Derivative-Free Optimization

Clément W. Royer

Machine Learning and AI for Economics and Finance Summer School

June 9, 2021

First and foremost

- Associate professor @ Dauphine-PSL;
- Researcher @ LAMSADE and PRAIRIE.
- Research: Nonlinear optimization and applications to data science;
- Focus:
  - Nonconvex problems;
  - **Derivative-free algorithms**.

What is this lecture about?

- Derivative-free optimization (DFO)?
- Black-box optimization?
- Surrogate-based optimization?
- Response surface methodology/Design of experiments?
- Automated machine learning?
- Hyperparameter tuning?

What is this lecture about?

- Derivative-free optimization (DFO)?
- Black-box optimization?
- Surrogate-based optimization?
- Response surface methodology/Design of experiments?
- Automated machine learning?
- Hyperparameter tuning?

- **All of the above** provided no derivatives are used;
- **More of the first two** (my background);
- A bit of the last two (this summer school).

Outline

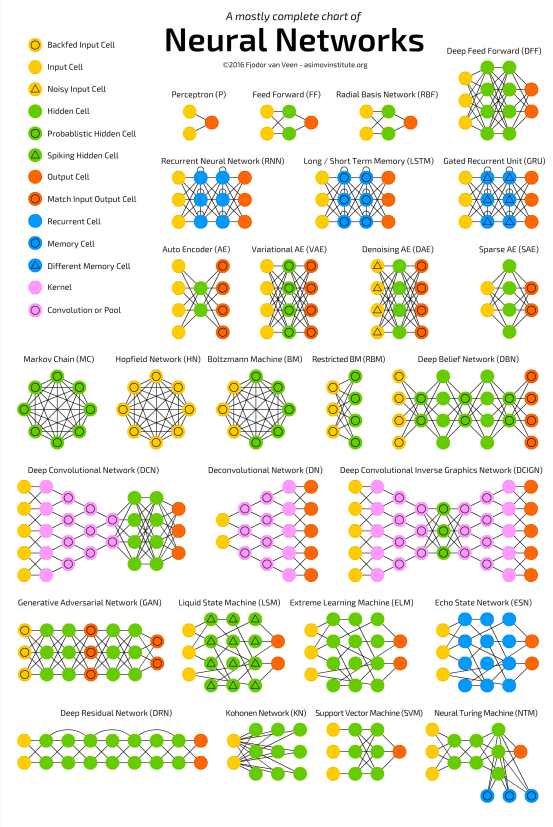
Outline

Outline

A modern challenge

Say you want to train a neural network...

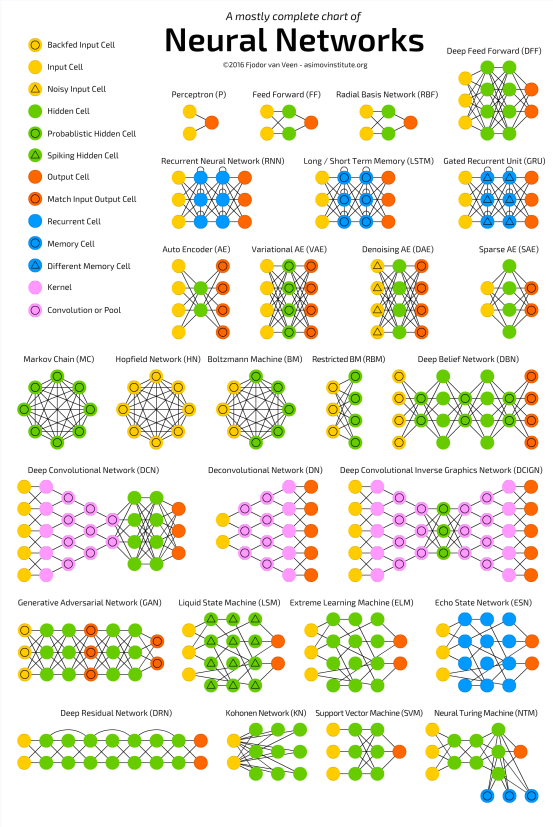
- What is your architecture? (Convolutional, Recurrent,etc)
- What is your training algorithm? (Adam, RMSProp, SG,etc)
- How do you choose your learning rate?



A modern challenge

Say you want to train a neural network...

- What is your architecture? (Convolutional, Recurrent,etc)
- What is your training algorithm? (Adam, RMSProp, SG,etc)
- How do you choose your learning rate?



Hyperparameter optimization

- Each change of hyperparameter involves another round of training (hours, days of CPU time + money!);
- Integer/Categorical/Continuous variables.

Goal: Reach automated ML!

Meanwhile...

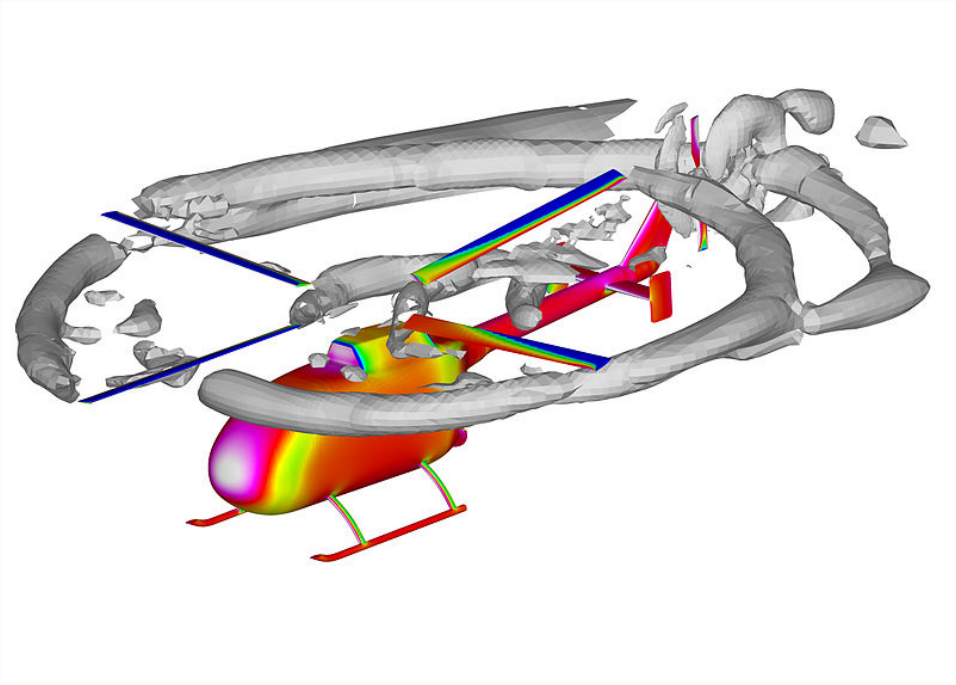
Scientific computing

- Extensive use of computer simulation in physics-based applications (aerospace engineering, chemical processes, climate);
- Very expensive runs;
- Need for parameter calibration.

Simulation-based optimization

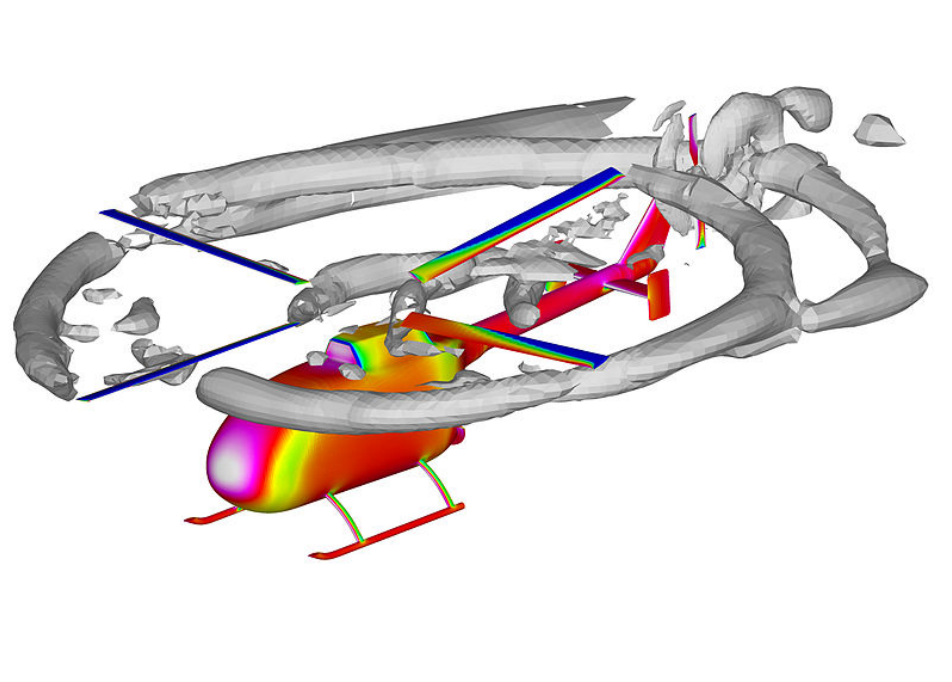
- Stemmed from the need to optimize with very expensive evaluations;
- Relied partly on standard optimization techniques.

Classical example: Rotor helicopter design (Booker et al. 1998)



- About 30 parameters;
- 1 simulation: 2 weeks of computational fluid dynamics simulation;
- A simulation failed 60% of the time.

Classical example: Rotor helicopter design (Booker et al. 1998)



- About 30 parameters;
- 1 simulation: 2 weeks of computational fluid dynamics simulation;
- A simulation failed 60% of the time.

Ubiquitous in multidisciplinary optimization:

- Several codes interfaced;
- Numerical simulations;
- Large amount of calculation, possible failures.

The problems we want to look at

Common features in automated ML and simulation-based optimization

- Require significant amount of computing power;
- Choosing the best parameters is not easy;
- The application people want the best value possible in order to build/deploy the system in real-world settings!

Our setup

- **Selecting the best parameters can be posed as an optimization problem**;
- The objective function in this problem is very expensive.

A definition of DFO

Derivative-free optimization problem

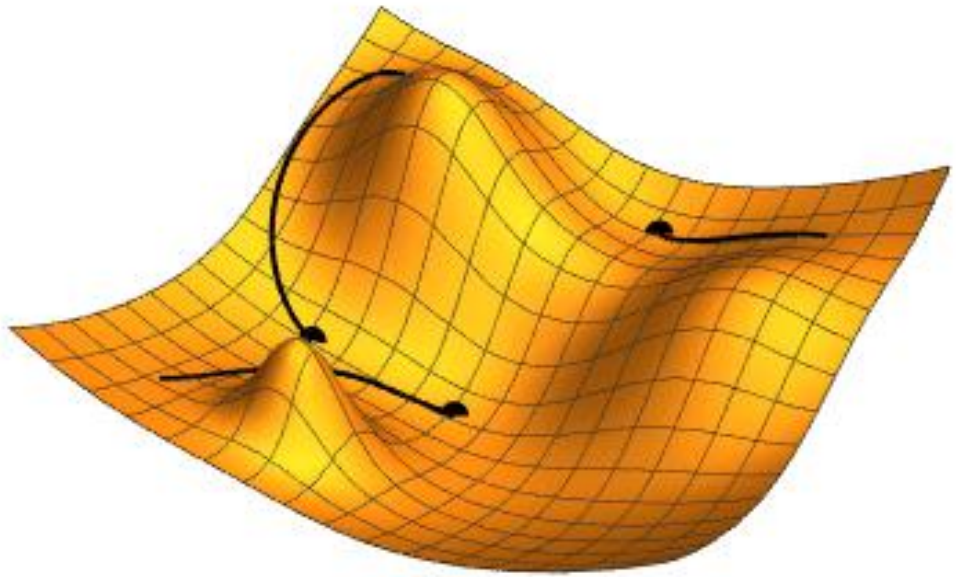
Some derivatives are unavailable for optimization purposes.

A definition of DFO

Derivative-free optimization problem

Some derivatives are unavailable for optimization purposes.

- Optimization is really connected to derivatives (gradient descent, optimality conditions);
- **Some**: It only takes one missing derivative!
- **Unavailable**: They may or may not exist!



A warning

If you can afford derivatives you should use them!

- Handcoding still exists;
- Finite differences can sometimes by used;
- Automatic differentiation is very powerful these days.

Most derivative-based solvers work with inexact derivatives.

A warning

If you can afford derivatives you should use them!

- Handcoding still exists;
- Finite differences can sometimes by used;
- Automatic differentiation is very powerful these days.

Most derivative-based solvers work with inexact derivatives.

Derivatives not affordable

- Complex phenomena $\Rightarrow$ Programming prone to errors;
- Costly/noisy evaluations $\Rightarrow$ Problem for finite differences;
- Source code not available, company-owned $\Rightarrow$ No hope for AD.

Outline

Problem

Our focus

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

- $\boldsymbol{x}$: variables;
- f: objective function;
- $\mathcal{F}$: set of feasible points.

Our assumptions

- f is bounded from below on $\mathcal{F}$: $f(\boldsymbol{x}) \geq f_{\text{low}}$;
- f is evaluated as a result of complex/long calculations:
  - Training a neural network/a language model;
  - Running a market simulation;
  - Taking a blood sample from a patient.

What do we want to do?

Find the optimal parameters?

- What does that even mean to be optimal without derivatives?
- Global optimality only possible under some assumptions (convexity) or if you wait forever.

What do we want to do?

Find the optimal parameters?

- What does that even mean to be optimal without derivatives?
- Global optimality only possible under some assumptions (convexity) or if you wait forever.

Find better parameters

- Any improvement can be valuable (and translate into efficiency/money);
- Initial configurations can be pretty good if designed by experts.

What do we want to do?

Find the optimal parameters?

- What does that even mean to be optimal without derivatives?
- Global optimality only possible under some assumptions (convexity) or if you wait forever.

Find better parameters

- Any improvement can be valuable (and translate into efficiency/money);
- Initial configurations can be pretty good if designed by experts.

Get provable guarantees

- Certifies that a method is worth using, serves as comparison;
- A popular metric these days: **worst-case complexity**.

Worst-case complexity (for this lecture)

Definition

Given

- A convergence criterion;
- A tolerance $\epsilon > 0$;
- An iterative algorithm;

bound the **worst-case number of function calls** required to satisfy the convergence criterion up to a tolerance ϵ.
The bound as a function of ϵ is called the **worst-case complexity** of the algorithm.

Worst-case complexity (for this lecture)

Definition

Given

- A convergence criterion;
- A tolerance $\epsilon > 0$;
- An iterative algorithm;

bound the **worst-case number of function calls** required to satisfy the convergence criterion up to a tolerance ϵ.
The bound as a function of ϵ is called the **worst-case complexity** of the algorithm.

Convergence criteria

- Differentiable f: $\|\nabla f(\boldsymbol{x}_k)\| \leq \epsilon$;
- Convex f: $f(\boldsymbol{x}_k) - f^* \leq \epsilon$.

Outline

Outline

Grid/Random search

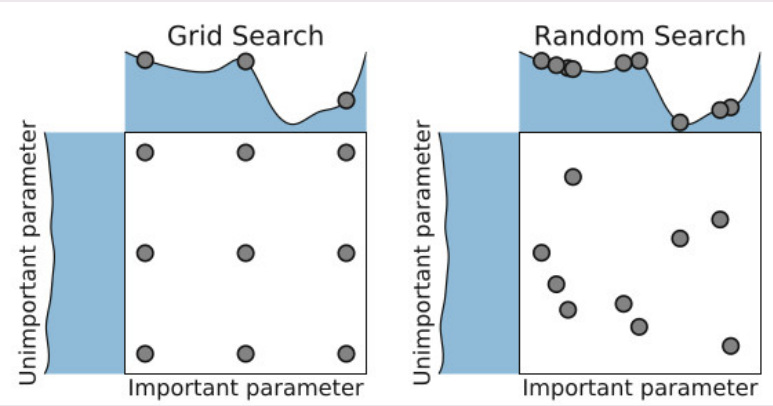
Goal: Solve $\text{minimize}_{\boldsymbol{x} \in \mathcal{F}} f(\boldsymbol{x})$ where only accesses to evaluations of f are available.

Basic search algorithm

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

1. Compute a new point $\boldsymbol{x}_{k+1}$ and $f(\boldsymbol{x}_{k+1})$.
2. If $f(\boldsymbol{x}_{k+1}) < f(\hat{\boldsymbol{x}_k})$ set $\hat{\boldsymbol{x}}_{k+1} = \boldsymbol{x}_{k+1}$, otherwise set $\hat{\boldsymbol{x}}_{k+1} = \hat{\boldsymbol{x}_k}$.
3. If evaluation budget exceeded stop, otherwise increment k by one.

- **Grid search**: Predefined set of values;
- **Random search**: Draw $\boldsymbol{x}_{k+1}$ at random.



Guarantees for random search

Theorem

Let $\hat{x}_K = \text{argmin}_{k=1,\ldots,K} f(\boldsymbol{x}_k)$ the best point obtained by random search, and $f^* = \min_{\boldsymbol{x}\in\mathcal{F}} f(\boldsymbol{x})$. Then

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

if

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

- Very generic result;
- Can require a lot of iterations/evaluations of f;
- All focus on exploration.

Direct search (DS)

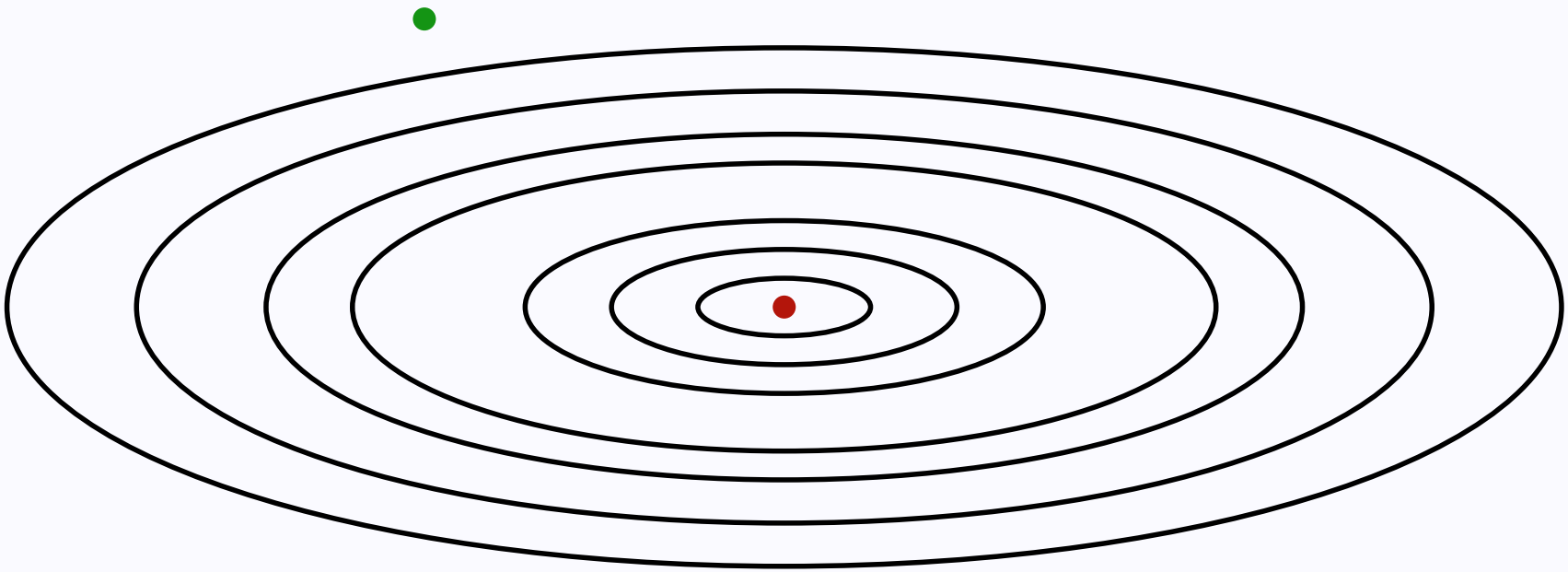
Going beyond grid/random search

- Work well in small dimensions;
- Fully exploratory algorithms;
- Only asymptotic guarantees.

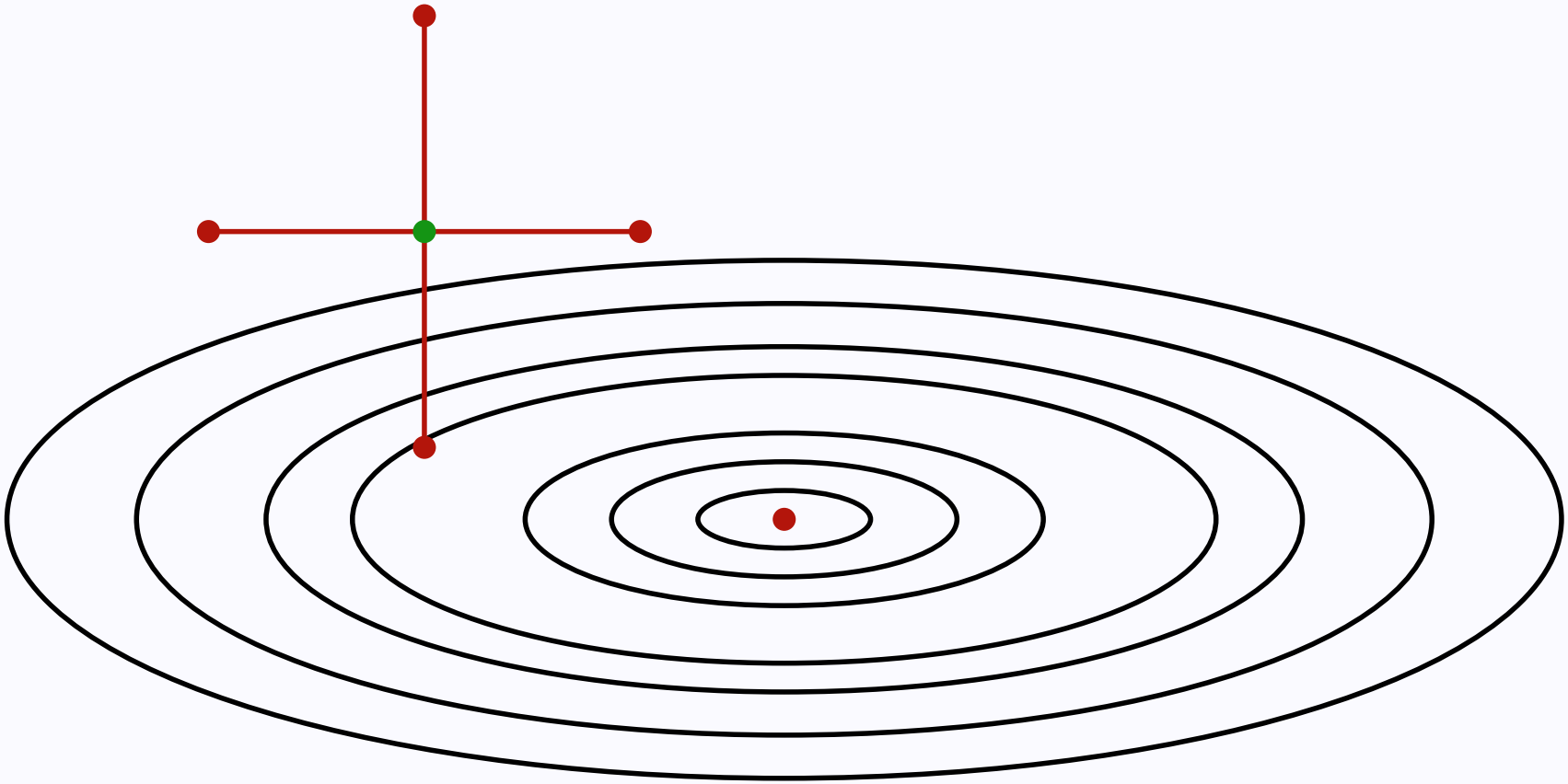
Direct search

- Early appearance: 1960s, convergence theory: 1990s.
- Attractive: simplicity, parallel potential;
- One method (Nelder-Mead, 1965) has more than 125000 citations and is still the default DFO method in MATLAB!

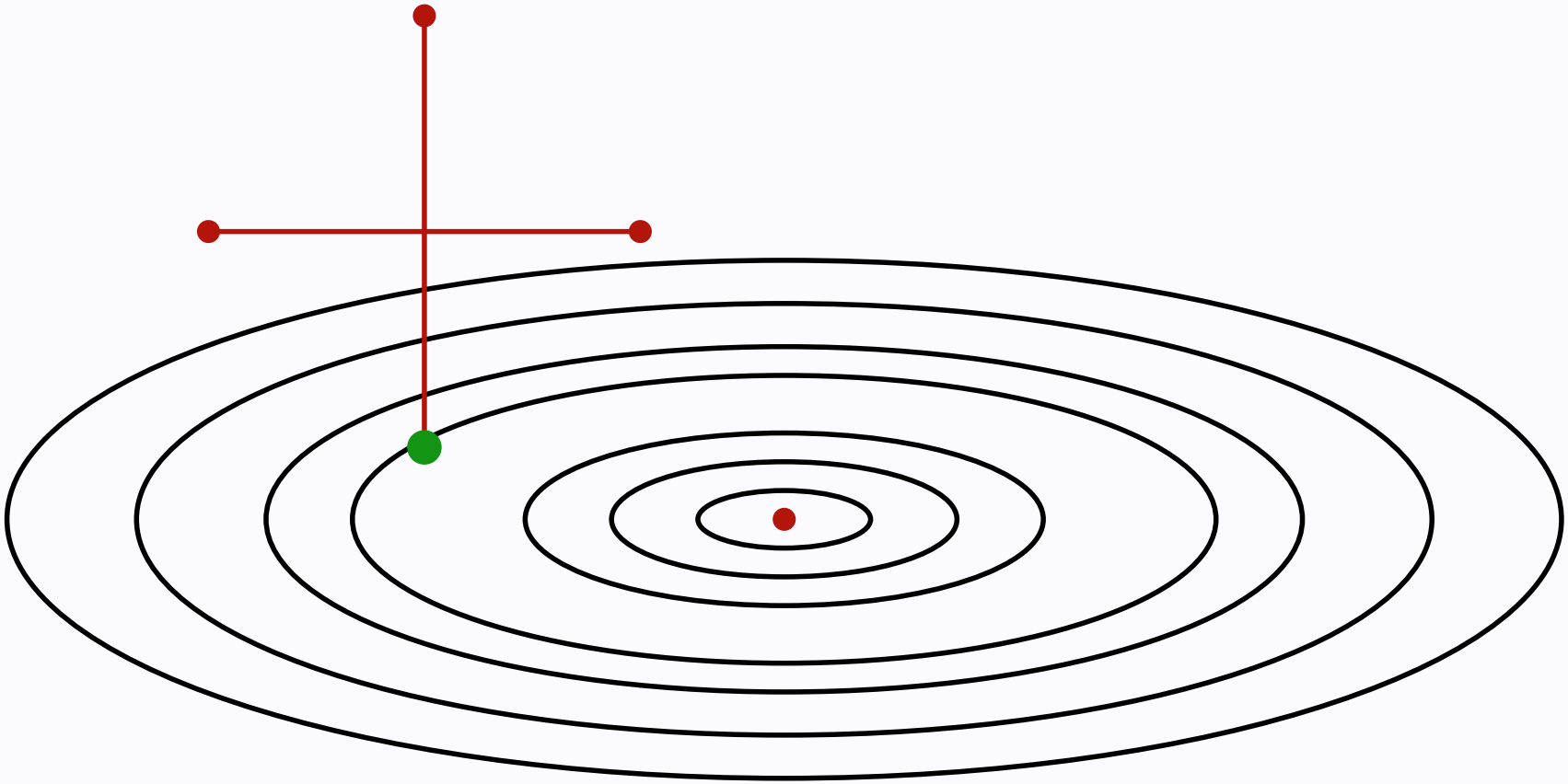
An example of DS : Coordinate Search



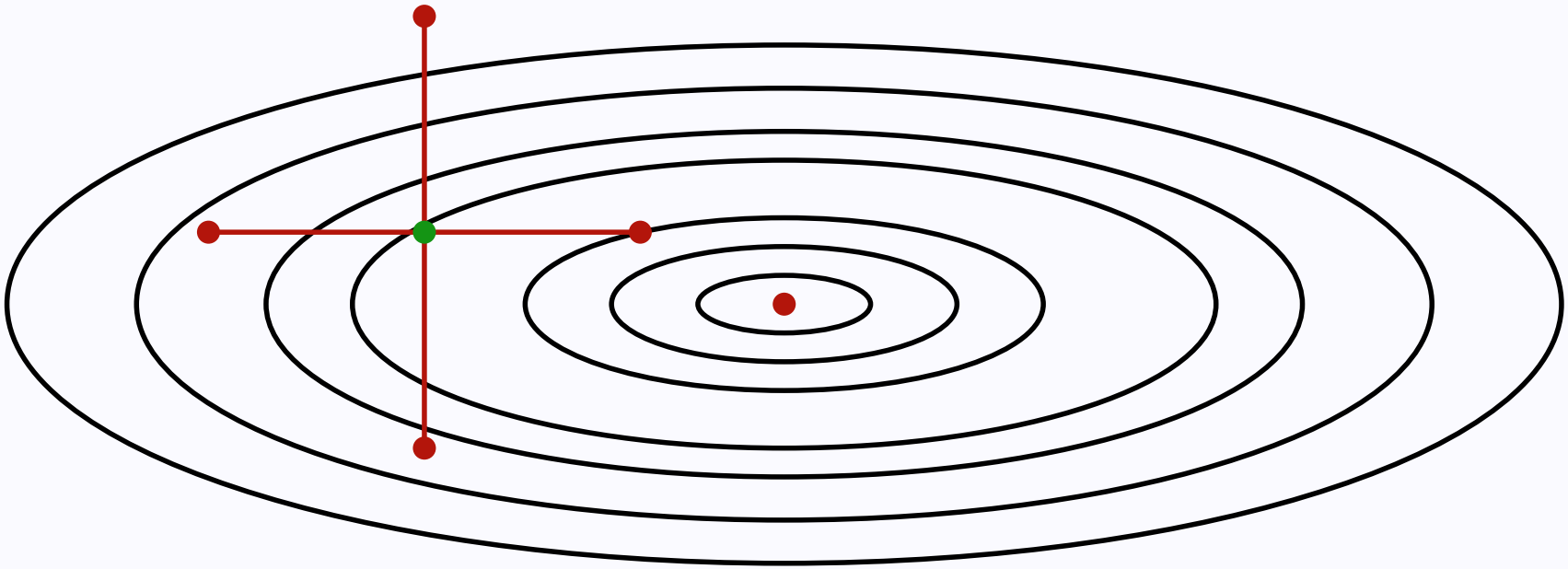
An example of DS : Coordinate Search



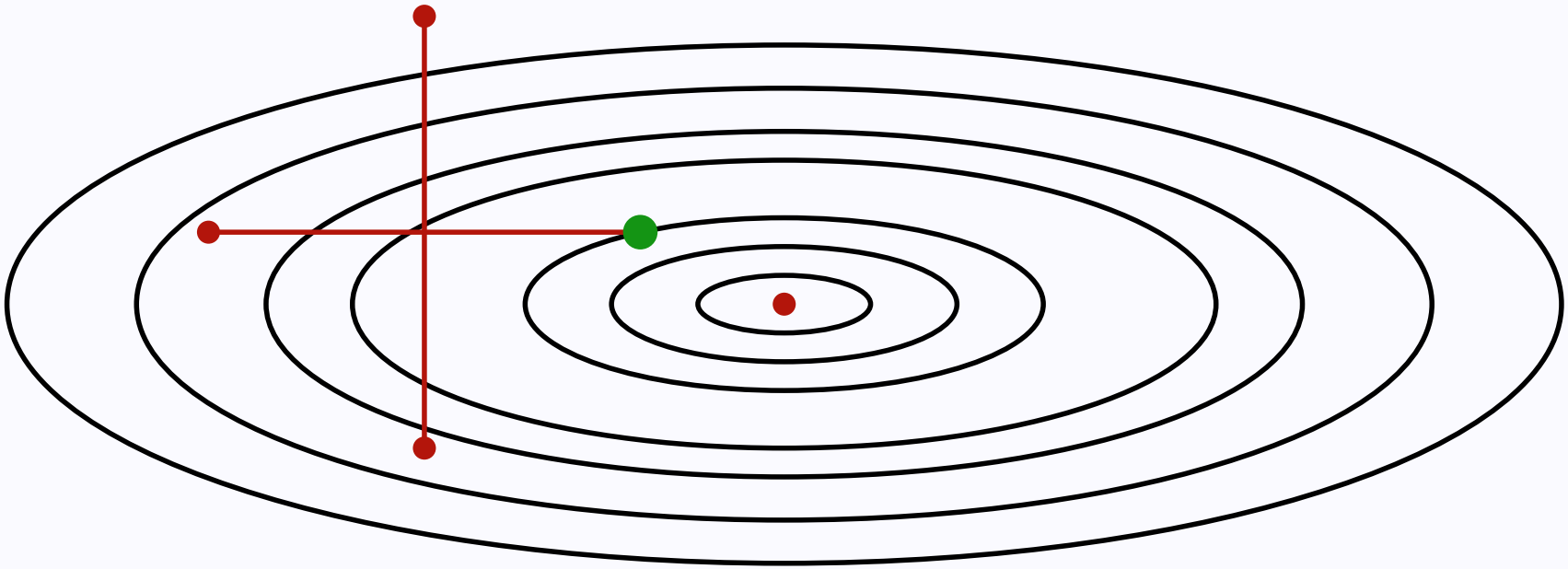
An example of DS : Coordinate Search



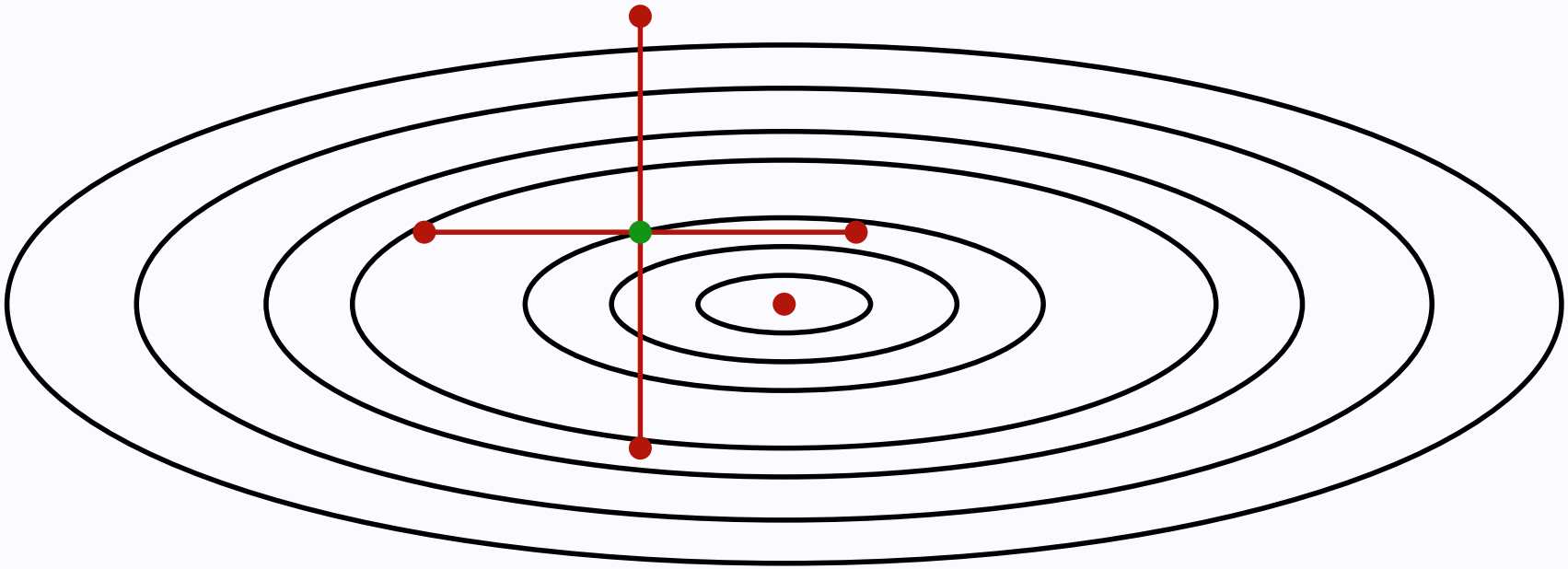
An example of DS : Coordinate Search



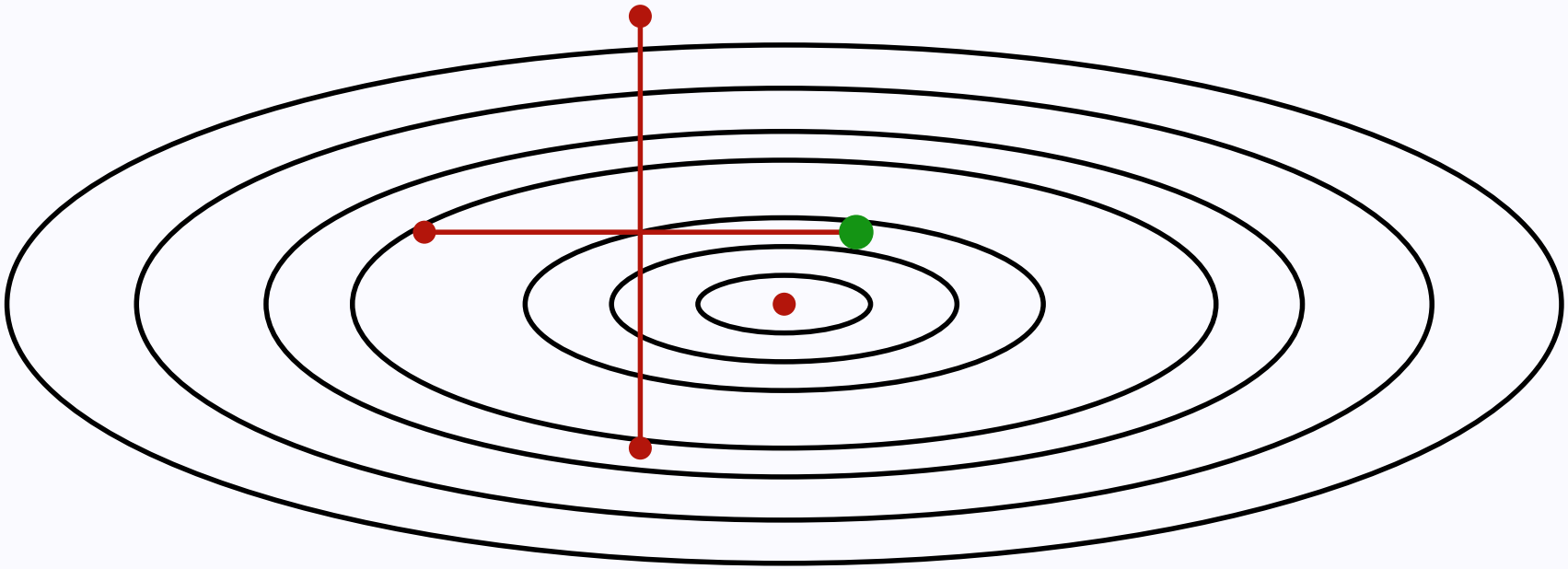
An example of DS : Coordinate Search



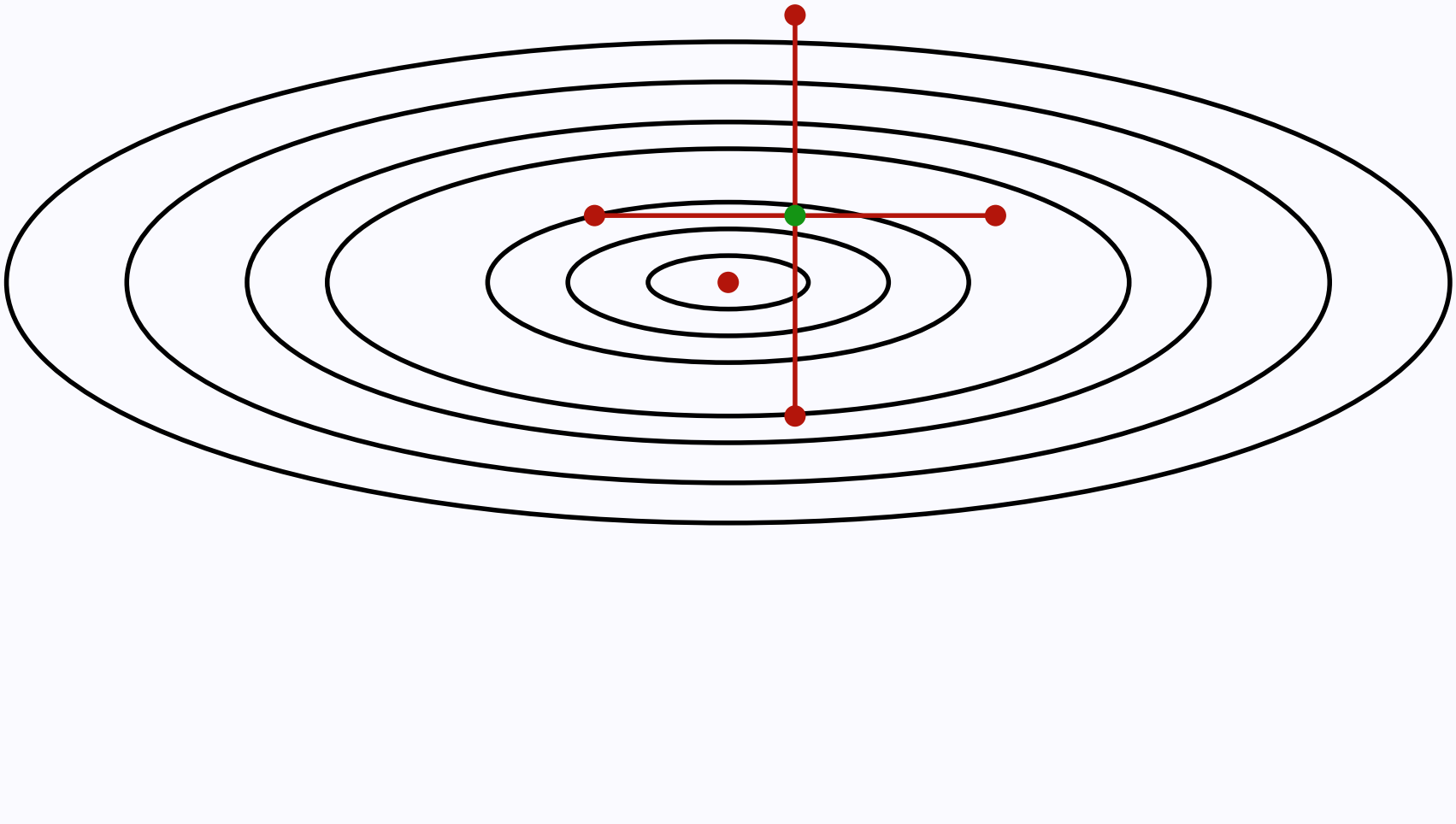
An example of DS : Coordinate Search



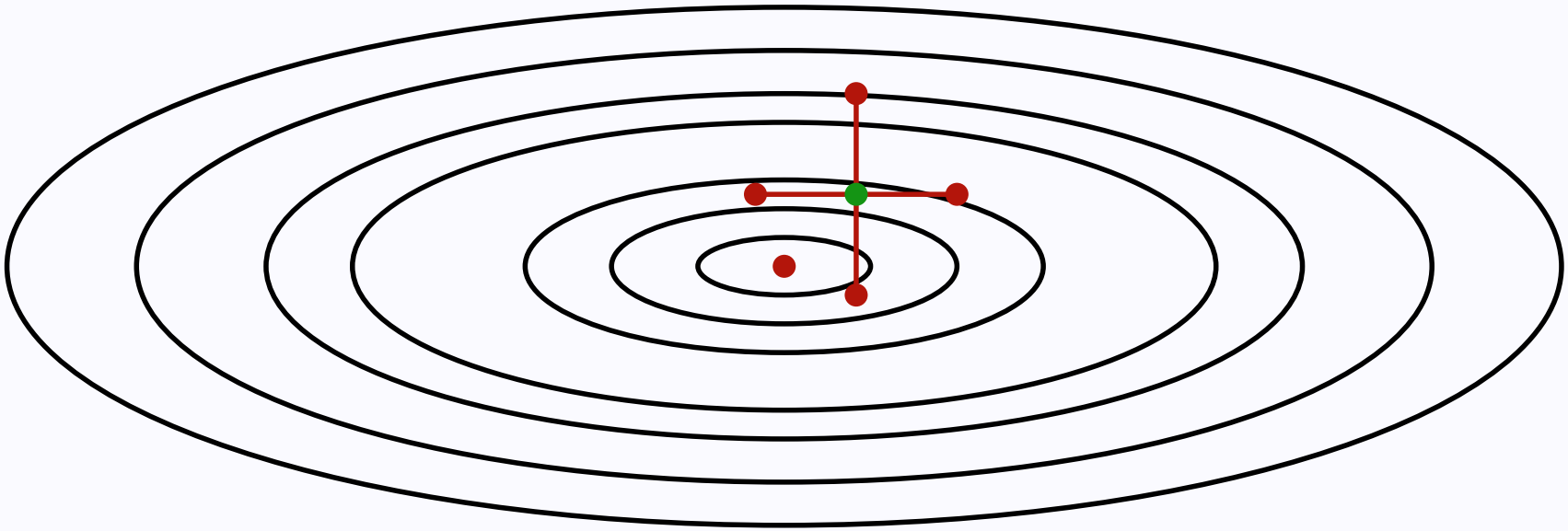
An example of DS : Coordinate Search



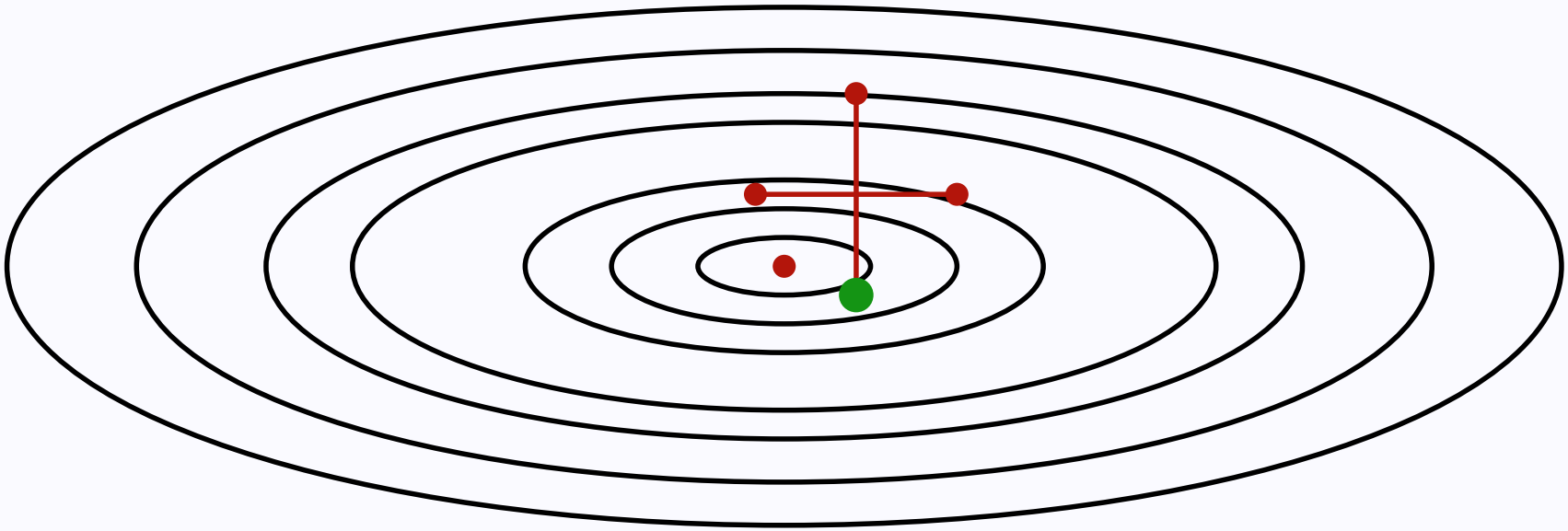
An example of DS : Coordinate Search



An example of DS : Coordinate Search



An example of DS : Coordinate Search



Basic direct-search framework

1. **Initialization:** Set $\boldsymbol{x}_0 \in \mathbb{R}^n, \alpha_0 > 0$.
2. **For** $k = 0, 1, 2, \ldots$
   - Choose a set $\mathcal{D}_k \subset \mathbb{R}^n$ of r directions.

Basic direct-search framework

1. **Initialization:** Set $\boldsymbol{x}_0 \in \mathbb{R}^n, \alpha_0 > 0$.
2. **For** $k = 0, 1, 2, \ldots$
   - Choose a set $\mathcal{D}_k \subset \mathbb{R}^n$ of r directions.
   - **If** it exists $\boldsymbol{d}_k \in \mathcal{D}_k$ so that

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

   then set $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$ and $\alpha_{k+1} \geq \alpha_k$ *(successful iteration)*.

Basic direct-search framework

1. **Initialization:** Set $\boldsymbol{x}_0 \in \mathbb{R}^n, \alpha_0 > 0$.
2. **For** $k = 0, 1, 2, \ldots$
   - Choose a set $\mathcal{D}_k \subset \mathbb{R}^n$ of r directions.
   - **If** it exists $\boldsymbol{d}_k \in \mathcal{D}_k$ so that

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

     then set $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$ and $\alpha_{k+1} \geq \alpha_k$ *(successful iteration)*.
   - **Otherwise** set $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k$ and $\alpha_{k+1} := 0.5\alpha_k$ *(unsuccessful iteration)*.

Basic direct-search framework

1. **Initialization:** Set $\boldsymbol{x}_0 \in \mathbb{R}^n, \alpha_0 > 0$.
2. **For** $k = 0, 1, 2, \ldots$
   - Choose a set $\mathcal{D}_k \subset \mathbb{R}^n$ of r directions.
   - **If** it exists $\boldsymbol{d}_k \in \mathcal{D}_k$ so that

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

     then set $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$ and $\alpha_{k+1} \geq \alpha_k$ *(successful iteration)*.
   - **Otherwise** set $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k$ and $\alpha_{k+1} := 0.5\alpha_k$ *(unsuccessful iteration)*.

Basic direct-search framework

1. **Initialization:** Set $\boldsymbol{x}_0 \in \mathbb{R}^n, \alpha_0 > 0$.
2. **For** $k = 0, 1, 2, \ldots$
   - Choose a set $\mathcal{D}_k \subset \mathbb{R}^n$ of r directions.
   - **If** it exists $\boldsymbol{d}_k \in \mathcal{D}_k$ so that

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

     then set $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$ and $\alpha_{k+1} \geq \alpha_k$ *(successful iteration)*.
   - **Otherwise** set $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k$ and $\alpha_{k+1} := 0.5\alpha_k$ *(unsuccessful iteration)*.

Key aspects

- Sufficient decrease condition;
- Choice of $\mathcal{D}_k$, value of r.

Cosine measure and PSS

A measure of set quality

For a set $\mathcal{D} \subset \mathbb{R}^n$ and $\boldsymbol{v} \in \mathbb{R}^n \setminus \{0\}$, the cosine measure of $\mathcal{D}$ at $\boldsymbol{v}$ is

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

Cosine measure and PSS

A measure of set quality

For a set $\mathcal{D} \subset \mathbb{R}^n$ and $\boldsymbol{v} \in \mathbb{R}^n \setminus \{0\}$, the cosine measure of $\mathcal{D}$ at $\boldsymbol{v}$ is

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

- When $\text{cm}(\mathcal{D}, \boldsymbol{v}) > 0$, $\boldsymbol{v}$ makes an acute angle with some $\boldsymbol{d} \in \mathcal{D}$.

Cosine measure and PSS

A measure of set quality

For a set $\mathcal{D} \subset \mathbb{R}^n$ and $\boldsymbol{v} \in \mathbb{R}^n \setminus \{0\}$, the cosine measure of $\mathcal{D}$ at $\boldsymbol{v}$ is

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

- When $\text{cm}(\mathcal{D}, \boldsymbol{v}) > 0$, $\boldsymbol{v}$ makes an acute angle with some $\boldsymbol{d} \in \mathcal{D}$.
- Ensuring $\text{cm}(\mathcal{D}, \boldsymbol{v}) > 0 \;\; \forall \boldsymbol{v} \neq 0$ requires $\mathcal{D}$ to be a Positive Spanning Set $\Rightarrow |\mathcal{D}| \geq n + 1$ vectors.

Example

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

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

Worst-case complexity in deterministic direct search

Assumption: It exists $\kappa \in (0,1)$ such that $\forall k,\ \text{cm}(\mathcal{D}_k, \boldsymbol{v}) \geq \kappa\ \forall \boldsymbol{v} \in \mathbb{R}^n$, with $|\mathcal{D}_k| = r$.

Theorem

Let $\epsilon \in (0,1)$ and N_ϵ be the number of function evaluations needed to satisfy $\|\nabla f(\boldsymbol{x}_k)\| < \epsilon$. Then,

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

Worst-case complexity in deterministic direct search

Assumption: It exists $\kappa \in (0,1)$ such that $\forall k,\ \mathrm{cm}(\mathcal{D}_k, \boldsymbol{v}) \geq \kappa\ \forall \boldsymbol{v} \in \mathbb{R}^n$, with $|\mathcal{D}_k| = r$.

Theorem

Let $\epsilon \in (0,1)$ and N_ϵ be the number of function evaluations needed to satisfy $\|\nabla f(\boldsymbol{x}_k)\| < \epsilon$. Then,

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

- Choosing $\mathcal{D}_k = \mathcal{D}_\oplus$, one has $\kappa = 1/\sqrt{n}, r = 2n$, and the bound becomes

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

- The n^2 cannot be improved **deterministically**.

Outline

Randomized direct-search techniques

Deterministic direct search

- Using fixed sets of directions gives limited exploration;
- Strong dependency on the dimension ($\mathcal{O}(n^2\epsilon^{-2})$ complexity).

Randomized direct-search techniques

Deterministic direct search

- Using fixed sets of directions gives limited exploration;
- Strong dependency on the dimension ($\mathcal{O}(n^2\epsilon^{-2})$ complexity).

Randomized techniques

- Randomized direct search: Use a random set of directions $\boldsymbol{D}_k$ such that

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

Randomized direct-search techniques

Deterministic direct search

- Using fixed sets of directions gives limited exploration;
- Strong dependency on the dimension ($\mathcal{O}(n^2\epsilon^{-2})$ complexity).

Randomized techniques

- Randomized direct search: Use a random set of directions $\boldsymbol{D}_k$ such that

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

- Nesterov's random search (inspired by Gaussian smoothing): Draw $\boldsymbol{u}_k \sim \mathcal{N}(0, \boldsymbol{I})$ and use

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

Randomized direct-search techniques

Deterministic direct search

- Using fixed sets of directions gives limited exploration;
- Strong dependency on the dimension ($\mathcal{O}(n^2\epsilon^{-2})$ complexity).

Randomized techniques

- Randomized direct search: Use a random set of directions $\boldsymbol{D}_k$ such that

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

- Nesterov's random search (inspired by Gaussian smoothing): Draw $\boldsymbol{u}_k \sim \mathcal{N}(0, \boldsymbol{I})$ and use

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

- For both: Complexity improved to $\mathcal{O}(n\epsilon^{-2})$!

Stochastic derivative-free optimization

$$\min_{\boldsymbol{x} \in \mathcal{F}} f(\boldsymbol{x})$$

New assumptions

- f only available through a stochastic oracle $\tilde{f}(\boldsymbol{x}; \boldsymbol{\xi})$;
- The vector $\boldsymbol{\xi}$ is a random quantity lying in a compact set Ξ;
- Typical : $\tilde{f}(\cdot; \boldsymbol{\xi})$ convex in $\boldsymbol{x}$ for every realization of $\boldsymbol{\xi}$.
- Minimum of f attained at $\boldsymbol{x}_*$.

Stochastic derivative-free optimization

$$\min_{\boldsymbol{x} \in \mathcal{F}} f(\boldsymbol{x})$$

New assumptions

- f only available through a stochastic oracle $\tilde{f}(\boldsymbol{x}; \boldsymbol{\xi})$;
- The vector $\boldsymbol{\xi}$ is a random quantity lying in a compact set Ξ;
- Typical : $\tilde{f}(\cdot; \boldsymbol{\xi})$ convex in $\boldsymbol{x}$ for every realization of $\boldsymbol{\xi}$.
- Minimum of f attained at $\boldsymbol{x}_*$.

- Stochastic oracle $\leftrightarrow$ Bandit feedback;
- Interesting connections with bandit/online optimization literature.

Multi-armed bandit classical setting

- Discrete set of arms $\{1, \ldots, A\}$;
- At every iteration k, a player plays an arm $\boldsymbol{x}_k$, nature draws $\boldsymbol{\xi}_k$, yielding a reward $f(\boldsymbol{x}_k; \xi_k)$;
- **Expected cumulative regret:**

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

Infinite-armed bandits (Auer, 2002)

1. Player plays $\boldsymbol{x}_k$, nature draws $\boldsymbol{\xi}_k$ from Ξ compact set;
2. Player observes $f(\boldsymbol{x}_k; \boldsymbol{\xi}_k)$.

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

Bandit feedback methods

One-point methods

- Draw $\boldsymbol{u}_k \sim \mathcal{U}(\mathbb{S}^{n-1})$ and use

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

- Best known complexity: $\mathcal{O}(n\epsilon^{-3})$ for convex problems.

Bandit feedback methods

One-point methods

- Draw $\boldsymbol{u}_k \sim \mathcal{U}(\mathbb{S}^{n-1})$ and use

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

- Best known complexity: $\mathcal{O}(n\epsilon^{-3})$ for convex problems.

Two/Multi-point methods

- Assumption: Can use the same $\boldsymbol{\xi}$ to perform several evaluations.
- Draw $\boldsymbol{u}_k \sim \mathcal{U}(\mathbb{S}^{n-1})$ and use

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

- Best known-complexity: $\mathcal{O}(n\epsilon^{-2})$ for convex problems.

Direct search: Summarizing

My point of view

- Direct-search methods sample but do not seek to construct a gradient;
- Decreasing the objective is the main goal;
- Classical methods have evolved closer to random search via randomization.

Interesting connections

- Geometry: Positive spanning sets;
- Bandit/Online optimization: For stochastic settings in particular.

Software suggestions

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

- Developed at Polytechnique Montréal (Canada) since 2009;
- C++/Matlab versions;
- Multiple features: categorical, constraints, etc;
- HyperNOMAD (2019): Extension applied to optimize architectures and hyperparameters of neural networks.

Bandit-based optimization methods

- Hyperband (Jamieson et al 2016), BOHB (Falkner et al 2018): combine bandit approaches with other tools from Bayesian optimization;
- In both cases, doing more than just sampling is helpful...

Outline

Outline

Beyond the direct-search approach

What we saw before

- Direct-search techniques do exploration...
- ...with a bit of **local exploitation**.
- Typically re-sample (especially for randomized methods), do not re-use information from the past.

Model-based DFO

- Uses **past** evaluations to construct a model of the objective function;
- Can re-use points and be significantly cheaper than finite differences.

Trust-region DFO algorithm

- Goal: minimize$_{\boldsymbol{x} \in \mathbb{R}^n} f(\boldsymbol{x})$;
- Evaluations of f available but expensive, f smooth.

Trust-region DFO algorithm

- Goal: minimize$_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x})$;
- Evaluations of f available but expensive, f smooth.

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

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

- Compute a model $\boldsymbol{s} \mapsto m_k(\boldsymbol{x}_k + \boldsymbol{s})$ of f at $\boldsymbol{x}_k$;
- Compute a step $\boldsymbol{s}_k \approx \text{argmin}_{\|\boldsymbol{s}\|\leq\delta_k} m_k(\boldsymbol{x}_k + \boldsymbol{s})$;

Trust-region DFO algorithm

- Goal: minimize$_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x})$;
- Evaluations of f available but expensive, f smooth.

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

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

- Compute a model $\boldsymbol{s} \mapsto m_k(\boldsymbol{x}_k + \boldsymbol{s})$ of f at $\boldsymbol{x}_k$;
- Compute a step $\boldsymbol{s}_k \approx \text{argmin}_{\|\boldsymbol{s}\|\leq\delta_k} m_k(\boldsymbol{x}_k + \boldsymbol{s})$;
- Evaluate $\rho_k = \frac{f(\boldsymbol{x}_k) - f(\boldsymbol{x}_k + \boldsymbol{s}_k)}{m_k(\boldsymbol{x}_k) - m_k(\boldsymbol{x}_k + \boldsymbol{s}_k)}$.

Trust-region DFO algorithm

- Goal: minimize$_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x})$;
- Evaluations of f available but expensive, f smooth.

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

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

- Compute a model $\boldsymbol{s} \mapsto m_k(\boldsymbol{x}_k + \boldsymbol{s})$ of f at $\boldsymbol{x}_k$;
- Compute a step $\boldsymbol{s}_k \approx \text{argmin}_{\|\boldsymbol{s}\|\leq\delta_k} m_k(\boldsymbol{x}_k + \boldsymbol{s})$;
- Evaluate $\rho_k = \frac{f(\boldsymbol{x}_k) - f(\boldsymbol{x}_k + \boldsymbol{s}_k)}{m_k(\boldsymbol{x}_k) - m_k(\boldsymbol{x}_k + \boldsymbol{s}_k)}$.
- If $\rho_k \geq \eta$, set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{s}_k$ and $\delta_{k+1} \geq \delta_k$.

Trust-region DFO algorithm

- Goal: minimize$_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x})$;
- Evaluations of f available but expensive, f smooth.

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

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

- Compute a model $\boldsymbol{s} \mapsto m_k(\boldsymbol{x}_k + \boldsymbol{s})$ of f at $\boldsymbol{x}_k$;
- Compute a step $\boldsymbol{s}_k \approx \operatorname{argmin}_{\|\boldsymbol{s}\|\leq\delta_k} m_k(\boldsymbol{x}_k + \boldsymbol{s})$;
- Evaluate $\rho_k = \frac{f(\boldsymbol{x}_k) - f(\boldsymbol{x}_k + \boldsymbol{s}_k)}{m_k(\boldsymbol{x}_k) - m_k(\boldsymbol{x}_k + \boldsymbol{s}_k)}$.
- If $\rho_k \geq \eta$, set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{s}_k$ and $\delta_{k+1} \geq \delta_k$.
- Otherwise, set $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$ and $\delta_{k+1} = \delta_k/2$.

Model quality

Goal: Approximate a smooth function f with a model m.

- Taylor-like error bounds on the approximation;
- Local: will hold over a ball ($\approx$ trust region).

Fully linear models (Conn, Scheinberg, Vicente '08)

The model m is a κ-fully linear model of f at $(\boldsymbol{x}, \delta)$ if for any $\boldsymbol{y} \in B(\boldsymbol{x}, \delta)$,

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

Building fully linear models

- $\mathcal{P}_n^d$: space of polynomial functions on $\mathbb{R}^n$ of degree $\leq d$, $\dim \mathcal{P}_n^d = q + 1$;
- $\Phi = \{\phi_0(\cdot), \ldots, \phi_q(\cdot)\}$: basis of $\mathcal{P}_n^d$;
- $\mathcal{Y} = \{\boldsymbol{y}^0, \ldots, \boldsymbol{y}^p\}$: interpolation set, $p + 1$ points in $\mathbb{R}^n$;

Building fully linear models

- $\mathcal{P}_n^d$: space of polynomial functions on $\mathbb{R}^n$ of degree $\leq d$, $\dim \mathcal{P}_n^d = q + 1$;
- $\Phi = \{\phi_0(\cdot), \ldots, \phi_q(\cdot)\}$: basis of $\mathcal{P}_n^d$;
- $\mathcal{Y} = \{\boldsymbol{y}^0, \ldots, \boldsymbol{y}^p\}$: interpolation set, $p + 1$ points in $\mathbb{R}^n$;
- **Goal:** model $m(\boldsymbol{x}) = \sum_{i=0}^{q} \alpha_i \phi_i(\boldsymbol{x})$ such that

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

- Reformulated as $M(\Phi, \mathcal{Y})\alpha \approx f(\mathcal{Y})$, with

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

Building fully linear models (2)

Polynomial regression models

Compute α^* solution of

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

and set $m(\boldsymbol{x}) = \sum_{i=0}^{q} \alpha_i^* \phi_i(\boldsymbol{x})$.

Main result

If $\mathcal{Y} \subset B(\boldsymbol{y}^0, \delta)$ is *poised*, then m is fully linear in $B(\boldsymbol{y}^0, \delta)$.

Ex)

- Linear interpolation/regression case $p = q = n$;
- $\mathcal{Y} = \{\boldsymbol{y}^0, \boldsymbol{y}^1, \ldots, \boldsymbol{y}^n\}$ vertices of a simplex in $\mathbb{R}^n$.

Complexity analysis

Key assumptions

For every iteration k,

- m_k is κ-fully linear with $\kappa > 0$;
- m_k is built using at most r new evaluations of f in $B(\boldsymbol{x}_k, \delta_k)$.

Complexity result

To achieve $\inf_{0 \leq \ell \leq k} \|\nabla f(\boldsymbol{x}_\ell)\| < \epsilon$, the method requires at most

- $\mathcal{O}(\kappa^2 \epsilon^{-2})$ iterations;
- $\mathcal{O}(r\, \kappa^2 \epsilon^{-2})$ function evaluations.

Ex) Linear interpolation/regression: $r = \mathcal{O}(n), \kappa = \mathcal{O}(\sqrt{n})$
$\Rightarrow$ Evaluation complexity in $\mathcal{O}(n^2 \epsilon^{-2})$.

Model quality

Goal: Approximate a smooth function f with a model m.

- Taylor-like error bounds on the approximation;
- Local: will hold over a ball ($\approx$ trust region).

Model quality

Goal: Approximate a smooth function f with a model m.

- Taylor-like error bounds on the approximation;
- Local: will hold over a ball ($\approx$ trust region).

Fully quadratic models (Conn, Scheinberg, Vicente '08)

The model m_k is a κ-fully quadratic model of f at $(\boldsymbol{x}_k, \delta_k)$ if for any $\boldsymbol{y} \in B(\boldsymbol{x}_k, \delta_k)$,

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

Building fully quadratic models in practice

Popular choices

Model m_k built using values of f at $\mathcal{Y}_k = \{\boldsymbol{x}_k, \boldsymbol{y}^1, \ldots, \boldsymbol{y}^r\}$:

- Interpolation/Regression polynomials;
- Radial basis functions (Gaussian kernels).

Main argument

Good geometry of $\mathcal{Y}_k \Rightarrow m_k$ fully quadratic in $B(\boldsymbol{x}_k, \delta_k)$.

Ex) Quadratic interpolation with $r = \mathcal{O}(n^2)$ samples

$$\mathcal{Y}_k = \left\{\boldsymbol{x}_k, \{\boldsymbol{x}_k \pm \delta_k \boldsymbol{e}_i\}_{i=1}^n, \{\boldsymbol{x}_k + \delta_k \tfrac{\boldsymbol{e}_i + \boldsymbol{e}_j}{2}\}_{1 \le i < j \le n}\right\}.$$

Building fully quadratic models in practice

Popular choices

Model m_k built using values of f at $\mathcal{Y}_k = \{\boldsymbol{x}_k, \boldsymbol{y}^1, \ldots, \boldsymbol{y}^r\}$:

- Interpolation/Regression polynomials;
- Radial basis functions (Gaussian kernels).

Main argument

Good geometry of $\mathcal{Y}_k \Rightarrow m_k$ fully quadratic in $B(\boldsymbol{x}_k, \delta_k)$.

Ex) Quadratic interpolation with $r = \mathcal{O}(n^2)$ samples

$$\mathcal{Y}_k = \left\{\boldsymbol{x}_k, \{\boldsymbol{x}_k \pm \delta_k \boldsymbol{e}_i\}_{i=1}^n, \{\boldsymbol{x}_k + \delta_k \tfrac{\boldsymbol{e}_i + \boldsymbol{e}_j}{2}\}_{1 \le i < j \le n}\right\}.$$

In practice

- Reuse previous points;
- Control geometry with criticality step;
- May still require r new samples per iteration.

Key assumptions

For every iteration k,

- m_k is κ-fully quadratic with $\kappa > 0$;
- m_k is built using at most r new evaluations of f in $B(\boldsymbol{x}_k, \delta_k)$.

Key results

- $\delta_k \to 0$;
- If $\boldsymbol{x}_k$ not stationary and δ_k small enough, the step is accepted.

Complexity analysis

Assumption: Steps chosen to yield best iteration complexity ($\mathcal{O}(\epsilon_g^{-3/2})$).

Complexity result

To achieve $\|\nabla f(\boldsymbol{x}_k)\| \leq \epsilon_g$ and $\nabla^2 f(\boldsymbol{x}_k) \succeq -\epsilon_g^{1/2} I$, the method requires at most

- $\mathcal{O}(\kappa^3 \epsilon_g^{-3/2})$ iterations;
- $\mathcal{O}(r\, \kappa^3 \epsilon_g^{-3/2})$ function evaluations.

Complexity analysis

Assumption: Steps chosen to yield best iteration complexity ($\mathcal{O}(\epsilon_g^{-3/2})$).

Complexity result

To achieve $\|\nabla f(\boldsymbol{x}_k)\| \leq \epsilon_g$ and $\nabla^2 f(\boldsymbol{x}_k) \succeq -\epsilon_g^{1/2} I$, the method requires at most

- $\mathcal{O}(\kappa^3 \epsilon_g^{-3/2})$ iterations;
- $\mathcal{O}(r\, \kappa^3 \epsilon_g^{-3/2})$ function evaluations.

Example: Quadratic interpolation/regression

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

Note: Exploiting structure

Behind the black box

- For simplicity, we focused on expensive objectives;
- In many cases, part of the objective actually has gradients;
- In general, always better to exploit any known structure of the model!

Example: Least squares

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

- Derivatives of r unknown but derivatives of f partially known:

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

- Can help build more accurate models by leveraging the problem structure.

Outline

Challenges with the deterministic approach

Using fully linear models

- In practice, reuse function values/using less than $n+1$ works well;
- In theory, need $\mathcal{O}(n)$ function evaluations to certify fully linearity (Scheinberg and Toint '09).

Idea (Bandeira et al, '14)

Suppose that models are only fully linear with some probability.

- Generates random processes;
- Analyzed with martingale-type arguments.

A probabilistic property

Recall: The model m_k is κ-fully linear at $(\boldsymbol{x}_k, \delta_k)$ if for all $\boldsymbol{y} \in \mathcal{B}(\boldsymbol{x}_k, \delta_k)$:

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

A probabilistic property

Recall: The model m_k is κ-fully linear at $(\boldsymbol{x}_k, \delta_k)$ if for all $\boldsymbol{y} \in \mathcal{B}(\boldsymbol{x}_k, \delta_k)$:

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

Probabilistic models

A **random model** sequence $\{m_k\}$ is said to be (p, κ)-fully linear if:

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

Probabilistic complexity

Key assumptions

For every iteration k,

- $\{m_k\}_k$ is (p, κ)-fully linear with $p > 1/2$;
- m_k is built using at most r new evaluations of f in $B(x_k, \delta_k)$.

Theorem (Gratton, R., Vicente, Zhang '18)

Let $\epsilon \in (0, 1)$ and N_ϵ the number of evaluations needed to have $\|\nabla f(x_k)\| \leq \epsilon$. Then, $N_\epsilon = \mathcal{O}(r\kappa^2\epsilon^{-2})$ with high probability.

Subsampling case

$$f(\boldsymbol{x}) = \frac{1}{m}\sum_{i=1}^{m} f_i(\boldsymbol{x}), \nabla f_i \text{ available but not } \nabla f$$

- Generate $S \subset \{1, \ldots, N\}$ at random;
- Form $m(\boldsymbol{x} + \boldsymbol{x}) = f(\boldsymbol{x}) + \frac{1}{|S|}\sum_{i \in S} \nabla f_i(\boldsymbol{x})^\top \boldsymbol{s}$;
- With $|S| = \mathcal{O}(\delta^{-2})$, the model is probabilistically fully linear on $B(\boldsymbol{x}, \delta)$.

Probabilistic models in practice

Subsampling case

$$f(\boldsymbol{x}) = \frac{1}{m}\sum_{i=1}^{m} f_i(\boldsymbol{x}), \nabla f_i \text{ available but not } \nabla f$$

- Generate $S \subset \{1, \ldots, N\}$ at random;
- Form $m(\boldsymbol{x} + \boldsymbol{x}) = f(\boldsymbol{x}) + \frac{1}{|S|}\sum_{i \in S} \nabla f_i(\boldsymbol{x})^\top \boldsymbol{s}$;
- With $|S| = \mathcal{O}(\delta^{-2})$, the model is probabilistically fully linear on $B(\boldsymbol{x}, \delta)$.

Probabilistic fully quadratic models

- Deterministically, requires $\mathcal{O}(n^2)$ evaluations;
- If objective Hessian is sparse, achieved (w. h. p.) in $\mathcal{O}\left(n(\log n)^4\right)$ evaluations using techniques from compressed sensing.

Bayesian optimization and models

Bayesian optimization paradigm

1. At every iteration, fit a posterior distribution to the existing observations;
2. Find next point by maximizing an acquisition function;
3. Repeat until the evaluation budget is exhausted.

Bayesian optimization paradigm

1. At every iteration, fit a posterior distribution to the existing observations;
2. Find next point by maximizing an acquisition function;
3. Repeat until the evaluation budget is exhausted.

- Popular among statisticians, great for uncertainty quantification;
- Does not scale up well to high-dimension, must be able to maximize the acquisition function;
- In spirit, a model-based paradigm based on radial basis functions:

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

$\Rightarrow$ Those are fully linear models!

Model based: Summarizing

Building models

- Exploits the history of the algorithm;
- Efficient implementations much cheaper than finite differences;
- The best variants exploit as much structure as they can.

Are those popular?

- Metamodels/Surrogates are ubiquitous in scientific computing;
- Bayesian optimization builds models too!

Model-based software

Michael Powell's codes

- Originally written in Fortran, still some of the most robust codes to date;
- PDFO (https://www.pdfo.net/) provides Python and MATLAB interfaces;
- Use explored in adversarial training.

Other software

- POUNDERS: Derivative-free least-squares (structure);
- ORBIT: Radial basis function trust region;
- Numerous packages in Python/R for surrogate/Bayesian optimization.

A quick summary

Derivative-free optimization

- When derivatives not available;
- Many names these days;
- Focus: Use as few evaluations as possible;
- These problems are important (and cool)!

DFO for ML and ML for DFO

- Randomized DFO techniques use ML-like techniques;
- Automated ML efficient with principled approaches (like DFO).

Things I left out

Discrete variables

- Most implementations handle them in some fashion, even categorical;
- Theory is spread over communities.

Genetic/Evolutionary algorithms

- Popular in practice, can be quite efficient in small dimension;
- But require many evaluations (not quite our setup);

Notable method: CMA-ES (ask Eric Benhamou!).

Black-box constraints

- Taxonomy of constraints in DFO (ex: relaxable);
- Many paradigms from optimization theory implemented. One classical approach is to use an extreme barrier $f(\boldsymbol{x}) = \infty$ if $\boldsymbol{x} \notin \mathcal{F}$.

References

- 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.

More references

- 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.

That's all!

Thank you for your attention!

clement.royer@dauphine.psl.eu

Image credits

- https://towardsdatascience.com/
- https://commons.wikimedia.org/
- M. Feurer and F. Hutter, *Hyperparameter Optimization*, in *Automated Machine Learning*, Springer, 2019.