

Une introduction à l'optimisation pour le machine learning

Rodolphe Le Riche¹, Dédji Brian Whannou², Kévin Kpakpo Akouété³, Espéran Padonou⁴, Armance Darrobers⁵, Loris Cros⁵

¹ CNRS LIMOS à Mines Saint-Étienne, France

² UBS Group AG

³ Descartes Underwriting

⁴ Fondation Vallet

⁵ CentraleSupélec

Juillet 2025
École d'Été en Intelligence Artificielle
Fondation Vallet
Cotonou, Bénin



Préambule

Ce cours a été donné lors d'une école d'été sur l'IA à Godomey, Bénin, en juillet-août 2025. L'école était organisée par l'ONG Bénin Excellence et la Fondation Vallet (cf. <https://www.fondationvallet.org/eeia>).

- Le cours présente les concepts de base d'optimisation
- destiné à un public intéressé par le machine learning
- avec un niveau équivalent à une année après le baccalauréat
- à travers des exemples codés en python from scratch
- Limite : les algorithmes ne sont pas ceux utilisés en deep learning de pointe, mais les concepts principaux liés à la descente de gradient sont présentés.

Le code, les diapositives et le sujet de projet sont disponibles sur :

<https://github.com/ML-for-B-E/Optimisation>

Plan du cours

Une introduction à l'optimisation pour l'apprentissage automatique

1 Introduction

- Objectifs, remerciements
- Formulation du problème d'optimisation
- Exemples d'utilisation de l'optimisation
- Concepts mathématiques de base pour l'optimisation

2 Algorithme de la descente la plus raide

- Algorithme de la descente la plus raide à pas fixe
- Recherche de pas (line search)

3 Recherches améliorées basées sur le gradient

- Directions de recherche pour accélération
- Un mot sur les contraintes
- Rendre la méthode plus globale : redémarrages

4 Application aux réseaux de neurones

5 Bibliographie



Optimisation = une formulation quantitative de la décision

L'optimisation est une¹ manière de modéliser mathématiquement une décision.

$$x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \in \mathbb{R}^n$$
$$\min_{x \in S} f(x)$$



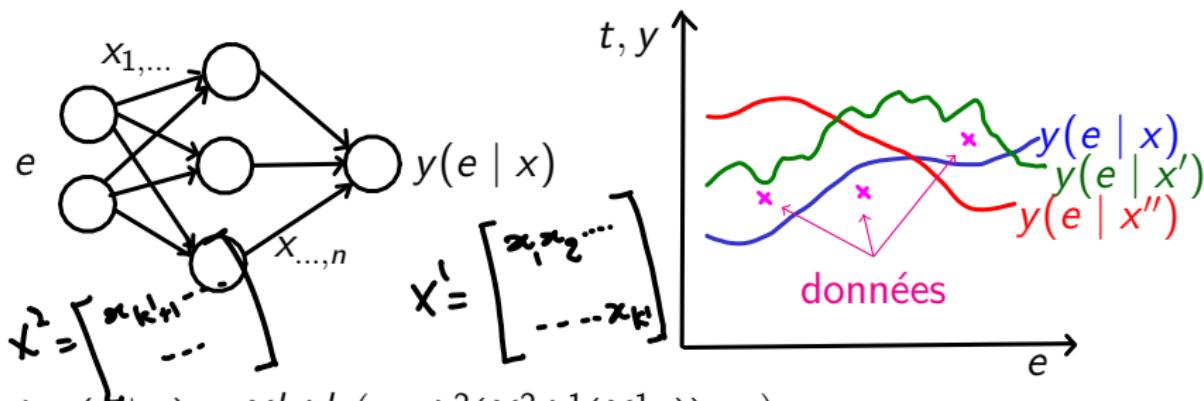
- x : vecteur des paramètres (ou variables) de décision : dimensions, investissements, réglages d'une machine ou d'un programme, etc.
- $f(x)$: coût associé à la décision x
- S : ensemble des valeurs possibles de x (espace de recherche)

¹non unique, incomplète quand on considère les êtres humains ou la vie

Exemple d'optimisation : Apprentissage

Réseau de neurones comme fonction paramétrée

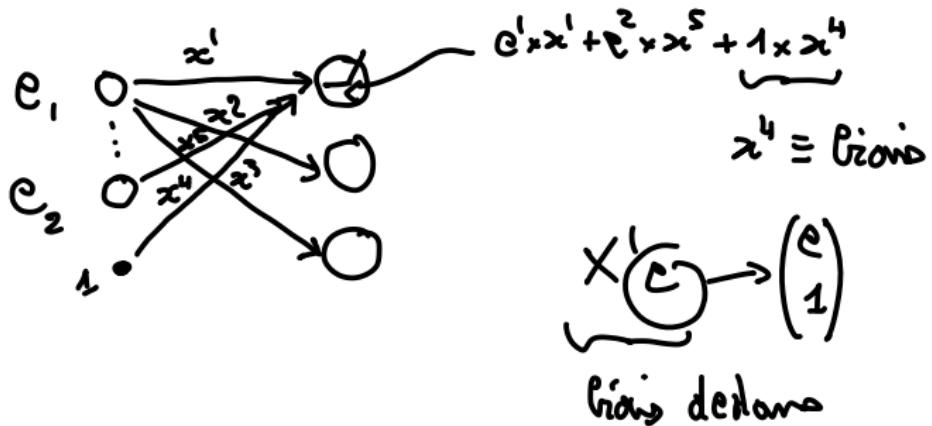
Trouver x , les poids et les biais du réseau de neurones (NN), de sorte que la fonction $y(\cdot | x)$ se rapproche des données observées.



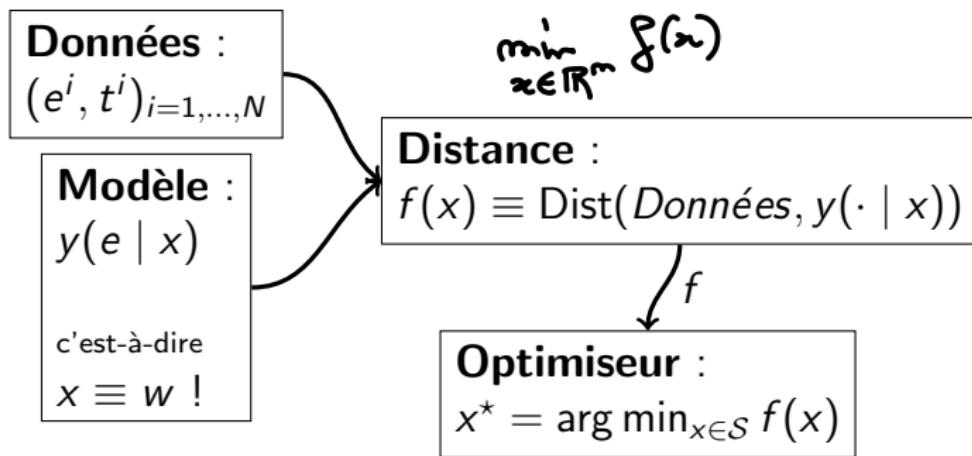
$$\text{où } y(e | x) = X^L \Phi^L (\dots \Phi^2(X^2 \Phi^1(X^1 e)) \dots),$$

X^1, \dots, X^L sont les poids/biais organisés en matrices pour chacune des L couches,

Φ^i est un vecteur de fonctions d'activation (ReLU, linéaire, sigmoïde, leaky ReLU) pour la i -ième couche.



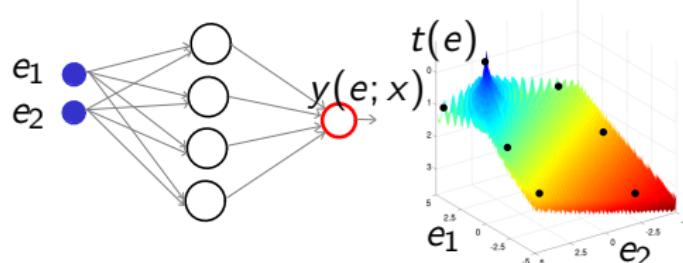
Apprentissage comme optimisation : vue d'ensemble



\Rightarrow **Modèle appris :** $y(\cdot; x^*)$

Exemple d'optimisation : régression par RN

Apprendre une fonction à partir d'un ensemble discret et limité d'observations



- e entrées ou inputs, $t(e) \in \mathbb{R}$: fonction cible à apprendre
- Ensemble de **données** observées : $(e^i, t^i \equiv t(e^i))$, $i = 1, \dots, N$
- **Modèle** $y(e; x)$: un réseau de neurones avec entrées e et poids biais x , censé approximer $t(e)$
- **Distance données-modèle** sous forme d'erreur quadratique :

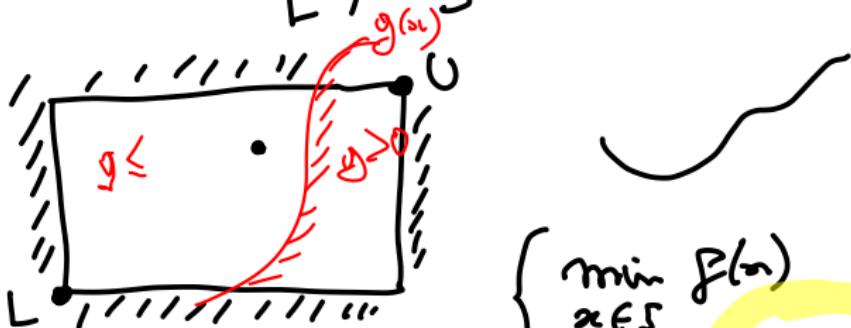
$$\min_x f(x)$$

$$f(x) = \frac{1}{2N} \sum_{i=1}^N (t^i - y(e^i; x))^2$$

$$\min_{x \in S} f(x) \quad f: \mathbb{R}^m \rightarrow \mathbb{R}$$

$[L, U]$ $\subset \mathbb{R}^m$

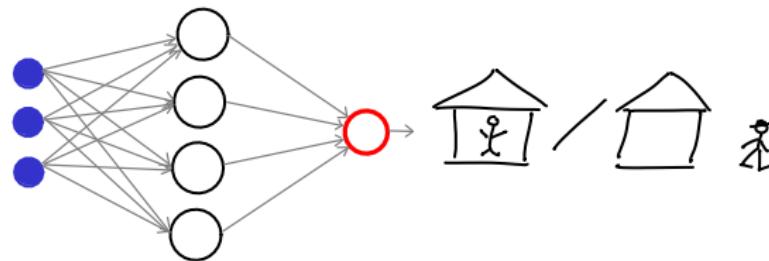
$m=2$



$\left\{ \begin{array}{l} \min_{x \in S} f(x) \\ \text{tel que } g(x) \leq 0 \end{array} \right.$

Exemple d'optimisation : classification par RN (1/3)

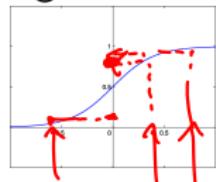
Exemple : prédire si une personne reste à la maison ou sort, en fonction de sa longitude, latitude et température = un problème de classification à 2 classes (dedans/dehors).



- e : entrées (ex: e_1 = longitude, e_2 = latitude, e_3 = température)
- $t(e) \in \{0, 1\}$: fonction cible à apprendre ($t = 1$ si la personne reste, $t = 0$ sinon)
- Ensemble de données observées: (e^i, t^i) , $i = 1, \dots, N$

Exemple d'optimisation : classification par RN (2/3)

- $y(e; x)$: Sortie du réseau de neurones $\in \mathbb{R} \neq \{0, 1\}$: l'espace de la fonction à apprendre est discret, celui du réseau est continu 😞
- Astuce : prédire la probabilité que $t(e) = 1$. Cette probabilité est dans $[0, 1]$ qui est continu... mais borné 😊
- Astuce de la régression logistique : passer la sortie du RN dans



une sigmoïde pour obtenir $y(\cdot; x) \in [0, 1]$ 😊

- Avantage : le modèle a une interprétation probabiliste. $y(e; x)$ est la probabilité que $t(e) = 1$, $t(e)$ est une variable de Bernoulli.

Exemple d'optimisation : classification par RN (3/3)

Erreur d'entropie croisée (logvraisemblance) comme distance modèle-données :

$$f(x) = - \sum_{i=1}^N \{t^i \log(y(e^i; x)) + (1 - t^i) \log(1 - y(e^i; x))\}$$

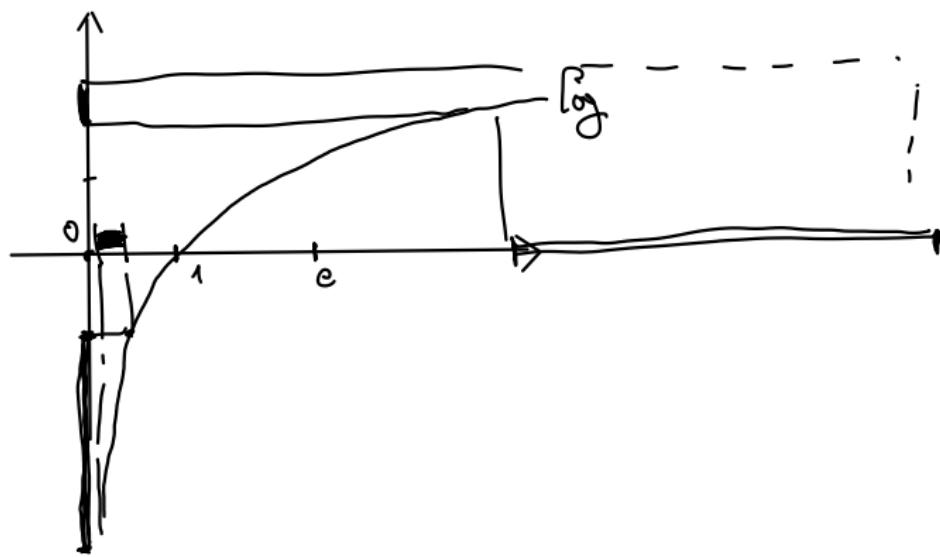
Démonstration :

$$\min_{x \in \mathcal{X}} f(x)$$

- $y(e; x)$ est la probabilité que $t(e) = 1$, $1 - y(e; x)$ est la proba que $t(e) = 0$, $0 \leq y(e; x) \leq 1$.
- La probabilité de t sachant e peut s'écrire $y(e; x)^t \times (1 - y(e; x))^{1-t}$
- La vraisemblance des N observations i.i.d. est $\prod_{i=1}^N [y(e^i; x)^{t^i} \times (1 - y(e^i; x))^{1-t^i}]$, à maximiser
- On transforme la vraisemblance en erreur, à minimiser, en prenant $-\log(\text{vraisemblance})$

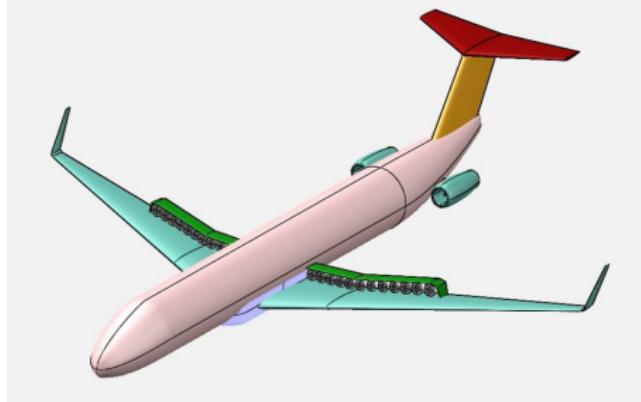
$$\begin{array}{c} e^i \\ \swarrow \quad \searrow \end{array} \begin{array}{l} t^i = 1 \\ t^i = 0 \end{array} \begin{array}{l} y(e^i; x) \\ p \end{array}$$





Autres exemples d'utilisation de l'optimisation

Exemple d'optimisation : conception



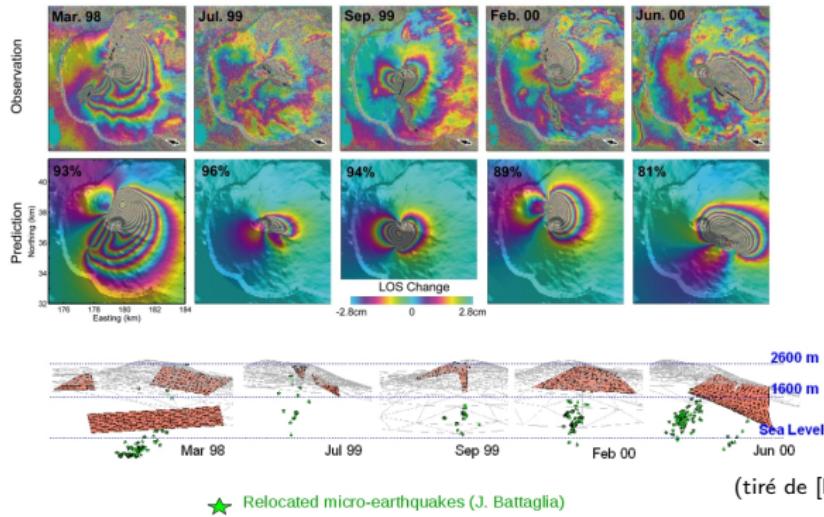
(tiré de [Sgueglia et al., 2018])

x = paramètres de l'avion (ici propulsion électrique distribuée)

$f()$ = $-1 \times$ métrique de performance (agrégation de $-1 \times$ portée, coût, distance de décollage, ...)

Au minimum, la conception est “optimale”.

Exemple d'optimisation : identification de modèle



x = position du barrage, géométrie, pression interne

$f()$ = distance entre les mesures (du satellite RADARSAT-1) et le modèle (éléments de frontière, calcul non trivial)

Au minimum, le modèle correspond le mieux aux mesures et devrait représenter le phénomène.

Exemple d'optimisation : débruitage d'image

$$\min_x f(x) \quad , \quad f(x) = \frac{1}{2} \sum_{i=1}^{N_{\text{pixels}}} (y_i - x_i)^2 + \lambda \sum_{i=1}^{N_{\text{pixels}}} \sum_{j \text{ proche de } i} |x_i - x_j|$$

$\lambda \geq 0$ constante de régularisation

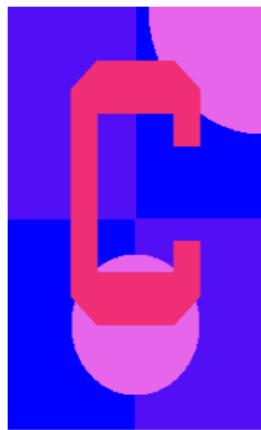
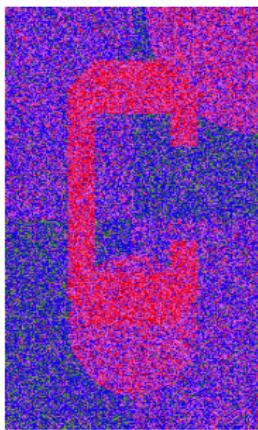
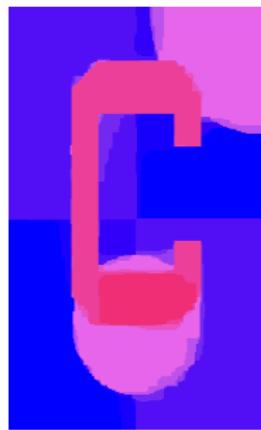


image cible



bruitée (observée)
 $= y_i$



dénoyautée (optimisée)
 $= x^*$

(tiré de [Ravikumar and Singh, 2017])

Concepts mathématiques de base pour l'optimisation

1 Introduction

- Objectifs, remerciements
- Formulation du problème d'optimisation
- Exemples d'utilisation de l'optimisation
- Concepts mathématiques de base pour l'optimisation

2 Algorithme de la descente la plus raide

- Algorithme de la descente la plus raide à pas fixe

- Recherche de pas (line search)

3 Recherches améliorées basées sur le gradient

- Directions de recherche pour accélération
- Un mot sur les contraintes
- Rendre la méthode plus globale : redémarrages

4 Application aux réseaux de neurones

5 Bibliographie

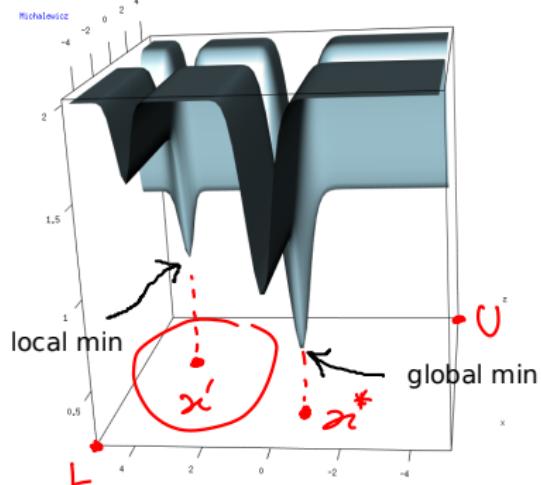
Optimum local versus global

$$g(x) = f(x_1, x_2)$$

$m=2$

$$\min_{x \in S \subset \mathbb{R}^n} f(x)$$

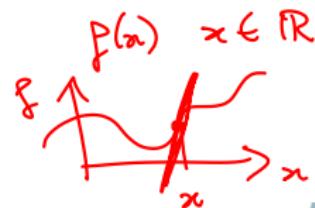
[L, U]



Code Python pour générer un tel graphique 3D disponible dans le dossier Code, fichier 3D_plots.py

Gradient d'une fonction

Gradient d'une fonction = direction de la montée la plus rapide = vecteur des dérivées partielles



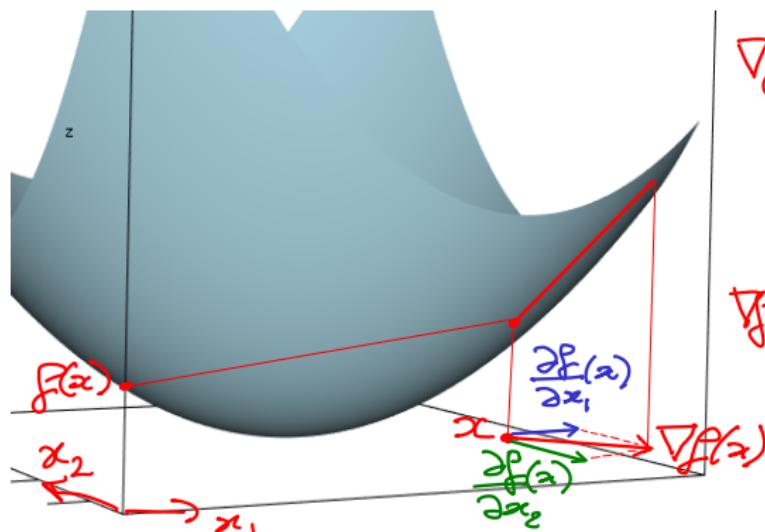
$$\nabla f(x) = \begin{pmatrix} \frac{\partial f}{\partial x_1}(x) \\ \dots \\ \frac{\partial f}{\partial x_n}(x) \end{pmatrix}$$

$$f: \mathbb{R}^n \rightarrow \mathbb{R}$$

$$f(x) = 1 + x_1 x_2 + 3 x_2^2$$

$$\nabla f(x) = \begin{pmatrix} \frac{\partial f}{\partial x_1}(x) \\ \frac{\partial f}{\partial x_2}(x) \end{pmatrix}$$

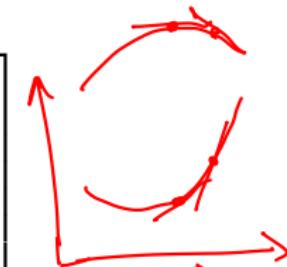
$$\nabla f(x) = \begin{pmatrix} x_2 \\ x_1 + 6x_2 \end{pmatrix}$$



Hessienne d'une fonction

$$f(x) = 1 + x_1 x_2 + 3x_2^2 \quad \nabla f(x) = \begin{pmatrix} x_2 = \frac{\partial f}{\partial x_1} \\ x_1 + 6x_2 = \frac{\partial f}{\partial x_2} \end{pmatrix} \quad \nabla^2 f(x) = \begin{bmatrix} 0 & 1 \\ 1 & 6 \end{bmatrix}$$

C'est la matrice des dérivées secondes,

$$\nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \frac{\partial^2 f(x)}{\partial x_2^2} & \cdots & \frac{\partial^2 f(x)}{\partial x_2 \partial x_n} \\ \cdots & \cdots & \ddots & \cdots \\ \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} & \frac{\partial^2 f(x)}{\partial x_2 \partial x_n} & \cdots & \frac{\partial^2 f(x)}{\partial x_n^2} \end{bmatrix}$$


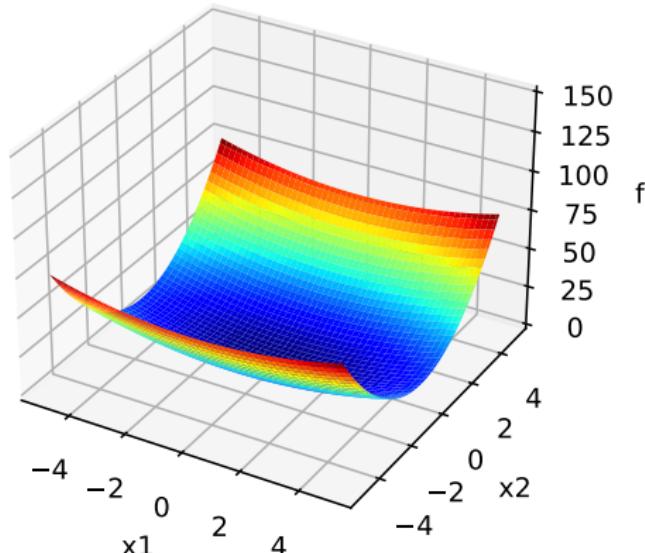
= matrice des courbures = gradient du gradient.

Fonction quadratique et Hessienne I

$$f(x) = \frac{1}{2} x^T H x, \quad \nabla^2 f(x) = H$$

une bonne approximation de ce qui se passe pour toute fonction au voisinage de la convergence quadratic

$l \times l$ $l \times m$ $m \times m$ $m \times l$

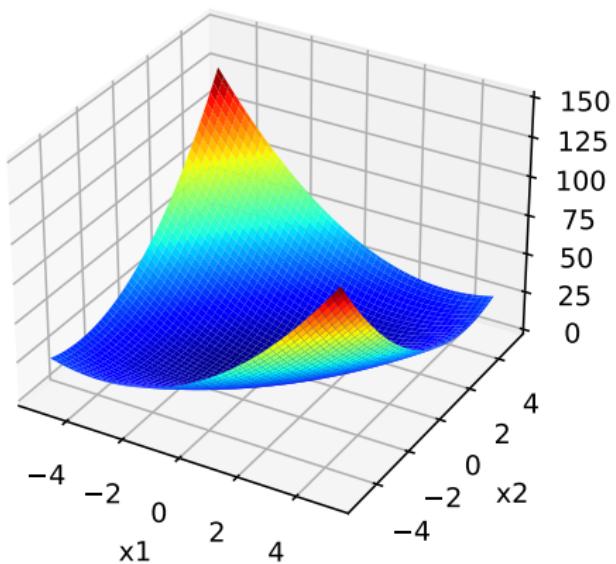


$$\frac{\partial^2 f}{\partial x_1^2}$$
$$H = \begin{bmatrix} 1 & 0 \\ 0 & 5 \end{bmatrix}$$

(devinez les valeurs propres et vecteurs propres)

Fonction quadratique et Hessienne II

quadratic



le même tourné de 45°

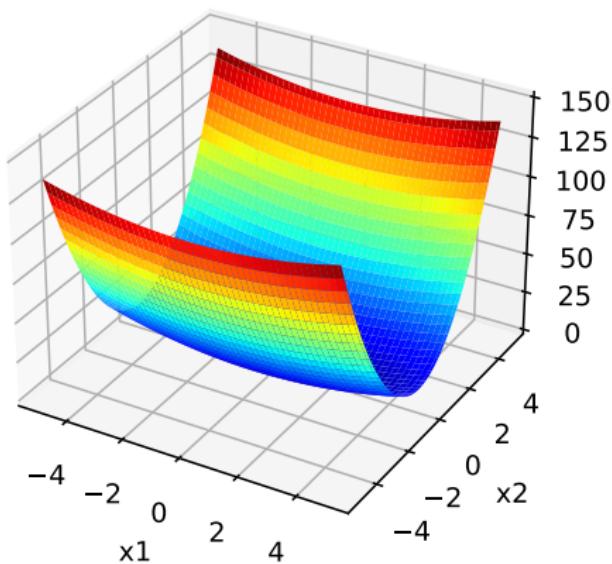
$$H = \begin{bmatrix} 3 & -2 \\ -2 & 3 \end{bmatrix}$$

vect. propres = $\begin{bmatrix} \sqrt{2}/2 & -\sqrt{2}/2 \\ \sqrt{2}/2 & \sqrt{2}/2 \end{bmatrix}$

val. propres = [1, 5]

Fonction quadratique et Hessienne III

quadratic

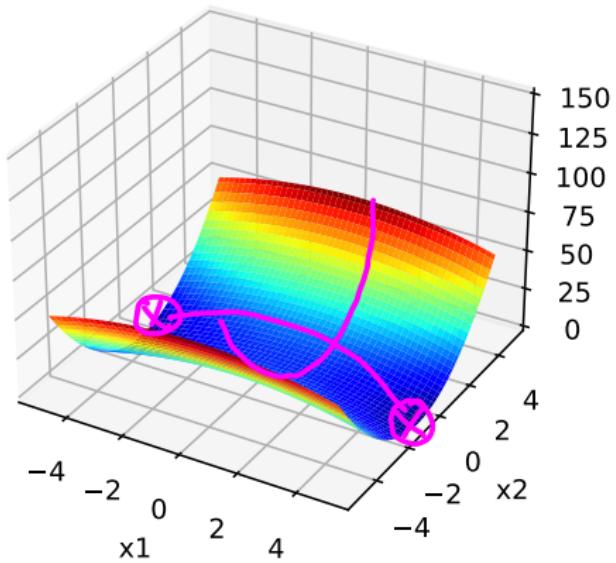


courbure accrue
 f (nombre de condition)

$$H = \begin{bmatrix} 1 & 0 \\ 0 & 10 \end{bmatrix}$$

Fonction quadratique et Hessienne IV

quadratic



$$\min_{\mathbf{x}} \tilde{f}(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T H \mathbf{x}$$

Hessienne non définie positive

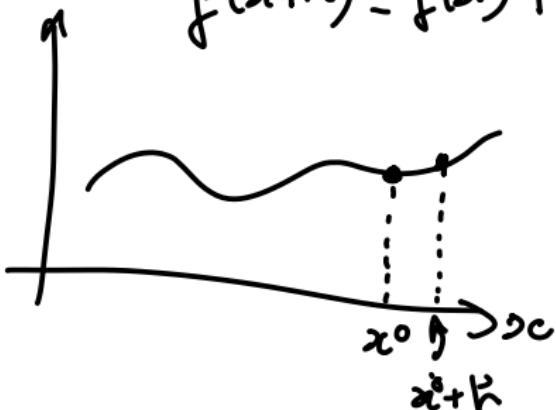
$$H = \begin{bmatrix} -1 & 0 \\ 0 & 5 \end{bmatrix}$$

quel est le problème ?

$m=1$

$$f(x) = f(x^0) + \frac{df(x^0)}{dx} (x - x^0) + \frac{1}{2} \frac{\cancel{df(x^0)}}{\cancel{dx}} (x - x^0)^2$$

$$f(x^0 + h) = f(x^0) + \frac{df(x^0)}{dx} h + \frac{1}{2} \frac{\cancel{df(x^0)}}{\cancel{dx}} h^2$$



h est petit $\ll 1$

$$h^2 \ll h$$

$$f(x^0 + h) \approx f(x^0) + \frac{df(x^0)}{dx} h$$

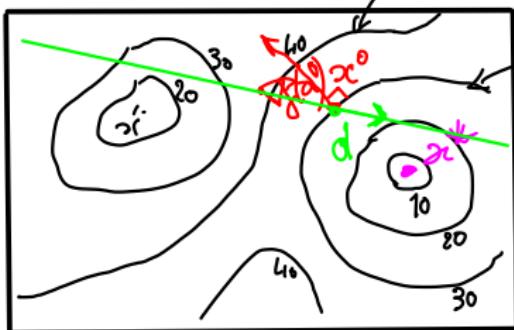
m

$$\underbrace{f(x^0 + h)}_{m \times 1} \approx \underbrace{f(x^0)}_{1 \times 1} + \underbrace{\nabla f(x^0)^T}_{1 \times 1} \cdot \underbrace{h}_{m \times 1}$$

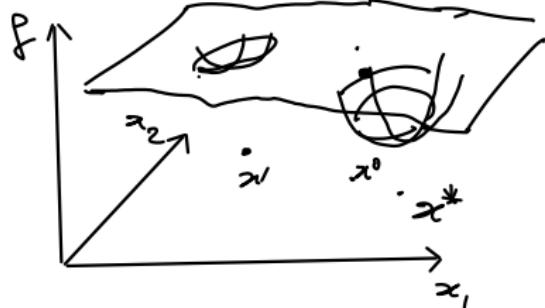
$$\langle \nabla f(x^0), h \rangle$$

$$\Rightarrow = \sum_{i=1}^m \frac{\partial f(x^0)}{\partial x_i} \cdot h_i$$

$m=2$



$$f(z) = f(z^0)$$



$$z^0 \dots z^0 + \alpha d \quad \|d\| = 1 \quad R = \alpha d$$

$\frac{m \times 1}{m \times 1}$

$\frac{1 \times 1}{1 \times 1}$

$$f(z^0 + \alpha d) \approx f(z^0) + \nabla f(z^0)^T \cdot (\alpha d)$$

α 0.5 + 0.5t

$$f(z^0 + \alpha d) = f(z^0) + \alpha \nabla f(z^0)^T \cdot d$$

$\frac{m \times 1}{m \times 1} \quad \frac{1 \times 1}{1 \times 1} \quad \frac{1 \times m}{1 \times m} \quad \frac{m \times 1}{m \times 1}$

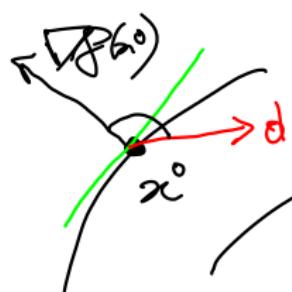
$1 \times 1 \quad 1 \times 1 \quad 1 \times 1 \quad 1 \times 1$

Direction de descente :

$$f(x^0 + \alpha d) \approx f(x^0) + \alpha \nabla f(x^0)^T \cdot d \quad ?$$

$\nabla f(x^0)^T \cdot d < 0$

condition de descente



$$\| \nabla f(x^0) \| \cdot \| d \| \cdot \underbrace{\cos(\nabla f(x^0), d)}_{=1} < 0$$

$$< 0$$

d t.q. $\cos(\nabla f(x^0), d) > 0$
 $\nabla f(x^0)^T \cdot d > 0$

d direction de montée
 $\nabla f(x^0)^T \cdot (-d) < 0$

$$\underbrace{(\nabla f(x^0))^T}_{1 \times m} \cdot \underbrace{\alpha}_{m \times 1}$$

$$\begin{aligned} \nabla f(x^0) &\longleftrightarrow \alpha \\ \alpha &\longleftrightarrow \beta \end{aligned}$$

$$\|\alpha\| = \sqrt{\sum_{i=1}^m \alpha_i^2}$$

$$\underbrace{b}_{m \times 1}, \underbrace{\alpha}_{m \times 1} \in \mathbb{R}^m$$

Product scalaire entre α et β

$$\begin{aligned} \alpha^T \cdot \beta &= \langle \alpha, \beta \rangle \\ &= \sum_{i=1}^m \alpha_i \beta_i \end{aligned}$$

$$= \|\alpha\| \cdot \|\beta\| \cdot \cos(\alpha, \beta)$$

Condition nécessaire d'optimisation

$$f(x^0 + \alpha d) \approx f(x^0) + \underbrace{\nabla f(x^0)}_d^\top d$$

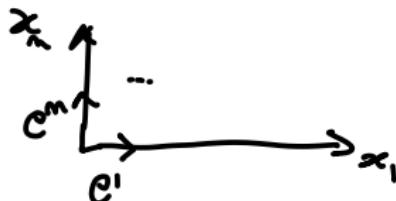


$$\forall d \quad \nabla f(x^0)^\top d = 0$$

$$\boxed{\nabla f(x^0) = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}}$$

Estimation du gradient par différences finies

$$d = e^i$$



$$e^i = \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}_{i\text{ème}}$$

$$\begin{aligned} f(x^0 + \alpha e^i) &\approx f(x^0) + \alpha \nabla f(x^0)^T \cdot e^i \\ &\approx f(x^0) + \alpha \frac{\partial f(x^0)}{\partial x_i} \end{aligned}$$

$$\boxed{\frac{\partial f(x^0)}{\partial x_i} \approx \frac{f(x^0 + \alpha e^i) - f(x^0)}{\alpha}}$$

α petit
 $i = 1, \dots, m$

x^i ← ID, "mom" du point, n° de l'itération
 x^i_j ← composante, $1 \leq j \leq m$

Approximation numérique du gradient

Par différences finies avant

$$\frac{\partial f(x)}{\partial x_i} \approx \frac{f(x + he^i) - f(x)}{h}$$

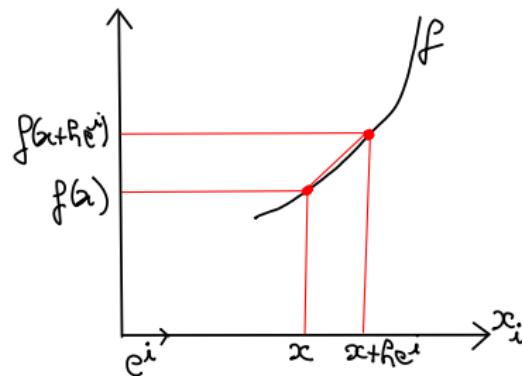
Démonstration : par Taylor,

$$f(x + he^i) = f(x) + he^{i^\top} \cdot \nabla f(x) + \frac{h^2}{2} e^{i^\top} \nabla^2 f(x +$$

$$\rho h e^i) e^i, \quad \rho \in]0, 1[$$

$$\frac{\partial f(x)}{\partial x_i} = \frac{f(x + he^i) - f(x)}{h} - \frac{h}{2} e^{i^\top} \nabla^2 f(x + \rho h e^i) e^i$$

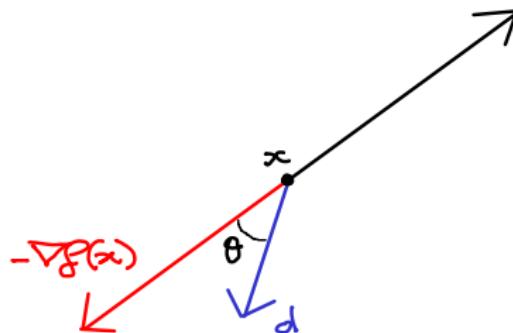
et en faisant tendre h vers zéro \square



Autres schémas (meilleurs mais plus complexes à implémenter) : différences centrées, différentiation automatique (ex. : TensorFlow ou PyTorch), différentiation (semi-)analytique (ex. : rétropropagation dans les réseaux de neurones).

Direction de descente

Une direction de recherche d formant un angle aigu avec $-\nabla f(x)$ est une direction de descente, c'est-à-dire que pour un pas suffisamment petit, f diminue assurément !

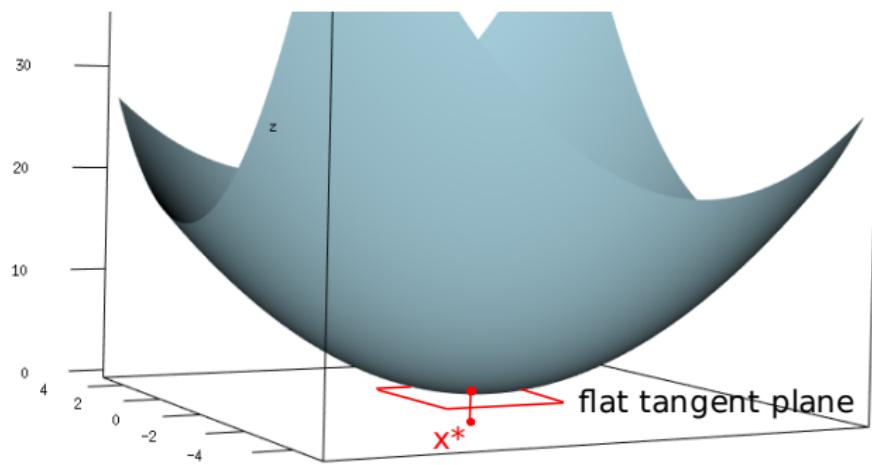


Démonstration : par Taylor, $\forall \alpha, \exists \epsilon \in [0, 1]$ tel que
 $f(x + \alpha d) = f(x) + \alpha d^\top \cdot \nabla f(x) + \frac{\alpha^2}{2} d^\top \nabla^2 f(x + \alpha \epsilon d) d$
 $\lim_{\alpha \rightarrow 0^+} \frac{f(x + \alpha d) - f(x)}{\alpha} = d^\top \cdot \nabla f(x) = -1 \times \|\nabla f(x)\| \cos(d, -\nabla f(x))$
est négatif si le cosinus est positif \square

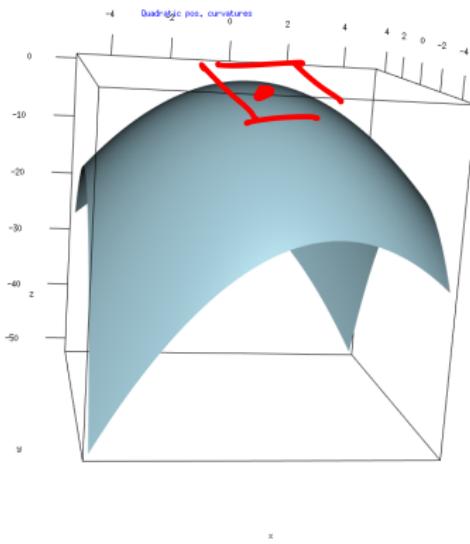
Condition nécessaire d'optimalité (1)

Une condition nécessaire pour qu'une fonction différentiable ait un minimum en x^* est que celle-ci soit plate en ce point, c'est-à-dire que son gradient soit nul :

$$x^* \in \arg \min_{x \in \mathcal{S}} f(x) \Rightarrow \nabla f(x^*) = 0$$

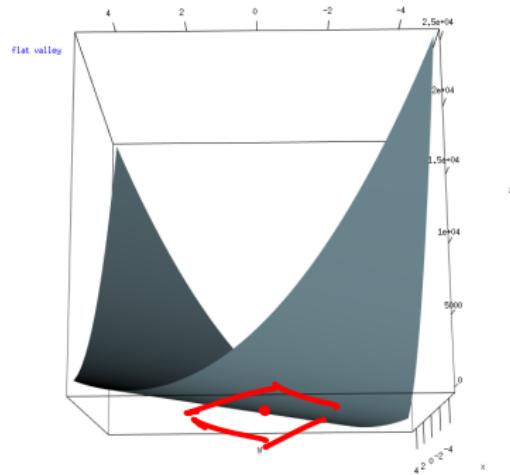


Condition nécessaire d'optimalité (2)



nécessaire mais pas suffisante (fonctionne aussi pour un maximum)

Condition nécessaire d'optimalité (3)

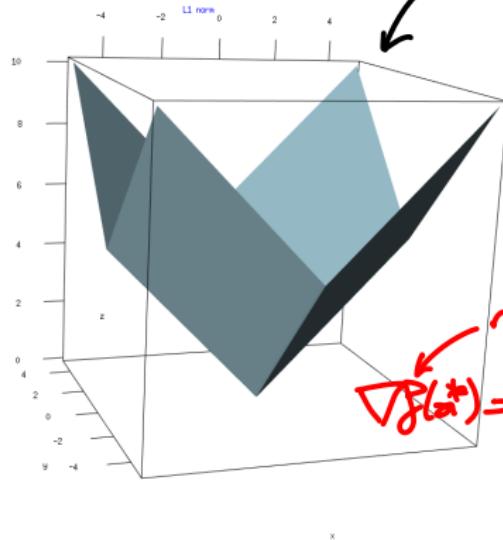


$\nabla f(x^*) = 0$ ne garantit pas l'unicité de x^* (vallée plate)

Condition nécessaire d'optimalité (4)

$$\sum_{i=1}^n |\alpha_i| = \|\alpha\|_1$$

L_1



non-différentiable

$$\nabla f(x^*) = 0$$

$\nabla f()$ n'est pas défini partout, exemple avec la norme $L1 = \sum_{i=1}^n |x_i|$

Plan du cours

Introduction à l'optimisation pour l'apprentissage automatique

1 Introduction

- Objectifs, remerciements
- Formulation du problème d'optimisation
- Exemples d'utilisation de l'optimisation
- Concepts mathématiques de base pour l'optimisation

2 Algorithme de la descente la plus raide

- Algorithme de la descente la plus raide à pas fixe
- Recherche de pas (line search)

3 Recherche linéaire

- Recherches améliorées basées sur le gradient
- Directions de recherche pour accélération
- Un mot sur les contraintes
- Rendre la méthode plus globale : redémarrages

4 Application aux réseaux de neurones

5 Bibliographie



Optimiseurs comme algorithmes itératifs

On cherche $x^* \in \arg \min_{x \in \mathcal{S}} f(x)$, $\mathcal{S} = \mathbb{R}^n$

- Sauf cas particuliers (ex. : problèmes quadratiques convexes), la solution n'est pas obtenue analytiquement via les conditions d'optimalité ($\nabla f(x^*) = 0$ + conditions d'ordre supérieur).
- On utilise typiquement des algorithmes itératifs : x^{i+1} dépend des itérés précédents x^1, \dots, x^i et de leurs valeurs f .
- Souvent, calculer $f(x^i)$ demande plus de temps que l'algorithme d'optimisation lui-même.
- Qualités recherchées chez un optimiseur : robustesse, rapidité de convergence. Il faut un compromis entre ces deux.

Algorithme de la descente de gradient à pas fixe

Répéter les étapes dans la direction de la descente la plus raide,

$-\nabla f(x^t)$ [Cauchy, 1847, Curry, 1944].

La taille des pas est proportionnelle à la norme du gradient.

Require: $f()$, $\bar{\alpha} \in]0, 1]$, x^1 , ϵ^{step} , ϵ^{grad} , i^{\max}

$i \leftarrow 0$, $f^{\text{bestSoFar}} \leftarrow \text{max_double}$

repeat

$i \leftarrow i + 1$

calculer $f(x^i)$ et $\nabla f(x^i)$

if $f(x^i) < f^{\text{bestSoFar}}$ **then**

mettre à jour $x^{\text{bestSoFar}}$ et $f^{\text{bestSoFar}}$ avec l'itéré courant

end if

direction : $d^i = -\frac{\nabla f(x^i)}{\|\nabla f(x^i)\|}$

$\|\nabla f(x^i)\| \leftarrow 1$ scalaire

pas : $x^{i+1} = x^i + \bar{\alpha} \|\nabla f(x^i)\| d^i$

$$\left\| \frac{\nabla f(x^i)}{\|\nabla f(x^i)\|} \right\| = 1 = \|d^i\|$$

until $i > i^{\max}$ **or** $\|x^i - x^{i-1}\| \leq \epsilon^{\text{step}}$ **or** $\frac{\|\nabla f(x^i)\|}{\sqrt{n}} \leq \epsilon^{\text{grad}}$

return $x^{\text{bestSoFar}}$ et $f^{\text{bestSoFar}}$



$$f(x^i + \alpha d) \approx f(x^i) + \alpha \underbrace{\nabla f(x^i)^T \cdot d}$$

$$\|\nabla f(x^i)\| \cdot \underbrace{\|d\|}_{=1} \cdot \cos(\nabla f(x^i), d)$$

$$d = \frac{-\nabla f(x^i)}{\|\nabla f(x^i)\|}$$

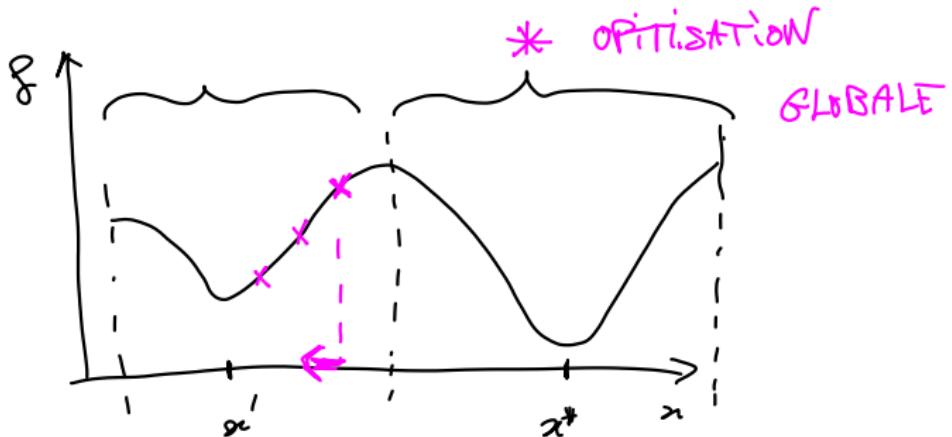


$$\underbrace{\cos(\nabla f(x^i), -\nabla f(x^{i-1})/\|_{i-1})}_{=-1}$$

$$\left\{ \begin{array}{l} \min_d \nabla f(x^i)^T \cdot d \\ \|d\| = 1 \end{array} \right. \Rightarrow d = -\frac{\nabla f(x^i)}{\|\nabla f(x^i)\|}$$

$$\frac{dx}{dt} = -\nabla f(x)$$

optimization
nu co.
sys. dynamique.



(organisation du code) (1)

- Dans `src/optimcourse` : fichiers ressources pour l'optimisation
 - `gradient_descent.py`, `restarted_gradient_descent.py` : algorithmes de descente basés sur le gradient ; la version actuelle avec pas fixe, et les versions à venir (autre direction, avec recherche linéaire), une version avec redémarrages aléatoires.
 - `random_search.py` : un algorithme de recherche aléatoire.
 - `test_functions.py` : une collection de fonctions de test.
 - `3D_plots.py` : trace une fonction à deux dimensions en 3D + courbes de niveau.
 - `optim_utilities.py` : routines supplémentaires.

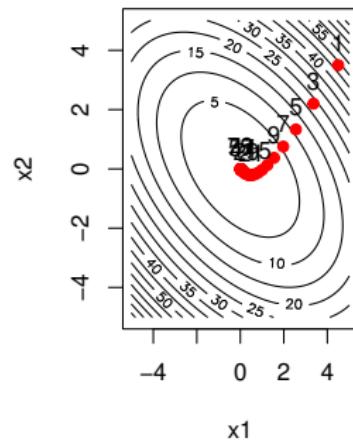
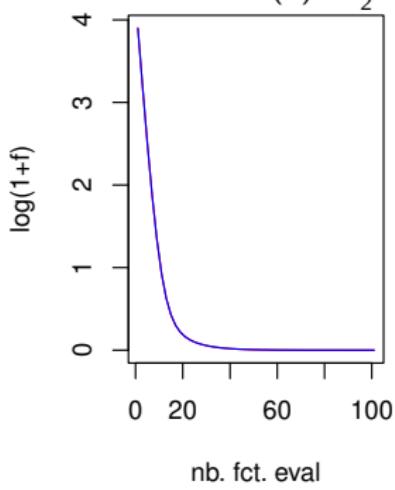
(organisation du code) (2)

- Dans `src/optimcourse` : fichiers ressources pour le réseau de neurones codé à partir de zéro
 - `activation_functions.py` : collection de fonctions d'activation.
 - `forward_propagation.py` : collection de routines pour la propagation avant dans un réseau de neurones.
- Dans `notebooks` : notebooks et script principal pour lancer les algorithmes de descente et l'apprentissage des réseaux de neurones.

Algorithme de descente de gradient à pas fixe (2)

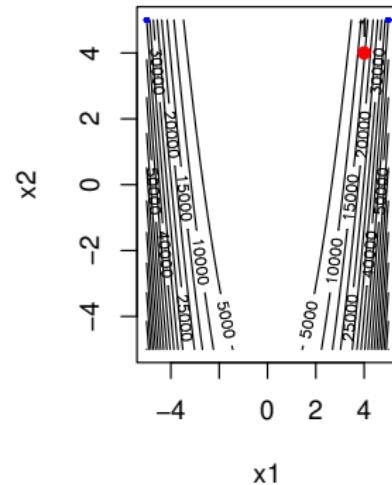
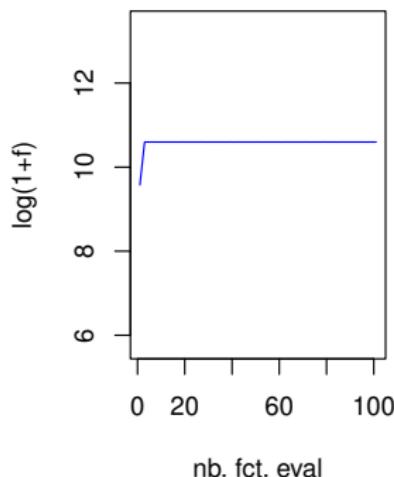
- Le choix du facteur de taille de pas $\bar{\alpha}$ est crucial : plus la fonction est pentue, plus $\bar{\alpha}$ doit être petit. Valeur par défaut = 0.1
- Le vrai code (cf. `gradient_descent.R`) est un peu plus long car il est nécessaire d'enregistrer les points visités.

$$f(x) = \frac{1}{2}x^T H x, \quad H \text{ définie positive}$$



Algorithme de descente de gradient à pas fixe (3)

$\bar{\alpha} = 0,1$ sur la fonction de Rosenbrock (en forme de banane) en dimension $d = 2$, exemple de divergence :



nb. fct. eval

$$x^* = (1, 1) , \quad f(x^*) = 0$$

Descente avec pas optimal

À chaque itération, on cherche la meilleure taille de pas dans la direction de descente² d^i (qui pour l'instant est $-\nabla f(x^i)/\|\nabla f(x^i)\|$, mais c'est plus général).

Même algorithme qu'avant, il suffit de modifier l'instruction **pas** :

Require: ...

(gradient à zéro :
 $\alpha^i = \frac{1}{\|\nabla f(x^i)\|}$)

initialisations, mais pas de α cette fois ...

repeat

incrémenter i , calculer $f(x^i)$ et $\nabla f(x^i)$...

direction : $d^i = -\nabla f(x^i)/\|\nabla f(x^i)\|$ ou toute autre direction de **descente**

pas : $\alpha^i = \arg \min_{\alpha > 0} f(x^i + \alpha d^i)$
 $x^{i+1} = x^i + \alpha^i d^i$

until critère d'arrêt

return meilleur point trouvé

²Si d^i n'est pas une direction de descente, alors $-d^i$ l'est. Preuve laissée en exercice.

Recherche en ligne approximative (1)

Notation : lors de la recherche en ligne à l'itération i ,

$$\begin{aligned}x &= x^i + \alpha d^i & x_j &= x_j^i + \alpha d_j^i & \frac{\partial x_j}{\partial \alpha} &= d_j^i \\f(\alpha) &= f(x^i + \alpha d^i) \\ \frac{df(0)}{d\alpha} &= \sum_{j=1}^n \frac{\partial f(x^i)}{\partial x_j} \frac{\partial x_j}{\partial \alpha} = \sum_{j=1}^n \frac{\partial f(x^i)}{\partial x_j} d_j^i & \text{with } \frac{\partial f(x^i)}{\partial x_j} &= \nabla f(x^i)^\top \cdot d^i\end{aligned}$$

En pratique, optimiser parfaitement α^i est trop coûteux et peu utile
⇒ on approime la recherche en ligne par une condition de décroissance suffisante :

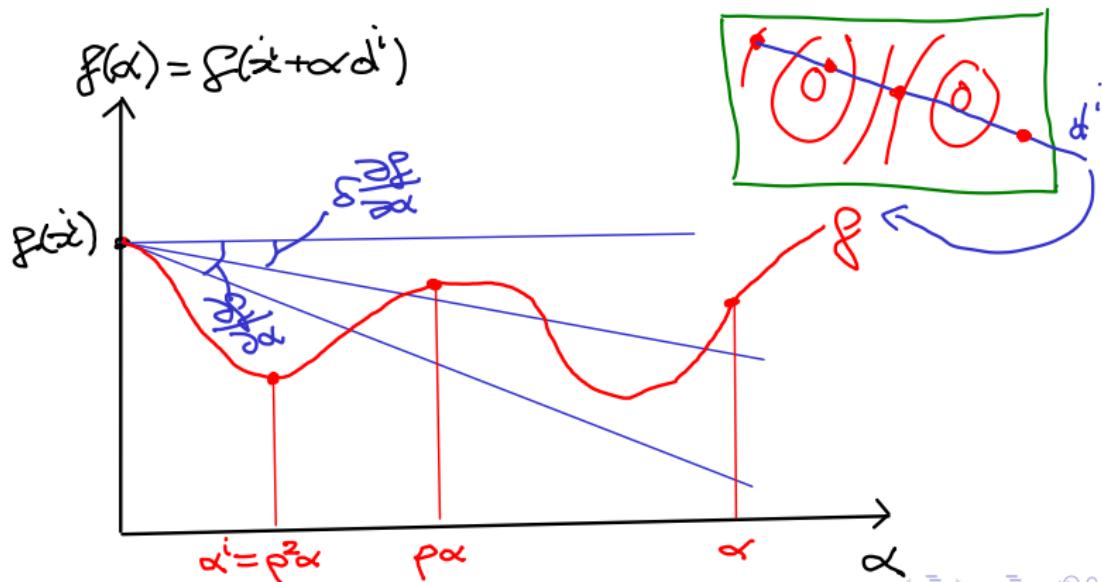
$$\text{trouver } \alpha^i \text{ tel que } f(x^i + \alpha^i d^i) < f(x^i) + \delta \alpha^i \nabla f(x^i)^\top \cdot d^i$$

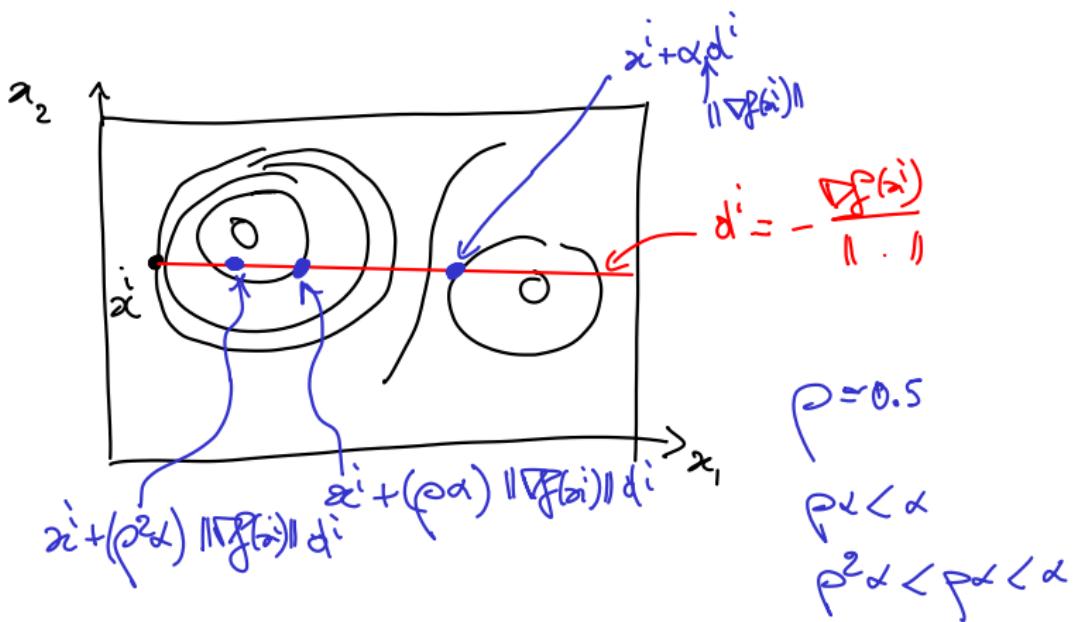
où $\delta \in [0, 1]$, c'est-à-dire atteindre une proportion δ du progrès prévu par le développement de Taylor d'ordre 1.

Recherche en ligne approximative (2)

Condition de décroissance suffisante réécrite avec la notation de recherche en ligne :

trouver α^i tel que $f(\alpha^i) < f(x^i) + \delta\alpha^i \frac{df(0)}{d\alpha}$





Recherche en ligne approximative (3)

À l'itération i :

Recherche en ligne par rétrogradation (Armijo)

Require: d^i une direction de descente, x^i , $\delta \in [0, 1]$, $\rho \in]0, 1[$,

$C > 0$

(valeurs par défaut : $\delta = 0.1$, $\rho = 0.5$, $C = 1$)

initialiser le pas : $\alpha = \max(C \times \|\nabla f(x^i)\|, \sqrt{n}/100)$

while $f(x^i + \alpha d^i) \geq f(x^i) + \delta \alpha \nabla f(x^i)^\top d^i$ **do**

 réduire le pas : $\alpha \leftarrow \rho \times \alpha$

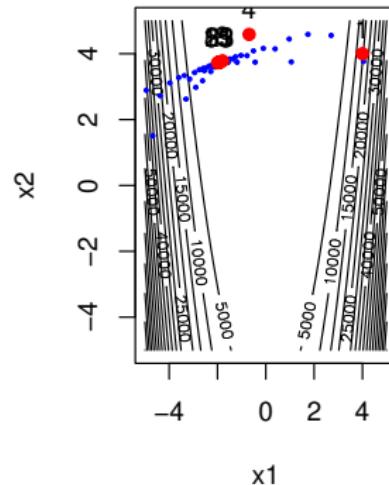
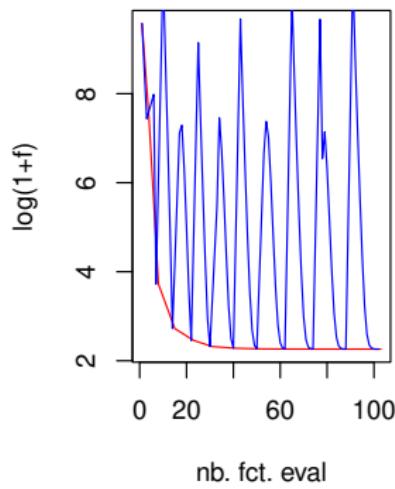
end while

return $\alpha^i \leftarrow \alpha$

Dorénavant, on utilise la recherche en ligne, et le nombre d'appels à f n'est plus égal au numéro d'itération puisque plusieurs évaluations de la fonction peuvent être faites au cours d'une même itération pendant la recherche linéaire.

Recherche en ligne approximative (4)

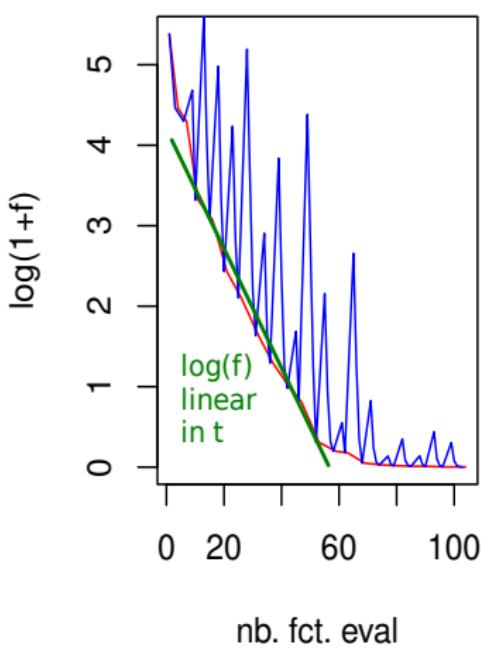
Regardons ce que fait la recherche en ligne sur $f(x) = \text{Rosenbrock}$, où un pas fixe faisait diverger



C'est mieux, mais pas parfait : les oscillations ralentissent fortement les progrès.

Vitesse de convergence du gradient

$f(x) = \frac{1}{2}x^\top Hx$ dans $n = 10$ dimensions, $H > 0$, non aligné avec les axes, nombre de condition = 10.



Empiriquement (pour les démonstrations et plus d'informations cf. [Ravikumar and Singh, 2017]) : sur des fonctions convexes et différentiables, la recherche par gradient avec recherche en ligne progresse à une vitesse telle que $f(x^t) \propto \xi\gamma^t$ où $\gamma \in [0, 1[$. Équivalent à dire que pour obtenir $f(x^t) < \varepsilon$, il faut $t > \mathcal{O}(\log(1/\varepsilon))$. $\log f(x^t) \propto t \log(\gamma) + \log(\xi) \Rightarrow \log(\gamma) < 0$ pente de la courbe verte.

$$\xi\gamma^t < \varepsilon \Leftrightarrow t > \frac{\log(\varepsilon) - \log(\xi)}{\log(\gamma)} = \frac{-1}{\log(\gamma)} \log(\xi/\varepsilon)$$
$$\Rightarrow t > \mathcal{O}(\log(1/\varepsilon)).$$

Oscillations de la descente de gradient

La recherche en ligne parfaite consiste à résoudre

$$\alpha^i = \arg \min_{\alpha > 0} f(\alpha) \text{ où } f(\alpha) = f(x^i + \alpha d^i)$$

Conditions nécessaires pour le pas optimal :

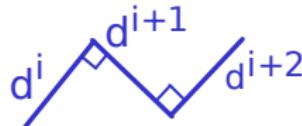
$$\frac{df(\alpha^i)}{d\alpha} = \sum_{j=1}^n \frac{\partial f(x^i + \alpha^i d^i)}{\partial x_j} \frac{\partial x_j}{\partial \alpha} = \nabla f(x^{i+1})^\top \cdot d^i = 0$$

Si la direction est le gradient, alors

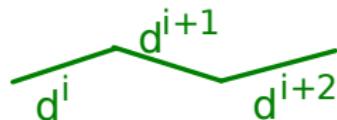


$-d^{i+1}^\top \cdot d^i = 0$ c'est-à-dire d^{i+1} et d^i sont perpendiculaires

gradient
does



less oscillations
seems better



Plan du cours

Introduction à l'optimisation pour l'apprentissage automatique

1 Introduction

- Objectifs, remerciements
- Formulation du problème d'optimisation
- Exemples d'utilisation de l'optimisation
- Concepts mathématiques de base pour l'optimisation

2 Algorithme de la descente la plus raide

- Algorithme de la descente la plus raide à pas fixe
- Recherche de pas (line search)
- Recherche linéaire

3 Recherches améliorées basées sur le gradient

- Directions de recherche pour accélération
- Un mot sur les contraintes
- Rendre la méthode plus globale : redémarrages

4 Application aux réseaux de neurones

5 Bibliographie



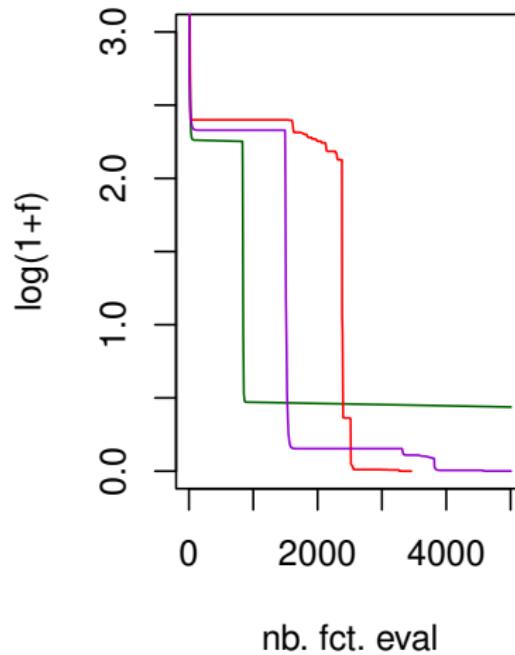
Changer la direction de recherche

Les recherches par gradient améliorées modifient légèrement (mais de manière importante) la direction de recherche par rapport à l'opposé du gradient :

- **Momentum** : la direction de recherche = opposé du gradient déplacé légèrement vers la direction de recherche précédente.
- **Nesterov** [Nesterov, 1983] : la direction de recherche = direction du momentum avec une anticipation du point où sera évalué le prochain gradient.
- **Adam** [Kingma and Ba, 2014] : méthode de pointe en apprentissage profond. Méthode du gradient stochastique avec adaptation indépendante de chaque variable basée sur le momentum.

Comparaison des méthodes (1)

Rosenbrock, $d = 2$: capacité à gérer des vallées incurvées



vert = gradient, rouge = momentum, violet = NAG

Plan du cours

Introduction à l'optimisation pour l'apprentissage automatique

1 Introduction

- Objectifs, remerciements
- Formulation du problème d'optimisation
- Exemples d'utilisation de l'optimisation
- Concepts mathématiques de base pour l'optimisation

2 Algorithme de la descente la plus raide

- Algorithme de la descente la plus raide à pas fixe
- Recherche de pas (line search)
- Recherche linéaire

3 Recherches améliorées basées sur le gradient

- Directions de recherche pour accélération
- Un mot sur les contraintes
- Rendre la méthode plus globale : redémarrages

4 Application aux réseaux de neurones

5 Bibliographie



Un mot sur les contraintes

$$\begin{cases} \min_{x \in \mathcal{S}} f(x) & , \quad \mathcal{S} = \mathbb{R}^n \\ \text{tel que } g_i(x) \leq 0 & , \quad i = 1, m \end{cases}$$

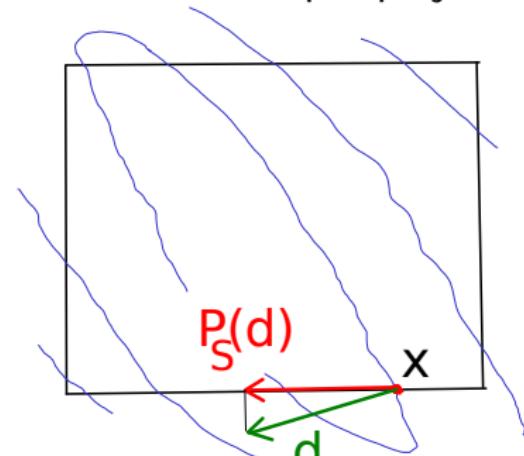
Contraintes de bornes

\mathcal{S} est un hypercube de \mathbb{R}^n , $\mathcal{S} = [LB, UB] \subset \mathbb{R}^n$.

Il peut être décrit par des contraintes, $g_{2i-1}(x) := LB_i - x_i \leq 0$, $g_{2i}(x) := x_i - UB_i \leq 0$, $i = 1, \dots, d$ mais ces contraintes sont si simples qu'elles peuvent être directement traitées par projection.

Si x^i est à une borne et que la direction de recherche d^i l'amène en dehors^a de $\mathcal{S} = [LB, UB]$, projeter le vecteur direction de recherche sur la borne active.

Exercice : comment coder cela ?



^aCela peut même arriver pour une fonction convexe dans un \mathcal{S} convexe, comme le montre le dessin.

Gestion des contraintes par pénalisation (1)

$$\begin{cases} \min_{x \in S \subset \mathbb{R}^d} f(x) \\ \text{tel que } g(x) \leq 0 \end{cases}$$

(notation vectorielle pour les contraintes)

Nous donnons deux techniques pour agréger f et les g_i en une nouvelle fonction objectif (à minimiser).

Fonction de pénalité externe : pénaliser les points qui ne satisfont pas les contraintes

$$f_r(x) = f(x) + r [\max(0, g(x))]^2 , \quad r > 0$$

- Avantages : simple, $\nabla f_r()$ continue au voisinage de la frontière des contraintes (si f et g le sont)
- Inconvénients : convergence par le domaine infaisable (d'où le nom externe), besoin de trouver un r suffisamment grand pour réduire l'infaisabilité, mais pas trop grand à cause des problèmes numériques (forte courbure au voisinage de la contrainte)

Gestion des contraintes par pénalisation (2)

Lagrangien : pour les problèmes sans écart de dualité³, par exemple les problèmes convexes, il existe des multiplicateurs de Lagrange λ^* tels que

$$x^* \in \arg \min_{x \in S} L(x; \lambda^*)$$

$$\text{où } L(x; \lambda^*) := f(x) + \lambda^* g(x)$$

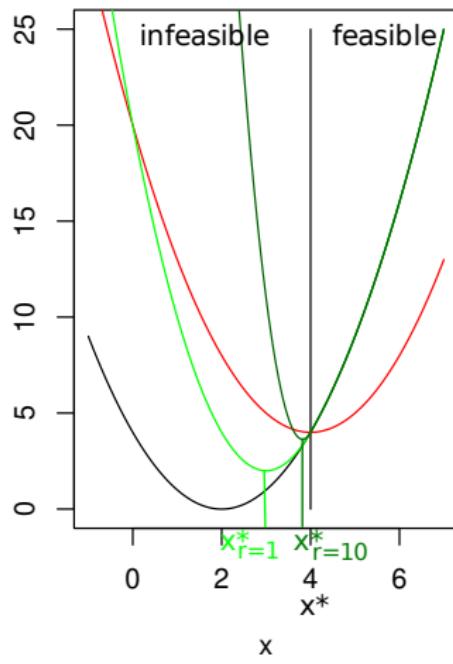
Le Lagrangien $L(\cdot; \lambda^*)$ est (en l'absence d'écart de dualité) une fonction de pénalité valide.

- Avantages : la dualité fournit un moyen de calculer λ^* , conduit à une solution faisable.
- Inconvénients : estimer λ^* a un coût numérique. Pour la plupart des problèmes avec optima locaux, il existe un écart de dualité
⇒ on s'appuie alors sur les lagrangiens augmentés⁴.

³cf. dualité, hors programme de ce cours

Gestion des contraintes par pénalisations (3)

Exemple : $f(x) = (x - 2)^2$, $g(x) = 4 - x \leq 0$, $x^* = 4$, problème convexe



f et g en noir, $L(x; \lambda^* = 4)$ en rouge, pénalité extérieure $f_r()$ avec $r = 1$ et 10 en vert clair et foncé, respectivement.

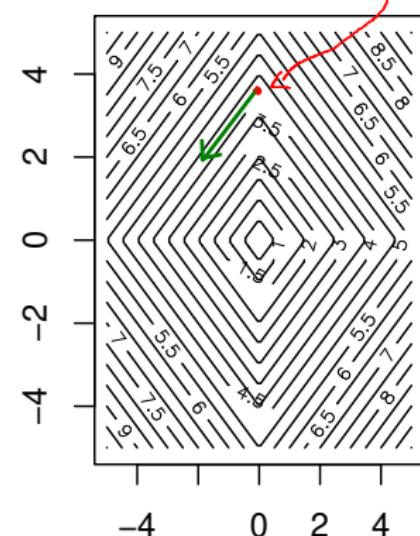
La lagrangienne est ici une pénalité valide.

Quand r augmente, $x_r^* \rightarrow x^*$ mais la courbure de $f_r()$ augmente.

Commentaires sur les algorithmes de descente basés sur le gradient

Utilisation sur des fonctions non différentiables : théoriquement peut converger vers un point qui n'est pas un minimum même pour des fonctions convexes (par ex. si un itéré se trouve à un point anguleux). Cela arrive rarement en pratique. Essayez la fonction $f(x) = \sum_{i=1}^n |x_i|$ ("norme L1") avec le code.

forward finite difference estimation to the gradient:
no progress, stops at



Principal défaut : peut rester bloqué dans des minima locaux.

Redémarrages des recherches locales

Principe simple : redémarrer les recherches par descente à partir de points initiaux choisis aléatoirement.

Utiliser l'aléa pour rendre les recherches déterministes par descente plus robustes.

Un compromis entre 2 extrêmes : local vs global, recherche linéaire vs recherche volumique, spécifique (aux fonctions unimodales différentiables) vs sans hypothèse, efficace vs très lent.

Implémentation simpliste au coût multiplié par nb_restarts :

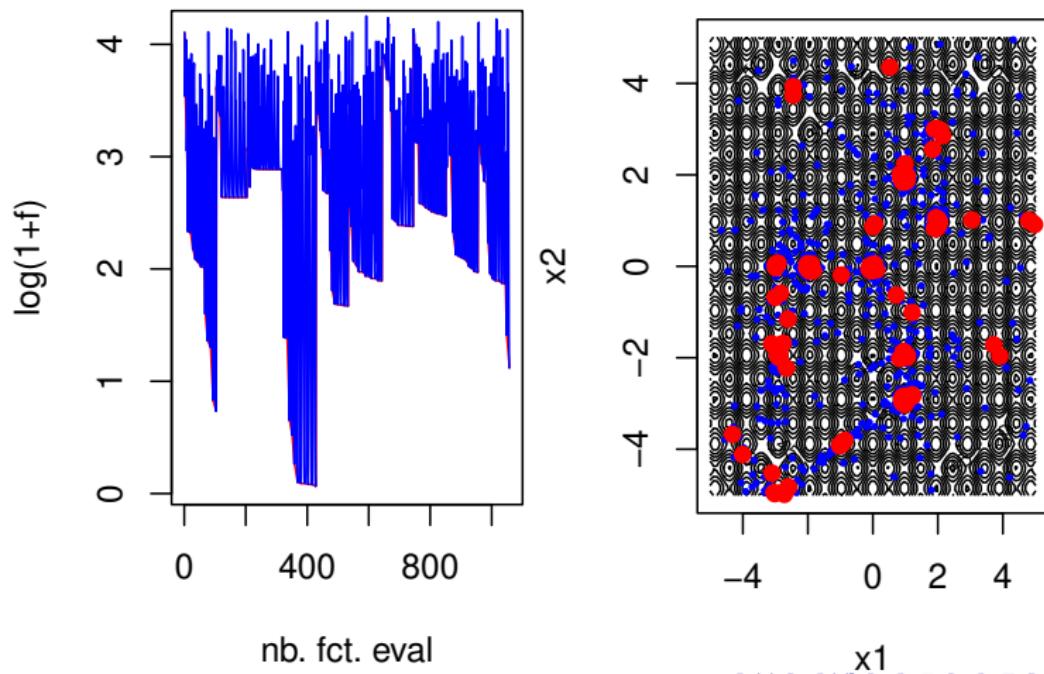
Require: budget, nb_restarts

```
for i in 1 to nb_restarts do
    xinit <- runif(n=d,min=LB,max=UB)
    res<-gradient_descent(xinit,budget=budget/nb_restarts)
    mettre à jour les résultats globaux de la recherche
end for
```

Redémarrages des recherches locales : exemple

Exécution du fichier `restarted_descent`.

```
fun <- rastrigin, d <- 2, budget <- 1000, nb_restart <-  
10 :
```



Application aux réseaux de neurones

Les applications pratiques sont disponibles via le notebook project sur github,

cf. <https://github.com/ML-for-B-E/Optimisation/notebook/project.ipynb>

Conclusions

- L'optimisation numérique est une technique fondamentale pour la prise de décision quantitative, la modélisation statistique, l'apprentissage automatique, ...
- L'engouement pour l'apprentissage automatique a conduit à de très nombreux algorithmes d'optimisation que nous n'avons pas abordés dans ce cours introductif : voir par exemple [Sun et al., 2019, Sra et al., 2012].
- Egalement non traitée mais en émergence : l'optimisation bayésienne pour le réglage des hyperparamètres (constantes de régularisation, nombre de couches de réseaux de neurones, types de neurones, paramètres des algorithmes basés sur le gradient) [Snoek et al., 2012].

Références bibliographiques pour le cours

Ce cours est basé sur

- [Ravikumar and Singh, 2017] : une présentation détaillée et à jour des principaux algorithmes d'optimisation convexe pour l'apprentissage automatique (niveau fin licence, bac +3)
- [Minoux, 2008] : un manuel classique d'optimisation, écrit avant la mode ML mais toujours utile (niveau fin licence / bac+3)
- [Bishop, 2006] : un ouvrage de référence en apprentissage automatique avec quelques pages sur l'optimisation (niveau fin licence / bac+3)
- [Schmidt et al., 2007] : techniques de régularisation L1 (article de recherche)
- [Sun, 2019] : revue des méthodes d'optimisation et bonnes pratiques pour le réglage des réseaux de neurones.

Le contenu de ces références a été simplifié pour ce cours.



Références I

-  Bishop, C. M. (2006).
Pattern recognition and machine learning.
-  Cauchy, A. L. (1847).
Méthode générale pour la résolution des systèmes d'équations simultanées.
Comp. Rend. Sci. Paris, 25(1847):536–538.
-  Curry, H. B. (1944).
The method of steepest descent for non-linear minimization problems.
Quarterly of Applied Mathematics, 2(3):258–261.
-  Fukushima, Y., Cayol, V., Durand, P., and Massonnet, D. (2010).
Evolution of magma conduits during the 1998–2000 eruptions of piton de la fournaise volcano, réunion island.
Journal of Geophysical Research: Solid Earth, 115(B10).
-  Kingma, D. P. and Ba, J. (2014).
Adam: A method for stochastic optimization.
arXiv preprint arXiv:1412.6980.
-  Minoux, M. (2008).
Programmation mathématique. Théorie et algorithmes.
Lavoisier.

Références II



Nesterov, Y. (1983).

A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$.

In *Doklady an USSR*, volume 269, pages 543–547.



Ravikumar, P. and Singh, A. (2017).

Convex optimization.

<http://www.cs.cmu.edu/~pradeepr/convexopt/>.



Schmidt, M., Fung, G., and Rosales, R. (2007).

Fast optimization methods for l1 regularization: A comparative study and two new approaches.

In *European Conference on Machine Learning*, pages 286–297. Springer.



Sgueglia, A., Schmollgruber, P., Bartoli, N., Atinault, O., Benard, E., and Morlier, J. (2018).

Exploration and sizing of a large passenger aircraft with distributed ducted electric fans.

In *2018 AIAA Aerospace Sciences Meeting*, page 1745.



Snoek, J., Larochelle, H., and Adams, R. P. (2012).

Practical bayesian optimization of machine learning algorithms.

Advances in neural information processing systems, 25.

Références III

-  Sra, S., Nowozin, S., and Wright, S. J. (2012).
Optimization for machine learning.
Mit Press.
-  Sun, R. (2019).
Optimization for deep learning: theory and algorithms.
arXiv preprint arXiv:1912.08957.
-  Sun, S., Cao, Z., Zhu, H., and Zhao, J. (2019).
A survey of optimization methods from a machine learning perspective.
IEEE transactions on cybernetics, 50(8):3668–3681.