Optimisation sans (?) dérivées

Clément Royer

Certificat Chef de Projet IA - Université Paris Dauphine-PSL

13 octobre 2022

Exemples d'introduction : SVM linéaire

$$\text{minimiser}_{\boldsymbol{x}\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} \max\{1 - b_i \boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x}, 0\} + \frac{\lambda}{2}\|\boldsymbol{x}\|^2.$$

- $\boldsymbol{a}_i \in \mathbb{R}^d$, $b_i \in \{-1, 1\}$;
- Reformulation en programme quadratique possible.

Exemples d'introduction : SVM linéaire

$$\text{minimiser}_{\boldsymbol{x}\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} \max\{1 - b_i \boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x}, 0\} + \frac{\lambda}{2}\|\boldsymbol{x}\|^2.$$

- $\boldsymbol{a}_i \in \mathbb{R}^d$, $b_i \in \{-1, 1\}$;
- Reformulation en programme quadratique possible.

Appliquer le gradient stochastique

- Intéressant dans le cas de données massives ($n \gg 1$);
- **Problème :** la fonction n'est pas partout dérivable !

Réseaux de neurones peu profond

Un réseau de neurones pour la régression

$$\text{minimiser}_{\boldsymbol{x}\in\mathbb{R}^d} f(\boldsymbol{x}) := \frac{1}{2n}\|\boldsymbol{W}_3\left(\boldsymbol{W}_2\boldsymbol{\sigma}(\boldsymbol{W}_1\boldsymbol{a}_i + \boldsymbol{v}_1) + \boldsymbol{v}_2\right) - \boldsymbol{b}_i\|^2.$$

- $\boldsymbol{a}_i \in \mathbb{R}^{d_a}$, $\boldsymbol{b}_i \in \mathbb{R}^{d_b}$.
- $\boldsymbol{\sigma}$: Activation non linéaire appliquée à chaque composante d'un vecteur.
- $\boldsymbol{x}$ concaténation des paramètres $\boldsymbol{W}_3, \boldsymbol{W}_2$, $\boldsymbol{v}_2$, $\boldsymbol{W}_1$, $\boldsymbol{v}_1$.

Réseaux de neurones peu profond

Un réseau de neurones pour la régression

$$\text{minimiser}_{\boldsymbol{x}\in\mathbb{R}^d} f(\boldsymbol{x}) := \frac{1}{2n}\|\boldsymbol{W}_3\left(\boldsymbol{W}_2\boldsymbol{\sigma}(\boldsymbol{W}_1\boldsymbol{a}_i + \boldsymbol{v}_1) + \boldsymbol{v}_2\right) - \boldsymbol{b}_i\|^2.$$

- $\boldsymbol{a}_i \in \mathbb{R}^{d_a}$, $\boldsymbol{b}_i \in \mathbb{R}^{d_b}$.
- $\boldsymbol{\sigma}$: Activation non linéaire appliquée à chaque composante d'un vecteur.
- $\boldsymbol{x}$ concaténation des paramètres $\boldsymbol{W}_3, \boldsymbol{W}_2$, $\boldsymbol{v}_2$, $\boldsymbol{W}_1$, $\boldsymbol{v}_1$.

Entraînement via le gradient stochastique

- Si $\boldsymbol{\sigma}$ est $\mathcal{C}^1$ (ex: tanh), tout est $\mathcal{C}^1$: formule pour $\nabla f(\boldsymbol{x})$?
- Si $\boldsymbol{\sigma}$ n'est pas $\mathcal{C}^1$ (ex: ReLU), que faire ?

Exemples d'introduction : SVM linéaire (2)

$$\text{minimiser}_{\boldsymbol{x}\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} \max\{1 - b_i \boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x}, 0\} + \frac{\lambda}{2}\|\boldsymbol{x}\|^2.$$

- $\boldsymbol{a}_i \in \mathbb{R}^d$, $b_i \in \{-1, 1\}$.
- Régularisation en $\frac{\lambda}{2}\|\boldsymbol{x}\|^2$.

Exemples d'introduction : SVM linéaire (2)

$$\text{minimiser}_{\boldsymbol{x}\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} \max\{1 - b_i \boldsymbol{a}_i^{\mathrm{T}}\boldsymbol{x}, 0\} + \frac{\lambda}{2}\|\boldsymbol{x}\|^2.$$

- $\boldsymbol{a}_i \in \mathbb{R}^d$, $b_i \in \{-1, 1\}$.
- Régularisation en $\frac{\lambda}{2}\|\boldsymbol{x}\|^2$.

Régularisation

- Compromis entre un terme d'attache aux données et une structure souhaitée.
- Questions :
  - Quelle est la meilleure valeur de λ ?
  - Peut-on optimiser sur λ ?

Les sujets de ce matin

Calcul des dérivées

- Différentiation automatique;
- Logiciels.

Pas de notion de dérivée

- Que faire dans ce contexte ?
- Quels algorithmes ?

Optimisation sans dérivées

- Quels problèmes ?
- Quelles méthodes ?

Plan

Table des matières

Sommaire

Rappel : le gradient

Dans la suite, on considère une fonction $f : \mathbb{R}^d \to \mathbb{R}$ de classe $\mathcal{C}^1$.

Gradient

Soit une fonction $f : \mathbb{R}^d \to \mathbb{R}$ de classe $\mathcal{C}^1$. Le **gradient de f en $\boldsymbol{x}$** est donné par

$$\nabla f(\boldsymbol{x}) = \left[\frac{\partial f}{\partial x_i}(\boldsymbol{x})\right]_{1 \leq i \leq d} \in \mathbb{R}^d.$$

Cadre plus général

On considère une fonction **lisse** (ou douce, ou *smooth*) $f : \mathbb{R}^d \to \mathbb{R}^m$.

Matrice jacobienne

$f : \mathbb{R}^d \to \mathbb{R}^m$ est dérivable en $\boldsymbol{x} \in \mathbb{R}^d$ si il existe une matrice $\boldsymbol{J}_f(\boldsymbol{x}) \in \mathbb{R}^{m \times d}$ telle que

$$\lim_{\substack{\boldsymbol{z} \to \boldsymbol{x} \\ \boldsymbol{z} \neq \boldsymbol{x}}} \frac{\|f(\boldsymbol{z}) - f(\boldsymbol{x}) - \boldsymbol{J}_f(\boldsymbol{x})(\boldsymbol{z} - \boldsymbol{x})\|}{\|\boldsymbol{z} - \boldsymbol{x}\|} = 0.$$

$\boldsymbol{J}_f(\boldsymbol{x})$ s'appelle la (matrice) **jacobienne** de f en $\boldsymbol{x}$.

Cadre plus général

On considère une fonction **lisse** (ou douce, ou *smooth*) $f : \mathbb{R}^d \to \mathbb{R}^m$.

Matrice jacobienne

$f : \mathbb{R}^d \to \mathbb{R}^m$ est dérivable en $\boldsymbol{x} \in \mathbb{R}^d$ si il existe une matrice $\boldsymbol{J}_f(\boldsymbol{x}) \in \mathbb{R}^{m \times d}$ telle que

$$\lim_{\substack{\boldsymbol{z} \to \boldsymbol{x} \\ \boldsymbol{z} \neq \boldsymbol{x}}} \frac{\|f(\boldsymbol{z}) - f(\boldsymbol{x}) - \boldsymbol{J}_f(\boldsymbol{x})(\boldsymbol{z} - \boldsymbol{x})\|}{\|\boldsymbol{z} - \boldsymbol{x}\|} = 0.$$

$\boldsymbol{J}_f(\boldsymbol{x})$ s'appelle la (matrice) **jacobienne** de f en $\boldsymbol{x}$.

Cas particuliers

- $m = 1$: la jacobienne se ramène à un vecteur $\nabla f(\boldsymbol{x}) = \boldsymbol{J}_f(\boldsymbol{x})^{\mathrm{T}}$, que l'on appelle le vecteur gradient;
- $n = m = 1$: la jacobienne est équivalente à un scalaire $f'(x) = \nabla f(\boldsymbol{x}) = \boldsymbol{J}_f(x)$, que l'on appelle la dérivée de f en $\boldsymbol{x}$.

Règles de calcul

Théorème : Dérivée d'une fonction composée

Supposons que $f : \mathbb{R}^d \to \mathbb{R}^m$ s'écrive sous la forme

$$f(\boldsymbol{x}) = \phi_2 \circ \phi_1(\boldsymbol{x}) = \phi_2(\phi_1(\boldsymbol{x})).$$

avec $\phi_1 : \mathbb{R}^d \to \mathbb{R}^{m_1}$, $\phi_2 : \mathbb{R}^{m_2} \to \mathbb{R}^m$. Alors, si ϕ_1 et ϕ_2 sont $\mathcal{C}^1$, f est aussi $\mathcal{C}^1$ avec

$$\boldsymbol{J}_f(\boldsymbol{x}) = \boldsymbol{J}_{\phi_2}(\phi_1(\boldsymbol{x}))\boldsymbol{J}_{\phi_1}(\boldsymbol{x}).$$

Règles de calcul

Théorème : Dérivée d'une fonction composée

Supposons que $f : \mathbb{R}^d \to \mathbb{R}^m$ s'écrive sous la forme

$$f(\boldsymbol{x}) = \phi_2 \circ \phi_1(\boldsymbol{x}) = \phi_2(\phi_1(\boldsymbol{x})).$$

avec $\phi_1 : \mathbb{R}^d \to \mathbb{R}^{m_1}$, $\phi_2 : \mathbb{R}^{m_2} \to \mathbb{R}^m$. Alors, si ϕ_1 et ϕ_2 sont $\mathcal{C}^1$, f est aussi $\mathcal{C}^1$ avec

$$\boldsymbol{J}_f(\boldsymbol{x}) = \boldsymbol{J}_{\phi_2}(\phi_1(\boldsymbol{x}))\boldsymbol{J}_{\phi_1}(\boldsymbol{x}).$$

Extensions

- Compositions multiples : $f(\boldsymbol{x}) = \phi \circ \phi_L \circ \phi_{L-1} \circ \cdots \circ \phi_1(\boldsymbol{x})$;
- Fonctions progressives :

$$f(\boldsymbol{x}) = \phi\left(\phi_L\left(\boldsymbol{x}_L, \phi_{L-1}\left(\boldsymbol{x}_{L-1}, \ldots, \phi_2\left(\boldsymbol{x}_2, \phi_1(\boldsymbol{x}_1)\right)\ldots\right)\right)\right).$$

Approches de calcul de dérivée

Calcul à la main

- Risque d'erreurs;
- Doit être refait dès que le modèle change.

Différentiation symbolique

- Renvoie une expression mathématique (cf Wolfram Alpha);
- Doit être recalculée si le modèle change.

Différentiation numérique

- Exemple : Différences finies.
- Requiert plusieurs évaluations de fonction.

Différentiation automatique

Principe

- Pour une itération de descente de gradient/de gradient stochastique, besoin de la valeur du gradient **en le point courant x_k**.
- La **valeur** $\nabla f(x_k)$ peut se calculer numériquement.

Utilisation de graphes

Principe

- Graphe de calcul d'une fonction;
- Différentiation automatique via le graphe de calcul et la règle de composition.

Exemple

Sommaire

Sur les fonctions non lisses

Définition

Une fonction est dite **non lisse** si elle n'est pas dérivable partout.
NB: Non lisse $\neq$ Discontinue.

Exemples de fonctions non lisses

- $w \mapsto |w|$ de $\mathbb{R}$ dans $\mathbb{R}$;
- $w \mapsto \|w\|_1 = \sum_{i=1}^{d} |w_i|$ de $\mathbb{R}^d$ dans $\mathbb{R}$;
- ReLU: $w \mapsto \max\{w, 0\}$ de $\mathbb{R}^d$ dans $\mathbb{R}$.

Optimisation non lisse

- Un algorithme d'optimisation lisse (ex : descente de gradient) est basé sur des dérivées;
- Non lisse $\Leftrightarrow$ Pas de dérivée en certains points;
- On utilise des notions plus générales de dérivée.

Alternatives

- Si possible, reformuler en un problème lisse :
  Ex) minimiser$_{w\in\mathbb{R}}\ |w|$ se ré-écrit

  minimiser$_{w,t^+,t^-\in\mathbb{R}}\ t^+ - t^-$ subject to $w = t^+ - t^-, t^+ \geq 0, t^- \geq 0.$

- Si la *fonction* est lipschitzienne, elle possède un gradient en presque tous les points (mais souvent pas en les minima).
  Ex) Fonction d'activation des réseaux de neurones
  $ReLU(\boldsymbol{w}) = [\max\{w_i, 0\}]$.

Définition

Soit $f : \mathbb{R}^d \to \mathbb{R}$ une fonction convexe. Un vecteur $\boldsymbol{g} \in \mathbb{R}^d$ est un **sous-gradient** de f en $\boldsymbol{x} \in \mathbb{R}^d$ si

$$\forall \boldsymbol{z} \in \mathbb{R}^d, \qquad f(\boldsymbol{z}) \geq f(\boldsymbol{x}) + \boldsymbol{g}^{\mathrm{T}}(\boldsymbol{z} - \boldsymbol{x}).$$

L'ensemble des sous-gradients de f en $\boldsymbol{x}$ s'appelle le *sous-différentiel* de f en $\boldsymbol{x}$: on le note $\partial f(\boldsymbol{x})$.

- Si f dérivable en $\boldsymbol{x}$, $\partial f(\boldsymbol{x}) = \{\nabla f(\boldsymbol{x})\}$;
- $0 \in \partial f(\boldsymbol{x}) \Leftrightarrow \boldsymbol{x}$ minimum de f!

Exemple : Soit $f : \mathbb{R} \to \mathbb{R}$, $f(x) = |x|$.

$$\partial f(x) = \begin{cases} -1 & \text{si } x < 0 \\ 1 & \text{si } x > 0 \\ [-1, 1] & \text{si } x = 0. \end{cases}$$

Méthode de sous-gradient

Itération pour minimiser$_{\boldsymbol{w}}$ $f(\boldsymbol{w})$, f convex nonsmooth

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha_k \boldsymbol{g}_k, \qquad \boldsymbol{g}_k \in \partial f(\boldsymbol{w}_k).$$

- Dépend du choix du sous-gradient;
- Choix de α_k plus technique (f peut croître dans la direction d'un sous-gradient !).

À retenir

- Certaines méthodes dites "de gradient" se basent en fait sur des sous-gradients;
- Ceux-ci sont bien compris pour des problèmes simples, et dans le cas convexe.

Sommaire

Problème régularisé et exemples

$$\text{minimiser}_{\boldsymbol{x}\in R^d} \underbrace{f(\boldsymbol{x})}_{\textit{perte}} + \underbrace{\lambda\Omega(\boldsymbol{x})}_{\textit{regularisation}} .$$

où $\lambda > 0$ est un paramètre de régularisation.

Exemple : Régularisation *ridge* ou écrêtée

$$\underset{\boldsymbol{x}\in\mathbb{R}^d}{\text{minimize}} \; f(\boldsymbol{x}) + \frac{\lambda}{2}\|\boldsymbol{x}\|^2.$$

Interprétations :

- Revient à ajouter une contrainte sur $\|\boldsymbol{x}\|^2 = \sum_{i=1}^d x_i^2$;
- Favorise les vecteurs $\boldsymbol{x}$ avec des composantes uniformément faibles en amplitude;
- Réduit la variance des solutions par rapport aux données;
- Problème fortement convexe pour λ suffisamment grand.

Problème régularisé et exemples (2)

Régularisation ℓ_0

- Objectif : Trouver $\boldsymbol{x} \in \mathbb{R}^d$ qui colle aux données avec le plus de coefficients nuls possible;
- Problème idéal : $\min_{\boldsymbol{x}} f(\boldsymbol{x}) + \lambda \|\boldsymbol{x}\|_0$, avec $\|\boldsymbol{v}\|_0 := |\{i | [\boldsymbol{v}]_i \neq 0\}|$. Mais la fonction $\|\cdots\|_0$ est non lisse, discontinue et introduit de la combinatoire.

Problème régularisé et exemples (2)

Régularisation ℓ_0

- Objectif : Trouver $\boldsymbol{x} \in \mathbb{R}^d$ qui colle aux données avec le plus de coefficients nuls possible;
- Problème idéal : $\min_{\boldsymbol{x}} f(\boldsymbol{x}) + \lambda \|\boldsymbol{x}\|_0$, avec $\|\boldsymbol{v}\|_0 := |\{i | [\boldsymbol{v}]_i \neq 0\}|$. Mais la fonction $\|\cdots\|_0$ est non lisse, discontinue et introduit de la combinatoire.

Régularisation LASSO

LASSO=Least Absolute Shrinkage and Selection Operator

$$\min_{\boldsymbol{x} \in \mathbb{R}^d} f(\boldsymbol{x}) + \lambda \|\boldsymbol{x}\|_1, \quad \|\boldsymbol{x}\|_1 = \sum_{i=1}^{d} |x_i|.$$

- $\|\cdot\|_1$ convexe, continue, norme;
- Non lisse mais possède des sous-gradients.

Résolution

Cadre : Optimisation composite

$$\underset{\boldsymbol{x}\in\mathbb{R}^d}{\text{minimize}}\, f(\boldsymbol{x}) + \lambda\Omega(\boldsymbol{x}).$$

- $f \in \mathcal{C}^{1,1}$;
- Ω convexe.

Approche proximale

- Classique en optimisation : remplacer un problème par une suite de sous-problèmes plus simples;
- Ici on exploite la "douceur" de f et la structure de Ω pour construire des problèmes que l'on peut résoudre **efficacement**.

Méthode de gradient proximale

Itération

$$\boldsymbol{x}_{k+1} = \underset{\boldsymbol{x}\in\mathbb{R}^d}{\operatorname{argmin}} \left\{ f(\boldsymbol{x}_k) + \nabla f(\boldsymbol{x}_k)^{\mathrm{T}}(\boldsymbol{x} - \boldsymbol{x}_k) + \frac{1}{2\alpha_k}\|\boldsymbol{x} - \boldsymbol{x}_k\|_2^2 + \lambda\Omega(\boldsymbol{x}) \right\}.$$

- Si $\Omega \equiv 0$, la solution est $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k)$: **c'est l'itération de la descente de gradient !**
- En général, une itération coûte un calcul de gradient + **1 résolution de sous-problème**.

Propriétés

- Complexité/Vitesses de convergence;
- Règles de choix de longueur de pas;
- Variantes accélérées/avec sous-gradients.

Exemple de méthode proximale : ISTA

Contexte

- Résoudre $\text{minimize}_{\boldsymbol{x} \in \mathbb{R}^d} f(\boldsymbol{x}) + \lambda \|\boldsymbol{x}\|_1$;
- Problème classique en traitement du signal/de l'image;
- La méthode du gradient proximal a une **forme explicite**.

Exemple de méthode proximale : ISTA

Contexte

- Résoudre $\text{minimize}_{\boldsymbol{x}\in\mathbb{R}^d} f(\boldsymbol{x}) + \lambda\|\boldsymbol{x}\|_1$;
- Problème classique en traitement du signal/de l'image;
- La méthode du gradient proximal a une **forme explicite**.

Itération ISTA : Iterative Soft-Thresholding Algorithm

Définit $\boldsymbol{x}_{k+1}$ par coordonnées: pour tout $i \in \{1, \ldots, d\}$,

$$[\boldsymbol{x}_{k+1}]_i = \begin{cases} [\boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k)]_i + \alpha_k\lambda & \text{si } [\boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k)]_i < -\alpha_k\lambda \\ [\boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k)]_i - \alpha_k\lambda & \text{si } [\boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k)]_i > \alpha_k\lambda \\ 0 & \text{si } [\boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k)]_i \in [-\alpha_k\lambda, \alpha_k\lambda]. \end{cases}$$

Exemple de méthode proximale : ISTA (2)

Mise à jour dans ISTA

- Part du pas de gradient $\boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k)$;
- Applique l'opérateur de *soft-thresholding* $s_{\alpha_k \lambda}(\bullet)$ à chaque coordonnée, avec

$$s_\mu(t) = \begin{cases} t + \mu & \text{si } t < -\mu \\ t - \mu & \text{si } t > \mu \\ 0 & \text{sinon.} \end{cases}$$

- Favorise les composantes nulles.

Variantes de ISTA

- Changement de longueur de pas;
- Ajout de momentum : FISTA (la plus utilisée en pratique).

En bref: Dérivées et optimisation

Problèmes lisses (avec dérivée)

- Différentiation automatique pour problèmes complexes;
- Outil-clé en apprentissage (backpropagation);
- On calcule juste une valeur !

Problèmes non lisses

- Notion de dérivée généralisée;
- Théorie et algorithmes associés;
- Permet notamment l'optimisation composite.

Plan

Terminologie

Quand on parle d'optimisation sans dérivées, on parle de...

1. Derivative-free optimization (DFO);

Terminologie

Quand on parle d'optimisation sans dérivées, on parle de...

1. Derivative-free optimization (DFO);
2. Black-box optimization;

Terminologie

Quand on parle d'optimisation sans dérivées, on parle de...

1. Derivative-free optimization (DFO);
2. Black-box optimization;
3. Surrogate-based optimization;

Terminologie

Quand on parle d'optimisation sans dérivées, on parle de...

1. Derivative-free optimization (DFO);
2. Black-box optimization;
3. Surrogate-based optimization;
4. Response surface methodology/Design of experiments;

Terminologie

Quand on parle d'optimisation sans dérivées, on parle de...

1. Derivative-free optimization (DFO);
2. Black-box optimization;
3. Surrogate-based optimization;
4. Response surface methodology/Design of experiments;
5. Automated machine learning;

Terminologie

Quand on parle d'optimisation sans dérivées, on parle de...

1. Derivative-free optimization (DFO);
2. Black-box optimization;
3. Surrogate-based optimization;
4. Response surface methodology/Design of experiments;
5. Automated machine learning;
6. Hyperparameter tuning.

Terminologie

Quand on parle d'optimisation sans dérivées, on parle de...

1. Derivative-free optimization (DFO);
2. Black-box optimization;
3. Surrogate-based optimization;
4. Response surface methodology/Design of experiments;
5. Automated machine learning;
6. Hyperparameter tuning.

Nous allons parler de...

- **Tout cela** dans un même cadre;
- **Des avancées pour 1+2**;
- Du lien avec 5+6.

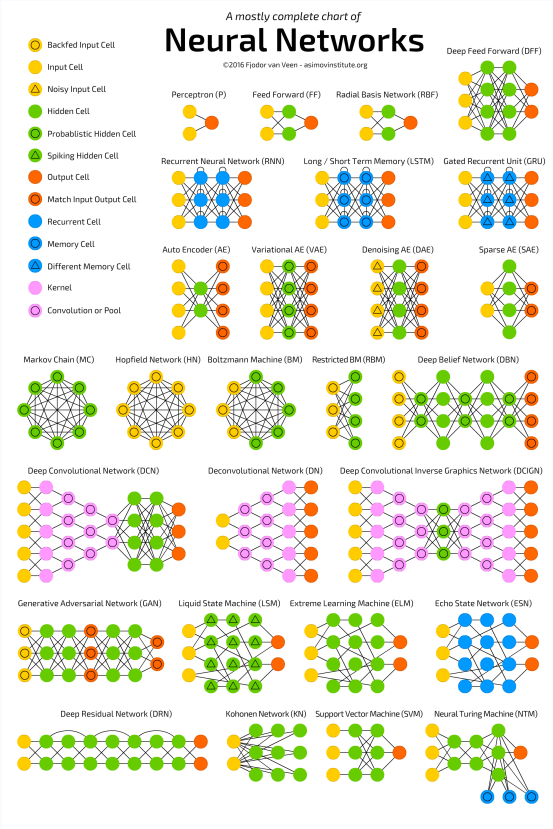
1. Différentiation
   - Calcul de gradient
   - Analyse non lisse
   - Application à la régularisation

2. L'optimisation sans dérivées
   - Exemples et définition
   - Méthodes de recherche directe
   - Méthodes basées sur des modèles

Ce dont tout le monde parle : IA automatisée

Entraînons un réseau de neurones...

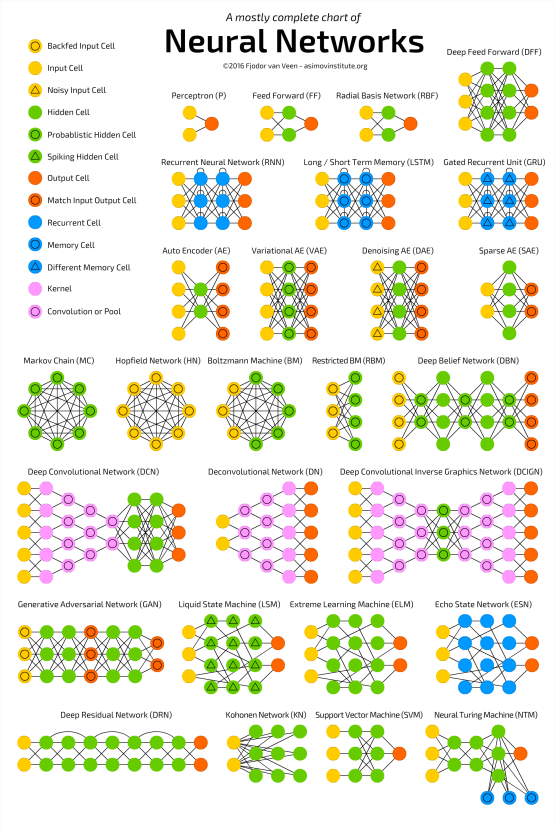
- Quelle architecture ? (nombre de couches, types de couches, etc)
- Quel algorithme d'entraînement ? (Adam, RMSProp, SGD, etc)
- Quelles options pour l'algorithme (*learning rate*, etc)?



Ce dont tout le monde parle : IA automatisée

Entraînons un réseau de neurones...

- Quelle architecture ? (nombre de couches, types de couches, etc)
- Quel algorithme d'entraînement ? (Adam, RMSProp, SGD, etc)
- Quelles options pour l'algorithm (*learning rate*, etc)?



Calibration d'hyperparamètres

- Chaque test d'une configuration correspond à un nouvel entraînement (heures/jours en temps CPU + argent !);
- Énormément de choix possibles.

Un autre point de vue

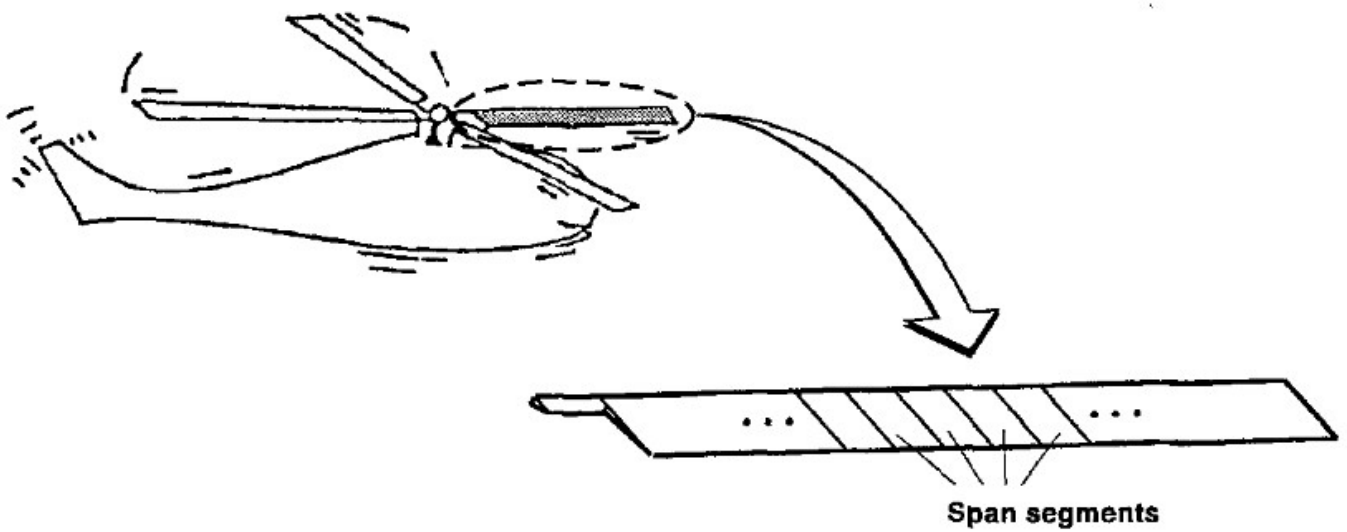
Calcul scientifique

- Usage intensif de la simulation par ordinateur (CFD, CAO) dans les applications de type physique (aéronautique, automobile, météorologie);
- Beaucoup de paramètres à calibrer.

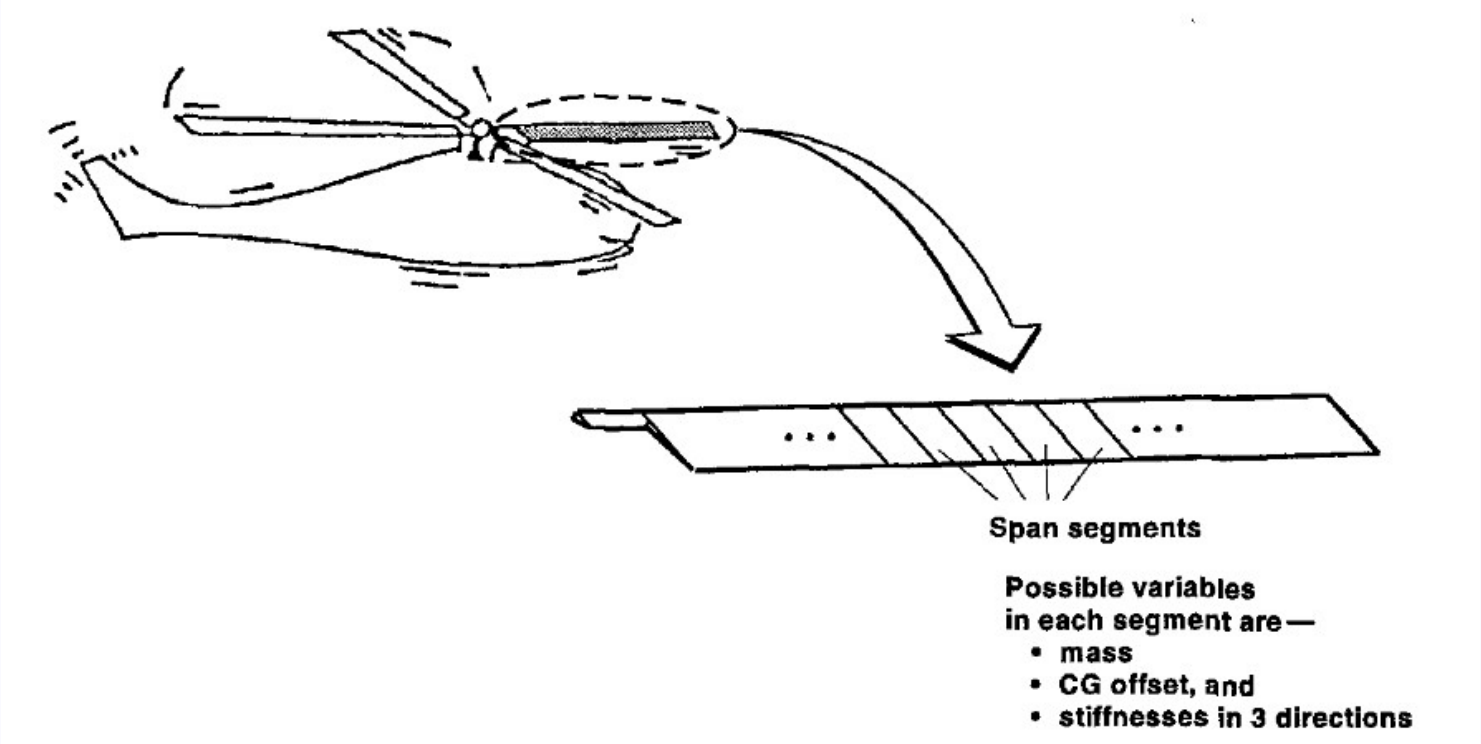
Optimisation de simulateurs

- Besoin : optimiser les paramètres de codes de simulations;
- Coût d'exécution des simulateurs élevé;
- Parfois plusieurs versions à coût variable (multifidélité).

Exemple classique : Design de pale d'hélicoptère (Booker et al. 1998)



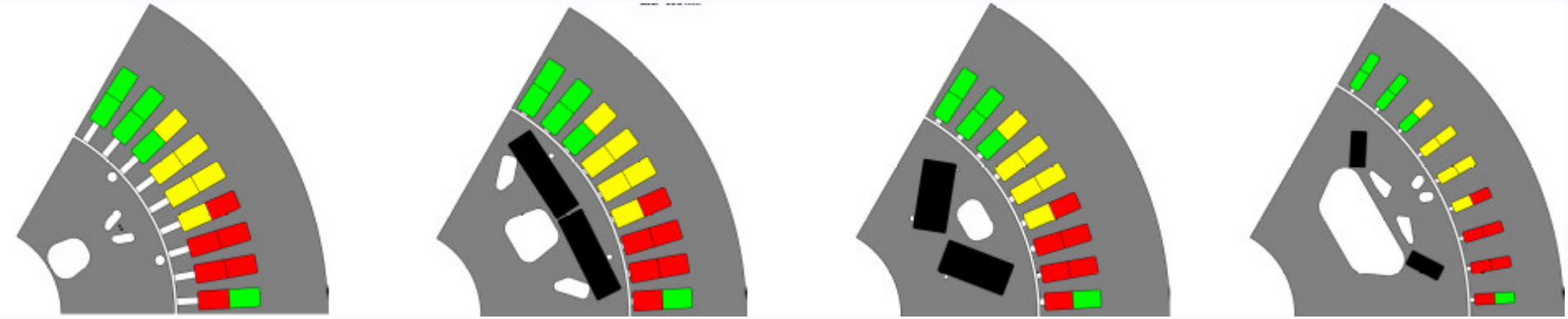
Exemple classique : Design de pale d'hélicoptère (Booker et al. 1998)



- 30 paramètres;
- 1 simulation : 2 semaines de calculs CFD;
- Échec de la simulation 60% du temps.

- Optimisation multi-disciplinaire : codes imbriqués;
- De la simulation numérique, beaucoup de calculs...
- qui peuvent échouer !

Moteur électrique (D. Gaudrie, Stellantis)



- Environ 50 paramètres (continus);
- Multiobjectif (3 objectifs), 6 fonctions de contraintes;
- La plupart des points ne sont pas réalisables !
- 1 simulation $\approx$ 5 minutes;
- Optimisation (par algorithmes génétiques) : 3 semaines !

Une classe de problèmes en commun

Points communs : IA automatisée et optimisation de simulateurs

- Effort de calcul conséquent, basé sur la simulation;
- Choix des meilleurs paramètres non trivial;
- Préliminaire à la construction/au déploiement du système.

En termes d'optimisation

- **Le choix des meilleurs paramètres peut se formuler comme un problème d'optimisation**;
- La fonction objectif de ce problème est très coûteuse à évaluer (en temps de simulation).

Une définition

Optimisation sans dérivées

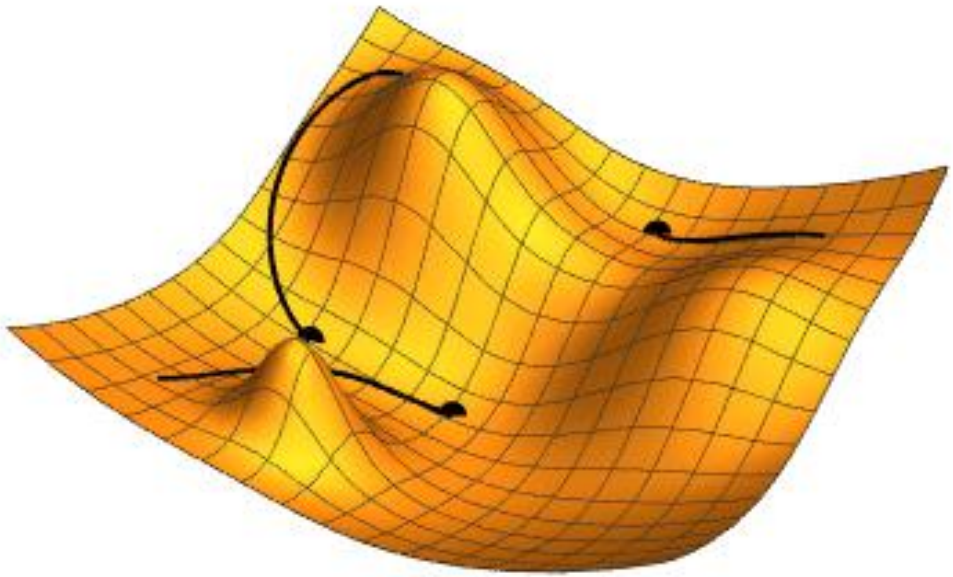
Certaines dérivées ne peuvent pas être exploitées pour optimiser.

Une définition

Optimisation sans dérivées

Certaines dérivées ne peuvent pas être exploitées pour optimiser.

- L'optimisation est fortement basée sur les dérivées (gradient, conditions d'optimalité);
- **Certaines** : il en suffit d'une !

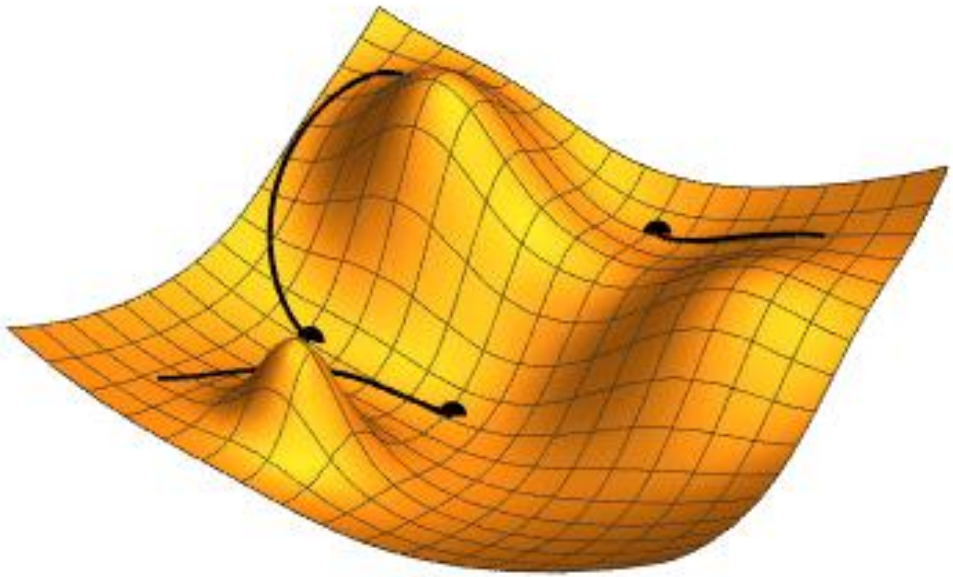


Une définition

Optimisation sans dérivées

Certaines dérivées ne peuvent pas être exploitées pour optimiser.

- L'optimisation est fortement basée sur les dérivées (gradient, conditions d'optimalité);
- **Certaines** : il en suffit d'une !
- **ne peuvent pas** : qu'elles existent ou non !



Avertissement

Toujours préférable d'utiliser les dérivées si possible !

Avertissement

Toujours préférable d'utiliser les dérivées si possible !

1. Expression analytique $\Rightarrow$ Dérivées explicites;

Avertissement

Toujours préférable d'utiliser les dérivées si possible !

1. Expression analytique $\Rightarrow$ Dérivées explicites;
2. Approximation numérique par différences finies;

Avertissement

Toujours préférable d'utiliser les dérivées si possible !

1. Expression analytique $\Rightarrow$ Dérivées explicites;
2. Approximation numérique par différences finies;
3. Puissance de la différentiation automatique/symbolique.

Avertissement

Toujours préférable d'utiliser les dérivées si possible !

1. Expression analytique ⇒ Dérivées explicites;
2. Approximation numérique par différences finies;
3. Puissance de la différentiation automatique/symbolique.

Dérivées non disponibles

1. Systèmes complexes ⇒ Risque d'erreur dans le code à la main;
2. Évaluations coûteuses/bruitées ⇒ Problème pour les différences finies;
3. Code propriétaire ⇒ Pas de diff. auto.

Ce dont on ne parlera pas

Algorithmes génétiques/évolutionnaires

- Souvent inspirés par la nature;
- Efficaces avec peu de variables et des évaluations peu coûteuses/parallélisables;
- Beaucoup d'heuristiques inspirées par la nature.

Ce dont on ne parlera pas

Algorithmes génétiques/évolutionnaires

- Souvent inspirés par la nature;
- Efficaces avec peu de variables et des évaluations peu coûteuses/parallélisables;
- Beaucoup d'heuristiques inspirées par la nature.

Une méthode remarquable : CMA-ES

- Maintient une matrice de covariance;
- Efficace et populaire;
- Récemment interprétée comme une méthode de gradient appliquée à de l'optimisation sur des distributions (chercheurs en IA).

Notre formulation

$$\text{minimiser}_{\boldsymbol{x} \in \mathbb{R}^d} f(\boldsymbol{x}) \quad \text{s. c.} \quad \boldsymbol{x} \in \mathcal{F}$$

- $\boldsymbol{x}$: variables;
- f: fonction objectif;
- $\mathcal{F}$: ensemble admissible.

Problème

Notre formulation

$$\text{minimiser}_{\boldsymbol{x} \in \mathbb{R}^d} f(\boldsymbol{x}) \quad \text{s. c.} \quad \boldsymbol{x} \in \mathcal{F}$$

- $\boldsymbol{x}$: variables;
- f: fonction objectif;
- $\mathcal{F}$: ensemble admissible.

Cadre de travail

- f minorée sur $\mathcal{F}$: $f(\boldsymbol{x}) \geq f_{\text{low}}$;
- Evaluations de f coûteuses :
  - Entraîner un réseau de neurones;
  - Prise de sang d'un patient;
  - Ramener un véhicule en usine.

Notre objectif ?

Trouver les paramètres optimaux ?

- Sans dérivées, pas de garantie d'optimalité locale...
- Optimalité globale seulement garantie sous d'autres hypothèses (convexité) ou si l'on attend indéfiniment.

Notre objectif ?

Trouver les paramètres optimaux ?

- Sans dérivées, pas de garantie d'optimalité locale...
- Optimalité globale seulement garantie sous d'autres hypothèses (convexité) ou si l'on attend indéfiniment.

Trouver une meilleure configuration ?

- Toute amélioration est bonne à prendre;
- Les configurations de départ peuvent être fixées par des experts, et difficiles à améliorer.

Notre objectif ?

Trouver les paramètres optimaux ?

- Sans dérivées, pas de garantie d'optimalité locale...
- Optimalité globale seulement garantie sous d'autres hypothèses (convexité) ou si l'on attend indéfiniment.

Trouver une meilleure configuration ?

- Toute amélioration est bonne à prendre;
- Les configurations de départ peuvent être fixées par des experts, et difficiles à améliorer.

Fournir des garanties

- Validation de principe, guide pour choisir des méthodes;
- Métrique du moment : **complexité**.

Complexité (pour cette séance)

Définition

A partir

- d'un critère de convergence/d'arrêt;
- d'une précision $\epsilon > 0$;
- d'un algorithme itératif $\{\boldsymbol{x}_k\}_k$;

borner le **nombre d'appels de fonction** requis dans le pire des cas pour satisfaire le critère avec précision ϵ.

Complexité (pour cette séance)

Définition

A partir

- d'un critère de convergence/d'arrêt;
- d'une précision $\epsilon > 0$;
- d'un algorithme itératif $\{\boldsymbol{x}_k\}_k$;

borner le **nombre d'appels de fonction** requis dans le pire des cas pour satisfaire le critère avec précision ϵ.
La borne (en tant que fonction de ϵ) s'appelle la **complexité au pire cas** de l'algorithme.

Complexité (pour cette séance)

Définition

A partir

- d'un critère de convergence/d'arrêt;
- d'une précision $\epsilon > 0$;
- d'un algorithme itératif $\{\boldsymbol{x}_k\}_k$;

borner le **nombre d'appels de fonction** requis dans le pire des cas pour satisfaire le critère avec précision ϵ.
La borne (en tant que fonction de ϵ) s'appelle la **complexité au pire cas** de l'algorithme.

Exemples de critère de convergence

- Gradient de f: $\|\nabla f(\boldsymbol{x}_k)\|$;
- Valeur de f: $f(\boldsymbol{x}_k)$.

Plan

Recherche par grille & aléatoire

But : Résoudre $\text{minimiser}_{\boldsymbol{x} \in \mathcal{F}} f(\boldsymbol{x})$ avec accès à f uniquement, budget limité.

Algorithme de recherche basique

Start with: $\hat{\boldsymbol{x}}_0 = \boldsymbol{x}_0 \in \mathcal{F}$, $f = f(\boldsymbol{x}_0)$, $k = 0$.

1. Calculer $\boldsymbol{x}_{k+1}$ et évaluer $f(\boldsymbol{x}_{k+1})$.

Recherche par grille & aléatoire

But : Résoudre minimiser$_{\boldsymbol{x}\in\mathcal{F}} f(\boldsymbol{x})$ avec accès à f uniquement, budget limité.

Algorithme de recherche basique

Start with: $\hat{\boldsymbol{x}}_0 = \boldsymbol{x}_0 \in \mathcal{F}$, $f = f(\boldsymbol{x}_0)$, $k = 0$.

1. Calculer $\boldsymbol{x}_{k+1}$ et évaluer $f(\boldsymbol{x}_{k+1})$.
2. Si $f(\boldsymbol{x}_{k+1}) < f(\hat{\boldsymbol{x}}_k)$ poser $\hat{\boldsymbol{x}}_{k+1} = \boldsymbol{x}_{k+1}$, sinon poser $\hat{\boldsymbol{x}}_{k+1} = \hat{\boldsymbol{x}}_k$.

Recherche par grille & aléatoire

But : Résoudre minimiser$_{\boldsymbol{x}\in\mathcal{F}} f(\boldsymbol{x})$ avec accès à f uniquement, budget limité.

Algorithme de recherche basique

Start with: $\hat{\boldsymbol{x}}_0 = \boldsymbol{x}_0 \in \mathcal{F}$, $f = f(\boldsymbol{x}_0)$, $k = 0$.

1. Calculer $\boldsymbol{x}_{k+1}$ et évaluer $f(\boldsymbol{x}_{k+1})$.
2. Si $f(\boldsymbol{x}_{k+1}) < f(\hat{\boldsymbol{x}}_k)$ poser $\hat{\boldsymbol{x}}_{k+1} = \boldsymbol{x}_{k+1}$, sinon poser $\hat{\boldsymbol{x}}_{k+1} = \hat{\boldsymbol{x}}_k$.
3. Si budget dépassé terminer, sinon incrémenter k de 1.

Recherche par grille & aléatoire

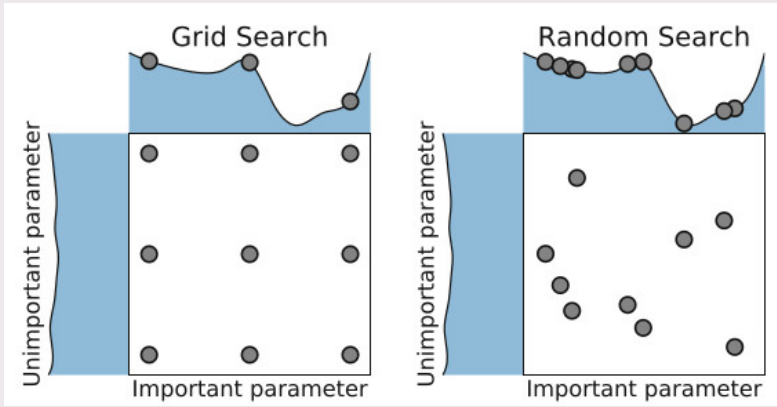
But : Résoudre minimiser$_{\boldsymbol{x}\in\mathcal{F}} f(\boldsymbol{x})$ avec accès à f uniquement, budget limité.

Algorithme de recherche basique

Start with: $\hat{\boldsymbol{x}}_0 = \boldsymbol{x}_0 \in \mathcal{F}$, $f = f(\boldsymbol{x}_0)$, $k = 0$.

1. Calculer $\boldsymbol{x}_{k+1}$ et évaluer $f(\boldsymbol{x}_{k+1})$.
2. Si $f(\boldsymbol{x}_{k+1}) < f(\hat{\boldsymbol{x}}_k)$ poser $\hat{\boldsymbol{x}}_{k+1} = \boldsymbol{x}_{k+1}$, sinon poser $\hat{\boldsymbol{x}}_{k+1} = \hat{\boldsymbol{x}}_k$.
3. Si budget dépassé terminer, sinon incrémenter k de 1.

- **Grille (Grid search)** : Valeurs des $\boldsymbol{x}_k$ fixées a priori;
- **Aléatoire (Random search)** : Tirer $\boldsymbol{x}_{k+1}$ au hasard.



Théorème

Soit $f^* = \min_{\boldsymbol{x} \in \mathcal{F}} f(\boldsymbol{x})$. Alors,

$$\mathbb{P}\left(f(\hat{\boldsymbol{x}}_K) \leq f^* + \epsilon\right) \geq p$$

après

$$K = \frac{\ln(p)}{\ln\left[\frac{\mu(\{x \in \mathcal{F} | f(x) > f^* + \epsilon\})}{\mu(\mathcal{F})}\right]}.$$

itérations.

Recherche aléatoire : garanties

Théorème

Soit $f^* = \min_{\boldsymbol{x} \in \mathcal{F}} f(\boldsymbol{x})$. Alors,

$$\mathbb{P}\left(f(\hat{\boldsymbol{x}}_K) \leq f^* + \epsilon\right) \geq p$$

après

$$K = \frac{\ln(p)}{\ln\left[\frac{\mu(\{x \in \mathcal{F} | f(x) > f^* + \epsilon\})}{\mu(\mathcal{F})}\right]}.$$

itérations.

- Plus : Valable pour f quelconque !
- Moins : Beaucoup d'itérations/d'évaluations de f ;
- Le budget est consommé en exploration.

Recherche directe *(direct search)*

Recherche exploratoire

- Valables en petite dimension;
- Algorithmes exploratoires.

Recherche directe *(direct search)*

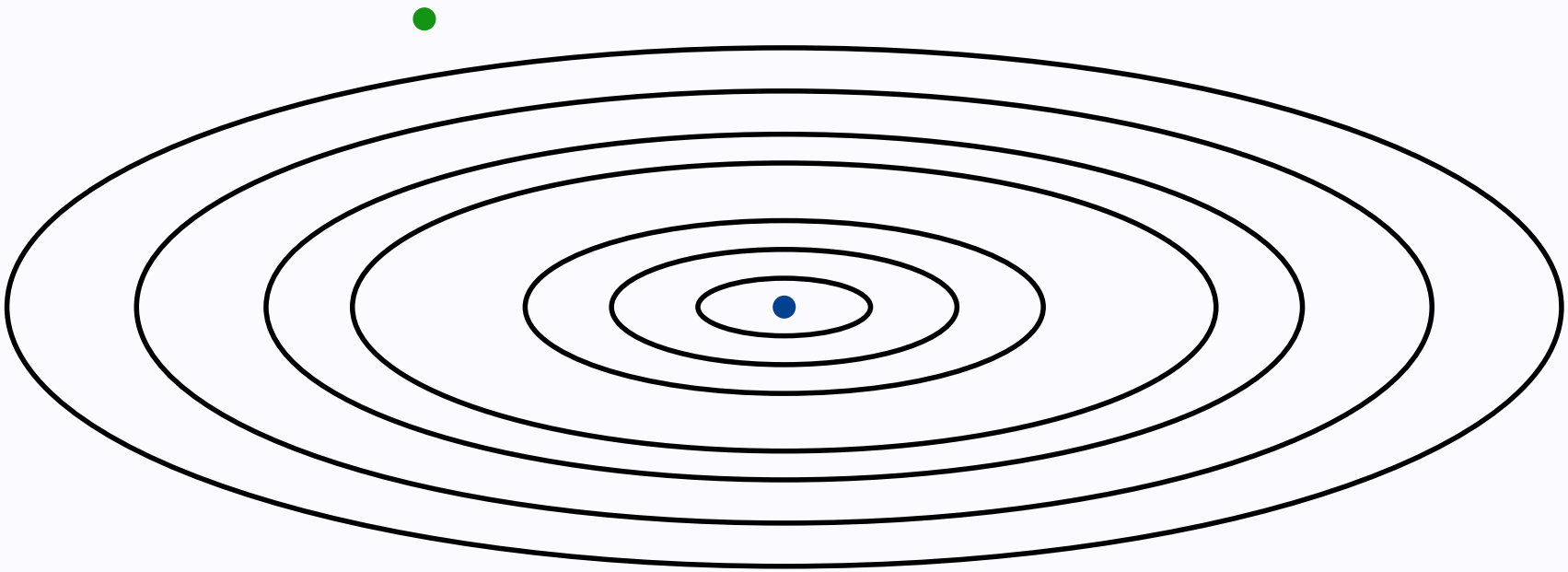
Recherche exploratoire

- Valables en petite dimension;
- Algorithmes exploratoires.

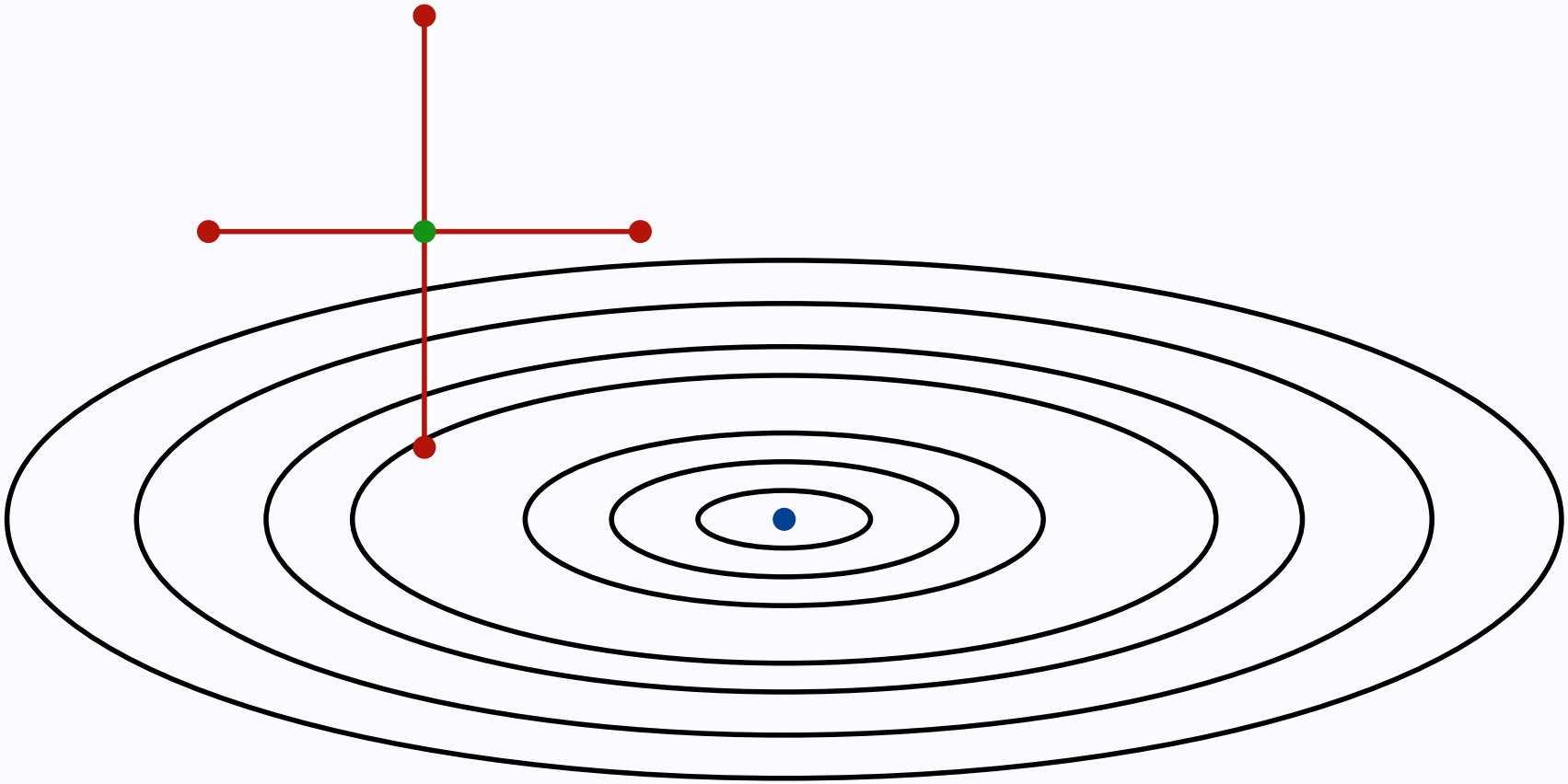
Recherche directe

- Origine : années 1960, théorie : années 1990;
- But : Aller au-delà de l'exploration pure.
- Intérêt : simplicité, parallélisme;
- La méthode du simplexe (Nelder-Mead, 1965) a plus de 125000 citations et est toujours la méthode sans dérivées de MATLAB!

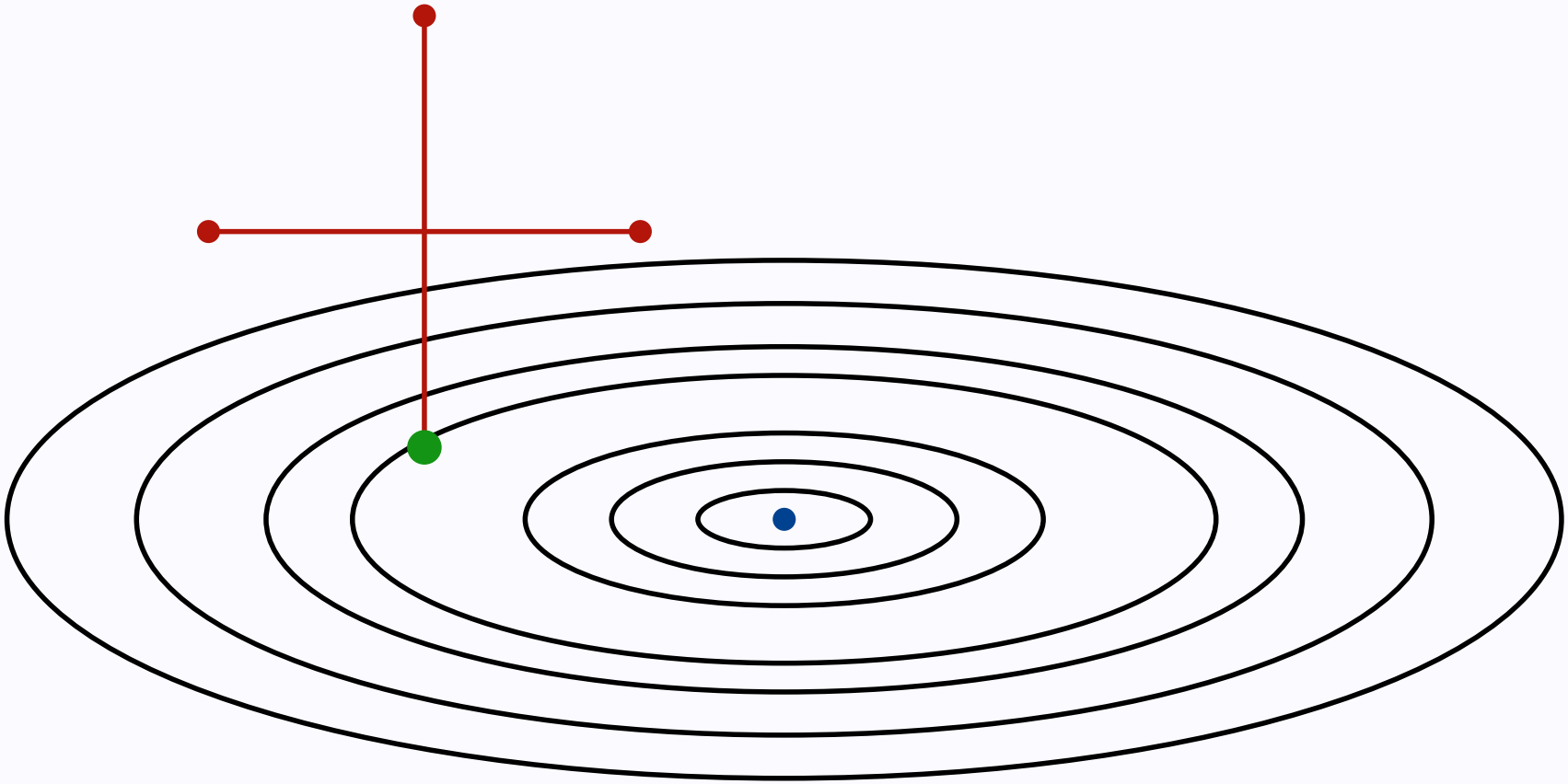
Exemple : Recherche par coordonnées



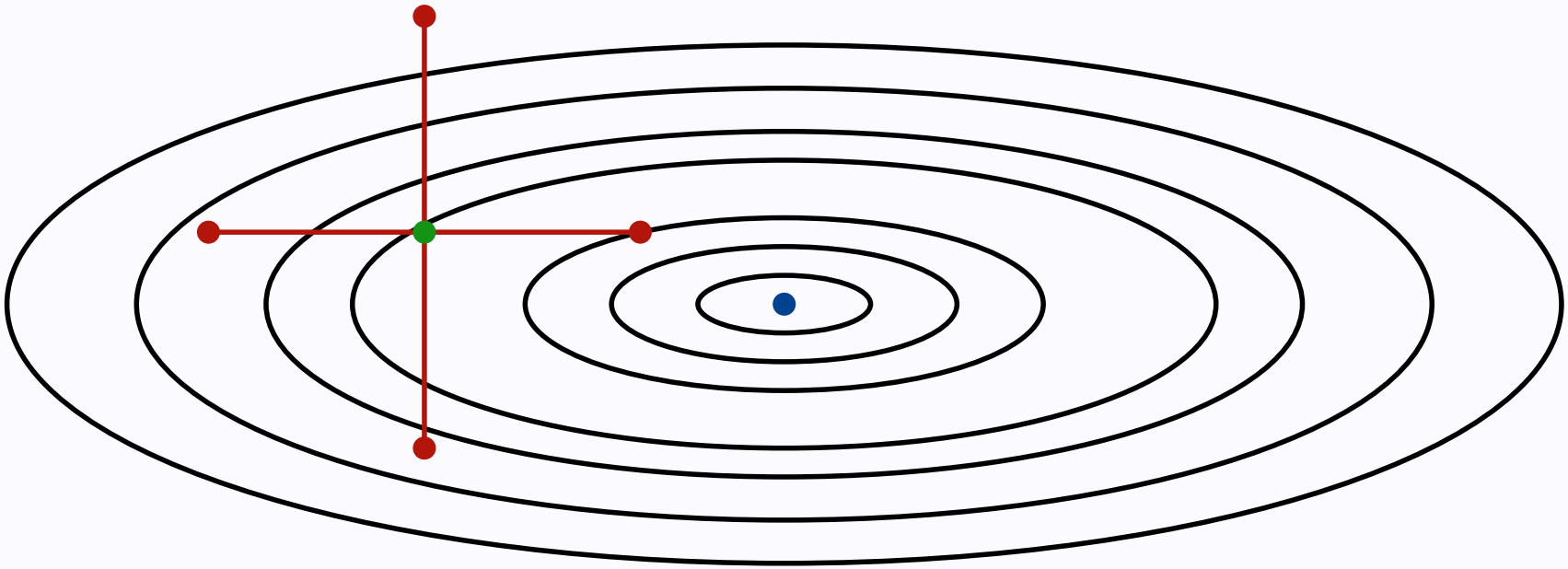
Exemple : Recherche par coordonnées



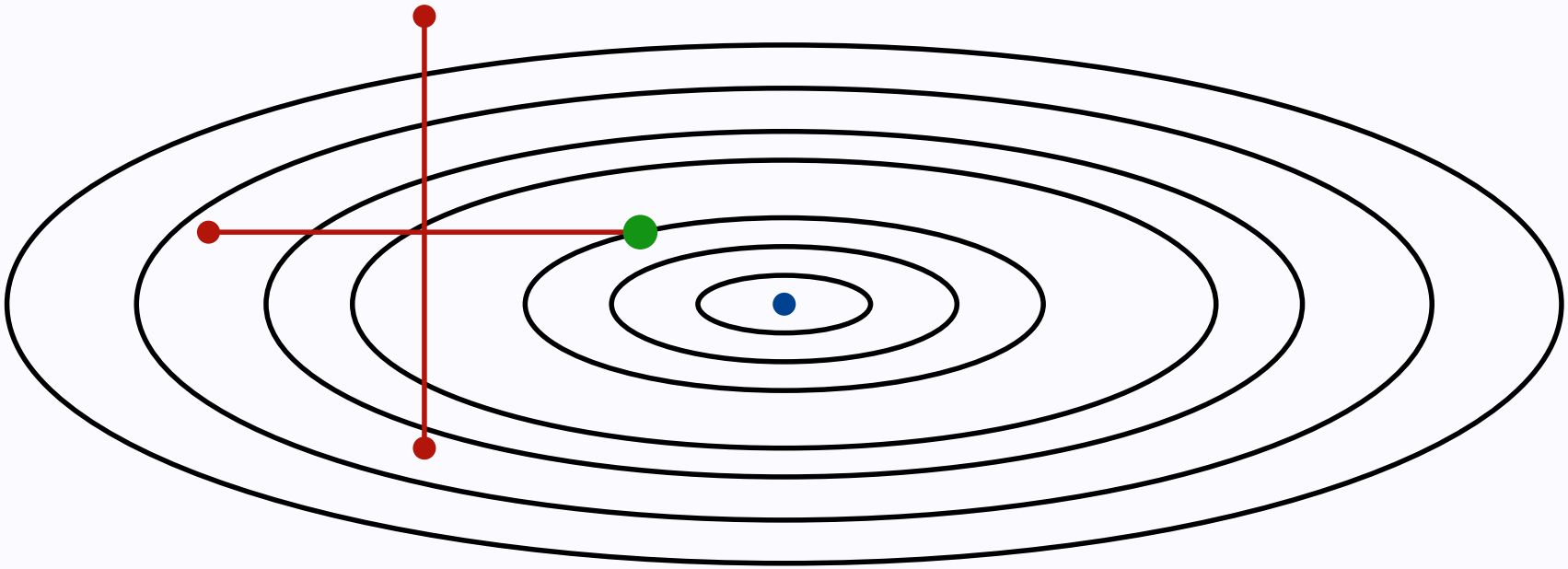
Exemple : Recherche par coordonnées



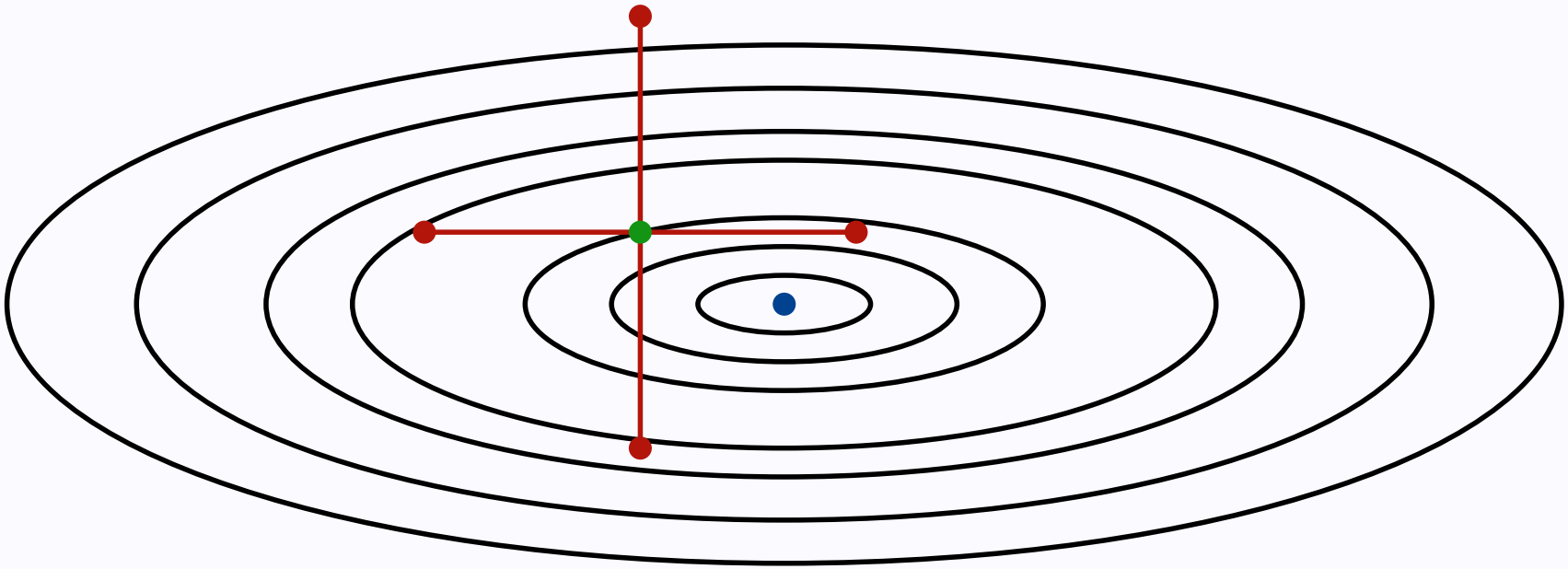
Exemple : Recherche par coordonnées



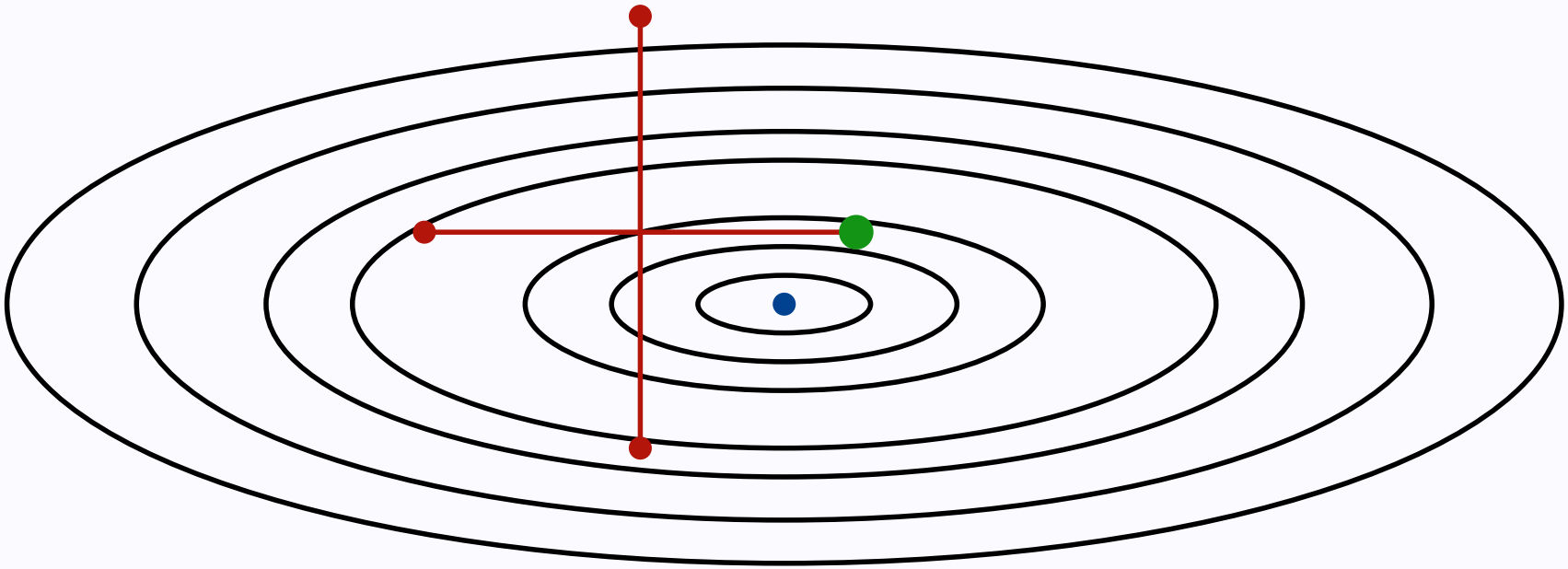
Exemple : Recherche par coordonnées



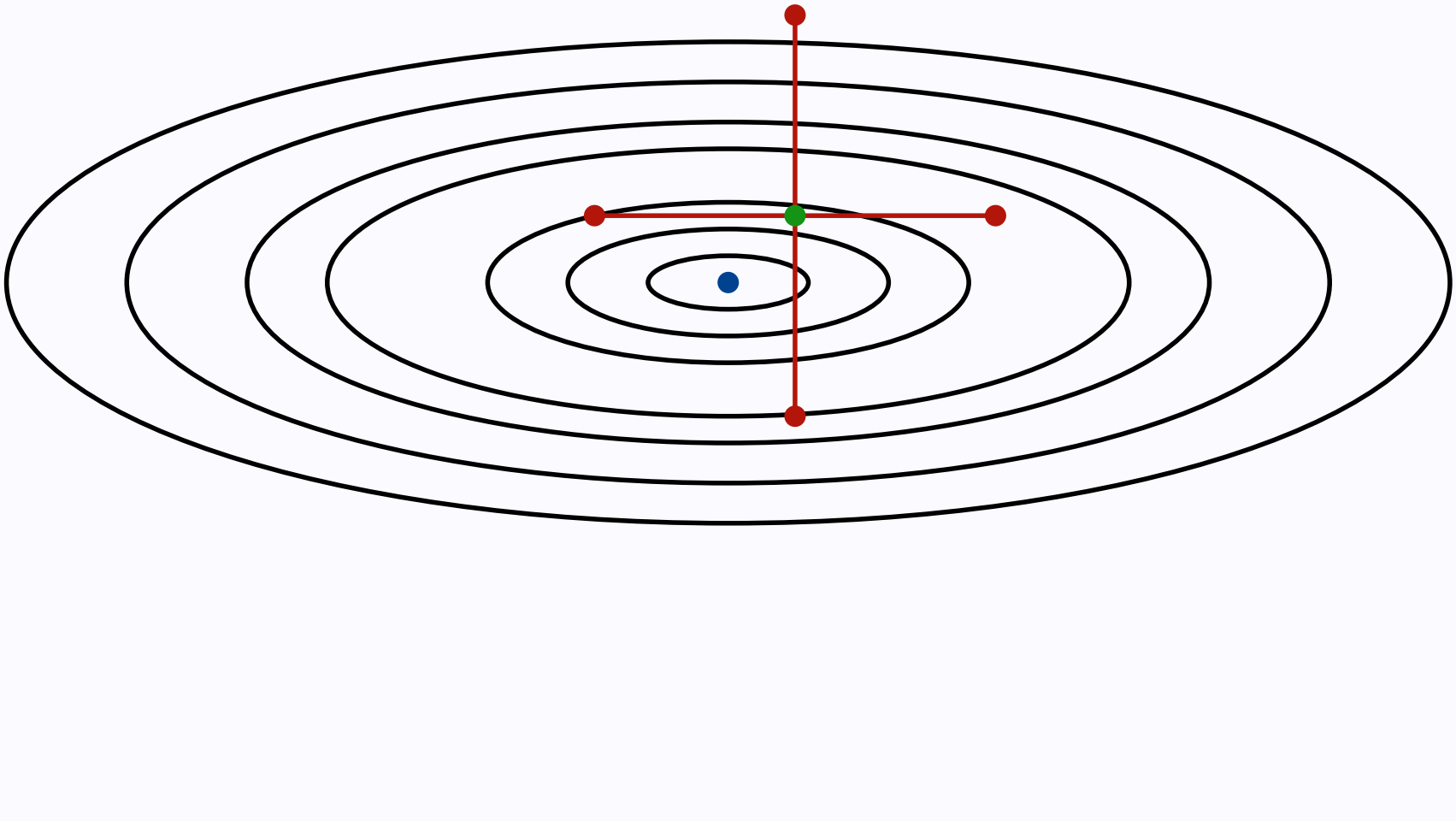
Exemple : Recherche par coordonnées



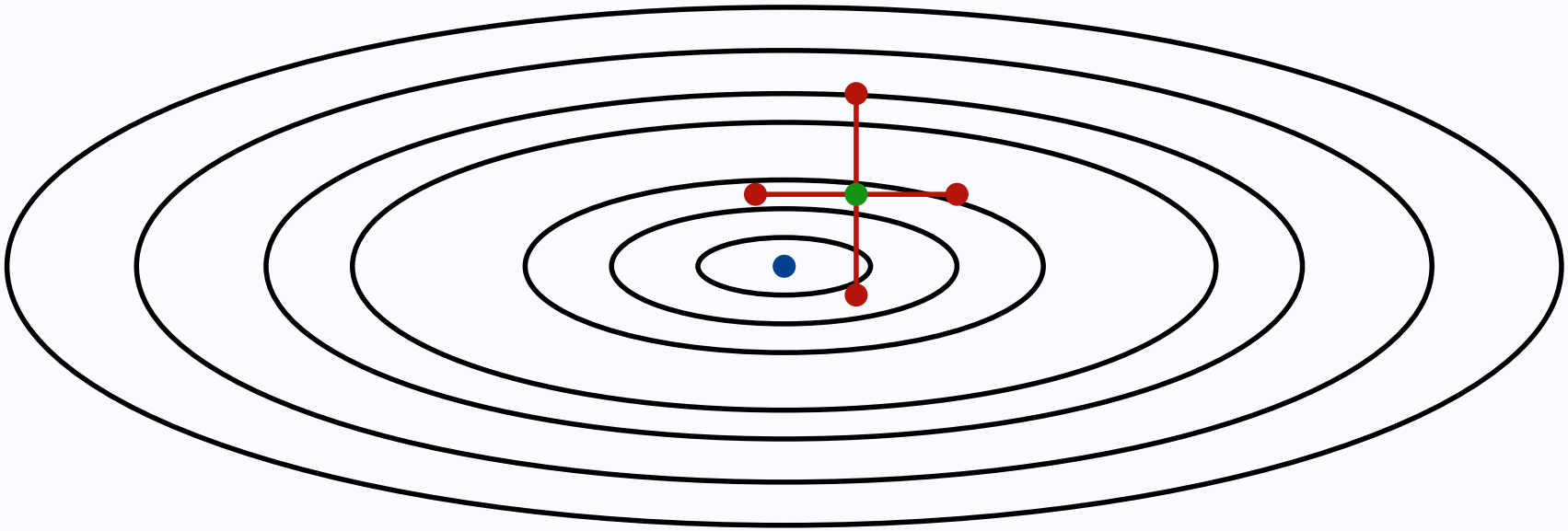
Exemple : Recherche par coordonnées



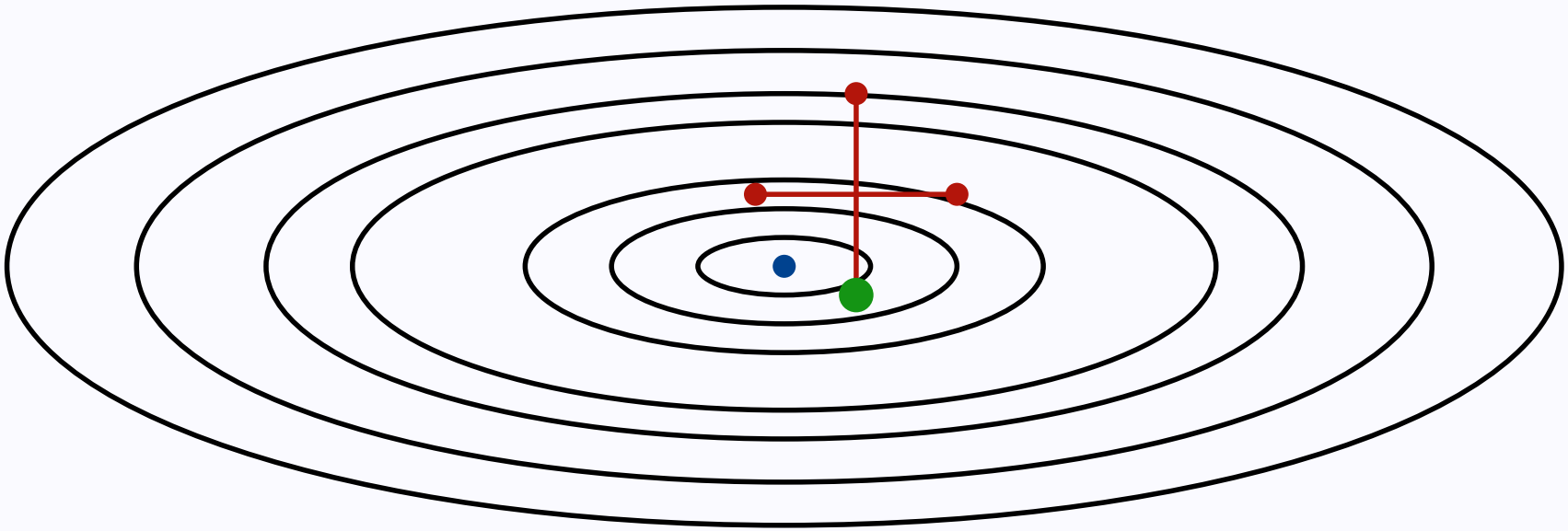
Exemple : Recherche par coordonnées



Exemple : Recherche par coordonnées



Exemple : Recherche par coordonnées



Algorithme de recherche directe basique

1. **Initialisation :** $x_0 \in \mathbb{R}^d, \alpha_0 > 0$.
2. **Pour** $k = 0, 1, 2, ...$
   - Choisir un ensemble $\mathcal{D}_k \subset \mathbb{R}^d$ de r directions.

Algorithme de recherche directe basique

1. **Initialisation :** $\boldsymbol{x}_0 \in \mathbb{R}^d, \alpha_0 > 0$.
2. **Pour** $k = 0, 1, 2, \ldots$
   - Choisir un ensemble $\mathcal{D}_k \subset \mathbb{R}^d$ de r directions.
   - **SI** il existe $\boldsymbol{d}_k \in \mathcal{D}_k$ tel que

$$f(\boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k) < f(\boldsymbol{x}_k) - \alpha_k^2,$$

   alors poser $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$ et $\alpha_{k+1} \geq \alpha_k$ *(itération réussie)*.

Algorithme de recherche directe basique

1. **Initialisation :** $\boldsymbol{x}_0 \in \mathbb{R}^d, \alpha_0 > 0$.
2. **Pour** $k = 0, 1, 2, ...$
   - Choisir un ensemble $\mathcal{D}_k \subset \mathbb{R}^d$ de r directions.
   - **SI** il existe $\boldsymbol{d}_k \in \mathcal{D}_k$ tel que

$$f(\boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k) < f(\boldsymbol{x}_k) - \alpha_k^2,$$

     alors poser $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$ et $\alpha_{k+1} \geq \alpha_k$ *(itération réussie)*.
   - **Sinon** poser $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k$ et $\alpha_{k+1} := 0.5\alpha_k$ *(itération non réussie)*.

Algorithme de recherche directe basique

1. **Initialisation :** $\boldsymbol{x}_0 \in \mathbb{R}^d, \alpha_0 > 0$.
2. **Pour** $k = 0, 1, 2, ...$
   - Choisir un ensemble $\mathcal{D}_k \subset \mathbb{R}^d$ de r directions.
   - **SI** il existe $\boldsymbol{d}_k \in \mathcal{D}_k$ tel que

$$f(\boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k) < f(\boldsymbol{x}_k) - \alpha_k^2,$$

     alors poser $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$ et $\alpha_{k+1} \geq \alpha_k$ *(itération réussie)*.
   - **Sinon** poser $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k$ et $\alpha_{k+1} := 0.5\alpha_k$ *(itération non réussie)*.

Algorithme de recherche directe basique

1. **Initialisation :** $\boldsymbol{x}_0 \in \mathbb{R}^d, \alpha_0 > 0$.
2. **Pour** $k = 0, 1, 2, \ldots$
   - Choisir un ensemble $\mathcal{D}_k \subset \mathbb{R}^d$ de r directions.
   - **SI** il existe $\boldsymbol{d}_k \in \mathcal{D}_k$ tel que

$$f(\boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k) < f(\boldsymbol{x}_k) - \alpha_k^2,$$

     alors poser $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$ et $\alpha_{k+1} \geq \alpha_k$ *(itération réussie)*.
   - **Sinon** poser $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k$ et $\alpha_{k+1} := 0.5\alpha_k$ *(itération non réussie)*.

Aspects cruciaux

- Condition de décroissance;
- Choix de $\mathcal{D}_k$, valeur de r.

Plus et moins de la recherche directe classique

Positif

- **Exploitation :** On se base sur le point courant pour choisir le suivant;
- Exploration : Mouvement contrôlé par une longueur de pas α_k.

Négatif

- **Directions :** Au moins $d + 1$ évaluations par itération pour converger;
- **Dépendance en d :** Dans la complexité, mais aussi dans le nombre d'évaluations à chaque itération.

Plus et moins de la recherche directe classique

Positif

- **Exploitation :** On se base sur le point courant pour choisir le suivant;
- Exploration : Mouvement contrôlé par une longueur de pas α_k.

Négatif

- **Directions :** Au moins $d + 1$ évaluations par itération pour converger;
- **Dépendance en d :** Dans la complexité, mais aussi dans le nombre d'évaluations à chaque itération.

Pratique moderne

- Utiliser des directions aléatoires;
- S'applique au cas bruité.

Optimisation sans dérivées stochastique

$$\text{minimiser}_{\boldsymbol{x} \in \mathcal{F}} f(\boldsymbol{x})$$

Nouvelles hypothèses

- f seulement disponible via un oracle stochastique $\tilde{f}(\boldsymbol{x}; \boldsymbol{\xi})$;
- Le vecteur $\boldsymbol{\xi}$ est une quantité aléatoire dans un ensemble Ξ;
- Typique : $\tilde{f}(\cdot; \boldsymbol{\xi})$ convexe en $\boldsymbol{x}$ pour toute réalisation de $\boldsymbol{\xi}$.
- Minimum de f atteint en $\boldsymbol{x}_*$.

Optimisation sans dérivées stochastique

$$\text{minimiser}_{\boldsymbol{x}\in\mathcal{F}}\, f(\boldsymbol{x})$$

Nouvelles hypothèses

- f seulement disponible via un oracle stochastique $\tilde{f}(\boldsymbol{x};\boldsymbol{\xi})$;
- Le vecteur $\boldsymbol{\xi}$ est une quantité aléatoire dans un ensemble Ξ;
- Typique : $\tilde{f}(\cdot;\boldsymbol{\xi})$ convexe en $\boldsymbol{x}$ pour toute réalisation de $\boldsymbol{\xi}$.
- Minimum de f atteint en $\boldsymbol{x}_*$.

- Oracle stochastique $\leftrightarrow$ "Bandit feedback";
- Liens modernes avec la littérature en IA sur les bandits et l'optimisation en ligne.

Cadre des bandits à plusieurs bras

- Ensemble de bras $\{1, \ldots, A\}$;
- À l'itération k, on joue le bras $\boldsymbol{x}_k$, $\boldsymbol{\xi}_k$ est tirée, on obtient $f(\boldsymbol{x}_k; \xi_k)$;
- **Regret cumulé espéré :**

$$\mathbb{E}\left[\sum_{k=0}^{K-1} f(\boldsymbol{x}_k; \xi_k)\right] - Kf(\boldsymbol{x}_*),$$

Bandits avec infinités de bras (Auer, 2002)

1. On joue $\boldsymbol{x}_k$, $\boldsymbol{\xi}_k$ généré;
2. On observe $f(\boldsymbol{x}_k; \boldsymbol{\xi}_k)$.

But (complexité) : $f(\bar{\boldsymbol{x}}_K) - f(\boldsymbol{x}_*) \leq \epsilon, \qquad \bar{\boldsymbol{x}}_K = \frac{1}{K}\sum_{k=0}^{K-1} \boldsymbol{x}_k.$

Méthodes à un point

Tirer $\boldsymbol{u}_k \sim \mathcal{U}(\mathbb{S}^{d-1})$ et utiliser

$$\frac{\tilde{f}(\boldsymbol{x}_k + \mu \boldsymbol{u}_k; \boldsymbol{\xi}_k)}{\mu} \boldsymbol{u}_k \quad \text{ou} \quad \frac{\tilde{f}(\boldsymbol{x}_k + \mu \boldsymbol{u}_k; \boldsymbol{\xi}_k^+) - \tilde{f}(\boldsymbol{x}_k - \mu \boldsymbol{u}_k; \boldsymbol{\xi}_k^-)}{\mu} \boldsymbol{u}_k$$

Méthodes deux/multi-pas

- Hypothèse : Le même $\boldsymbol{\xi}$ permet de faire plusieurs évaluations.
- Tirer $\boldsymbol{u}_k \sim \mathcal{U}(\mathbb{S}^{d-1})$ et prendre

$$\frac{\tilde{f}(\boldsymbol{x}_k + \mu_k \boldsymbol{u}_k; \boldsymbol{\xi}_k)}{\mu_k} \boldsymbol{u}_k \quad \text{or} \quad \frac{\tilde{f}(\boldsymbol{x}_k + \mu_k \boldsymbol{u}_k; \boldsymbol{\xi}_k) - \tilde{f}(\boldsymbol{x}_k - \mu_k \boldsymbol{u}_k; \boldsymbol{\xi}_k)}{\mu_k} \boldsymbol{u}_k$$

Logiciels de recherche directe

NOMAD/HyperNOMAD: https://github.com/bbopt/HyperNOMAD

- Dédié au départ aux problèmes physiques (HydroQuébec);
- C++/Matlab/Python;
- Gère de nombreuses difficultés non abordées ici :
  - Variables catégorielles, entières;
  - Contraintes plus ou moins relâchables.
- HyperNOMAD (2019) : Extension appliquée pour optimiser architectures et hyperparamètres de réseaux de neurones.

Méthode à base de bandits

- Hyperband (Jamieson et al 2016), BOHB (Falkner et al 2018): approches de bandits + optimisation bayésienne (cf ci-après);
- Garanties supérieures à la recherche aléatoire, applicables à un grand nombre de variables.

Plan

Aller plus loin que la recherche directe

Résumé de ce qui précède

- Les méthodes de recherche directe explorent...
- ...et certaines **exploitent** localement.
- Utilisation de nouveaux points (notamment pour les méthodes aléatoires), pas de ré-utilisation d'information antérieure.

Aller plus loin que la recherche directe

Résumé de ce qui précède

- Les méthodes de recherche directe explorent...
- ...et certaines **exploitent** localement.
- Utilisation de nouveaux points (notamment pour les méthodes aléatoires), pas de ré-utilisation d'information antérieure.

DFO basée sur des modèles

- Utilise des évaluations **passées** de la fonction pour en construire un modèle;
- Ré-utilise des points, beaucoup moins coûteux que des différences finies.

Algorithme : Régions de confiance sans dérivées

- But : minimiser$_{\boldsymbol{x} \in \mathbb{R}^d} f(\boldsymbol{x})$;
- Évaluations de f coûteuses.

Algorithme : Régions de confiance sans dérivées

- But : minimiser$_{\boldsymbol{x}\in\mathbb{R}^d}$ $f(\boldsymbol{x})$;
- Évaluations de f coûteuses.

Entrées : $x_0 \in \mathbb{R}^d$, $\eta \in (0,1), \delta_0 > 0$.

Pour $k = 0, 1, 2, \ldots$

- Calculer un modèle $\boldsymbol{s} \mapsto m_k(\boldsymbol{x}_k + \boldsymbol{s})$ de f en $\boldsymbol{x}_k$;
- Calculer $\boldsymbol{s}_k \approx \text{argmin}_{\|\boldsymbol{s}\|\leq\delta_k}\, m_k(\boldsymbol{x}_k + \boldsymbol{s})$;

Algorithme : Régions de confiance sans dérivées

- But : minimiser$_{\boldsymbol{x}\in\mathbb{R}^d}$ $f(\boldsymbol{x})$;
- Évaluations de f coûteuses.

Entrées : $x_0 \in \mathbb{R}^d$, $\eta \in (0,1), \delta_0 > 0$.

Pour $k = 0, 1, 2, \ldots$

- Calculer un modèle $\boldsymbol{s} \mapsto m_k(\boldsymbol{x}_k + \boldsymbol{s})$ de f en $\boldsymbol{x}_k$;
- Calculer $\boldsymbol{s}_k \approx \text{argmin}_{\|\boldsymbol{s}\|\leq\delta_k} m_k(\boldsymbol{x}_k + \boldsymbol{s})$;
- Évaluer $\rho_k = \frac{f(\boldsymbol{x}_k) - f(\boldsymbol{x}_k+\boldsymbol{s}_k)}{m_k(\boldsymbol{x}_k) - m_k(\boldsymbol{x}_k+\boldsymbol{s}_k)}$.

Algorithme : Régions de confiance sans dérivées

- But : minimiser$_{\boldsymbol{x}\in\mathbb{R}^d}$ $f(\boldsymbol{x})$;
- Évaluations de f coûteuses.

Entrées : $x_0 \in \mathbb{R}^d$, $\eta \in (0,1), \delta_0 > 0$.

Pour $k = 0, 1, 2, \ldots$

- Calculer un modèle $\boldsymbol{s} \mapsto m_k(\boldsymbol{x}_k + \boldsymbol{s})$ de f en $\boldsymbol{x}_k$;
- Calculer $\boldsymbol{s}_k \approx \text{argmin}_{\|\boldsymbol{s}\|\leq\delta_k} m_k(\boldsymbol{x}_k + \boldsymbol{s})$;
- Évaluer $\rho_k = \frac{f(\boldsymbol{x}_k) - f(\boldsymbol{x}_k + \boldsymbol{s}_k)}{m_k(\boldsymbol{x}_k) - m_k(\boldsymbol{x}_k + \boldsymbol{s}_k)}$.
- Si $\rho_k \geq \eta$, poser $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{s}_k$ et $\delta_{k+1} \geq \delta_k$.

Algorithme : Régions de confiance sans dérivées

- But : minimiser$_{\boldsymbol{x}\in\mathbb{R}^d}$ $f(\boldsymbol{x})$;
- Évaluations de f coûteuses.

Entrées : $x_0 \in \mathbb{R}^d$, $\eta \in (0,1), \delta_0 > 0$.

Pour $k = 0, 1, 2, \ldots$

- Calculer un modèle $\boldsymbol{s} \mapsto m_k(\boldsymbol{x}_k + \boldsymbol{s})$ de f en $\boldsymbol{x}_k$;
- Calculer $\boldsymbol{s}_k \approx \text{argmin}_{\|\boldsymbol{s}\|\leq\delta_k} m_k(\boldsymbol{x}_k + \boldsymbol{s})$;
- Évaluer $\rho_k = \frac{f(\boldsymbol{x}_k)-f(\boldsymbol{x}_k+\boldsymbol{s}_k)}{m_k(\boldsymbol{x}_k)-m_k(\boldsymbol{x}_k+\boldsymbol{s}_k)}$.
- Si $\rho_k \geq \eta$, poser $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{s}_k$ et $\delta_{k+1} \geq \delta_k$.
- Sinon, poser $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ et $\delta_{k+1} = \delta_k/2$.

Qualité du modèle

But : Approcher une fonction dérivable f par un modèle m.

- Bornes sur l'erreur d'approximation;
- Garanties locales, dans une **région de confiance**.

Qualité du modèle

But : Approcher une fonction dérivable f par un modèle m.

- Bornes sur l'erreur d'approximation;
- Garanties locales, dans une **région de confiance**.

Modèles pleinement linéaires

Le modèle m est κ-pleinement linéaire pour f en $(\boldsymbol{x}, \delta)$ si pour tout $\boldsymbol{y} \in B(\boldsymbol{x}, \delta)$,

$$\begin{aligned} |m(\boldsymbol{y}) - f(\boldsymbol{y})| &\leq \kappa\delta^2 \\ \|\nabla m(\boldsymbol{y}) - \nabla f(\boldsymbol{y})\| &\leq \kappa\delta. \end{aligned}$$

Construire des modèles pleinement linéaires

- $\mathcal{P}_d^n$: polynômes de degré n sur $\mathbb{R}^d$, $\dim \mathcal{P}_d^n = q + 1$;
- $\Phi = \{\phi_0(\cdot), \ldots, \phi_q(\cdot)\}$: base de $\mathcal{P}_d^n$;
- $\mathcal{Y} = \{\boldsymbol{y}^0, \ldots, \boldsymbol{y}^p\}$: ensemble de $p + 1$ points de $\mathbb{R}^d$;

Construire des modèles pleinement linéaires

- $\mathcal{P}_d^n$: polynômes de degré n sur $\mathbb{R}^d$, $\dim \mathcal{P}_d^n = q+1$;
- $\Phi = \{\phi_0(\cdot), \ldots, \phi_q(\cdot)\}$: base de $\mathcal{P}_d^n$;
- $\mathcal{Y} = \{\boldsymbol{y}^0, \ldots, \boldsymbol{y}^p\}$: ensemble de $p+1$ points de $\mathbb{R}^d$;
- **But :** modèle $m(\boldsymbol{x}) = \sum_{i=0}^q \alpha_i \phi_i(\boldsymbol{x})$ tels que

$$\forall j = 0, \ldots, p, \quad m(\boldsymbol{y}^j) \approx f(\boldsymbol{y}^j).$$

- Reformulé comme $M(\Phi, \mathcal{Y})\alpha \approx f(\mathcal{Y})$, avec

$$M(\Phi, \mathcal{Y}) = \begin{bmatrix} \phi_0(\boldsymbol{y}^0) & \cdots & \phi_q(\boldsymbol{y}^0) \\ \vdots & \vdots & \vdots \\ \phi_0(\boldsymbol{y}^p) & \cdots & \phi_q(\boldsymbol{y}^p) \end{bmatrix}, \quad f(\mathcal{Y}) = \begin{bmatrix} f(\boldsymbol{y}^0) \\ \vdots \\ f(\boldsymbol{y}^p) \end{bmatrix}.$$

Régression polynomiale

Calculer $\boldsymbol{\alpha}^*$ solution de

$$\text{minimiser}_{\boldsymbol{\alpha} \in \mathbb{R}^{q+1}} \|M(\Phi, \mathcal{Y})\alpha - f(\mathcal{Y})\|^2.$$

et prendre $m(\boldsymbol{x}) = \sum_{i=0}^{q} \boldsymbol{\alpha}_i^* \phi_i(\boldsymbol{x})$.

Construire des modèles pleinement linéaires

Régression polynomiale

Calculer $\boldsymbol{\alpha}^*$ solution de

$$\text{minimiser}_{\boldsymbol{\alpha} \in \mathbb{R}^{q+1}} \|M(\Phi, \mathcal{Y})\alpha - f(\mathcal{Y})\|^2.$$

et prendre $m(\boldsymbol{x}) = \sum_{i=0}^{q} \boldsymbol{\alpha}_i^* \phi_i(\boldsymbol{x})$.

Cas classique

- Interpolation/régression linéaire avec $p = q = d$ (**technique de base d'analyse de données**);
- $\mathcal{Y} = \{\boldsymbol{y}^0, \boldsymbol{y}^1, \ldots, \boldsymbol{y}^d\}$ sommets d'un simplexe de $\mathbb{R}^d$.

De meilleurs modèles

Qualité du modèle

But : Approcher une fonction **deux fois dérivable** f par un modèle m.

- Bornes sur l'erreur d'approximation;
- Garanties locales, dans une **région de confiance**.

De meilleurs modèles

Qualité du modèle

But : Approcher une fonction **deux fois dérivable** f par un modèle m.

- Bornes sur l'erreur d'approximation;
- Garanties locales, dans une **région de confiance**.

Modèles pleinement quadratiques

Le modèle m_k est κ-pleinement quadratique pour f sur $(\boldsymbol{x}_k, \delta_k)$ si pour tout $\boldsymbol{y} \in B(\boldsymbol{x}_k, \delta_k)$,

$$\begin{aligned} |m_k(\boldsymbol{y}) - f(\boldsymbol{y})| &\leq \kappa\delta_k^3 \\ \|\nabla m_k(\boldsymbol{y}) - \nabla f(\boldsymbol{y})\| &\leq \kappa\delta_k^2 \\ \|\nabla^2 m_k(\boldsymbol{y}) - \nabla^2 f(\boldsymbol{y})\| &\leq \kappa\delta_k. \end{aligned}$$

Construire des modèles pleinement quadratiques en pratique

Choix classiques

Modèles m_k construits à partir de valeurs de f en $\mathcal{Y}_k = \{\boldsymbol{x}_k, \boldsymbol{y}^1, \ldots, \boldsymbol{y}^r\}$:

- Interpolation/Régression;
- Radial basis functions (RBF, ou noyaux gaussiens).

Point clé

Bonne géométrie de $\mathcal{Y}_k \Rightarrow m_k$ pleinement quadratique sur $B(\boldsymbol{x}_k, \delta_k)$.

Ex) Interpolation quadratique avec $r = \mathcal{O}(n^2)$ échantillons
$\mathcal{Y}_k = \left\{\boldsymbol{x}_k, \{\boldsymbol{x}_k \pm \delta_k \boldsymbol{e}_i\}_{i=1}^n, \{\boldsymbol{x}_k + \delta_k \frac{\boldsymbol{e}_i + \boldsymbol{e}_j}{2}\}_{1 \leq i < j \leq n}\right\}$.

Construire des modèles pleinement quadratiques en pratique

Choix classiques

Modèles m_k construits à partir de valeurs de f en $\mathcal{Y}_k = \{\boldsymbol{x}_k, \boldsymbol{y}^1, \ldots, \boldsymbol{y}^r\}$:

- Interpolation/Régression;
- Radial basis functions (RBF, ou noyaux gaussiens).

Point clé

Bonne géométrie de $\mathcal{Y}_k \Rightarrow m_k$ pleinement quadratique sur $B(\boldsymbol{x}_k, \delta_k)$.

Ex) Interpolation quadratique avec $r = \mathcal{O}(n^2)$ échantillons
$\mathcal{Y}_k = \left\{\boldsymbol{x}_k, \{\boldsymbol{x}_k \pm \delta_k \boldsymbol{e}_i\}_{i=1}^n, \{\boldsymbol{x}_k + \delta_k \frac{\boldsymbol{e}_i + \boldsymbol{e}_j}{2}\}_{1 \leq i < j \leq n}\right\}$.

Dans la pratique

- On ré-utilise autant de points que possible!
- On contrôle la géométrie et on la corrige si besoin:

Construire des modèles pleinement quadratiques en pratique

Choix classiques

Modèles m_k construits à partir de valeurs de f en $\mathcal{Y}_k = \{\boldsymbol{x}_k, \boldsymbol{y}^1, \ldots, \boldsymbol{y}^r\}$:

- Interpolation/Régression;
- Radial basis functions (RBF, ou noyaux gaussiens).

Point clé

Bonne géométrie de $\mathcal{Y}_k \Rightarrow m_k$ pleinement quadratique sur $B(\boldsymbol{x}_k, \delta_k)$.

Ex) Interpolation quadratique avec $r = \mathcal{O}(n^2)$ échantillons
$\mathcal{Y}_k = \left\{\boldsymbol{x}_k, \{\boldsymbol{x}_k \pm \delta_k \boldsymbol{e}_i\}_{i=1}^n, \{\boldsymbol{x}_k + \delta_k \frac{\boldsymbol{e}_i + \boldsymbol{e}_j}{2}\}_{1 \leq i < j \leq n}\right\}$.

Dans la pratique

- On ré-utilise autant de points que possible!
- On contrôle la géométrie et on la corrige si besoin:
  - Peut demander r nouvelles valeurs;

Construire des modèles pleinement quadratiques en pratique

Choix classiques

Modèles m_k construits à partir de valeurs de f en $\mathcal{Y}_k = \{\boldsymbol{x}_k, \boldsymbol{y}^1, \ldots, \boldsymbol{y}^r\}$:

- Interpolation/Régression;
- Radial basis functions (RBF, ou noyaux gaussiens).

Point clé

Bonne géométrie de $\mathcal{Y}_k \Rightarrow m_k$ pleinement quadratique sur $B(\boldsymbol{x}_k, \delta_k)$.

Ex) Interpolation quadratique avec $r = \mathcal{O}(n^2)$ échantillons
$\mathcal{Y}_k = \left\{\boldsymbol{x}_k, \{\boldsymbol{x}_k \pm \delta_k \boldsymbol{e}_i\}_{i=1}^n, \{\boldsymbol{x}_k + \delta_k \frac{\boldsymbol{e}_i + \boldsymbol{e}_j}{2}\}_{1 \leq i < j \leq n}\right\}$.

Dans la pratique

- On ré-utilise autant de points que possible!
- On contrôle la géométrie et on la corrige si besoin:
  - Peut demander r nouvelles valeurs;
  - En pratique, bien plus économe.

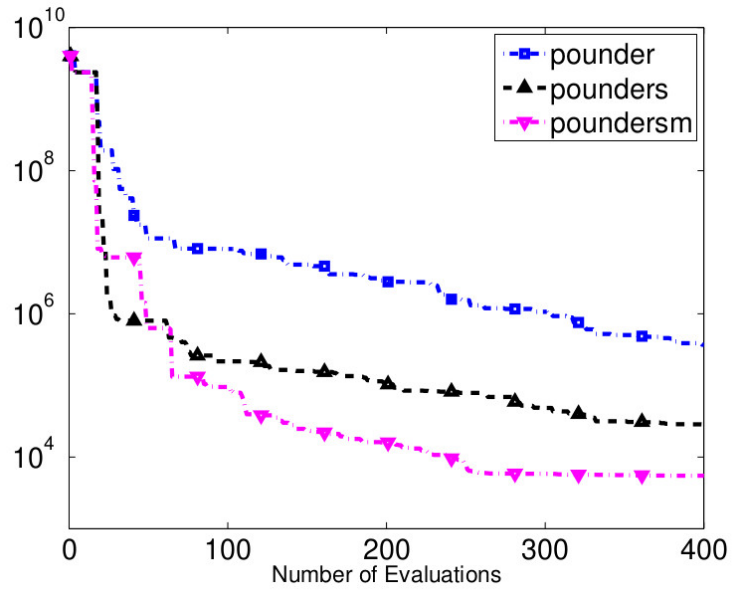
Remarque : Utiliser la structure

- Nous avons traité la fonction en “boîte noire”;
- Souvent on peut calculer les dérivées d'une partie de la fonction;
- Globalement, connaître la structure peut aider!

Remarque : Utiliser la structure

- Nous avons traité la fonction en “boîte noire”;
- Souvent on peut calculer les dérivées d'une partie de la fonction;
- Globalement, connaître la structure peut aider!

Ex) $\text{minimiser}_{\boldsymbol{x}\in\mathbb{R}^d} f(\boldsymbol{x}) = \frac{1}{2}\|r(\boldsymbol{x})\|^2 \quad r : \mathbb{R}^d \to \mathbb{R}^n.$



- Dérivées de r inaccessibles mais on a :

$$\nabla f(\boldsymbol{x}) = \boldsymbol{J}_r(\boldsymbol{x})^{\mathrm{T}} r(\boldsymbol{x}).$$

- On construit ainsi des modèles plus précis.

Sur l'approche déterministe

Avec des modèles linéaires

- En pratique, ré-utiliser des points/prendre moins de $d+1$ points marche;
- En théorie, il faut $\mathcal{O}(d)$ points pour être pleinement linéaires!

Sur l'approche déterministe

Avec des modèles linéaires

- En pratique, ré-utiliser des points/prendre moins de $d+1$ points marche;
- En théorie, il faut $\mathcal{O}(d)$ points pour être pleinement linéaires!

Idée

Supposer que les modèles sont pleinement linéaires en probabilité.

- Processus aléatoire;
- Analysé via des arguments statistiques (martingales).

Sous-échantillonnage

$$f(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N} f_i(\boldsymbol{x}), \nabla f_i \text{ disponible mais pas } \nabla f$$

- Choisir $S \subset \{1, \ldots, N\}$ aléatoirement;
- Poser $m(\boldsymbol{x}+\boldsymbol{x}) = f(\boldsymbol{x}) + \frac{1}{|S|}\sum_{i\in S} \nabla f_i(\boldsymbol{x})^\top \boldsymbol{s}$;
- Avec $|S| = \mathcal{O}(\delta^{-2})$, le modèle est pleinement linéaire en probabilité.

Sous-échantillonnage

$$f(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N} f_i(\boldsymbol{x}), \nabla f_i \text{ disponible mais pas } \nabla f$$

- Choisir $S \subset \{1, \ldots, N\}$ aléatoirement;
- Poser $m(\boldsymbol{x}+\boldsymbol{x}) = f(\boldsymbol{x}) + \frac{1}{|S|}\sum_{i\in S} \nabla f_i(\boldsymbol{x})^\top \boldsymbol{s}$;
- Avec $|S| = \mathcal{O}(\delta^{-2})$, le modèle est pleinement linéaire en probabilité.

Modèles pleinement quadratiques en proba.

- Déterministe : $\mathcal{O}(d^2)$ évaluations;
- Si la matrice hessienne est creuse, vrai (en proba.) avec $\mathcal{O}\left(d(\log d)^4\right)$ évaluations via des techniques de compression de signal.

Paradigme de l'optimisation bayésienne

1. À chaque itération, calculer une distribution a posteriori relativement aux évaluations connues;
2. Trouver un nouveau point en maximisant une fonction d'acquisition;
3. Répéter.

Paradigme de l'optimisation bayésienne

1. À chaque itération, calculer une distribution a posteriori relativement aux évaluations connues;
2. Trouver un nouveau point en maximisant une fonction d'acquisition;
3. Répéter.

- Populaire chez les *data scientists*:
- Les modèles sont basés sur des processus gaussiens et donc des fonctions RBF :

$$m_k(\boldsymbol{x}_k + \boldsymbol{s}) = \sum_{i=1}^{|\mathcal{Y}|} \exp(-\|\boldsymbol{y}_i - \boldsymbol{s}\|^2)$$

$\Rightarrow$ Ce sont des modèles pleinement linéaires !

Résumé des méthodes à modèles

Construire des modèles

- Utilise l'historique de l'algorithme;
- Implémentations efficaces meilleures qu'estimer les dérivées directement;
- Il faut exploiter de la structure s'il en existe.

Popularité des modèles

- Méta-modèles/*Surrogates* en calcul scientifique;
- Méta-modèles/processus gaussiens en optimisation bayésienne.

Logiciels

Codes de M. J. D. Powell

- En FORTRAN 77, mais toujours l'un des meilleurs codes disponibles;
- PDFO (https://www.pdfo.net/), interfaces Python et MATLAB;
- Récemment utilisé en IA (apprentissage adverse).

Autres codes

- POUNDERS : Moindres carrés sans dérivées (structure);
- ORBIT : Régions de confiance, modèle RBF;
- Packages Python/R pour l'optimisation de *surrogates*/ l'optimisation bayésienne.

Résumé

Calcul de gradient

- Via différentiation automatique.
- Backpropagation pour les réseaux de neurones.

En l'absence de gradients

- Dérivées généralisées (type sous-gradients).
- Algorithmes généralisés.
- Différentiation automatique.

Optimisation sans dérivées/DFO

- Pas de dérivées dans l'algorithme (ou pas toutes).
- Tout dépend de l'évaluation et de son coût.
- Application : Optimisation d'hyperparamètres.

Bibliographie

Différentiation

- A. Griewank and A. Walther, *Evaluating Derivatives: Principles and Techniques of Automatic Differentiation*. SIAM, 2008.
- Packages : `JAX`, `Autograd`.

Optimisation sans dérivées

- J. Larson, M. Menickelly and S. M. Wild, *Derivative-free optimization methods*, Acta Numerica, 2019.
- M. Feurer and F. Hutter, *Hyperparameter Optimization*, in *Automated Machine Learning*, Springer, 2019.
- P. I. Frazier, *Bayesian Optimization*, Tutorials in Operations Research, 2018.

Merci de votre attention !

clement.royer@lamsade.dauphine.fr

Crédits

- https://towardsdatascience.com/
- https://commons.wikimedia.org/
- A. J. Booker, J. E. Dennis Jr., P. D. Frank, D. B. Serafini and V. Torczon, *A rigorous framework for optimization of expensive functions by surrogates*, Structural Optim., 1999.
- M. Feurer and F. Hutter, *Hyperparameter Optimization*, in *Automated Machine Learning*, Springer, 2019.
- D. Gaudrie (Stellantis).
- S. M. Wild, *Beyond the Black Box in Derivative-Free and Simulation-Based Optimization*, SIAM Annual Meeting, 2016.