Machine learning for optimization (1/5)

Clément W. Royer

M2 MODO - 2024/2025

V2: January 20, 2025

Dauphine | PSL
UNIVERSITÉ PARIS

About this course

Resources

- Course webpage:
  `https://www.lamsade.dauphine.fr/~croyer/teachMLO.html`
- URL for the lab session (in Python):
  `https://tinyurl.com/yye8jje3`

Logistics

- Courses 1/3/5: In lab room, in Python, Mondays Jan. 13/20/27 8.30am-11.45am.
- Course 2: Regular classroom, Wednesday Jan. 15 1.45-5pm.
- Course 4: Regular classroom, Friday Jan. 24 1.45-5pm.
- Exam: Friday Feb 7, 10am-12pm (details TBA).

About this course ('ed)

Overall theme: **Applying learning techniques to improve optimization algorithms.**

About this course ('ed)

Overall theme: **Applying learning techniques to improve optimization algorithms.**

- Ongoing research topic⇒No definite answers!
- Lack of maturity⇒Hard to teach!
- Need optimization and ML concepts⇒A lot for 15h!

About this course ('ed)

Overall theme: **Applying learning techniques to improve optimization algorithms.**

- Ongoing research topic ⇒ No definite answers!
- Lack of maturity ⇒ Hard to teach!
- Need optimization and ML concepts ⇒ A lot for 15h!

My approach

- Build around use cases.
- Review the concepts around those use cases.
- Try things out (lab sessions)!

Tentative outline

- **Course 1** Simple optimization schemes (blackbox) and learning models (no neural networks).
  $\Rightarrow$ **Argue that ML techniques make sense in my field of research**

Tentative outline

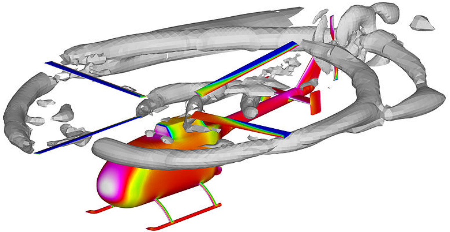
- **Course 1** Simple optimization schemes (blackbox) and learning models (no neural networks).
  $\Rightarrow$ **Argue that ML techniques make sense in my field of research**
- **Course 2** Implicit layers and optimization solvers.
- **Course 3** Implicit layers and optimization solvers (lab).
  $\Rightarrow$ **Solving a problem and learning at once**.

Tentative outline

- **Course 1** Simple optimization schemes (blackbox) and learning models (no neural networks).
  $\Rightarrow$ **Argue that ML techniques make sense in my field of research**
- **Course 2** Implicit layers and optimization solvers.
- **Course 3** Implicit layers and optimization solvers (lab).
  $\Rightarrow$ **Solving a problem and learning at once**.
- **Course 4** Learning with multiple problem instances.
- **Course 5** Learning with multiple problem instances (lab).
  $\Rightarrow$ **Leverage previous solves**.

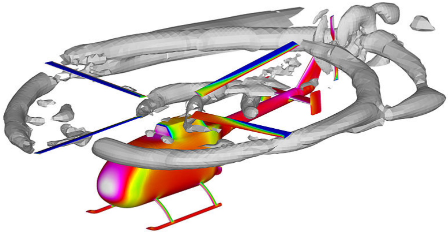
Roadmap

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.

Setup

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

Assumptions

- f bounded below;
- No differentiability/convexity requirement!

Blackbox/Derivative-free optimization

- **Derivatives unavailable for algorithmic use.**
- Only access to (possibly noisy) values of f.
- Function values often expensive to get.

Direct search

Direct-search paradigm

- Explore the space at every iteration through certain directions.
- Stops *polling* as soon as a better point is found.
- Use an adaptive stepsize to explore the space.

Coordinate search

- Use coordinate directions and their negatives.
- Classical:
  - Double the stepsize if a better point is found.
  - Halve the stepsize otherwise.

Basic coordinate search framework

Inputs: $\boldsymbol{x}_0 \in \mathbb{R}^n$, $0 < \alpha_0 \leq \alpha_{\max}$, $\mathcal{D} = [\boldsymbol{I}_n \ -\boldsymbol{I}_n]$.
Iteration k**:** Given $(\boldsymbol{x}_k, \alpha_k)$,

- If $\exists\ \boldsymbol{d}_k \in \mathcal{D}$ such that

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

  set $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$, $\alpha_{k+1} := \min\left\{2\alpha_k, \alpha_{\max}\right\}$.

Basic coordinate search framework

Inputs: $\boldsymbol{x}_0 \in \mathbb{R}^n$, $0 < \alpha_0 \leq \alpha_{\max}$, $\mathcal{D} = [\boldsymbol{I}_n \ -\boldsymbol{I}_n]$.

Iteration k: Given $(\boldsymbol{x}_k, \alpha_k)$,

- If $\exists \ \boldsymbol{d}_k \in \mathcal{D}$ such that

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

  set $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$, $\alpha_{k+1} := \min\{2\alpha_k, \alpha_{\max}\}$.
- Otherwise, set $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k$, $\alpha_{k+1} := \alpha_k/2$.

Basic coordinate search framework

Inputs: $\boldsymbol{x}_0 \in \mathbb{R}^n$, $0 < \alpha_0 \leq \alpha_{\max}$, $\mathcal{D} = [\boldsymbol{I}_n \ -\boldsymbol{I}_n]$.

Iteration k**:** Given $(\boldsymbol{x}_k, \alpha_k)$,

- If $\exists\ \boldsymbol{d}_k \in \mathcal{D}$ such that

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

  set $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$, $\alpha_{k+1} := \min\{2\alpha_k, \alpha_{\max}\}$.
- Otherwise, set $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k$, $\alpha_{k+1} := \alpha_k/2$.

Basic coordinate search framework

Inputs: $\boldsymbol{x}_0 \in \mathbb{R}^n$, $0 < \alpha_0 \leq \alpha_{\max}$, $\mathcal{D} = [\boldsymbol{I}_n \ -\boldsymbol{I}_n]$.

Iteration k: Given $(\boldsymbol{x}_k, \alpha_k)$,

- If $\exists\ \boldsymbol{d}_k \in \mathcal{D}$ such that

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

  set $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$, $\alpha_{k+1} := \min\{2\alpha_k, \alpha_{\max}\}$.
- Otherwise, set $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k$, $\alpha_{k+1} := \alpha_k/2$.

Basic coordinate search framework

Inputs: $\boldsymbol{x}_0 \in \mathbb{R}^n$, $0 < \alpha_0 \leq \alpha_{\max}$, $\mathcal{D} = [\boldsymbol{I}_n \ -\boldsymbol{I}_n]$.
Iteration k**:** Given $(\boldsymbol{x}_k, \alpha_k)$,

- If $\exists\ \boldsymbol{d}_k \in \mathcal{D}$ such that

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

  set $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$, $\alpha_{k+1} := \min\left\{2\alpha_k, \alpha_{\max}\right\}$.
- Otherwise, set $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k$, $\alpha_{k+1} := \alpha_k/2$.

- Can force f to decrease by a sufficient amount (e.g. α_k^2).
- Can change update rules on α_k.

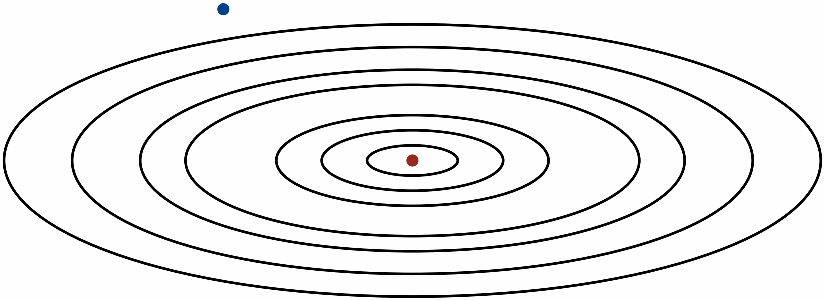


Illustration: Coordinate search

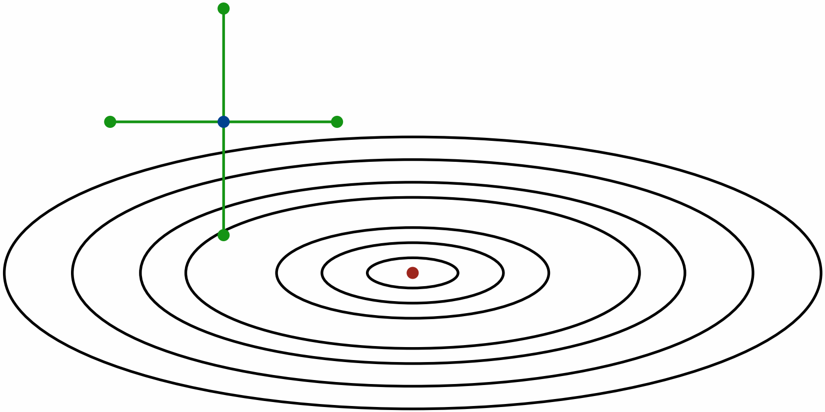


Illustration: Coordinate search

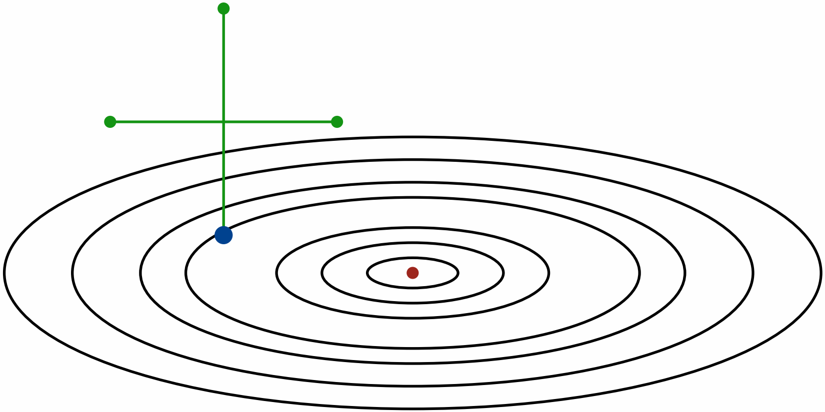


Illustration: Coordinate search

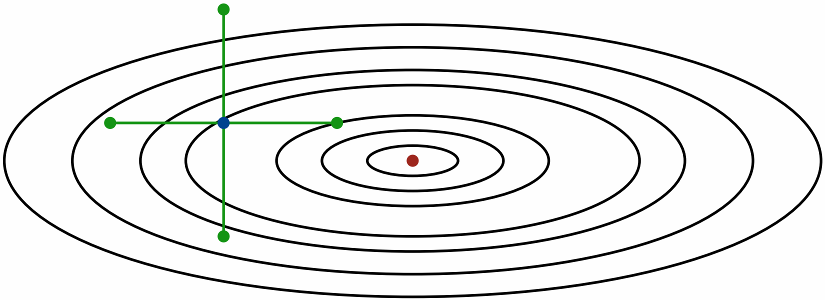


Illustration: Coordinate search

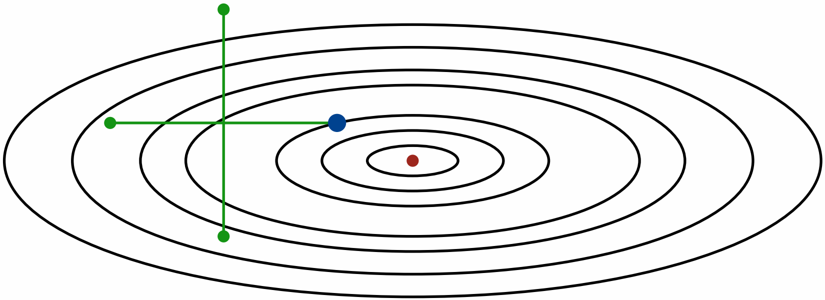


Illustration: Coordinate search

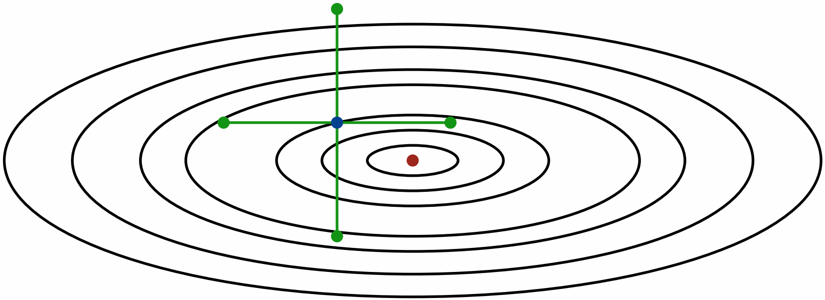


Illustration: Coordinate search

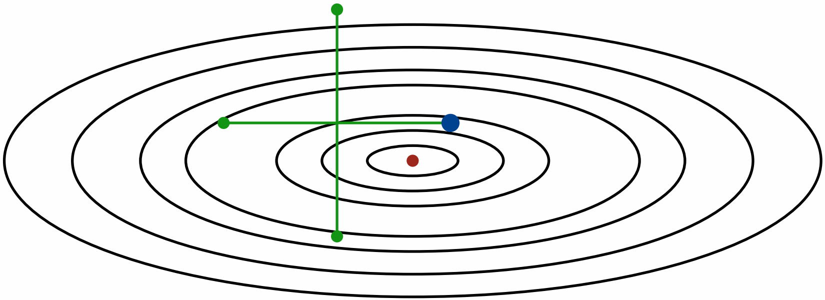


Illustration: Coordinate search

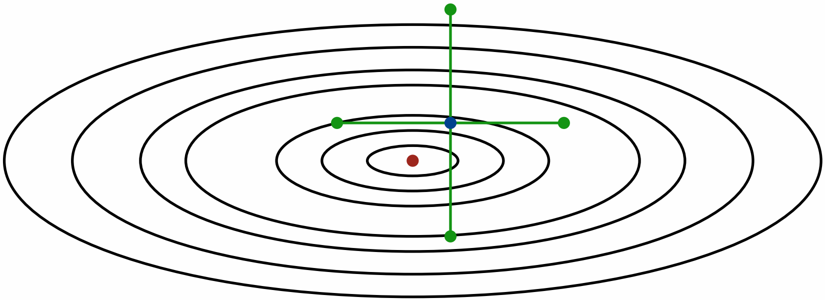


Illustration: Coordinate search

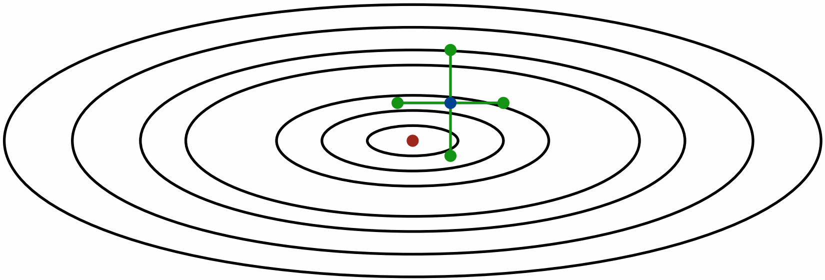
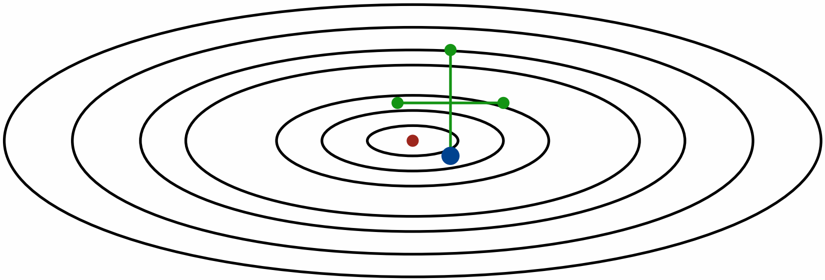


Illustration: Coordinate search



Lab time!

- Open the notebook (with Colab or Jupyter), and check that it works!
- Run coordinate search on the first blackbox function.

Roadmap

Beyond the direct-search approach

What we saw before

- Direct-search techniques do exploration...
- ...with a bit of **local exploitation**.
- Do not re-use information from the past.

Beyond the direct-search approach

What we saw before

- Direct-search techniques do exploration...
- ...with a bit of **local exploitation**.
- Do not re-use information from the past.

Surrogate modeling

- Uses **past** evaluations to construct a model of the objective function;
- Update the model at every iteration.
- **Typical choice:** Polynomial models.

Building polynomial models (general)

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

Building polynomial models (general)

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

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

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

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

Link with ML approaches

Building the model: A linear regression problem!

Find $\boldsymbol{w}^*$ solution of

$$\operatorname*{minimize}_{\boldsymbol{w} \in \mathbb{R}^{q+1}} \|M(\Phi, \mathcal{Y})\boldsymbol{w} - f(\mathcal{Y})\|^2.$$

and take $m(\boldsymbol{y}) = \sum_{i=0}^{q} w_i^* \phi_i(\boldsymbol{y})$.

Building the model: A linear regression problem!

Find $\boldsymbol{w}^*$ solution of

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

and take $m(\boldsymbol{y}) = \sum_{i=0}^{q} w_i^* \phi_i(\boldsymbol{y})$.

Linear regression

Given $\boldsymbol{A} \in \mathbb{R}^{m\times d}$ and $\boldsymbol{b} \in \mathbb{R}^m$, find

$$\boldsymbol{w}^* \in \underset{\boldsymbol{w}\in\mathbb{R}^d}{\text{argmin}} \frac{1}{2m}\|\boldsymbol{A}\boldsymbol{w} - \boldsymbol{b}\|^2.$$

- Ordinary least-squares/Maximum likelihood estimator.
- Standard routines available to solve this linear least-squares problem!

Simplest case: Linear models

Goal: Given $\mathcal{Y} = \{\boldsymbol{y}^0, \dots, \boldsymbol{y}^p\}$, build a model $m(\boldsymbol{y}) = v + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{y}$.

$$\underset{\substack{\boldsymbol{w}\in\mathbb{R}^n \\ v\in\mathbb{R}}}{\text{minimize}} \frac{1}{2(p+1)} \left\| M(\mathcal{Y}) \begin{bmatrix} v \\ \boldsymbol{w} \end{bmatrix} - f(\mathcal{Y}) \right\|^2.$$

- Basis functions: $\{1, [\boldsymbol{y}]_1, \dots, [\boldsymbol{y}]_n\}$.

$$M(\mathcal{Y}) = \begin{bmatrix} 1 & [\boldsymbol{y}^0]_1 & \cdots & [\boldsymbol{y}^0]_n \\ 1 & [\boldsymbol{y}^1]_1 & \cdots & [\boldsymbol{y}^1]_n \\ \vdots & \vdots & \vdots & \vdots \\ 1 & [\boldsymbol{y}^p]_1 & \cdots & [\boldsymbol{y}^p]_n \end{bmatrix}$$

- Function values: $f(\mathcal{Y}) = \left[f(\boldsymbol{y}^i)\right]_{i=0,\dots,p}$.

Using models in coordinate search

Inputs: $\boldsymbol{x}_0 \in \mathbb{R}^n$, $0 < \alpha_0 \leq \alpha_{\max}$, $\mathcal{D} = [\boldsymbol{I}_n \ -\boldsymbol{I}_n]$.

Iteration k**:** Given $(\boldsymbol{x}_k, \alpha_k)$,

- Build a linear model $m_k(\boldsymbol{s}) = v_k + \boldsymbol{w}_k^{\mathrm{T}} \boldsymbol{s}$ and set $\boldsymbol{s}_k = -\alpha_k \frac{\boldsymbol{w}_k}{\|\boldsymbol{w}_k\|}$.

Using models in coordinate search

Inputs: $\boldsymbol{x}_0 \in \mathbb{R}^n$, $0 < \alpha_0 \leq \alpha_{\max}$, $\mathcal{D} = [\boldsymbol{I}_n \ -\boldsymbol{I}_n]$.

Iteration k**:** Given $(\boldsymbol{x}_k, \alpha_k)$,

- Build a linear model $m_k(\boldsymbol{s}) = v_k + \boldsymbol{w}_k^{\mathrm{T}} \boldsymbol{s}$ and set $\boldsymbol{s}_k = -\alpha_k \frac{\boldsymbol{w}_k}{\|\boldsymbol{w}_k\|}$.
- If $f(\boldsymbol{x}_k + \boldsymbol{s}_k) < f(\boldsymbol{x}_k)$,
  set $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + \boldsymbol{s}_k$, $\alpha_{k+1} := \min\{2\alpha_k, \alpha_{\max}\}$.

Using models in coordinate search

Inputs: $\boldsymbol{x}_0 \in \mathbb{R}^n$, $0 < \alpha_0 \leq \alpha_{\max}$, $\mathcal{D} = [\boldsymbol{I}_n \ -\boldsymbol{I}_n]$.
Iteration k: Given $(\boldsymbol{x}_k, \alpha_k)$,

- Build a linear model $m_k(\boldsymbol{s}) = v_k + \boldsymbol{w}_k^{\mathrm{T}} \boldsymbol{s}$ and set $\boldsymbol{s}_k = -\alpha_k \frac{\boldsymbol{w}_k}{\|\boldsymbol{w}_k\|}$.
- If $f(\boldsymbol{x}_k + \boldsymbol{s}_k) < f(\boldsymbol{x}_k)$,
  set $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + \boldsymbol{s}_k$, $\alpha_{k+1} := \min\{2\alpha_k, \alpha_{\max}\}$.
- Else if $\exists\ \boldsymbol{d}_k \in \mathcal{D}$ such that

$$f(\boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k) < f(\boldsymbol{x}_k)$$

  set $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$, $\alpha_{k+1} := \min\{2\alpha_k, \alpha_{\max}\}$.

Using models in coordinate search

Inputs: $\boldsymbol{x}_0 \in \mathbb{R}^n$, $0 < \alpha_0 \leq \alpha_{\max}$, $\mathcal{D} = [\boldsymbol{I}_n \ -\boldsymbol{I}_n]$.

Iteration k: Given $(\boldsymbol{x}_k, \alpha_k)$,

- Build a linear model $m_k(\boldsymbol{s}) = v_k + \boldsymbol{w}_k^{\mathrm{T}}\boldsymbol{s}$ and set $\boldsymbol{s}_k = -\alpha_k \frac{\boldsymbol{w}_k}{\|\boldsymbol{w}_k\|}$.
- If $f(\boldsymbol{x}_k + \boldsymbol{s}_k) < f(\boldsymbol{x}_k)$,
  set $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + \boldsymbol{s}_k$, $\alpha_{k+1} := \min\{2\alpha_k, \alpha_{\max}\}$.
- Else if $\exists\ \boldsymbol{d}_k \in \mathcal{D}$ such that

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

  set $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$, $\alpha_{k+1} := \min\{2\alpha_k, \alpha_{\max}\}$.
- Otherwise, set $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k$, $\alpha_{k+1} := \alpha_k/2$.

Subtleties

- Linear model: $m_k(\boldsymbol{s}) = v_k + \boldsymbol{w}_k^{\mathrm{T}} \boldsymbol{s}$, computed by solving the linear regression problem

$$\operatorname*{minimize}_{\substack{\boldsymbol{w} \in \mathbb{R}^n \\ v \in \mathbb{R}}} \frac{1}{2(p+1)} \left\| M(\mathcal{Y}_k) \begin{bmatrix} v \\ \boldsymbol{w} \end{bmatrix} - f(\mathcal{Y}_k) \right\|^2.$$

with

$$M(\mathcal{Y}_k) = \begin{bmatrix} 1 & [\boldsymbol{y}^0]_1 & \cdots & [\boldsymbol{y}^0]_n \\ 1 & [\boldsymbol{y}^1]_1 & \cdots & [\boldsymbol{y}^1]_n \\ \vdots & \vdots & \vdots & \vdots \\ 1 & [\boldsymbol{y}^p]_1 & \cdots & [\boldsymbol{y}^p]_n \end{bmatrix}, f(\mathcal{Y}_k) = \begin{bmatrix} f(\boldsymbol{y}^0) \\ \vdots \\ f(\boldsymbol{y}^p) \end{bmatrix}.$$

Subtleties

- Linear model: $m_k(\boldsymbol{s}) = v_k + \boldsymbol{w}_k^{\mathrm{T}}\boldsymbol{s}$, computed by solving the linear regression problem

$$\operatorname*{minimize}_{\substack{\boldsymbol{w}\in\mathbb{R}^n \\ v\in\mathbb{R}}} \frac{1}{2(p+1)} \left\| M(\mathcal{Y}_k) \begin{bmatrix} v \\ \boldsymbol{w} \end{bmatrix} - f(\mathcal{Y}_k) \right\|^2 .$$

with

$$M(\mathcal{Y}_k) = \begin{bmatrix} 1 & [\boldsymbol{y}^0]_1 & \cdots & [\boldsymbol{y}^0]_n \\ 1 & [\boldsymbol{y}^1]_1 & \cdots & [\boldsymbol{y}^1]_n \\ \vdots & \vdots & \vdots & \vdots \\ 1 & [\boldsymbol{y}^p]_1 & \cdots & [\boldsymbol{y}^p]_n \end{bmatrix}, f(\mathcal{Y}_k) = \begin{bmatrix} f(\boldsymbol{y}^0) \\ \vdots \\ f(\boldsymbol{y}^p) \end{bmatrix}.$$

- Build $\mathcal{Y}_k$ by keeping only points within $B(\boldsymbol{x}_k, \alpha_k)$.

- Linear model: $m_k(\boldsymbol{s}) = v_k + \boldsymbol{w}_k^{\mathrm{T}}\boldsymbol{s}$, computed by solving the linear regression problem

$$\operatorname*{minimize}_{\substack{\boldsymbol{w}\in\mathbb{R}^n \\ v\in\mathbb{R}}} \frac{1}{2(p+1)} \left\| M(\mathcal{Y}_k) \begin{bmatrix} v \\ \boldsymbol{w} \end{bmatrix} - f(\mathcal{Y}_k) \right\|^2.$$

with

$$M(\mathcal{Y}_k) = \begin{bmatrix} 1 & [\boldsymbol{y}^0]_1 & \cdots & [\boldsymbol{y}^0]_n \\ 1 & [\boldsymbol{y}^1]_1 & \cdots & [\boldsymbol{y}^1]_n \\ \vdots & \vdots & \vdots & \vdots \\ 1 & [\boldsymbol{y}^p]_1 & \cdots & [\boldsymbol{y}^p]_n \end{bmatrix}, f(\mathcal{Y}_k) = \begin{bmatrix} f(\boldsymbol{y}^0) \\ \vdots \\ f(\boldsymbol{y}^p) \end{bmatrix}.$$

- Build $\mathcal{Y}_k$ by keeping only points within $B(\boldsymbol{x}_k, \alpha_k)$.
- Step: $\boldsymbol{s}_k = -\alpha_k \frac{\boldsymbol{w}_k}{\|\boldsymbol{w}_k\|} = \operatorname{argmin}_{\|\boldsymbol{s}\|\le\alpha_k} m_k(\boldsymbol{s})$.

Lab time!

- Augment coordinate search with linear regression models.
- Compare the performance on the toy example.

Roadmap

$$\underset{\boldsymbol{x} \in \mathbb{R}^n}{\text{minimize}} \, f(\boldsymbol{x}).$$

Hidden constraints

- Definition: $f(\boldsymbol{x}) = +\infty$ for some $\boldsymbol{x} \in \mathbb{R}^n$.
- Typical of objectives obtained using numerical codes that fail (CFD solve in helicopter problem).
- Recent attempts at learning those hidden constraints (El Amri et al '21).

Using multiple instances

Setup

- Several instances $f_1, \ldots, f_\ell$ with the same hidden constraint.
- Collection of pairs $(\boldsymbol{x}, f_i(\boldsymbol{x}))$ obtained from runs of an algorithm on $f_1, \ldots, f_\ell$ with a limited budget of function evaluations.

Using multiple instances

Setup

- Several instances $f_1, \dots, f_\ell$ with the same hidden constraint.
- Collection of pairs $(\boldsymbol{x}, f_i(\boldsymbol{x}))$ obtained from runs of an algorithm on $f_1, \dots, f_\ell$ with a limited budget of function evaluations.

Goal

Given a new instance $f_{\ell+1}$ with the same **hidden constraint**, can we use past information to predict satisfaction of the hidden constraint?

A tool: Logistic regression

Logistic regression

Given $\boldsymbol{A} \in \mathbb{R}^{m \times d}$ and $\boldsymbol{b} \in \{-1,+1\}^m$, find $\boldsymbol{w} \in \mathbb{R}^d$ such that (the sign of) $\boldsymbol{a}_i^\mathrm{T} \boldsymbol{w}$ predicts b_i.

$$\operatorname*{minimize}_{\boldsymbol{w} \in \mathbb{R}^d} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 + \exp\left(-b_i \boldsymbol{a}_i^\mathrm{T} \boldsymbol{w}\right)\right).$$

- Can be solved using nonlinear optimization techniques (L-BFGS).
- Idea: Build $\boldsymbol{A}$ and $\boldsymbol{b}$ using evaluations from instances $f_1, \dots, f_\ell$.

Lab time!

- Run coordinate search on several instances with the same hidden constraint.
- Build a classifier to learn which points to avoid.
- Test the method augmented with the classifier.

Summary: Course 1

Blackbox optimization

- Challenging optimization paradigm.
- Data: Function values (or lack thereof).

Two examples of using ML

- Learning surrogate models ⇒ Regression!
- Learning constraints from past instances ⇒ Classification!

References

- C. Audet, W. Hare, *Derivative-Free and Blackbox Optimization*, Springer, 2017.
- M. R. El Amri, C. Helbert, M. M. Zuniga, C. Prieur, D. Sinoquet, *Feasible set estimation under functional uncertainty by Gaussian Process modelling*, Physica D, 2023.
- J. Larson, M. Menickelly and S. M. Wild, *Derivative-free optimization methods*, Acta Numerica, 2019.