Optimisation sans dérivées

Clément Royer

Certificat Chef de Projet IA - Université Paris Dauphine-PSL

18 novembre 2021

Objectifs de la séance

Un tour d'horizon

- De techniques classiques en optimisation sans dérivées;
- Et de leurs liens avec l'IA.

Objectifs de la séance

Un tour d'horizon

- De techniques classiques en optimisation sans dérivées;
- Et de leurs liens avec l'IA.

Des révisions

- Les concepts de lundi nous seront utiles;
- Ils seront rappelés.

Objectifs de la séance

Un tour d'horizon

- De techniques classiques en optimisation sans dérivées;
- Et de leurs liens avec l'IA.

Des révisions

- Les concepts de lundi nous seront utiles;
- Ils seront rappelés.

En fonction du temps

- Illustrations numériques;
- Retour global.

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.

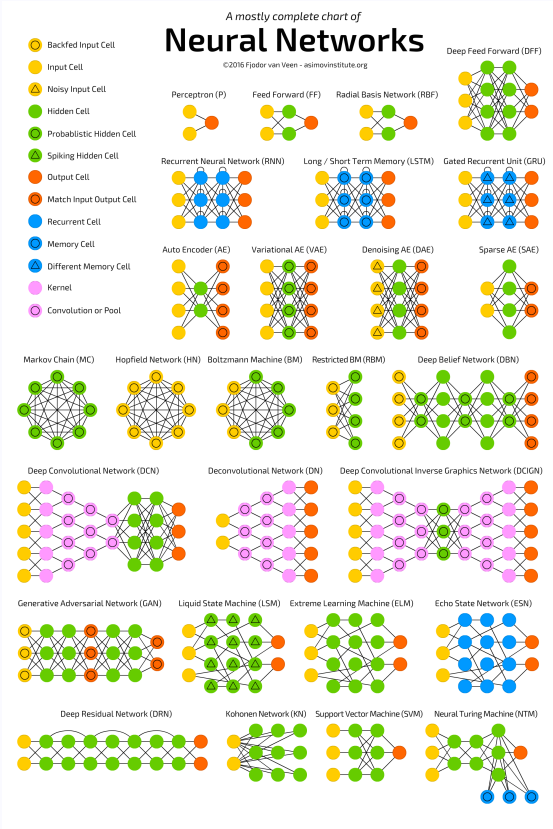
Plan

Plan

Ce dont tout le monde parle : IA automatisée

Entraînons un réseau de neurones...

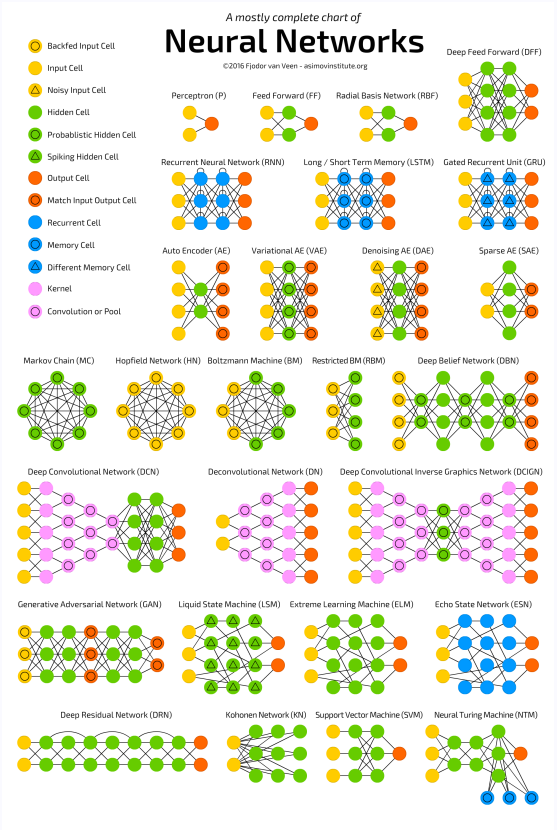
- Quelle architecture ? (nombre de couches, types de couches, etc)
- Quel algorithme d'entraînement ? (Adam, RMSProp, SGD, etc)
- Quelles options pour l'algorithme (*learning rate*, etc)?



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'algorithme (*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

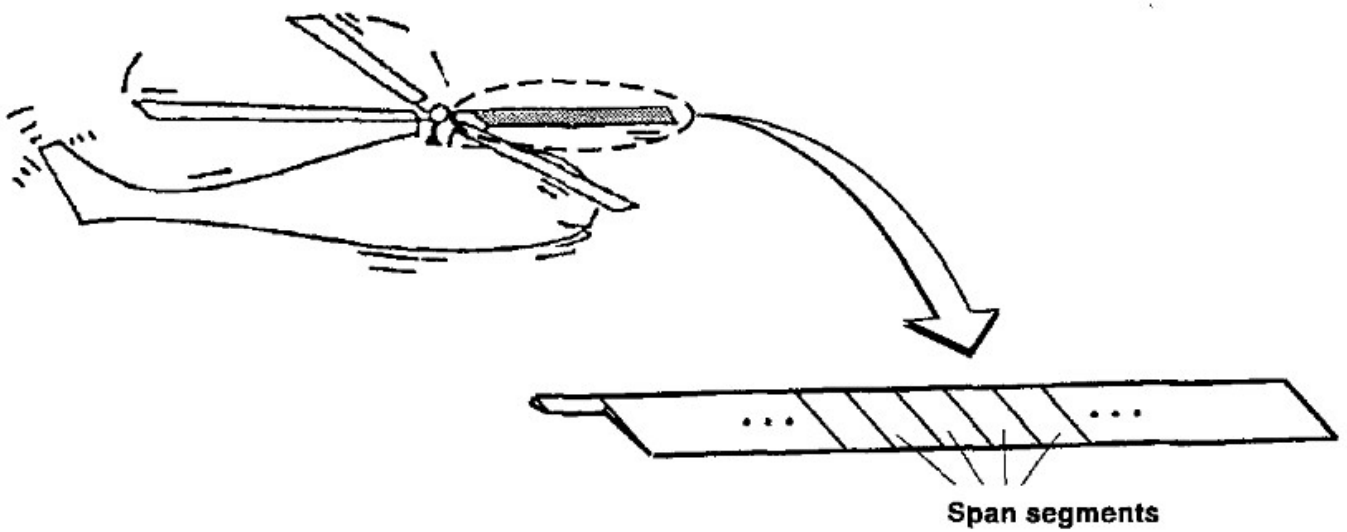
Calcul scientifique

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

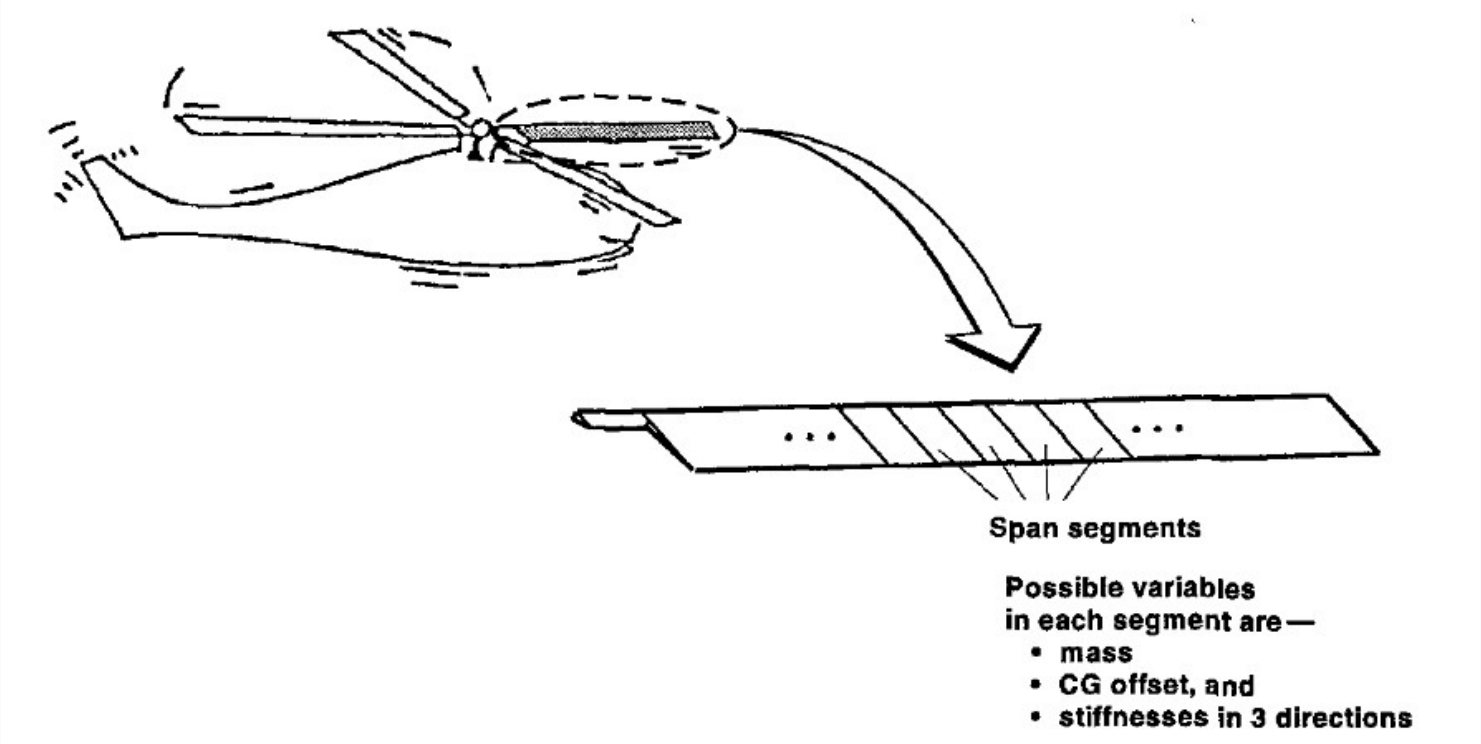
Optimisation de simulateurs

- 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)



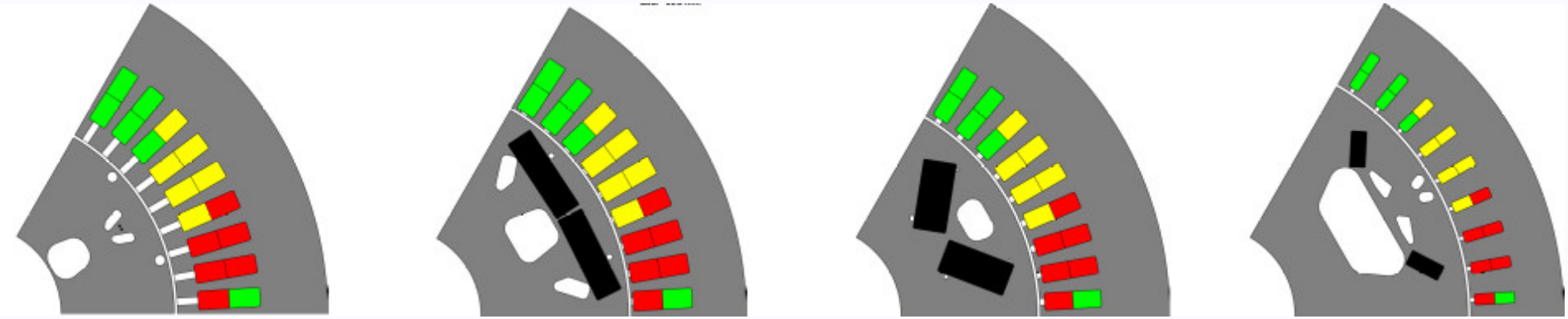
Exemple classique : Design de pale d'hélicoptère (Booker et al. 1998)



- 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

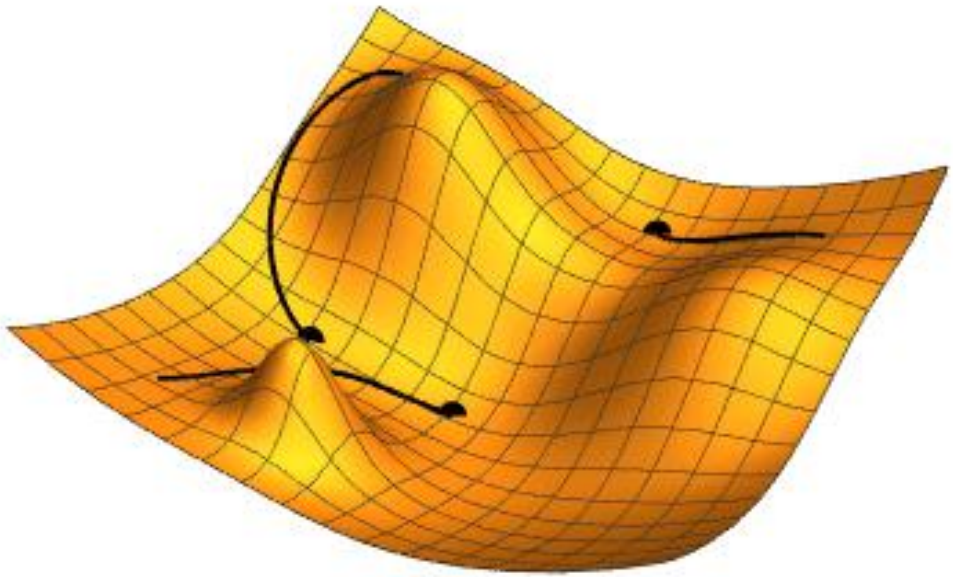
Certaines dérivées ne peuvent pas être exploitées pour optimiser.

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 !

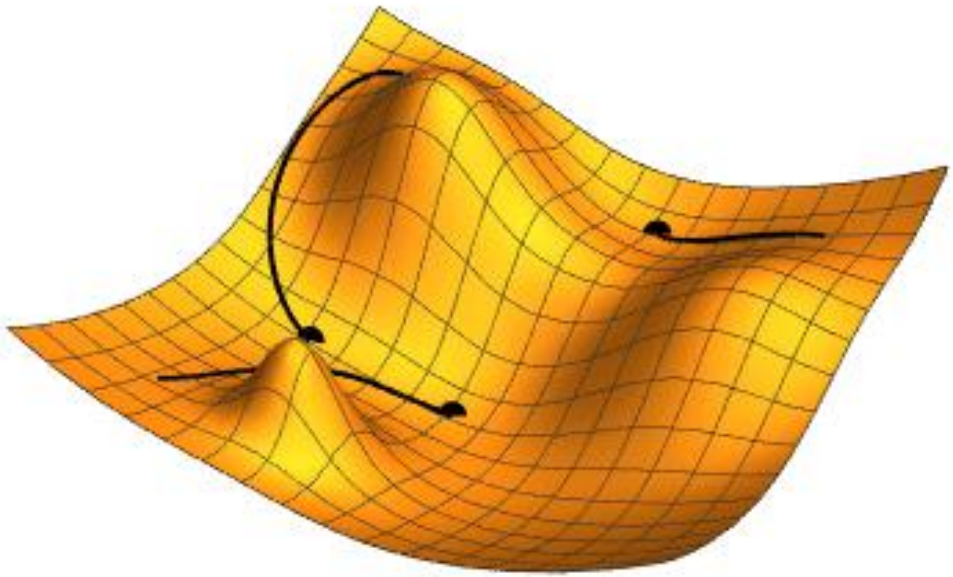


Une définition

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).

Plan

Notre formulation

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

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

Problème

Notre formulation

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

- $\boldsymbol{x}$: 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

Plan

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

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

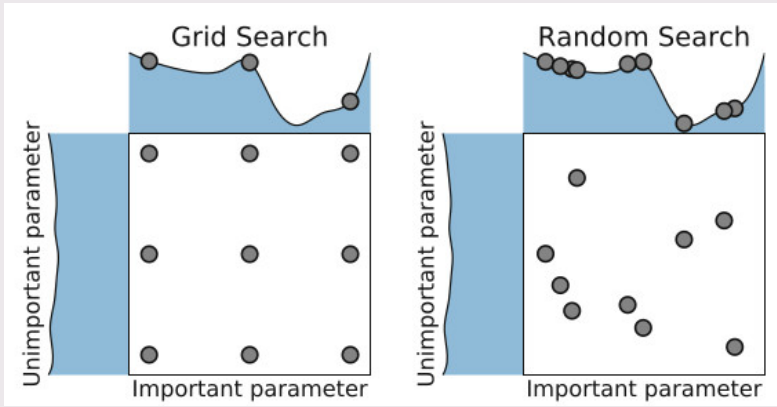
But : Résoudre $\text{minimiser}_{\boldsymbol{x} \in \mathcal{F}} f(\boldsymbol{x})$ avec accès à f uniquement, budget limité.

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



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.

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)*

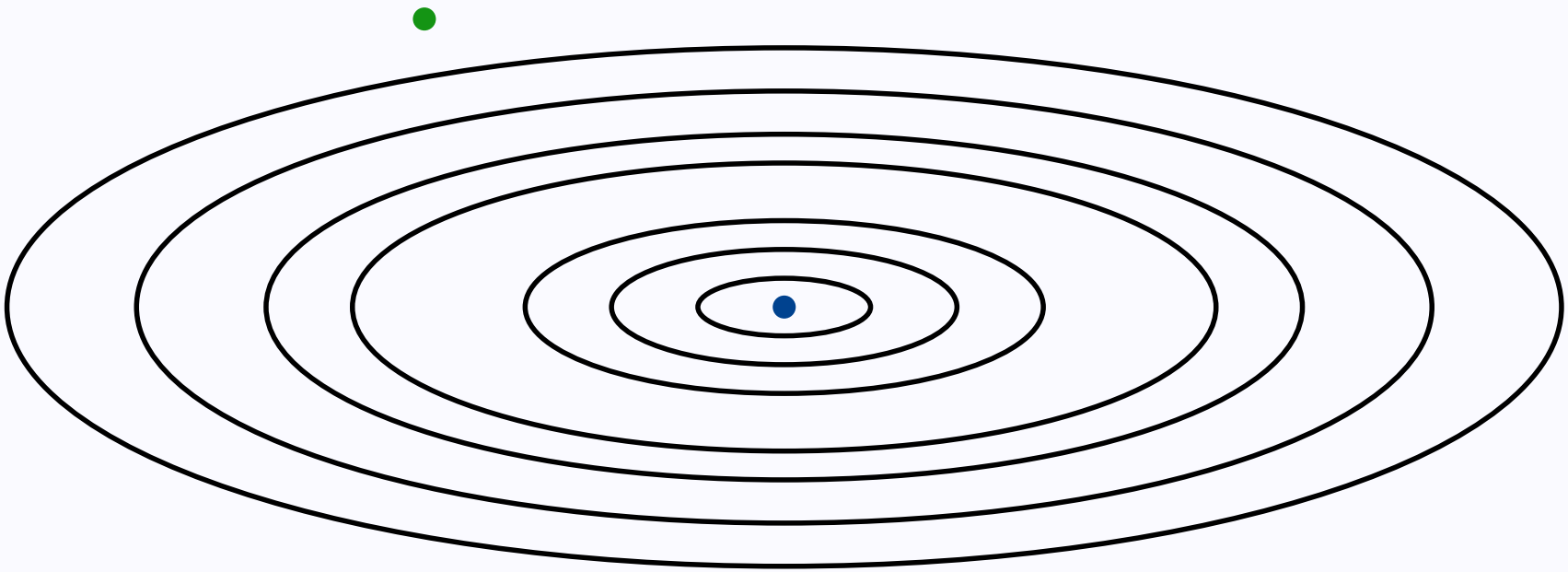
Recherche exploratoire

- Valables en petite dimension;
- Algorithmes exploratoires.

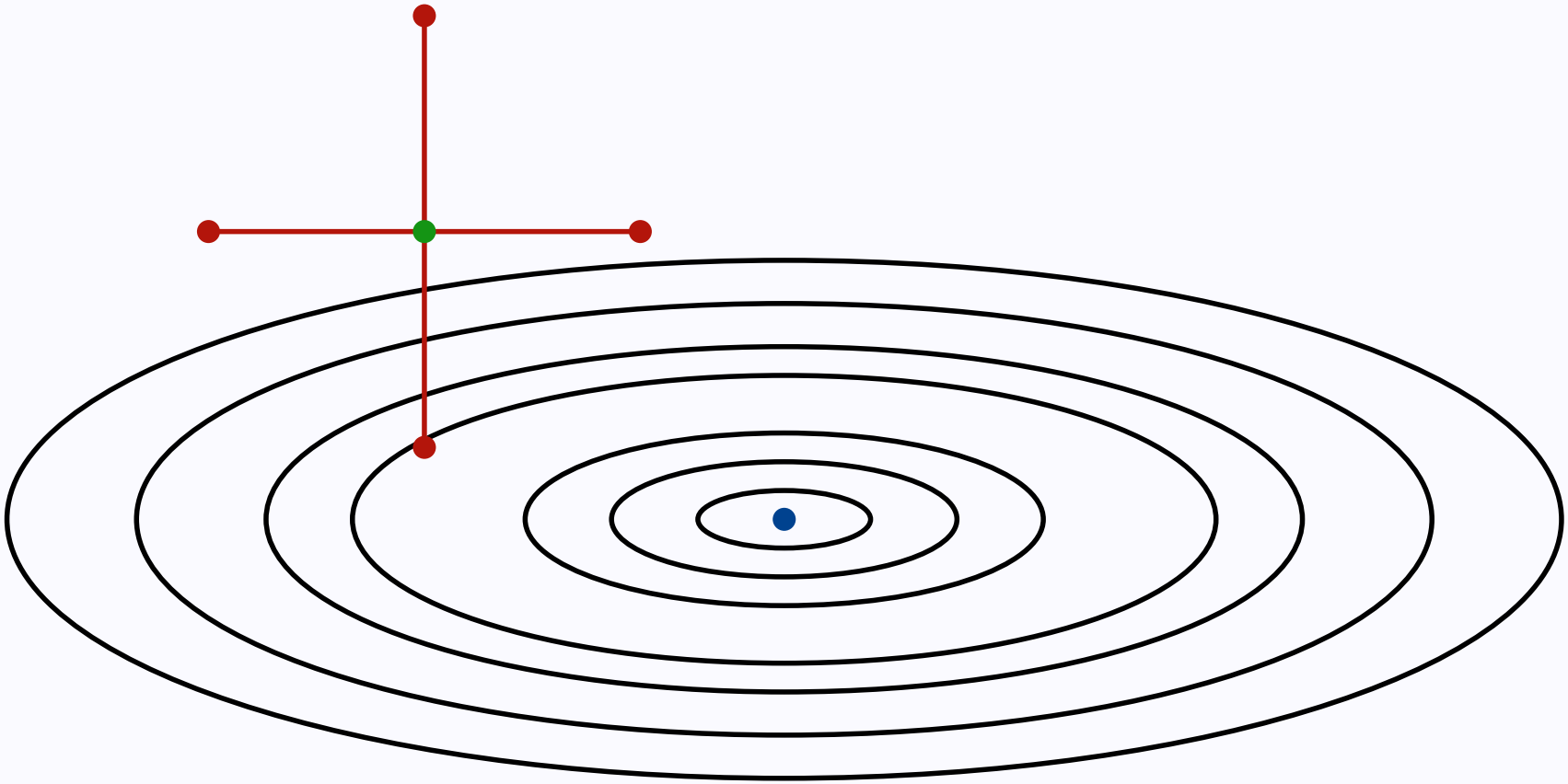
Recherche directe

- 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!

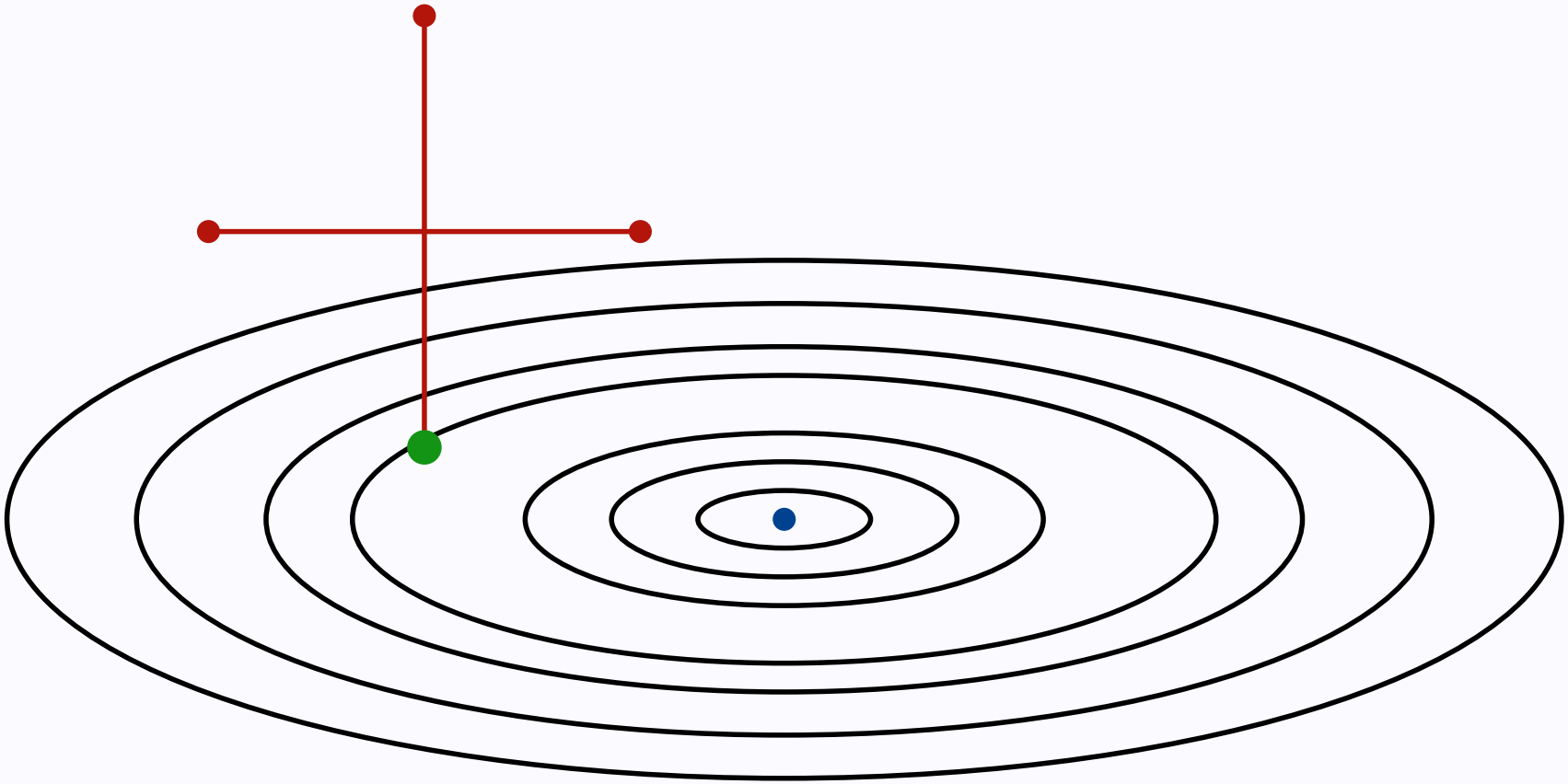
Exemple : Recherche par coordonnées



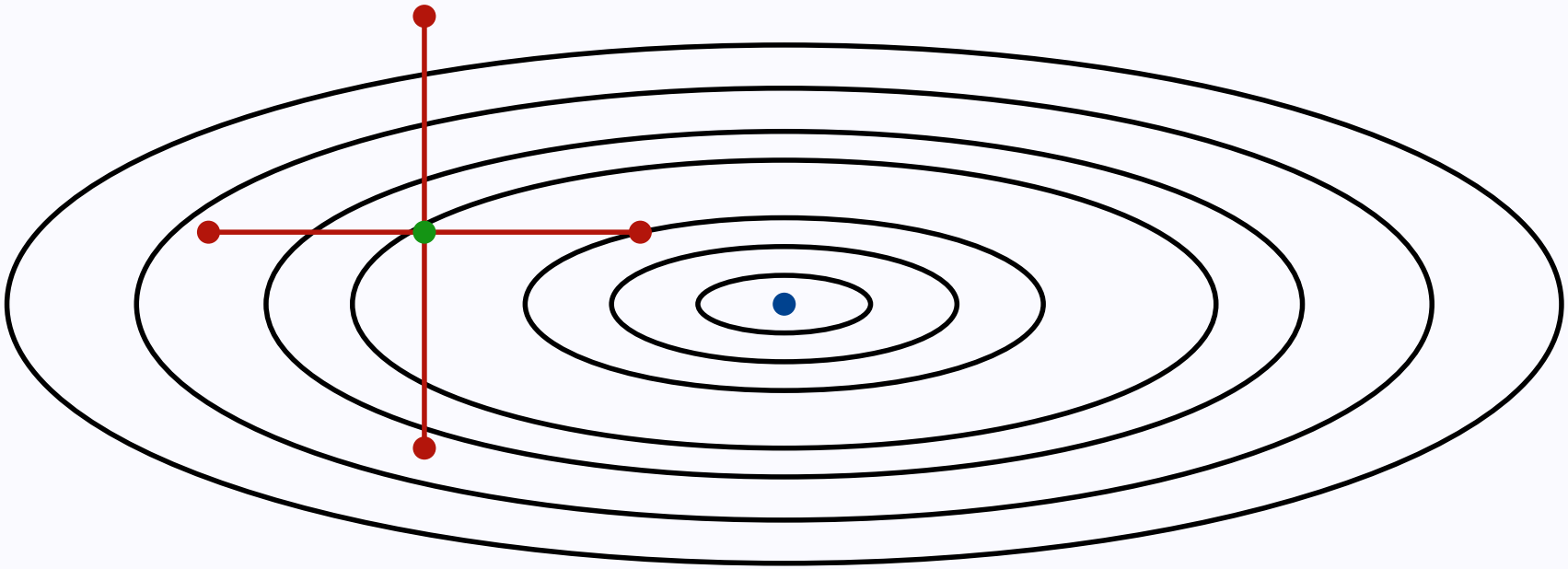
Exemple : Recherche par coordonnées



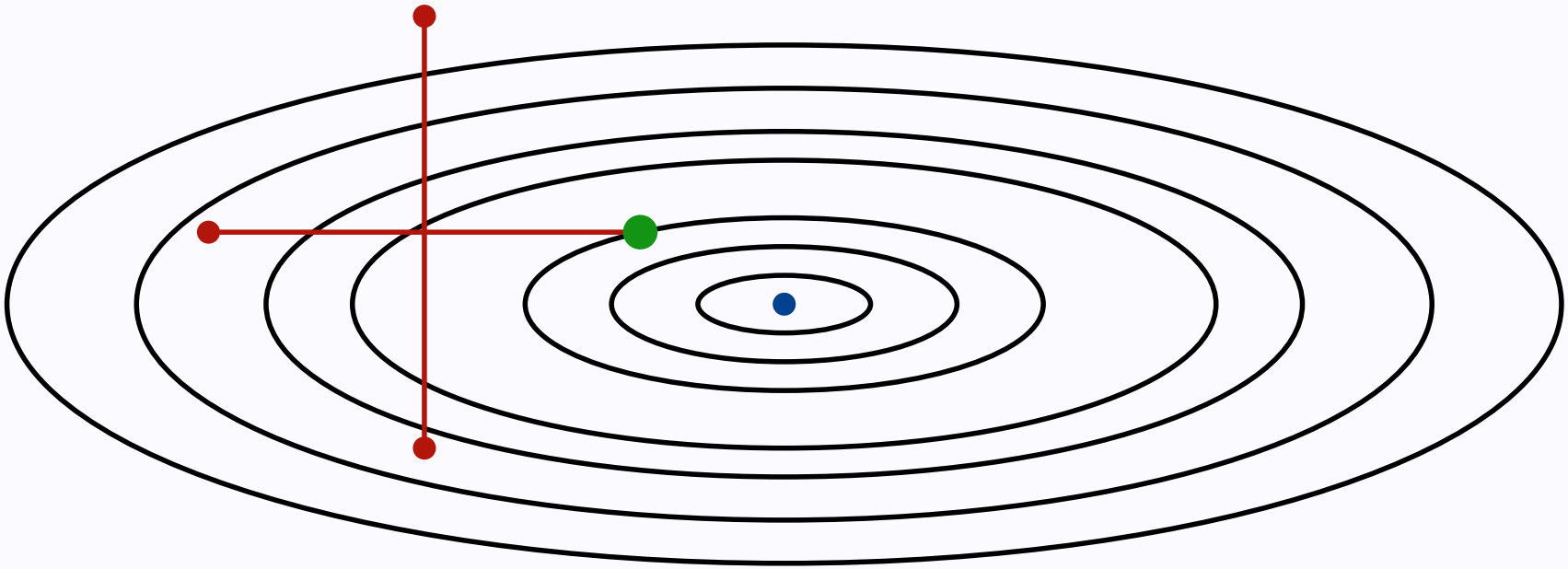
Exemple : Recherche par coordonnées



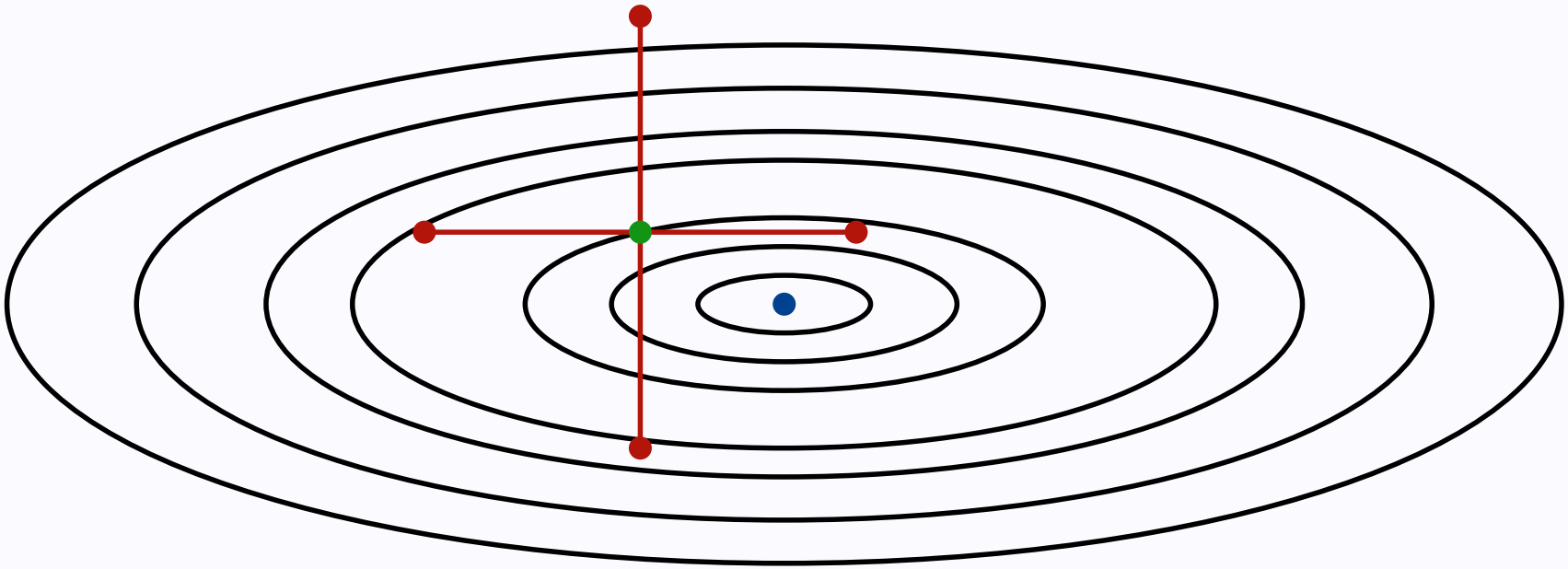
Exemple : Recherche par coordonnées



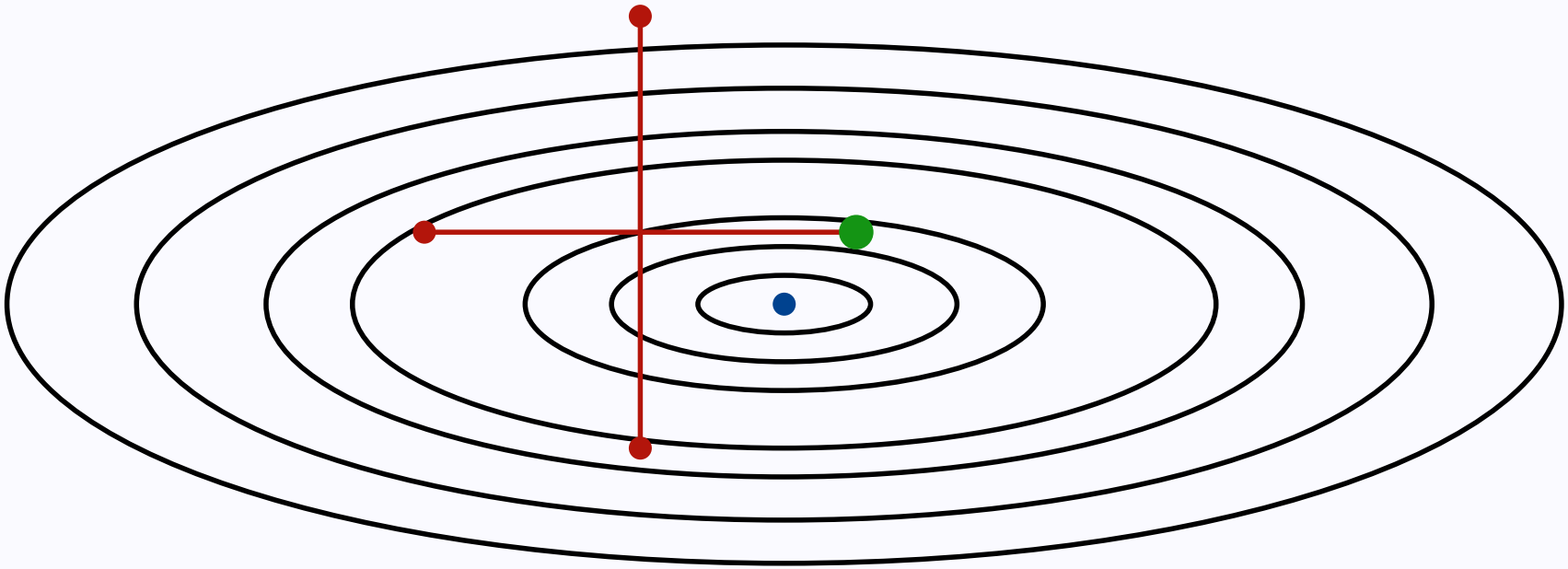
Exemple : Recherche par coordonnées



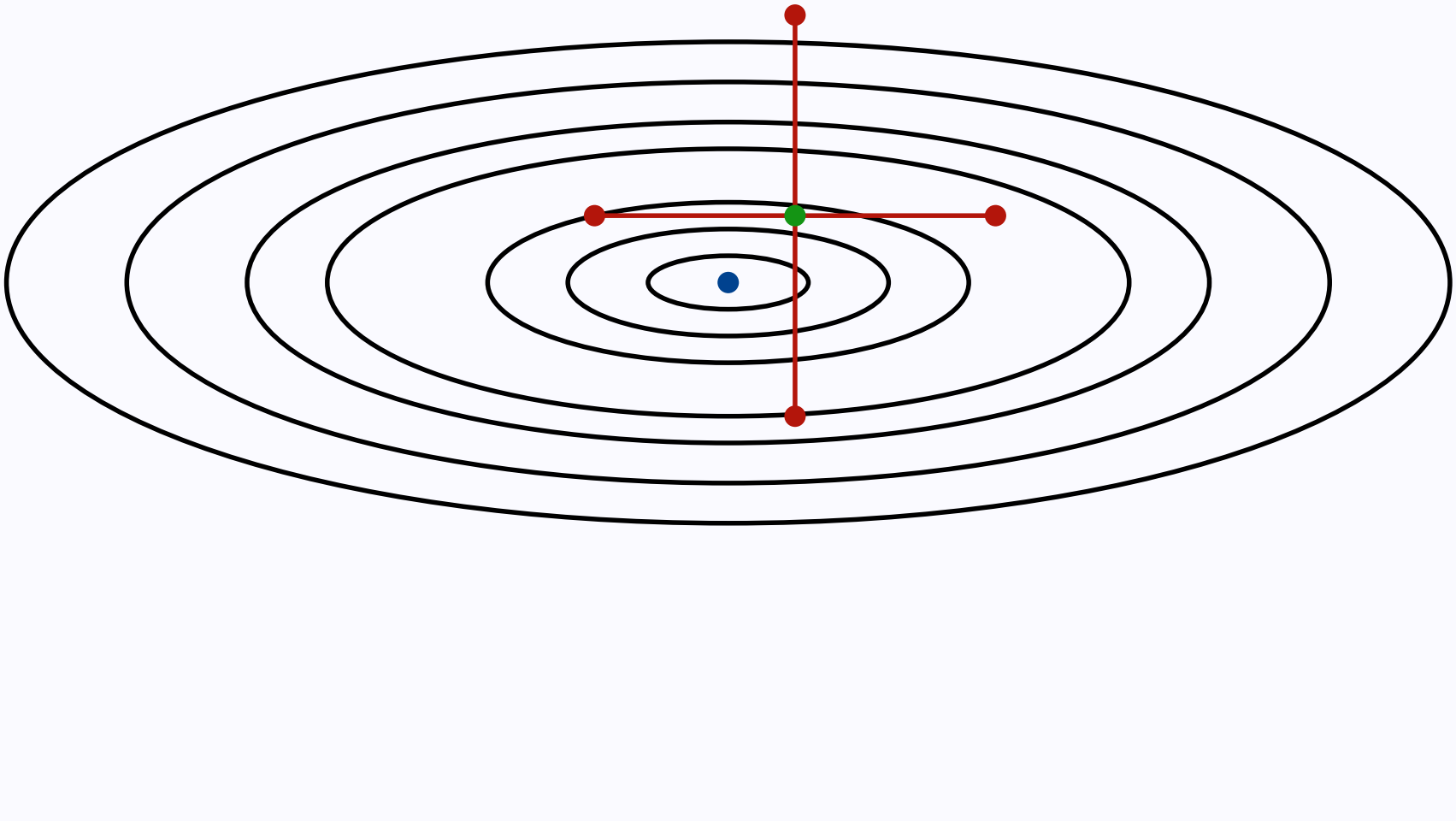
Exemple : Recherche par coordonnées



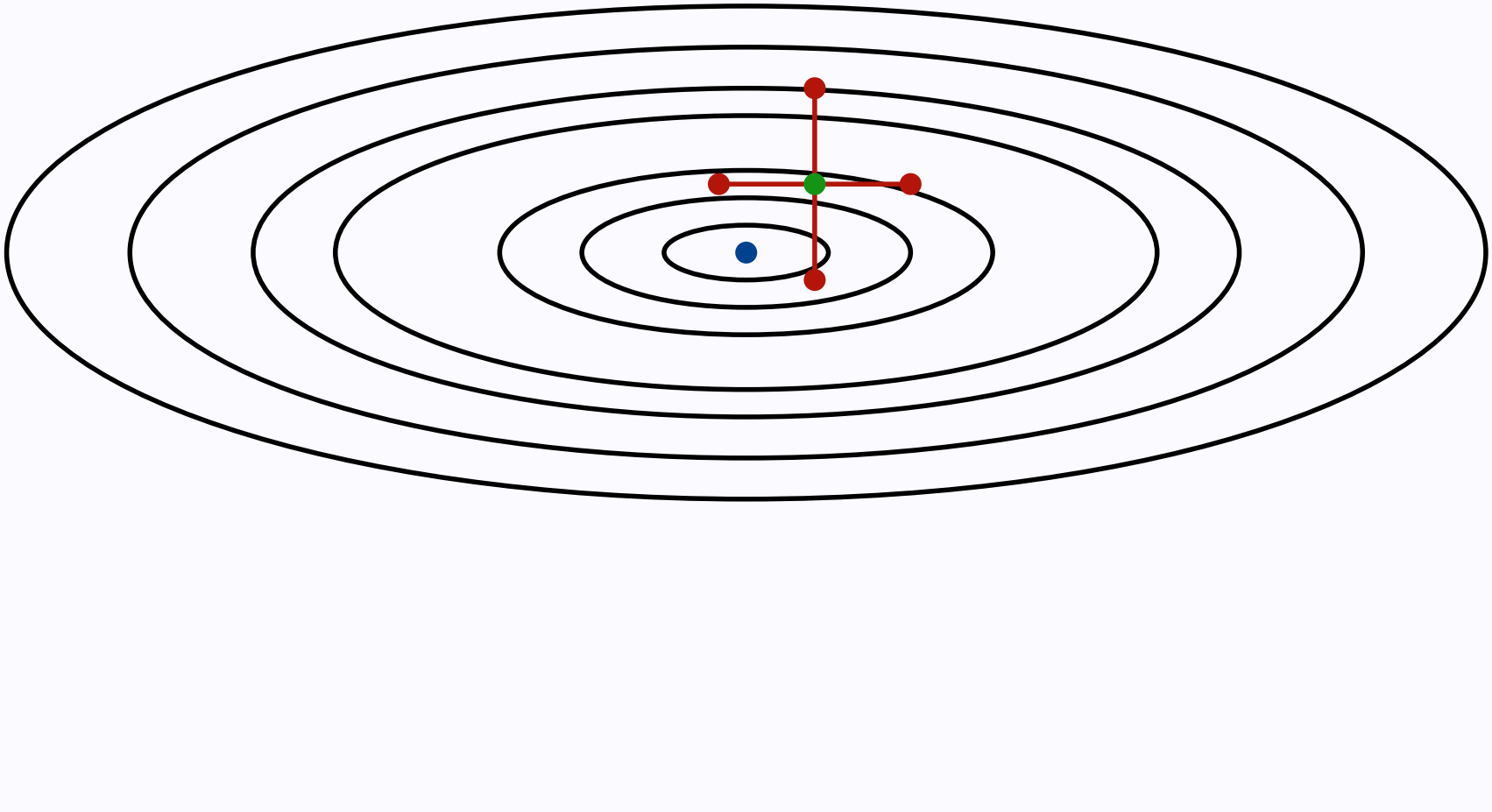
Exemple : Recherche par coordonnées



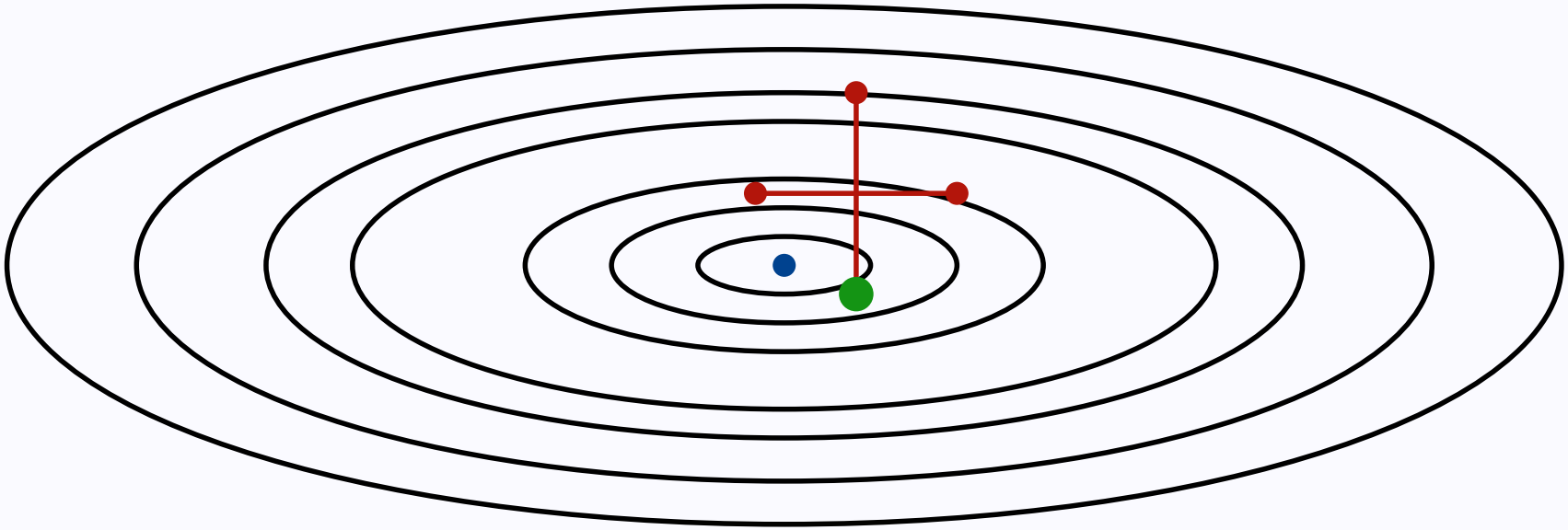
Exemple : Recherche par coordonnées



Exemple : Recherche par coordonnées



Exemple : Recherche par coordonnées



Algorithme de recherche directe basique

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

Algorithme de recherche directe basique

1. **Initialisation :** $\boldsymbol{x}_0 \in \mathbb{R}^n, \alpha_0 > 0$.
2. **Pour** $k = 0, 1, 2, \ldots$
   - Choisir un ensemble $\mathcal{D}_k \subset \mathbb{R}^n$ 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 :** $\boldsymbol{x}_0 \in \mathbb{R}^n, \alpha_0 > 0$.
2. **Pour** $k = 0, 1, 2, \ldots$
   - Choisir un ensemble $\mathcal{D}_k \subset \mathbb{R}^n$ 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 :** $\boldsymbol{x}_0 \in \mathbb{R}^n, \alpha_0 > 0$.
2. **Pour** $k = 0, 1, 2, \ldots$
   - Choisir un ensemble $\mathcal{D}_k \subset \mathbb{R}^n$ 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 :** $\boldsymbol{x}_0 \in \mathbb{R}^n, \alpha_0 > 0$.
2. **Pour** $k = 0, 1, 2, \ldots$
   - Choisir un ensemble $\mathcal{D}_k \subset \mathbb{R}^n$ 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.

Une mesure de qualité

Soit $\mathcal{D} \subset \mathbb{R}^n$ et $\boldsymbol{v} \in \mathbb{R}^n \setminus \{0\}$, la mesure cosinus de $\mathcal{D}$ en $\boldsymbol{v}$ est

$$\text{cm}(\mathcal{D}, \boldsymbol{v}) = \max_{\boldsymbol{d} \in \mathcal{D}} \frac{\boldsymbol{d}^\top \boldsymbol{v}}{\|\boldsymbol{d}\| \, \|\boldsymbol{v}\|}$$

Mesure cosinus et PSS

Une mesure de qualité

Soit $\mathcal{D} \subset \mathbb{R}^n$ et $\boldsymbol{v} \in \mathbb{R}^n \setminus \{0\}$, la mesure cosinus de $\mathcal{D}$ en $\boldsymbol{v}$ est

$$\text{cm}(\mathcal{D}, \boldsymbol{v}) = \max_{\boldsymbol{d} \in \mathcal{D}} \frac{\boldsymbol{d}^\top \boldsymbol{v}}{\|\boldsymbol{d}\| \, \|\boldsymbol{v}\|}$$

- Distance angulaire de l'analyse de données textuelles (`word2vec`);
- Ici sert à approcher tout vecteur de l'espace par un vecteur de $\mathcal{D}$;

Mesure cosinus et PSS

Une mesure de qualité

Soit $\mathcal{D} \subset \mathbb{R}^n$ et $\boldsymbol{v} \in \mathbb{R}^n \setminus \{0\}$, la mesure cosinus de $\mathcal{D}$ en $\boldsymbol{v}$ est

$$\text{cm}(\mathcal{D}, \boldsymbol{v}) = \max_{\boldsymbol{d} \in \mathcal{D}} \frac{\boldsymbol{d}^\top \boldsymbol{v}}{\|\boldsymbol{d}\| \, \|\boldsymbol{v}\|}$$

- Distance angulaire de l'analyse de données textuelles (`word2vec`);
- Ici sert à approcher tout vecteur de l'espace par un vecteur de $\mathcal{D}$;
- Meilleurs ensembles (PSS) : $\text{cm}(\mathcal{D}, \boldsymbol{v}) > 0 \quad \forall \boldsymbol{v} \neq 0$ (requiert $|\mathcal{D}| \geq n + 1$).

Mesure cosinus et PSS

Une mesure de qualité

Soit $\mathcal{D} \subset \mathbb{R}^n$ et $\boldsymbol{v} \in \mathbb{R}^n \setminus \{0\}$, la mesure cosinus de $\mathcal{D}$ en $\boldsymbol{v}$ est

$$\text{cm}(\mathcal{D}, \boldsymbol{v}) = \max_{\boldsymbol{d} \in \mathcal{D}} \frac{\boldsymbol{d}^\top \boldsymbol{v}}{\|\boldsymbol{d}\| \, \|\boldsymbol{v}\|}$$

- Distance angulaire de l'analyse de données textuelles (`word2vec`);
- Ici sert à approcher tout vecteur de l'espace par un vecteur de $\mathcal{D}$;
- Meilleurs ensembles (PSS) : $\text{cm}(\mathcal{D}, \boldsymbol{v}) > 0 \quad \forall \boldsymbol{v} \neq 0$ (requiert $|\mathcal{D}| \geq n + 1$).

Mesure cosinus et PSS

Une mesure de qualité

Soit $\mathcal{D} \subset \mathbb{R}^n$ et $\boldsymbol{v} \in \mathbb{R}^n \setminus \{0\}$, la mesure cosinus de $\mathcal{D}$ en $\boldsymbol{v}$ est

$$\text{cm}(\mathcal{D}, \boldsymbol{v}) \;=\; \max_{\boldsymbol{d} \in \mathcal{D}} \frac{\boldsymbol{d}^\top \boldsymbol{v}}{\|\boldsymbol{d}\| \, \|\boldsymbol{v}\|}$$

- Distance angulaire de l'analyse de données textuelles (`word2vec`);
- Ici sert à approcher tout vecteur de l'espace par un vecteur de $\mathcal{D}$;
- Meilleurs ensembles (PSS) : $\text{cm}(\mathcal{D}, \boldsymbol{v}) > 0 \quad \forall \boldsymbol{v} \neq 0$ (requiert $|\mathcal{D}| \geq n + 1$).

Exemple : $\mathcal{D}_\oplus = \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n, \text{-}\boldsymbol{e}_1, \ldots, \text{-}\boldsymbol{e}_n\}$

- $|\mathcal{D}_\oplus| = 2n$.
- $\forall \boldsymbol{v}, \quad \text{cm}\,(\mathcal{D}_\oplus, \boldsymbol{v}) \geq \frac{1}{\sqrt{n}}$.

Complexité en recherche directe

Hypothèse : Il existe $\kappa \in (0,1)$ tel que $\forall k,\ \mathrm{cm}(\mathcal{D}_k, \boldsymbol{v}) \geq \kappa\ \forall \boldsymbol{v} \in \mathbb{R}^n$, et $|\mathcal{D}_k| = r$ pour tout k.

Théorème

Soit $\epsilon \in (0,1)$ et N_ϵ le nombre d'appels à f pour satisfaire $\|\nabla f(\boldsymbol{x}_k)\| < \epsilon$. Alors

$$N_\epsilon \ \leq\ \mathcal{O}\left(r\,(\kappa\,\epsilon)^{-2}\right).$$

Complexité en recherche directe

Hypothèse : Il existe $\kappa \in (0,1)$ tel que $\forall k,\ \text{cm}(\mathcal{D}_k, \boldsymbol{v}) \geq \kappa\ \forall \boldsymbol{v} \in \mathbb{R}^n$, et $|\mathcal{D}_k| = r$ pour tout k.

Théorème

Soit $\epsilon \in (0,1)$ et N_ϵ le nombre d'appels à f pour satisfaire $\|\nabla f(\boldsymbol{x}_k)\| < \epsilon$. Alors

$$N_\epsilon \ \leq\ \mathcal{O}\left(r\,(\kappa\,\epsilon)^{-2}\right).$$

- Avec $\mathcal{D}_k = \mathcal{D}_\oplus$, on a $\kappa = 1/\sqrt{n}, r = 2n$, d'où

$$N_\epsilon \ \leq\ \mathcal{O}\left(n^2\,\epsilon^{-2}\right).$$

- Le facteur n^2 ne peut pas être amélioré avec une méthode **déterministe**.

Plus et moins de la recherche directe classique

Positif

Négatif

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

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

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

Plan

Recherche directe “randomisée”

Recherche déterministe

- Directions souvent fixes $\Rightarrow$ Peu d'exploration;
- Complexité en $\mathcal{O}(n^2\epsilon^{-2})$: forte dépendance en la dimension.

Recherche directe "randomisée"

Recherche déterministe

- Directions souvent fixes $\Rightarrow$ Peu d'exploration;
- Complexité en $\mathcal{O}(n^2\epsilon^{-2})$: forte dépendance en la dimension.

Techniques aléatoires

- Recherche directe probabiliste : Utiliser des directions aléatoires $\boldsymbol{D}_k$ telles que

$$\forall \boldsymbol{v}, \mathbb{P}(\mathrm{cm}(\boldsymbol{D}_k, \boldsymbol{v}) \geq \kappa | \boldsymbol{D}_0, \ldots, \boldsymbol{D}_{k-1}) \geq p.$$

Recherche directe "randomisée"

Recherche déterministe

- Directions souvent fixes $\Rightarrow$ Peu d'exploration;
- Complexité en $\mathcal{O}(n^2\epsilon^{-2})$: forte dépendance en la dimension.

Techniques aléatoires

- Recherche directe probabiliste : Utiliser des directions aléatoires $\boldsymbol{D}_k$ telles que

$$\forall \boldsymbol{v}, \mathbb{P}(\mathrm{cm}(\boldsymbol{D}_k, \boldsymbol{v}) \geq \kappa | \boldsymbol{D}_0, \ldots, \boldsymbol{D}_{k-1}) \geq p.$$

- Recherche aléatoire de Nesterov : Tirer $\boldsymbol{u}_k \sim \mathcal{N}(0, \boldsymbol{I})$ et prendre

$$\frac{f(\boldsymbol{x} + \mu \boldsymbol{u}) - f(x)}{\mu} \boldsymbol{u} \quad \text{ou} \quad \frac{f(\boldsymbol{x} + \mu \boldsymbol{u}) - f(x - \mu \boldsymbol{u})}{\mu} \boldsymbol{u}$$

Recherche directe "randomisée"

Recherche déterministe

- Directions souvent fixes $\Rightarrow$ Peu d'exploration;
- Complexité en $\mathcal{O}(n^2\epsilon^{-2})$: forte dépendance en la dimension.

Techniques aléatoires

- Recherche directe probabiliste : Utiliser des directions aléatoires $\boldsymbol{D}_k$ telles que

$$\forall \boldsymbol{v}, \mathbb{P}(\mathrm{cm}(\boldsymbol{D}_k, \boldsymbol{v}) \geq \kappa | \boldsymbol{D}_0, \ldots, \boldsymbol{D}_{k-1}) \geq p.$$

- Recherche aléatoire de Nesterov : Tirer $\boldsymbol{u}_k \sim \mathcal{N}(0, \boldsymbol{I})$ et prendre

$$\frac{f(\boldsymbol{x} + \mu \boldsymbol{u}) - f(x)}{\mu} \boldsymbol{u} \quad \text{ou} \quad \frac{f(\boldsymbol{x} + \mu \boldsymbol{u}) - f(x - \mu \boldsymbol{u})}{\mu} \boldsymbol{u}$$

- **Dans les deux cas : Complexité $\mathcal{O}(n\epsilon^{-2})$!**

Directions aléatoires

Idée

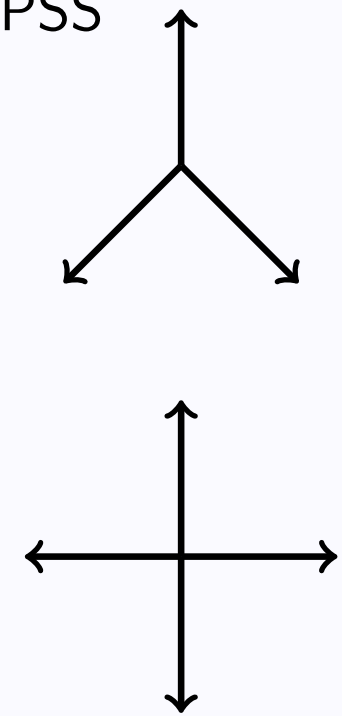
Utiliser des directions aléatoires en recherche directe.

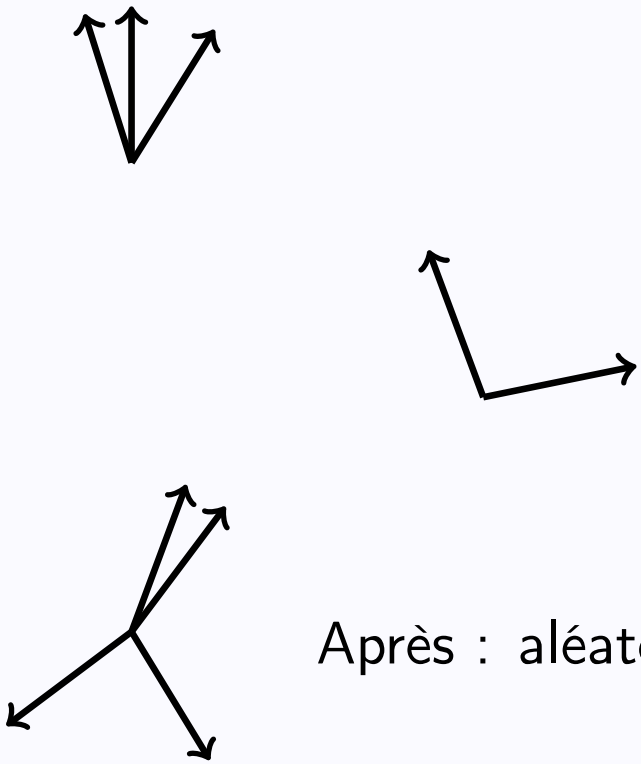
Directions aléatoires

Idée

Utiliser des directions aléatoires en recherche directe.

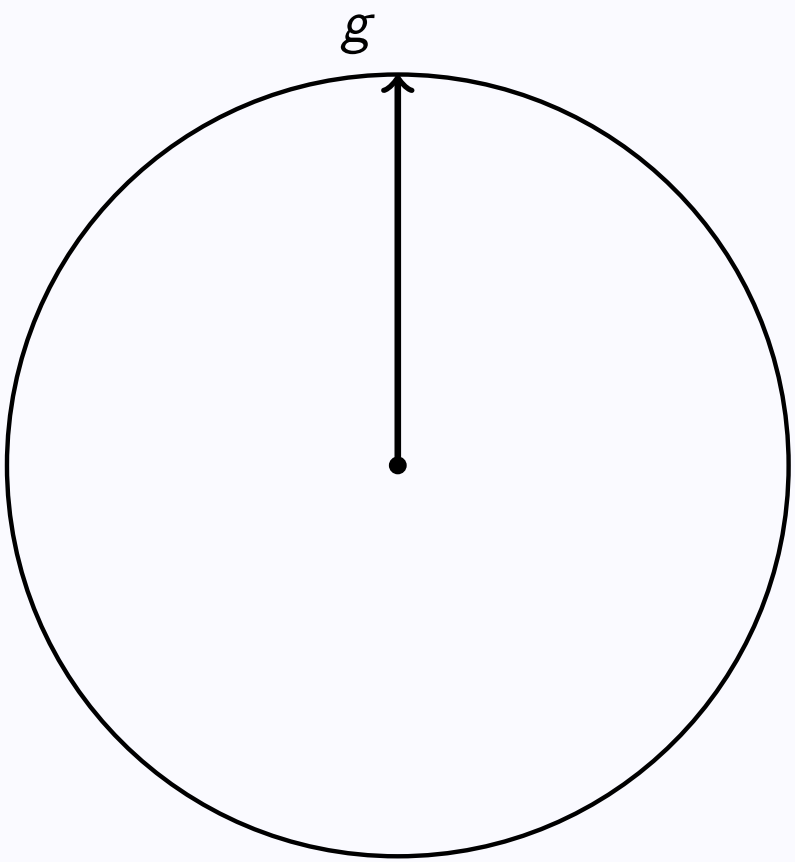
Avant : PSS



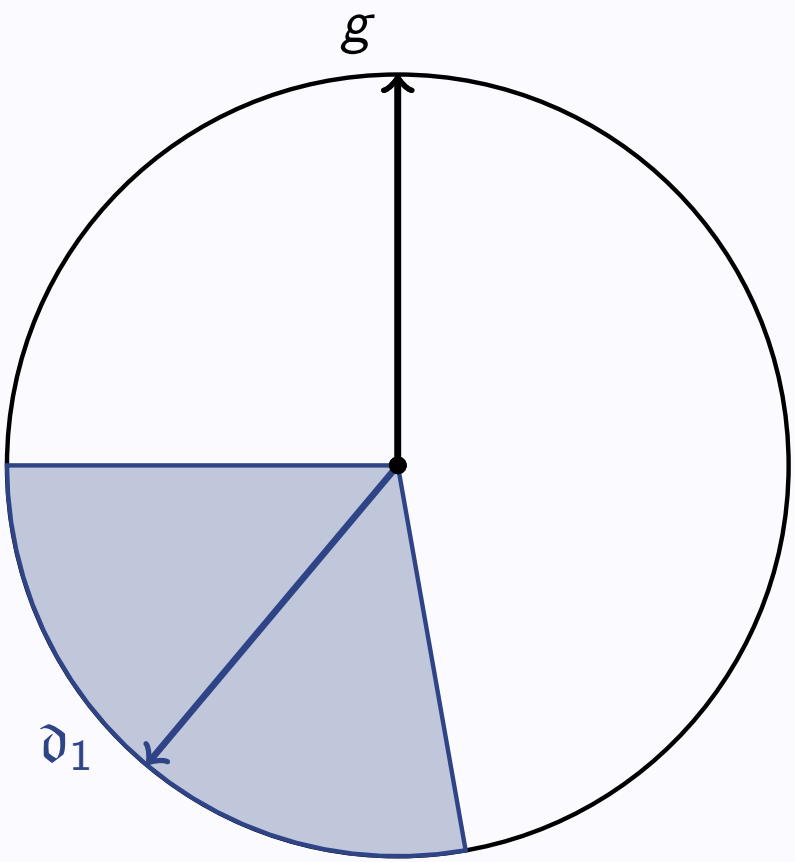


Après : aléatoire

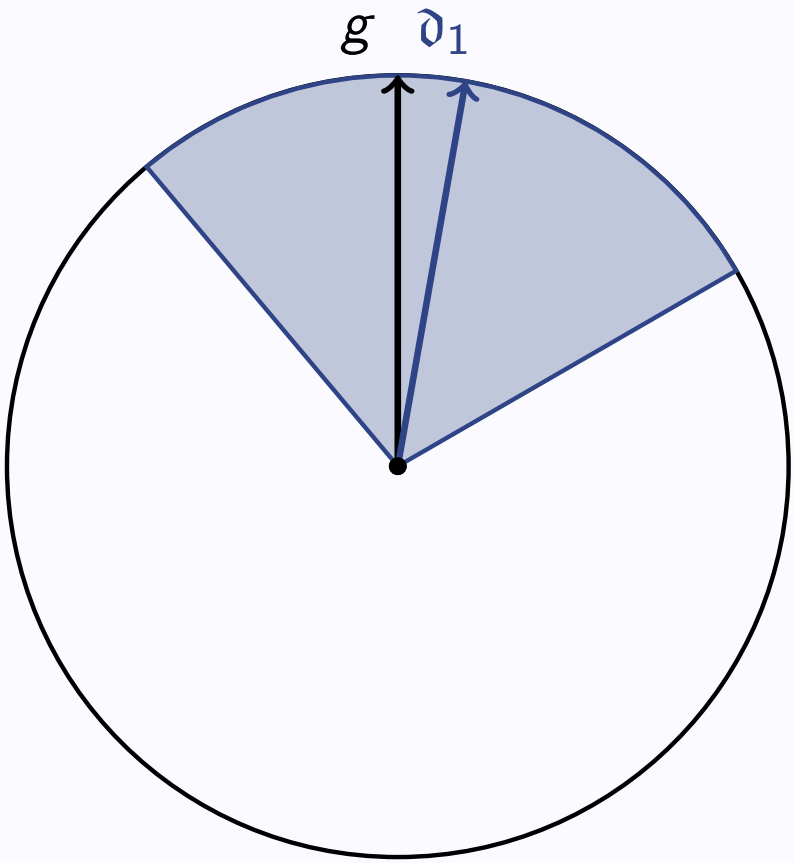
En pratique : deux directions (pas une !)



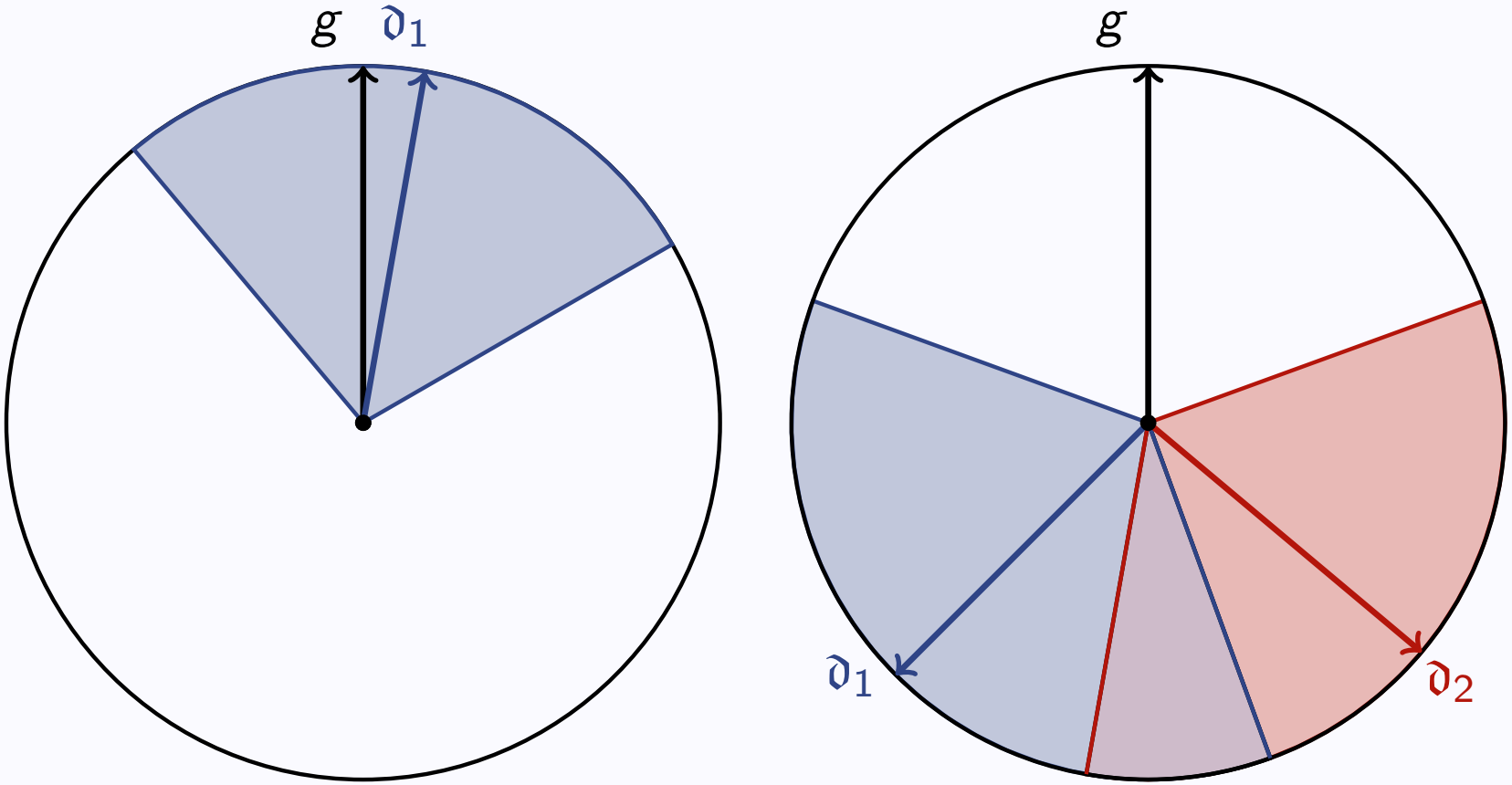
En pratique : deux directions (pas une !)



En pratique : deux directions (pas une !)

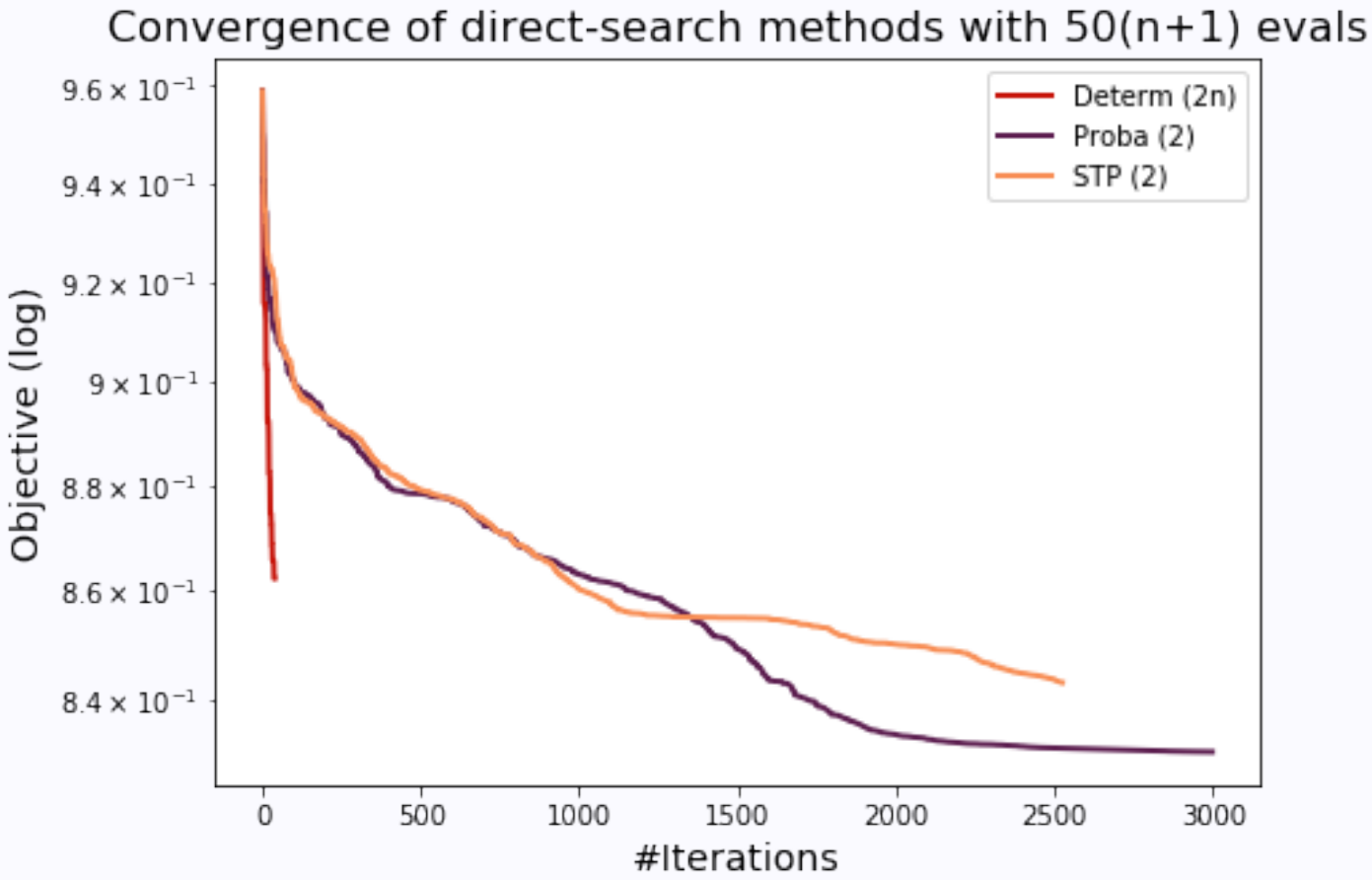


En pratique : deux directions (pas une !)



Et ça marche !

- Régression linéaire robuste avec perte de Tukey;
- Données synthétiques : $n = 100$, budget de $50(n+1)$ évals.



Plan

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 stochastic $\leftrightarrow$ "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 $\boldsymbol{u}_k \sim \mathcal{U}(\mathbb{S}^{n-1})$ et utiliser

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

Méthodes à un point

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

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

- Complexité : $\mathcal{O}(n\epsilon^{-3})$ pour problèmes convexes.

Méthodes à un point

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

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

- Complexité : $\mathcal{O}(n\epsilon^{-3})$ pour problèmes convexes.
- Bonne dépendance en n, moins en ϵ.

Méthodes à un point

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

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

- Complexité : $\mathcal{O}(n\epsilon^{-3})$ pour problèmes convexes.
- Bonne dépendance en n, moins en ϵ.

Méthodes à un point

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

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

- Complexité : $\mathcal{O}(n\epsilon^{-3})$ pour problèmes convexes.
- Bonne dépendance en n, moins en ϵ.

Méthodes deux/multi-pas

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

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

- Complexité : $\mathcal{O}(n\epsilon^{-2})$ pour problèmes convexes.

Résumé : recherche directe

Quelques idées-clés

- Exploration via décroissance de fonction (toujours un meilleur point;
- Échantillonnage mais pas de construction de dérivées;
- Les méthodes classiques rajoutent maintenant de l'aléatoire pour des raisons de performance en gardant des garanties.

Liens avec l'IA

- Géométriques : distance angulaire;
- Bandit/Optimisation en ligne : nouveaux outils stochastiques.

Logiciels

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.

Révisons ! Recherche directe

1. Quelle est la différence entre une recherche sur grille *(grid search)* et une recherche aléatoire *(random search)* ?
2. Comment améliorer la dépendance en la dimension d'une recherche directe standard ?
3. Sur quel concept d'IA se reposent les méthodes de "feedback" ?

Plan

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

Algorithme : Régions de confiance sans dérivées

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

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

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

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

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

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

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

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

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 $\delta_{k+1} = \delta_k/2$.

Analyse déterministe

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

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

Construire des modèles pleinement linéaires

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

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

Construire des modèles pleinement linéaires (2)

Régression polynomiale

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

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

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

Construire des modèles pleinement linéaires (2)

Régression polynomiale

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

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

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

Résultat

Si $\mathcal{Y} \subset B(\boldsymbol{y}^0, \delta)$ est *équilibré*, alors m est pleinement linéaire $B(\boldsymbol{y}^0, \delta)$.

Ex)

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

Hypothèse

À chaque itération k,

- m_k est κ-pleinement linéaire with $\kappa > 0$;
- m_k construit avec au plus r évaluations de f dans $B(\boldsymbol{x}_k, \delta_k)$.

Hypothèse

À chaque itération k,

- m_k est κ-pleinement linéaire with $\kappa > 0$;
- m_k construit avec au plus r évaluations de f dans $B(\boldsymbol{x}_k, \delta_k)$.

Résultat

On obtient $\|\nabla f(\boldsymbol{x}_k)\| < \epsilon$ en au plus

- $\mathcal{O}(\kappa^2 \epsilon^{-2})$ itérations;
- $\mathcal{O}(r\, \kappa^2 \epsilon^{-2})$ évaluations.

Ex) Interpolation/régression linéaire : $r = \mathcal{O}(n), \kappa = \mathcal{O}(\sqrt{n})$
$\Rightarrow$ Complexité en $\mathcal{O}(n^2 \epsilon^{-2})$.

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**.

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.

Complexité

Hypothèses

À chaque itération k,

- m_k est κ-pleinement quadratique, $\kappa > 0$;
- m_k construit avec au plus r nouvelles évaluations of f.

Complexité

Hypothèses

À chaque itération k,

- m_k est κ-pleinement quadratique, $\kappa > 0$;
- m_k construit avec au plus r nouvelles évaluations of f.

Résultat

On obtient $\|\nabla f(\boldsymbol{x}_k)\| \leq \epsilon_g$ et $\nabla^2 f(\boldsymbol{x}_k) \succeq -\epsilon_g^{1/2} \boldsymbol{I}_n$ en au plus

- $\mathcal{O}(\kappa^3 \epsilon_g^{-3/2})$ itérations;
- $\mathcal{O}(r\, \kappa^3 \epsilon_g^{-3/2})$ évaluations.

Complexité

Hypothèses

À chaque itération k,

- m_k est κ-pleinement quadratique, $\kappa > 0$;
- m_k construit avec au plus r nouvelles évaluations of f.

Résultat

On obtient $\|\nabla f(\boldsymbol{x}_k)\| \leq \epsilon_g$ et $\nabla^2 f(\boldsymbol{x}_k) \succeq -\epsilon_g^{1/2} \boldsymbol{I}_n$ en au plus

- $\mathcal{O}(\kappa^3 \epsilon_g^{-3/2})$ itérations;
- $\mathcal{O}(r\, \kappa^3 \epsilon_g^{-3/2})$ évaluations.

Exemple : Interpolation/régression

- $r = \mathcal{O}(n^2), \kappa = \mathcal{O}(n)$;
- Complexité en $\mathcal{O}(n^5 \epsilon_g^{-3/2})$.

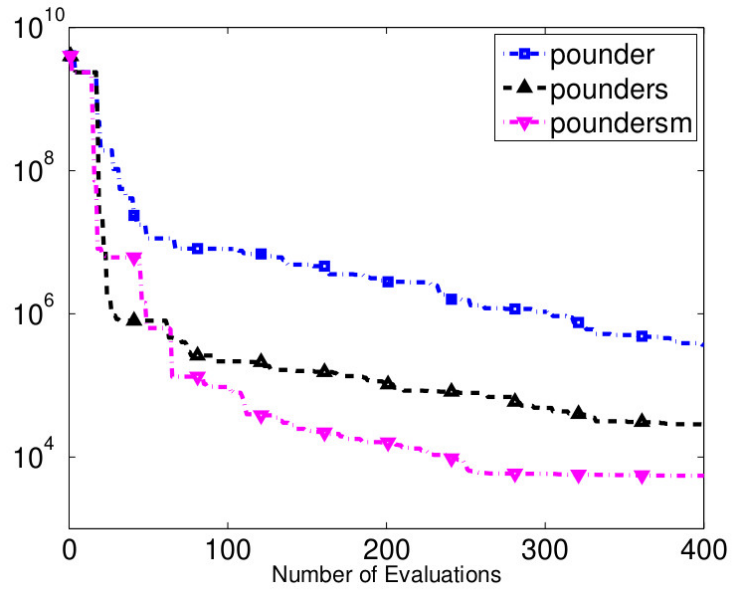
Remarque : Utiliser la structure

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



- 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.

Plan

Sur l'approche déterministe

Avec des modèles linéaires

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

Sur l'approche déterministe

Avec des modèles linéaires

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

Idée

Supposer que les modèles sont pleinement linéaires en probabilité.

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

Propriété probabiliste

Rappel : Le modèle m_k est κ-pleinement linéaire en $(\boldsymbol{x}_k, \delta_k)$ si pour tout $\boldsymbol{y} \in \mathcal{B}(\boldsymbol{x}_k, \delta_k)$:

$$|m_k(\boldsymbol{y}) - f(\boldsymbol{y})| \leq \kappa \delta_k^2, \quad \|\nabla m_k(\boldsymbol{y}) - \nabla f(\boldsymbol{y})\| \leq \kappa \delta_k$$

Propriété probabiliste

Rappel : Le modèle m_k est κ-pleinement linéaire en $(\boldsymbol{x}_k, \delta_k)$ si pour tout $\boldsymbol{y} \in \mathcal{B}(\boldsymbol{x}_k, \delta_k)$:

$$|m_k(\boldsymbol{y}) - f(\boldsymbol{y})| \leq \kappa \delta_k^2, \quad \|\nabla m_k(\boldsymbol{y}) - \nabla f(\boldsymbol{y})\| \leq \kappa \delta_k$$

Modèle probabiliste

Une **suite aléatoire** de modèles $\{m_k\}$ est (p, κ)-pleinement linéaire si

$$\begin{aligned} \mathbb{P}\left(m_0 \ \kappa\text{-plein. lin. }\right) &\geq p \\ \forall k \geq 1, \quad \mathbb{P}\left(m_k \ \kappa\text{-plein. lin. } \mid m_0, \ldots, m_{k-1}\right) &\geq p, \end{aligned}$$

Hypothèses

Pour chaque itération k,

- $\{m_k\}_k$ est (p, κ)-pleinement linéaire avec $p > 1/2$;
- m_k construit avec r nouveaux appels à f.

Complexité probabiliste

Hypothèses

Pour chaque itération k,

- $\{m_k\}_k$ est (p, κ)-pleinement linéaire avec $p > 1/2$;
- m_k construit avec r nouveaux appels à f.

Théorème

Soient $\epsilon \in (0, 1)$ et N_ϵ le nombre d'évaluations nécessaires pour obtenir $\|\nabla f(\boldsymbol{x}_k)\| \leq \epsilon$. Alors, $N_\epsilon = \mathcal{O}(r\kappa^2\epsilon^{-2})$ avec forte 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 probabilistes en pratique

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

Optimisation bayésienne et modèles

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.

Révisons ! Méthodes à modèles

1. En quoi les méthodes basées sur des modèles exploitent-elles plus l'information de l'objectif ?
2. Expliquer pourquoi les bornes de complexité sont très pessimistes pour les méthodes basées sur des modèles.
3. Donner un exemple de structure de problème qui permet d'utiliser certaines dérivées.

Résumé

Optimisation sans dérivées/DFO

- Pas de dérivées dans l'algorithme (ou pas toutes);
- Différents noms, même but;
- Tout dépend de l'évaluation.

Résumé

Optimisation sans dérivées/DFO

- Pas de dérivées dans l'algorithme (ou pas toutes);
- Différents noms, même but;
- Tout dépend de l'évaluation.

DFO et IA

- Les techniques DFO "randomisées" se basent sur des outils de l'IA;
- Le chemin vers l'IA automatique passe par des approches formalisées (comme en DFO).

Ce qu'il reste à faire

La grande dimension

- Challenge constant;
- Beaucoup d'efforts fournis en IA.

Ce qu'il reste à faire

La grande dimension

- Challenge constant;
- Beaucoup d'efforts fournis en IA.

Comprendre les algorithmes "randomisés"

- Rapport *RASC* du DOE (États-Unis);
- Approche systémique requise, nombreux points d'amélioration;
- Clé pour la grande dimension !

Là où l'IA peut aider

Apprendre un modèle des données

- Cadre d'apprentissage classique;
- Souci : peu de données;
- Utilisation de la structure du problème.

Là où l'IA peut aider

Apprendre un modèle des données

- Cadre d'apprentissage classique;
- Souci : peu de données;
- Utilisation de la structure du problème.

Fournir des outils mathématiques

- Statistique en grande dimension;
- Points de vue typiquement IA (ex : bandits).

Là où l'IA peut aider

Apprendre un modèle des données

- Cadre d'apprentissage classique;
- Souci : peu de données;
- Utilisation de la structure du problème.

Fournir des outils mathématiques

- Statistique en grande dimension;
- Points de vue typiquement IA (ex : bandits).

Rejoint les besoins de l'IA automatisée !

Bibliographie

- A. R. Conn, K. Scheinberg, L. N. Vicente, *Introduction to derivative-free optimization*, SIAM, 2009.
- C. Audet, W. Hare, *Derivative-Free and Blackbox Optimization*, Springer, 2017.
- 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.
- A. Flaxman, A. T. Kalai and H. B. McMahan, *Online convex optimization in the bandit setting: Gradient descent without a gradient*, Symposium on Discrete Algorithms (SODA), 2005.
- P. I. Frazier, *Bayesian Optimization*, Tutorials in Operations Research, 2018.

Plus de détails

- P. Auer, *Using Confidence Bounds for Exploitation-Exploration Trade-offs*, Journal of Machine Learning Research, 2002.
- A. Bandeira, K. Scheinberg and L. N. Vicente, *Convergence of trust-region methods based on probabilistic models*, SIAM Journal on Optimization, 2014.
- A.J. Booker, J.E. Dennis Jr., P.D. Frank, D.W. Moore and D.B. Serafini, *Managing surrogate objectives to optimize a helicopter rotor design – further ex- periments*, AIAA/ISMOO Symposium on Multidisciplinary Analysis and Optimization, 1998.
- A.R. Conn, K. Scheinberg and L.N. Vicente, *Geometry of interpolation sets in derivative free optimization*, Mathematical Programming, 2018.
- S. Gratton, C. W. Royer, L. N. Vicente and Z. Zhang, *Direct search based on probabilistic descent*, SIAM Journal on Optimization, 2015.
- S. Gratton, C. W. Royer, L. N. Vicente and Z. Zhang, *Complexity and global rates of trust-region methods based on probabilistic models*, IMA Journal of Numerical Analysis, 2018.
- K. Jamieson et al., *Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization*, Journal of Machine Learning Research, 2018.
- D. Lakhmiri et al., *HyperNOMAD*: https://github.com/bbopt/HyperNOMAD
- Yu. Nesterov and V. Spokoiny, *Random gradient-free minimization of convex functions*, Foundations of Computational Mathematics, 2017.
- T. M. Ragonneau and Z. Zhang, *PDFO*: https://www.pdfo.net/
- K. Scheinberg and Ph. L. Toint, *Self-correcting geometry in model-based algorithms for derivative-free unconstrained optimization*, SIAM Journal on Optimization, 2010.

Merci de votre attention !

clement.royer@dauphine.psl.eu

Crédits image

- 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.