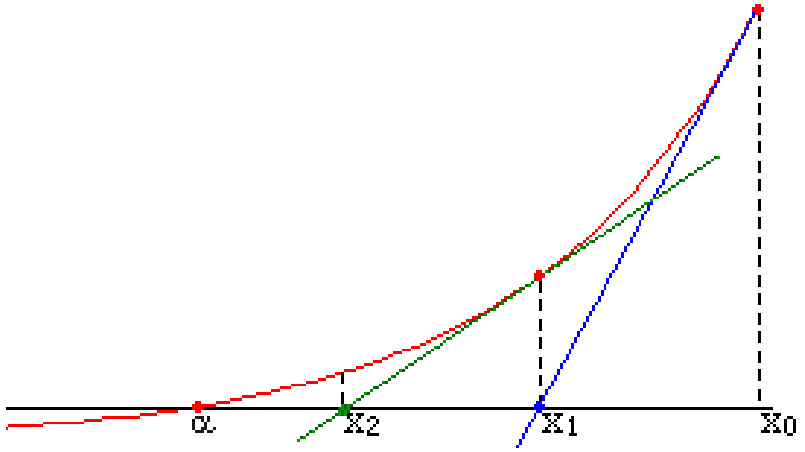
Méthode de Newton

Objectif

Résoudre une équation du type f(x) = 0

Hypothèse

f est dérivable et on connaît f'



Méthode

On approxime la fonction par sa tangente au point courant x_k

On résout f(x) = 0 avec cette approximation → nouveau point x_{k+1}

Formule itérative (x_0 choisi)

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \qquad \text{arrêt quand } |f(x_k)| < \epsilon$$

Méthode de Newton

Point de vue non géométrique

Développement de Taylor en x_k (ordre 1)

$$f(x_{k+1}) = f(x_k) + f'(x_k)(x_{k+1} - x_k) = 0 \quad \Rightarrow \quad x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

Note

Nécessité d'avoir un estimé initial x_0

En pratique : besoin de la dérivée seconde (zéro de la dérivée)

Se généralise aux dimensions supérieures :

- Inversion de la matrice jacobienne
- Efficace via résolution de systèmes d'équations linéaires

MÉTHODE DE NEWTON

Cas d'échec

$$f(x)=x^3-2x+2 \qquad f'(x)=3x^2-2$$

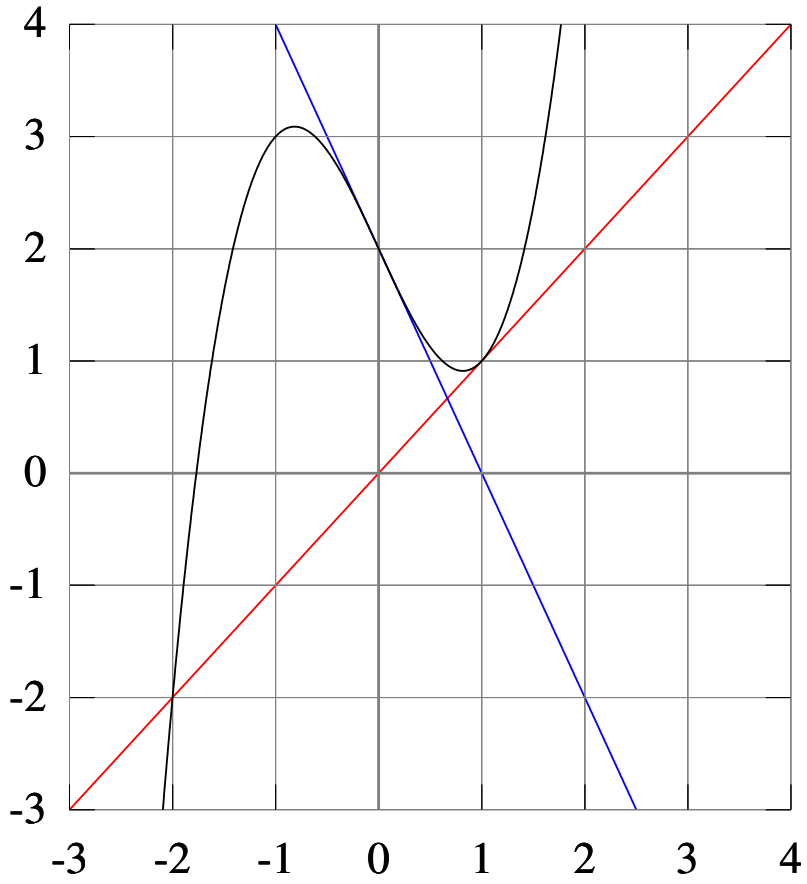
$$x_0=0 \Rightarrow f(0)=2, f'(0)=-2, x_1=1$$

$$x_1=1 \Rightarrow f(1)=1, f'(1)=1, x_2=0$$

Adaptation

Utilisation d'un pas

$$x_{k+1}=x_k-\lambda\frac{f(x_k)}{f'(x_k)}$$



DIMENSIONS SUPÉRIEURES : RELAXATION

Principe

Se ramener à la minimisation d'une fonction réelle à une variable

Pour cela on choisit une direction d_k

On trouve t_0 tel que $t_0 = \underset{t}{\operatorname{argmin}} f(x_k + t\, d_k)$

On met à jour $x_{k+1} = x_k + t_0 d_k$

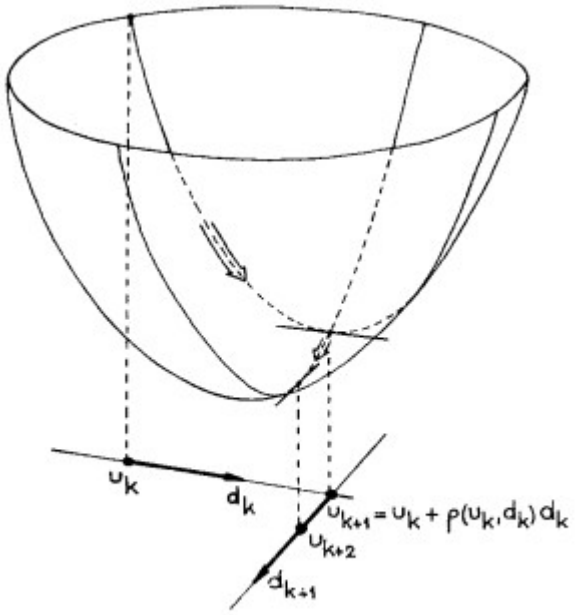


FIG. 8.4-1.

CHOIX DE LA DIRECTION

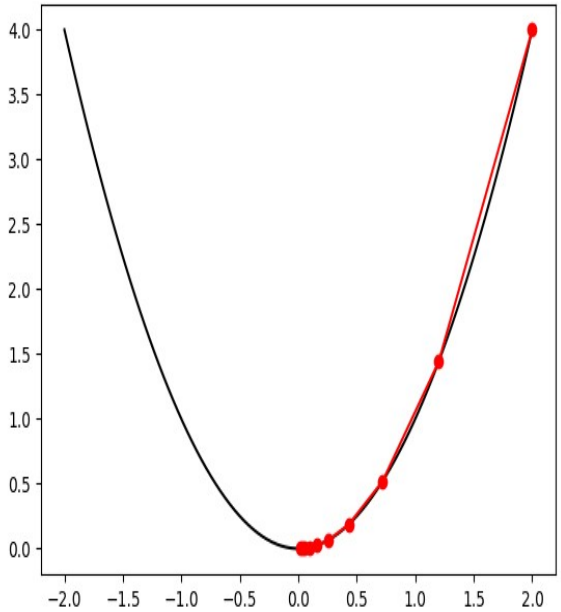
Plusieurs options

- On prend la direction d'une coordonnée → *coordinate descent*
  - Plusieurs stratégies possibles pour l'ordre de revue des coordonnées
  - Simple, mais peu efficace
- La direction du gradient (plus forte pente) → *(steepest) gradient descent*
  - Utilisation d'un pas $x_{k+1} = x_k - \lambda \nabla f(x_k)$
  - Pas d'apprentissage (*learning rate*) → stratégie pour le pas

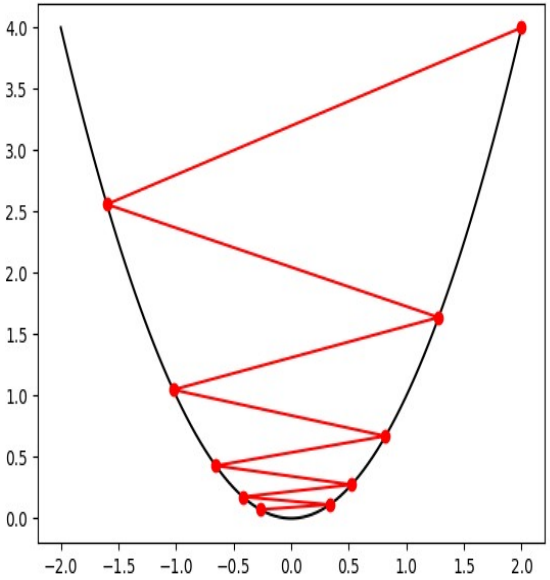
CHOISIR LE PAS

Fonction $f(x) = x^2$, pas constant

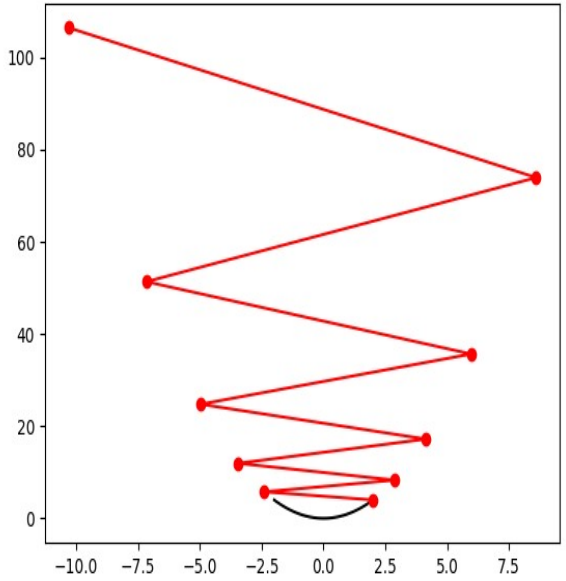
λ=0.2



λ=0.9



λ=1.1



Momentum

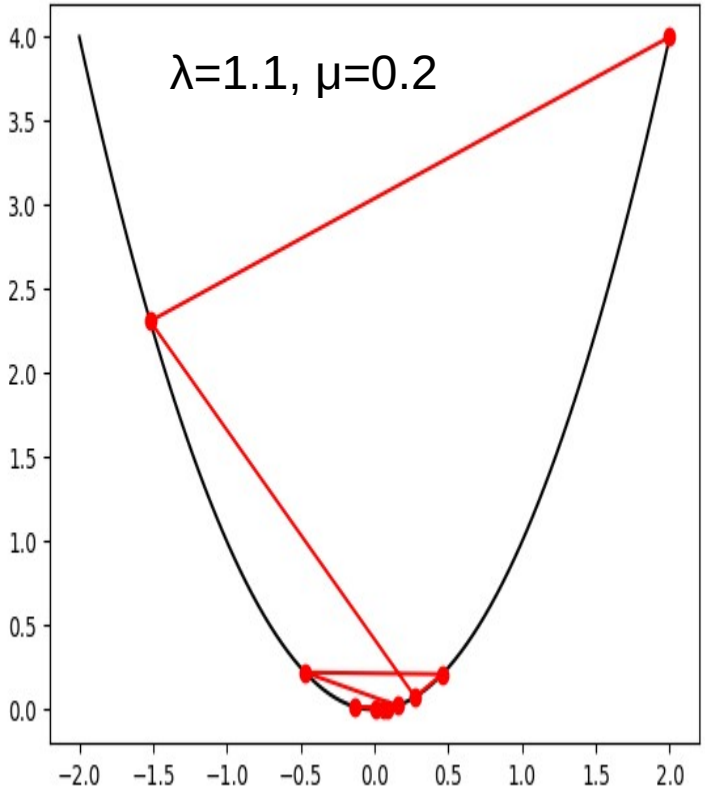
Objectif

Limiter l'influence d'un gradient particulièrement et localement fort

Souvent en cas de bruit

Garder une inertie de la dernière direction prise

$$d_{k+1} = (1-\mu)\nabla f(x_k) + \mu d_k$$



Méthode de Powell

Principe

Maintenir un ensemble de N directions

Méthode

N directions initiales

Recherche d'un minimum suivant chaque direction successivement

$x_k \rightarrow x_k+a_0d_0 \rightarrow x_k+a_0d_0+a_1d_1 \rightarrow \ldots x_k+a_0d_0+\ldots+a_{N-1}d_{N-1}$

Nouvelle direction $a_0d_0+\ldots+a_{N-1}d_{N-1}$

Ajout de cette nouvelle direction et retrait de celle avec a_i max

On itère

GRADIENT STOCHASTIQUE

Apprentissage supervisé

Critère = somme de valeurs du critère calculées pour chaque donnée

Ex : critère des moindres carrés

$$C(\theta) = \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i ; \theta))^2 = \frac{1}{n} \sum_{i=1}^{n} C_i(\theta)$$

Conséquence : lent !

Il faut passer en revue toute la base d'apprentissage pour obtenir une seule mise à jour des paramètres → optimisation par batch, une seule mise à jour du réseau par époque

Descente selon la moyenne des gradients sur chaque donnée

Principe du gradient stochastique

Mise à jour pour chaque donnée d'apprentissage

Le gradient de chaque C_i est une direction de descente successive

GRADIENT STOCHASTIQUE : CRITIQUE

Avantages

plus rapide, on exploite la redondance éventuelle dans les données

Gradient approximatif : on échappe plus facilement à des petits minima locaux

Inconvénients

gradient bruité, attention aux données aberrantes

Solution

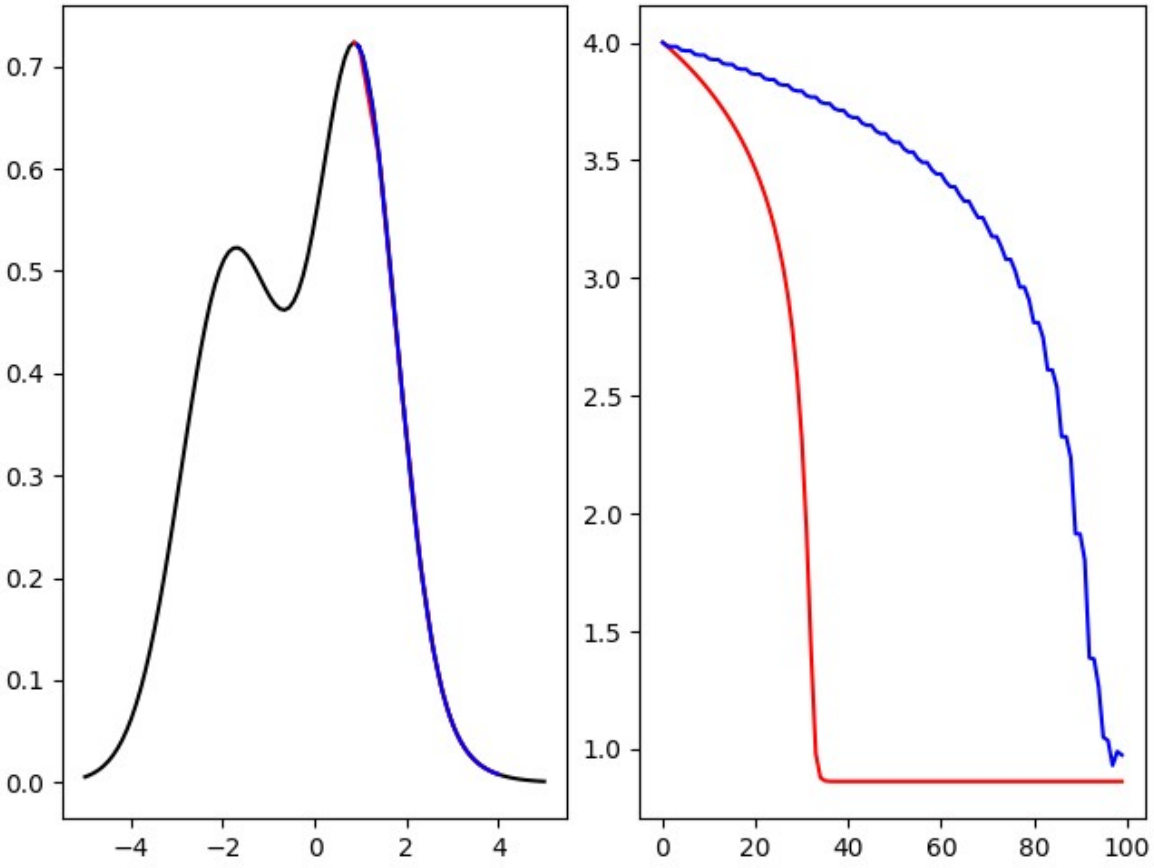
Minibatch : sous-ensemble des données (entre 1 et N=batch)

Direction = moyenne du gradient sur le minibatch

Époque = ensemble des minibatches, couvre toutes les données, minibatches non intersectants (minibatches = partition du batch)

GRADIENT STOCHASTIQUE : EXEMPLE

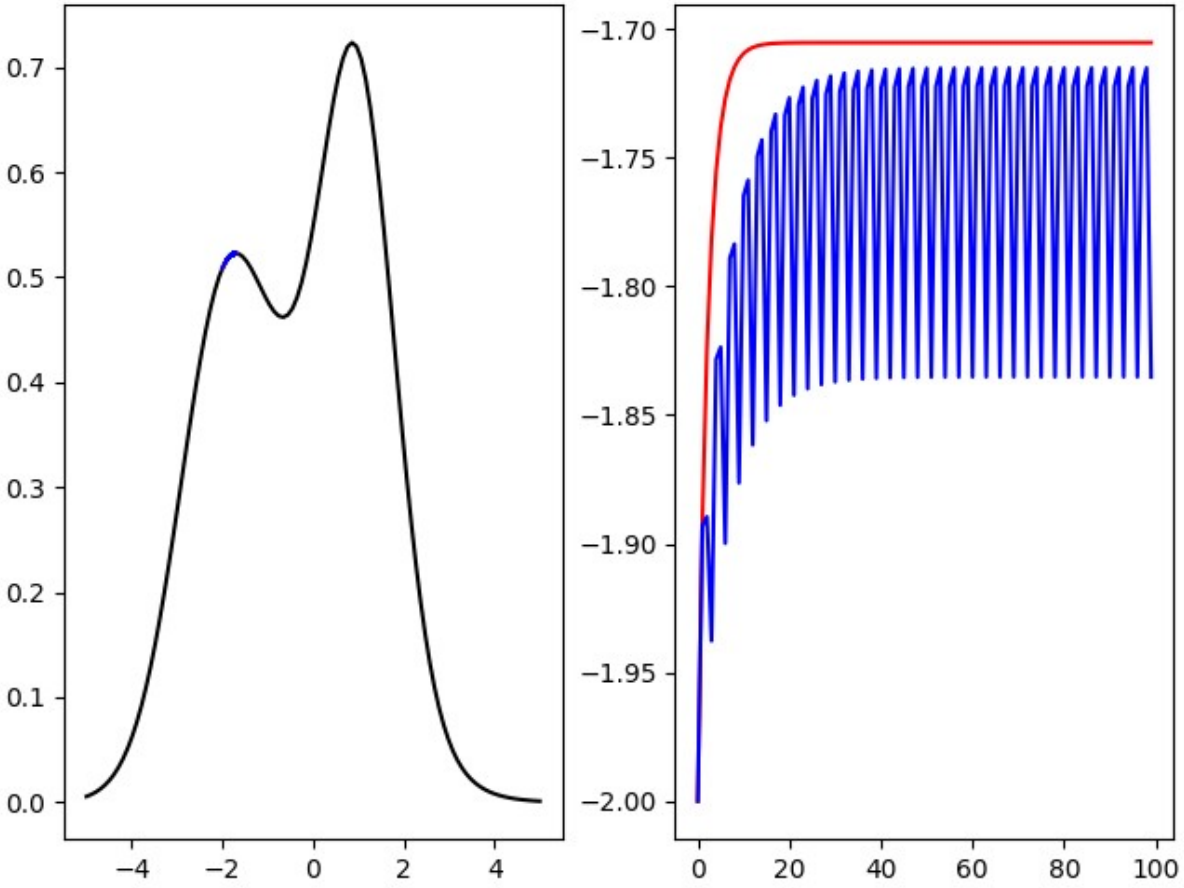
$f = f_0 + f_1 + f_2$ avec $f_0 = N(-2,1)$, $f_1 = N(0,1.5)$, $f_2 = N(1,0.8)$



x_0=4
Rouge : GD
Bleu : SGD
Gauche : f(x) et trajectoires
Droite : x_k (k en abscisse)

GRADIENT STOCHASTIQUE : EXEMPLE

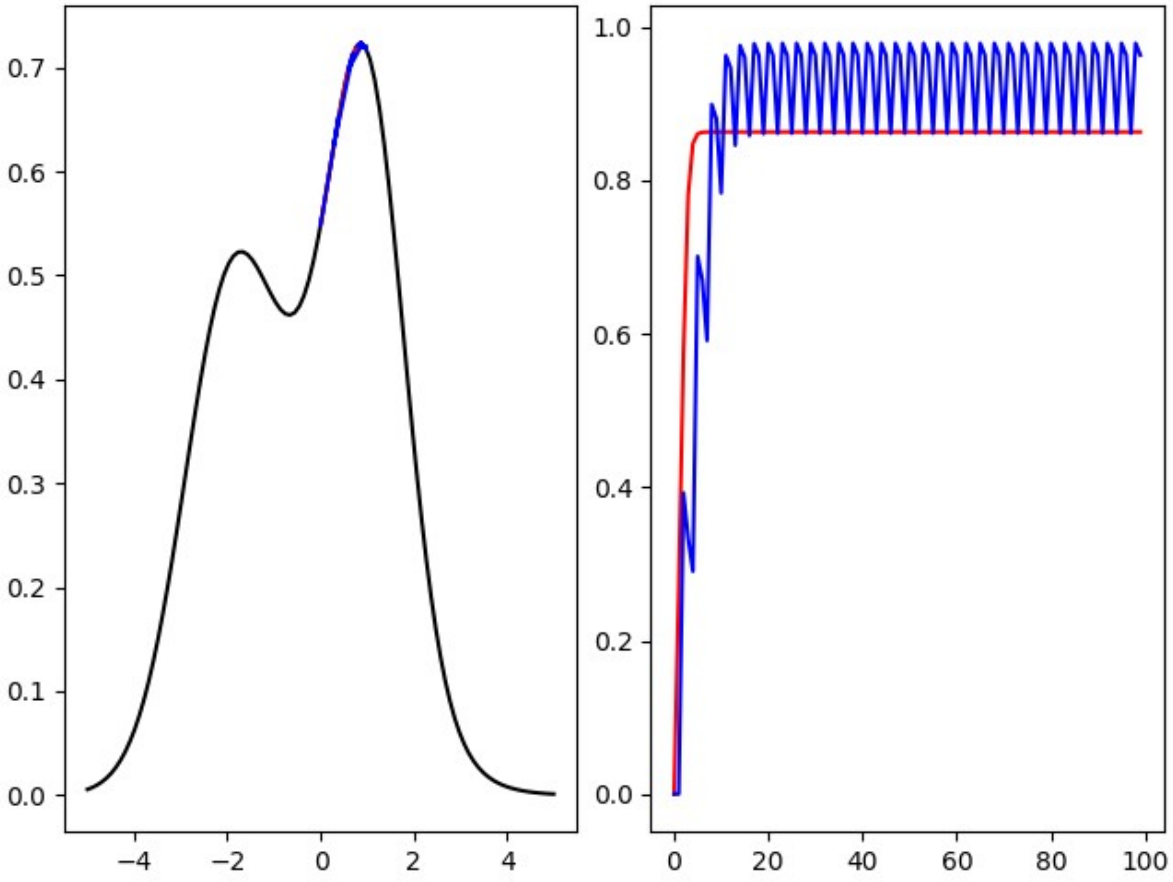
$f = f_0 + f_1 + f_2$ avec $f_0 = N(-2,1)$, $f_1 = N(0,1.5)$, $f_2 = N(1,0.8)$



x_0=-2
Rouge : GD
Bleu : SGD
Gauche : f(x) et trajectoires
Droite : x_k (k en abscisse)

GRADIENT STOCHASTIQUE : EXEMPLE

$f = f_0 + f_1 + f_2$ avec $f_0 = N(-2,1)$, $f_1 = N(0,1.5)$, $f_2 = N(1,0.8)$



$x_0=0$
Rouge : GD
Bleu : SGD
Gauche : f(x) et trajectoires
Droite : x_k (k en abscisse)

Plan du cours

Quelques méthodes d'optimisation

- Méthode sans dérivée explicite
- Méthode de Newton
- Descente de gradient
- Gradient stochastique

Calcul d'une dérivée

- Méthode directe
- Approximation numérique
- Différentiation automatique
- Application à la rétropropagation de gradient

Optimisation pratique

- Paramétrisation, pondération des coûts, équilibrage des données
- Intégration de contraintes

Calcul direct ou symbolique

Calcul direct

- Fonctions assez simples
- Requis par plusieurs algo d'optimisation (basés gradient voire hessien)
  - Cf scipy.optimize.minize(),et ses options jac, hess et hessp

Calcul symbolique

- Utilisation d'un logiciel de calcul symbolique (Maple,Sage,Mathematica,sympy)
- Traduction directe sous forme de fonction informatique
- Potentiellement expressions très complexes et moins efficaces qu'un calcul approché
- Limites si fonction trop complexe (par exemple définition par morceaux ou par cas (usage de if...))

CALCUL APPROCHÉ

Différences finies

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Différents schémas possibles

Forward difference $f'(x) \approx \frac{f(x+h) - f(x)}{h}$

Backward difference $f'(x) \approx \frac{f(x) - f(x-h)}{h}$

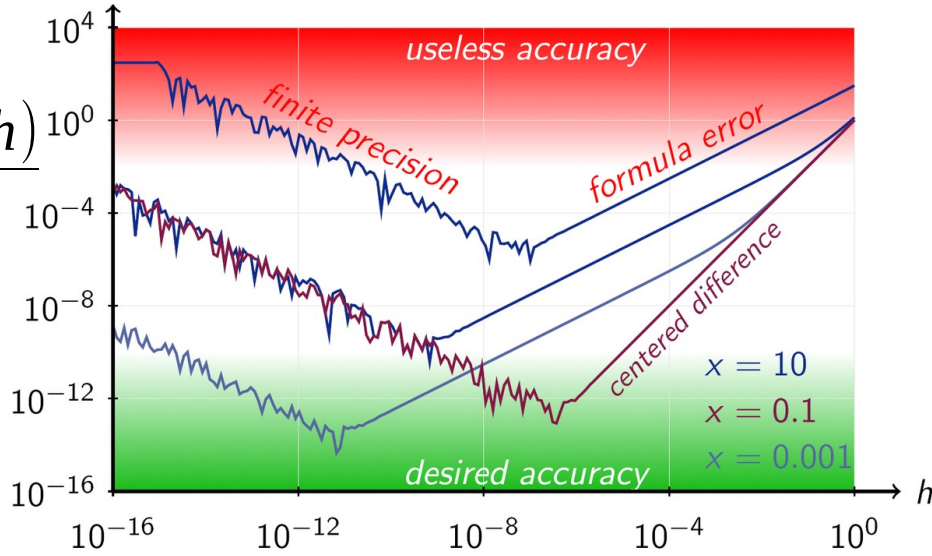
Central difference $f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$

Une infinité de formules...

Principale difficulté : choix de h

Le plus petit possible

Mais si trop petit → erreurs d'arrondis



Source : Wikipedia, Numerical differentiation

DIFFÉRENTIATION AUTOMATIQUE (*AUTOGRAD*)

Principe

Chaque fonction (ex fonction de coût) se programme à partir de fonctions (ex sin, exp) et opérateurs de base (ex +, *).

Dérivée connue explicitement pour chaque fonction/opérateur de base

Exploitation de la chaîne de dérivation (*chain rule*)

$$(f \circ g(x))' = (f(g(x)))' = f'(g(x))g'(x) \quad \text{soit} \quad \frac{\partial f}{\partial x} = \frac{\partial f}{\partial w}\frac{\partial w}{\partial x} \text{ avec } w = g(x)$$

En pratique

Code de calcul de la fonction et de sa dérivée

Chaque appel de la fonction appelle aussi celui de sa dérivée

Calcul de la valeur de la fonction et de celle de sa dérivée en un point (pas de formule explicite)

Exemple de DA

$$f(x)=\sin(\exp(x))^2$$

Décomposition

$$w_0=x \qquad \frac{\partial w_0}{\partial x}=1$$

$$w_1=\exp(w_0) \qquad \frac{\partial w_1}{\partial w_0}=\exp(w_0)$$

$$w_2=\sin(w_1) \qquad \frac{\partial w_2}{\partial w_1}=\cos(w_1)$$

$$w_3=w_2^2 \qquad \frac{\partial w_3}{\partial w_2}=2*w_2$$

Dérivée

$$f'(x)=\frac{\partial w_3}{\partial w_2}\frac{\partial w_2}{\partial w_1}\frac{\partial w_1}{\partial w_0}\frac{\partial w_0}{\partial x}$$

Exemple de DA

$$f(x)=\sin(\exp(x))^2$$

Calcul en x=0

$$w_0=0 \qquad \frac{\partial w_0}{\partial x}=1$$

$$w_1=\exp(0)=1 \qquad \frac{\partial w_1}{\partial w_0}=\exp(0)=1$$

$$w_2=\sin(1) \qquad \frac{\partial w_2}{\partial w_1}=\cos(1)$$

$$w_3=\sin^2(1) \qquad \frac{\partial w_3}{\partial w_2}=2\sin(1)$$

Dérivée

$$f'(0)=\frac{\partial w_3}{\partial w_2}\frac{\partial w_2}{\partial w_1}\frac{\partial w_1}{\partial w_0}\frac{\partial w_0}{\partial x}=2\sin(1)*\cos(1)*1*1=2\sin(1)\cos(1)$$

2 MODES DE CALCUL

Accumulation directe (mode avant)

On calcule la dérivée dès qu'on calcule la valeur de la fonction

Début $\frac{\partial x}{\partial x}=1$, puis calcul de $\frac{\partial w_{i+1}}{\partial x}=\frac{\partial w_{i+1}}{\partial w_i}\frac{\partial w_i}{\partial x}$

Calcul de toutes les dérivées par rapport à x

Accumulation inverse (mode arrière)

On calcule les valeurs de fonction (passe avant), puis on calcule les valeurs des dérivées (passe arrière)

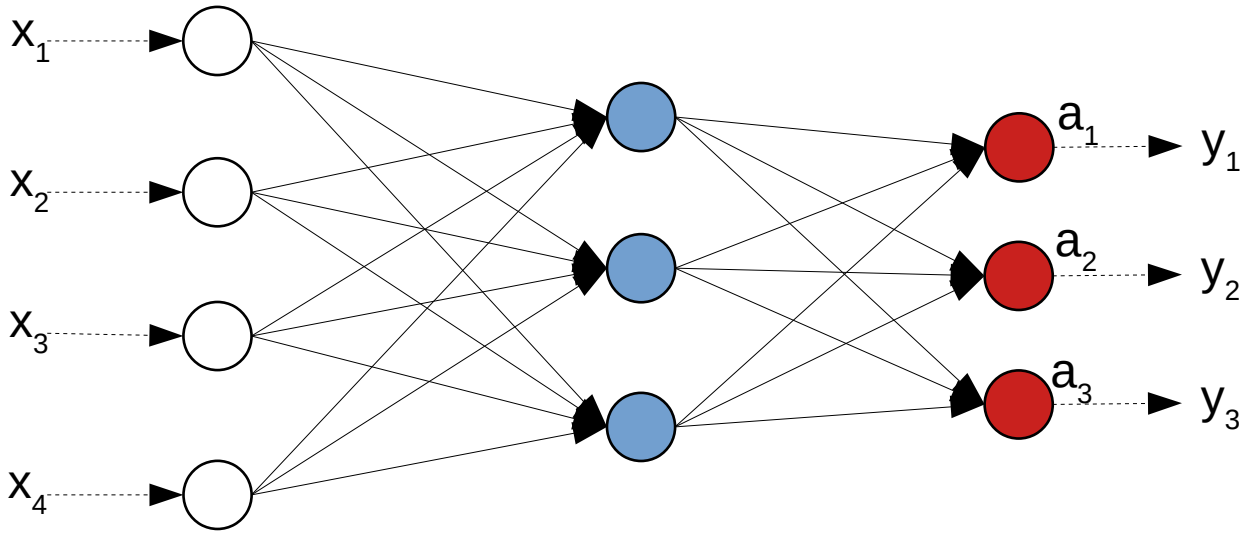
Début $\frac{\partial y}{\partial y}=1$, puis calcul de $\frac{\partial y}{\partial w_i}=\frac{\partial y}{\partial w_{i+1}}\frac{\partial w_{i+1}}{\partial w_i}$

Calcul des dérivées de la fonction par rapport à chaque variable

Intérêt : mise à jour des variables intermédiaires par descente de gradient

Pour aller plus loin : https://www.cs.toronto.edu/~rgrosse/courses/csc2541_2022/tutorials/tut01.pdf

PERCEPTRON MULTICOUCHES, MULTI-CLASSES



Pas de fonction d'activation en sortie, mais une normalisation SoftMax : $y_i = \frac{e^{a_i}}{\sum_k e^{a_k}}$

$w_{i,j}^{\ k}$: poids du neurone i de la couche k vers le neurone j de la couche k+1 (les biais sont notés $w_{0,j}^{\ k}$, et $z_0^{\ k}$ sont tels que $\sigma(z_0^{\ k})$ vaut 1)

$z_i^{\ k}$: signal en entrée du neurone i de la couche k, avant activation : $z_j^{k+1} = \sum_i w_{i,j}^k \sigma(z_i^k)$

$a_i = z_i^N$ où N est le nombre de couches

APPRENTISSAGE

Objectif :

trouver les poids (y compris biais) qui minimisent la fonction de coût L (*loss*)

$L = \sum_n L_n \rightarrow$ un coût pour chaque échantillon de la base d'apprentissage

Descente de gradient

Mise à jour des poids $w_{i,j}^k \leftarrow w_{i,j}^k - \lambda \frac{\partial L}{\partial w_{i,j}^k}(\boldsymbol{w})$

Calcul du gradient global par sommation $\frac{\partial L}{\partial w_{i,j}^k}(\boldsymbol{w}) = \sum_n \frac{\partial L_n}{\partial w_{i,j}^k}(\boldsymbol{w})$

DIFFÉRENTIATION

$$\frac{\partial L_n}{\partial w_{i,j}^k} = \frac{\partial z_j^{k+1}}{\partial w_{i,j}^k} \frac{\partial L_n}{\partial z_j^{k+1}}$$

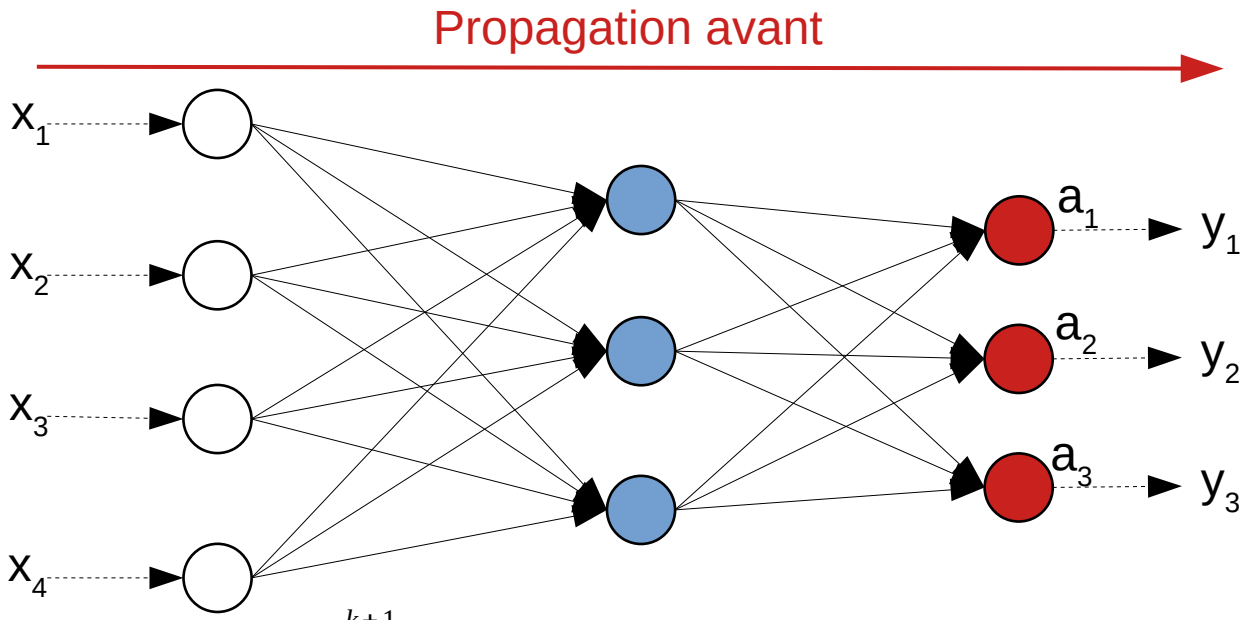
or $$z_j^{k+1} = \sum_i w_{i,j}^k \sigma\left(z_i^k\right) \quad \Rightarrow \quad \frac{\partial z_j^{k+1}}{\partial w_{i,j}^k} = \sigma\left(z_i^k\right)$$

et $$\frac{\partial L_n}{\partial z_j^{k+1}} = \sum_l \frac{\partial z_l^{k+2}}{\partial z_j^{k+1}} \frac{\partial L_n}{\partial z_l^{k+2}}$$

avec $$\frac{\partial z_l^{k+2}}{\partial z_j^{k+1}} = w_{j,l}^{k+1} \frac{\partial \sigma\left(z_j^{k+1}\right)}{\partial z_j^{k+1}} = w_{j,l}^{k+1} \sigma'\left(z_j^{k+1}\right)$$

d'où $$\frac{\partial L_n}{\partial z_j^{k+1}} = \sigma'\left(z_j^{k+1}\right) \sum_l w_{j,l}^{k+1} \frac{\partial L_n}{\partial z_l^{k+2}}$$

RÉTROPROPAGATION



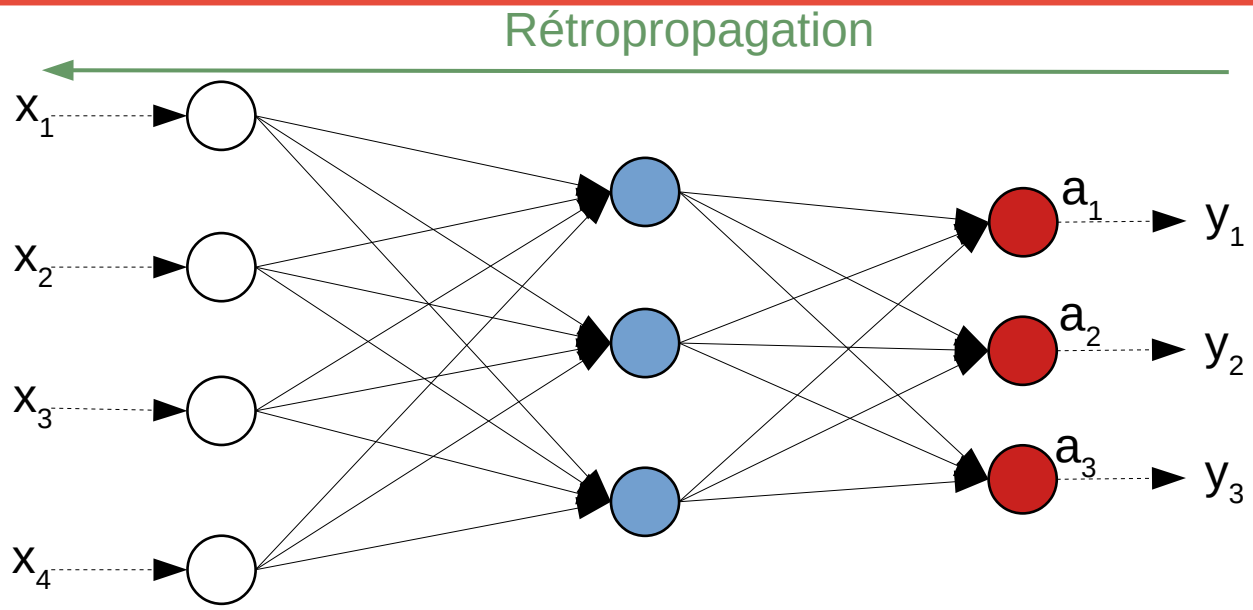
$$\frac{\partial L_n}{\partial w_{i,j}^k} = \frac{\partial z_j^{k+1}}{\partial w_{i,j}^k}\frac{\partial L_n}{\partial z_j^{k+1}} \quad \text{et} \quad \frac{\partial L_n}{\partial z_j^{k+1}} = \sigma'\left(z_j^{k+1}\right)\sum_l w_{j,l}^{k+1}\frac{\partial L_n}{\partial z_l^{k+2}}$$

Donnés par l'itération précédente : $w_{i,j}{}^k$

Pour chaque échantillon #n

Propagation avant : calcul des z_i^k, des $\sigma(z_i^k)$, des $\sigma'(z_i^k)$ et des $\hat{y}_i$

RÉTROPROPAGATION



$$\frac{\partial L_n}{\partial w_{i,j}^k} = \frac{\partial z_j^{k+1}}{\partial w_{i,j}^k}\frac{\partial L_n}{\partial z_j^{k+1}} \quad \text{et} \quad \frac{\partial L_n}{\partial z_j^{k+1}} = \sigma'\left(z_j^{k+1}\right)\sum_l w_{j,l}^{k+1}\frac{\partial L_n}{\partial z_l^{k+2}}$$

Donnés par l'itération précédente : $w_{i,j}^{\ k}$

Donnés par propagation avant : z_i^k, $\sigma(z_i^k)$, $\sigma'(z_i^k)$ et $\hat{y}_i$

Initialisation : $\frac{\partial L_n}{\partial z_i^N} = \frac{\partial L_n}{\partial y_i}\frac{\partial y_i}{\partial a_i}\frac{\partial a_i}{\partial z_i^N}$

Rétropropagation : $\forall k = N \ldots 1$, calcul de $\frac{\partial L_n}{\partial z_i^k}$ puis de $\frac{\partial L_n}{\partial w_{i,j}^k}$

APPRENTISSAGE (RAPPEL)

Objectif :

trouver les poids (y compris biais) qui minimisent la fonction de coût L (*loss*)

$L = \sum_n L_n \rightarrow$ un coût pour chaque échantillon de la base d'apprentissage

Descente de gradient

Calcul du gradient global par sommation $\frac{\partial L}{\partial w_{i,j}^k}(\boldsymbol{w}) = \sum_n \frac{\partial L_n}{\partial w_{i,j}^k}(\boldsymbol{w})$

Mise à jour des poids $w_{i,j}^k \leftarrow w_{i,j}^k - \lambda \frac{\partial L}{\partial w_{i,j}^k}(\boldsymbol{w})$

Plan du cours

Quelques méthodes d'optimisation

- Méthode sans dérivée explicite
- Méthode de Newton
- Descente de gradient
- Gradient stochastique

Calcul d'une dérivée

- Méthode directe
- Approximation numérique
- Différentiation automatique
- Application à la rétropropagation de gradient

Optimisation pratique

- Paramétrisation, pondération des coûts, équilibrage des données
- Intégration de contraintes

OPTIMISATION EN PRATIQUE

Deux nombreuses considérations à garder à l'esprit

Non-unicité des paramètres

Ex : unité $y = \lambda x \quad \Rightarrow \quad \frac{\partial L}{\partial y} = \frac{\partial L}{\partial x}\frac{\partial x}{\partial y} = \frac{1}{\lambda}\frac{\partial L}{\partial x}$

Plus généralement $y = f(x) \quad \Rightarrow \quad \frac{\partial L}{\partial y} = \frac{\partial L}{\partial x}\frac{\partial x}{\partial y} = \frac{1}{f'(x)}\frac{\partial L}{\partial x}$

Paramètres incommensurables

Ex : amplitudes de la translation et angles de rotation

Équilibre des composantes du gradient

Celles de plus forte influence (gradient fort) dominent la direction de descente (*exploding gradient*)

Éviter une influence d'amplitude moindre que la tolérance de convergence (*vanishing gradient*)

Diverses normalisations (ex Mahalanobis, mais ne suffit souvent pas)

OPTIMISATION EN PRATIQUE

Des fonctions de coût à plusieurs termes

- Ex : régularisation, hétérogénéité des paramètres (termes ad hoc, cf détection d'objets)
- Trouver une pondération qui équilibre les influences sur le gradient
- Poids fixes, mais peuvent être appris. Souvent essais-erreurs

Équilibre des données

- Ex : beaucoup de données positives, peu de négatives ; répartition inégale dans l'espace
- Pondération des coûts (L_n)
- Duplication/augmentation/synthèse de données
- Sélection des cas en surnombre pour équilibrer les batchs

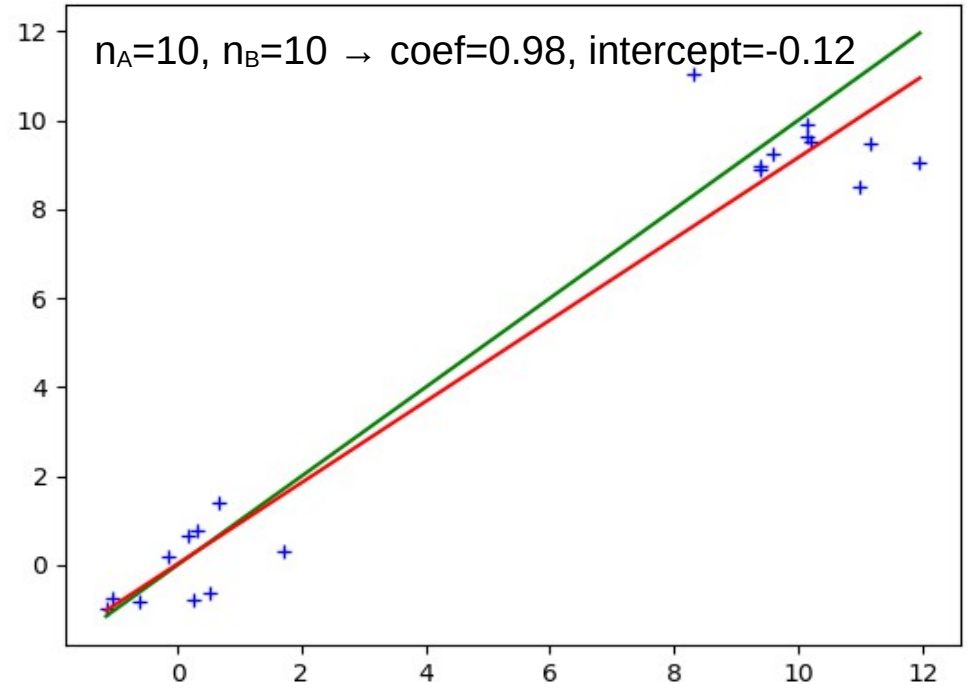
DÉSÉQUILIBRE DES DONNÉES

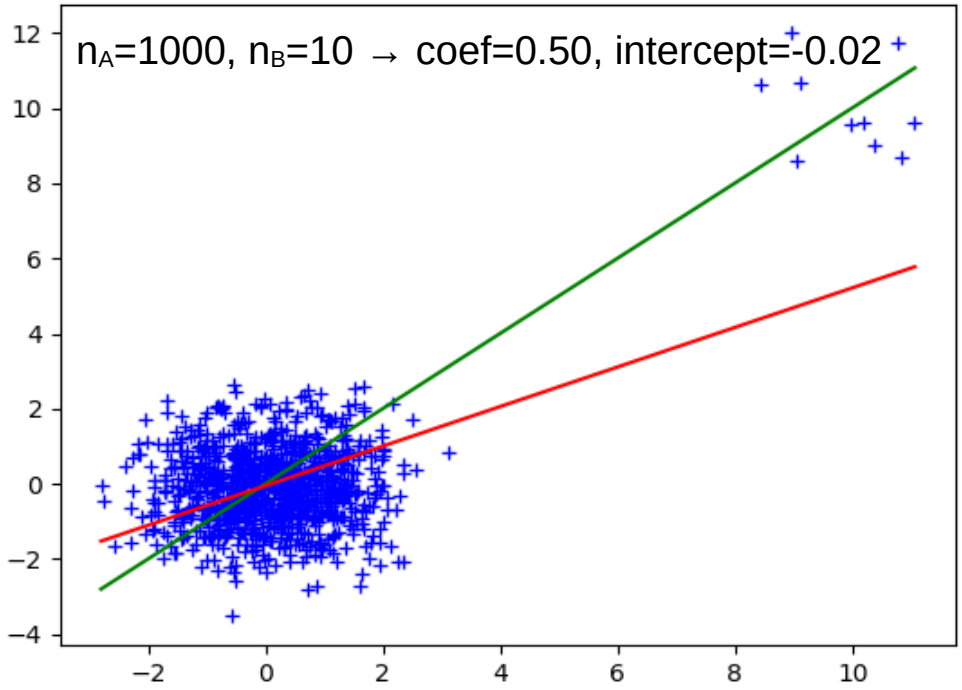
Régression linéaire

Point A(0,0) + n_A points bruités (N(0,1))

Point B(10,10) + n_B points bruités (N(0,1))

Régression idéale : y=x (coef=1, intercept=0, vert)





INTÉGRATION DE CONTRAINTES

Expression des contraintes

$$\min f(x) \quad \text{avec } x \text{ tel que} \quad g(x)=0 \quad \text{et } h(x)\geq 0$$

Méthode générale (hors cours)

- Si juste g(x)=0 → multiplicateur de Lagrange
- Sinon, plus compliqué

Deux cas pratiques

- X ≥ 0 → changement de variable y=ln(x)
- |X|=1 → renormalisation (souvent ok si petits pas)