MsC in Data Sciences & Business Analytics

Optimization Course

Gif-sur-Yvette
December 13, 2017

Charles SOUSSEN
charles.soussen@centralesupelec.fr

Summary

1. Introduction.
2. Unconstrained optimization.
   - Derivative-free algorithms.
   - Gradient-based algorithms.
   - Least squares algorithms.
3. Least-square problems.
4. Constrained optimization.
5. Global / discrete optimization.

Organization

8 courses:

- Course 1: lecture.
- Courses 2 to 8: 1h30 lecture and 1h30 practical session (matlab).
- Dec. 14, 2017: exam.

1. Introduction

General formulation (unconstrained)

Minimize $f(\boldsymbol{x})$.

Terminology:

- $\boldsymbol{x} \in \mathbb{R}^n$: variables.
- $f : \mathbb{R}^n \mapsto \mathbb{R}$: objective function (criterion).

Resolution:

- Existence and unicity of the solution?
- Which numerical algorithm?
- Algorithm complexity: computation time, memory storage.

General formulation (constrained)

Minimize $f(x)$ **subject to** $\begin{cases} g_i(x) \leq 0, & i = 1, \ldots, p \\ h_i(x) = 0, & i = 1, \ldots, q. \end{cases}$

Terminology:

- $x \in \mathbb{R}^n$: variables.
- $f : \mathbb{R}^n \mapsto \mathbb{R}$: objective function.
- $g_i : \mathbb{R} \mapsto \mathbb{R}$: inequality constraints.
- $h_i : \mathbb{R} \mapsto \mathbb{R}$: equality constraints.

Reformulation

Minimize $f(\boldsymbol{x})$ **subject to** $\boldsymbol{x} \in \mathcal{D}$.

Feasible set: $\mathcal{D} = \{\boldsymbol{x} \in \mathbb{R}^n, \, \boldsymbol{g}(\boldsymbol{x}) \leq \mathbf{0}_p, \, \boldsymbol{h}(\boldsymbol{x}) = \mathbf{0}_q\}$

- $\boldsymbol{g}(\boldsymbol{x}) = \begin{bmatrix} g_1(\boldsymbol{x}) \\ \vdots \\ g_p(\boldsymbol{x}) \end{bmatrix}$ gathers the p inequality constraints.
- $\boldsymbol{h}(\boldsymbol{x}) = \begin{bmatrix} h_1(\boldsymbol{x}) \\ \vdots \\ h_q(\boldsymbol{x}) \end{bmatrix}$ gathers the q equality constraints.

Example 1: Knapsack problem

Jo goes hitch hiking. The maximum weight allowed in his knapsack is W. Each article $i = 1, \ldots, n$ he can take weight w_i and has a usefulness u_i. What articles should be taken in the knapsack to maximize the usefulness?

Example. $W = 25$, and:

i	w_i	u_i
1	25.00	40
2	12.50	35
3	11.25	18
4	5.00	4
5	2.50	10
6	1.25	2

Mathematical formulation

1. What is the search space?

   $\boldsymbol{x} \in \mathcal{D} = \{0, 1\}^n$

   $x_i = 1 \Leftrightarrow$ bring article # i in the knapsack

2. What is the objective function?

   $$f(\boldsymbol{x}) = \sum_{i=1}^{n} w_i x_i$$

3. What constraints?

   $$\sum_{i=1}^{n} w_i x_i \leq W$$

$\Rightarrow$ *Combinatorial* (discrete) optimization problem.

Example 2: Traveling salesman problem

Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?

- Search space? $\mathcal{D}$: set of all permutations of $\{1, \ldots, n\}$.

$\Rightarrow$ *Combinatorial* (discrete) optimization problem.

Example 3: Ressource allocation: example

A factory produces two products P1 and P2 and has resources R1 (equipment), R2 (human), R3 (raw materials) in limited quantities.

P1 is sold for 6 euros per unit, P2 is sold for 4 euros per unit.

Example.

	P1	P2	Availability
R1	3	9	81
R2	4	5	55
R3	2	1	20
Profit	6	4	

Questions. Mathematical formulation? Feasible domain?

Example 3: Ressource allocation

- Bandwith allocation in wireless communication.
- Strategic planning: allocating scarce resources (*e.g.,* human resources) among the various projects or business units.
- *etc.*

Linear programming problems.

Example 4: Portfolio optimization

Choosing the proportions of various assets to be held in a portfolio, in such a way as to make the portfolio better than any other according to some criterion.

The criterion will combine considerations of:

- the expected value of the portfolio's rate of return
- the return dispersion
- other measures of financial risk.

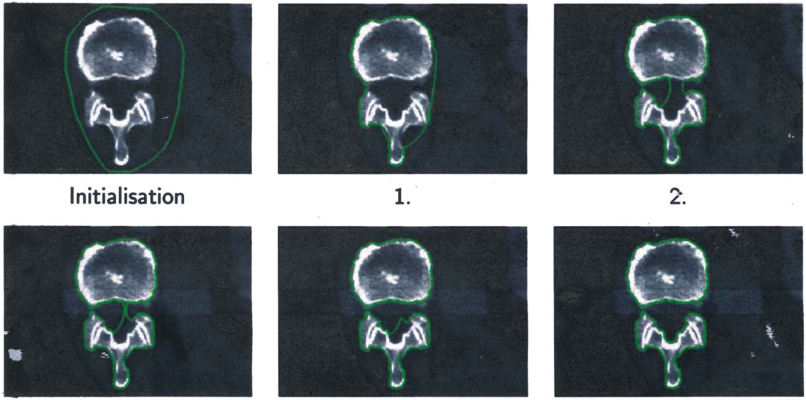
Example 5: Mechanical design of structures

Bridge topology optimization

- Minimize the total mass under rigidity constraints.
- Shape optimization: maximize the stability (oscillations due to car traffic, pedestrian or wind) of hanging bridges.

Example 6: Image segmentation using active contours

Contours actifs 2D *3/5*



Initialisation 1. 2.

3. 4. Résultat

Example 6: Image segmentation using active contours

A contour is defined as a parametric function $M = f(s)$.

- The contours lays in the contrast areas of the image.

  $\Rightarrow$ maximimize
  $$\int_s \|\nabla \boldsymbol{A}(f(s))\|_2^2 \, ds$$
  with respect to f.
- Find a smooth contour.

  $\Rightarrow$ constraint that the smoothness function R(f) is bounded.
- Formulation:

$$\min_f - \int_s \|\nabla \boldsymbol{A}(f(s))\|_2^2 \, ds \quad \text{subject to} \quad R(f) \leq A$$

Example 7: Image registration

$$\max_{T} S(I_t, T(I_s))$$

- T: geometrical transformation (translation, rotation, *etc.*).
- I_s: source image, I_t target image.
- S: measure of similarity.

Conclusion (part 1)

There is no generic numerical method to solve all kinds of problems!

The effectiveness of numerical optimization methods depends on:

- The differentiability of f.
- The possibility to compute efficiently its derivatives.
- The feasible set: discrete or continuous?
- The kinds of constraints (equalities / inequalities).
- The specific class of problem (linear programming, convex, *etc.*).
- The problem dimension.

Basic definitions

Local/global minimizers

$\boldsymbol{x}^\star \in \mathcal{D}$ is a global minimizer of f over $\mathcal{D}$ if $\forall \boldsymbol{x} \in \mathcal{D}$, $f(\boldsymbol{x}) \geq f(\boldsymbol{x}^\star)$.

$\boldsymbol{x} \in \mathcal{D}$ is a local minimizer of f over $\mathcal{D}$ if there exists a neighborhood $\mathcal{V}(\boldsymbol{x})$ of $\boldsymbol{x}$ such $\forall \boldsymbol{y} \in \mathcal{V}(\boldsymbol{x}) \cap \mathcal{D}$, $f(\boldsymbol{y}) \geq f(\boldsymbol{x})$.

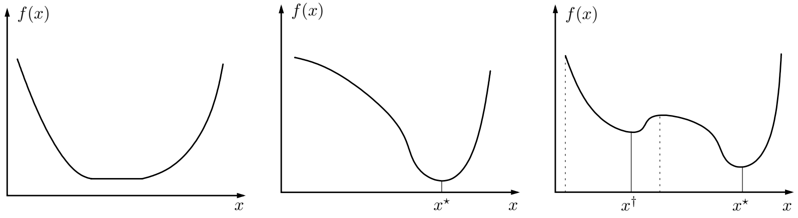
The minimum of f over $\mathcal{D}$ is the value $\min\limits_{\boldsymbol{x} \in \mathcal{D}} f(\boldsymbol{x})$.

Unimodality / multimodality

f is unimodal if f possesses a unique local minimizer.
Otherwise, f is multimodal.

Remark.

- Strict convexity $\Rightarrow$ unimodality.
- Convexity $\Rightarrow$ all local minimizers are global minimizers.



convex but multimodal — nonconvex but unimodal — nonconvex and multimodal

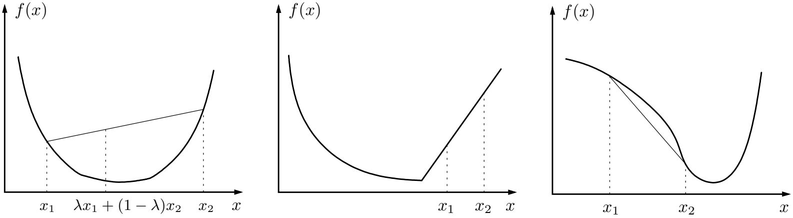
Convexity

f is convex if for any $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{D}$, and any $\lambda \in [0, 1]$,

$$f(\lambda \boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2) \;\leq\; \lambda f(\boldsymbol{x}_1) + (1-\lambda) f(\boldsymbol{x}_2)$$

f is strictly convex if for any $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{D}$, and any $\lambda \in]0, 1[$,

$$f(\lambda \boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2) < \lambda f(\boldsymbol{x}_1) + (1-\lambda) f(\boldsymbol{x}_2)$$



strictly convex | convex | nonconvex

Convexity: specific cases

Linear programming.

- $f(\boldsymbol{x}) = \boldsymbol{c}^T \boldsymbol{x} = \sum_i c_i x_i$ is convex.

Quadratic programming.

- $f(\boldsymbol{x}) = \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 = \sum_i (y_i - \boldsymbol{a}_i^T \boldsymbol{x})^2$ is convex.

- $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}^T \boldsymbol{x} = \frac{1}{2} \sum_i \sum_j a_{ij} x_j x_j - \sum_i b_i x_i$ is convex when $\boldsymbol{H}$ is symmetric positive semedefinite.

Iterative algorithms

Iterative algorithms.

Solve the minimization with an iterative algorithm which generates a sequence of iterates x_0, x_1, ..., x_n.

Convergence.

The sequence of iterates x_n converges to the solution $x^\star$ of the minimization problem.

- Global convergence: valid for all $x_0 \in \mathcal{D}$.
- Local convergence: valid for x_0 in a neighborhood of $x^\star$.

Rate of convergence

- Linear convergence: $\dfrac{\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^\star\|}{\|\boldsymbol{x}_k - \boldsymbol{x}^\star\|} \leq r$ for k sufficiently large.
- Quadratic convergence: $\dfrac{\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^\star\|}{\|\boldsymbol{x}_k - \boldsymbol{x}^\star\|^2} \leq r$ for k sufficiently large.
- Super-linear convergence: $\displaystyle\lim_{k\to\infty} \frac{\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^\star\|}{\|\boldsymbol{x}_k - \boldsymbol{x}^\star\|} = 0$.

Linear programming

Canonical form:

Minimize $\boldsymbol{c}^T \boldsymbol{x}$ subject to $\begin{cases} \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ \boldsymbol{x} \geq \boldsymbol{0} \end{cases}$

Standard form:

Minimize $\boldsymbol{c}^T \boldsymbol{x}$ subject to $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$

Remarks:

- Set $\boldsymbol{z} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{0}$. Standard form $\Rightarrow$ Canonical form.
- No closed-form solution.

Resolution:

- The simplex algorithm.
- Integer linear programming (discrete case).

Linear least squares

Given $\boldsymbol{y} \in \mathbb{R}^m$ and $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, find $\boxed{\boldsymbol{x}^\star \in \arg\min_{\boldsymbol{x}} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2}$

$\boldsymbol{x}^\star$ is a solution of

$$\begin{aligned} & \nabla_{\boldsymbol{x}} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 = \mathbf{0} \\ \Leftrightarrow\ & 2\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}) = \mathbf{0} \\ \Leftrightarrow\ & \boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{y} \end{aligned}$$

If $m \geq n$ and $\boldsymbol{A}$ is full column rank: $\boldsymbol{x}^\star = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{y}$.

Otherwise:

- infinity of solutions.
- Generalized inverse: solution having the minimum ℓ_2-norm:

Example. Find the best coefficients (a, b) of a linear model so as to reproduce at best observations y_i at times t_i.

Example. Data fitting using a polynom of degree p.

Quadratic programming

Given $\boldsymbol{c} \in \mathbb{R}^n$, $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$, $\boldsymbol{A} \in \mathbb{R}^{p\times n}$,

Minimize $\frac{1}{2}\boldsymbol{x}^T Q \boldsymbol{x} + \boldsymbol{c}^T \boldsymbol{x}$ **subject to** $A\boldsymbol{x} \leq \boldsymbol{b}$.

- p: number of constraints.
- j-th constraint: $\boldsymbol{a}_i^T \boldsymbol{x} \leq b_i$.
- Q is positive semi-definite $\implies$ convex problem.
- Q is positive definite: strictly convex problem.

Nonlinear least squares

Given $\boldsymbol{y} \in \mathbb{R}^m$ and $\boldsymbol{\varphi} : \mathbb{R}^n \mapsto \mathbb{R}^m$,

$$\textbf{Minimize } \|\boldsymbol{y} - \boldsymbol{\varphi}(\boldsymbol{x})\|_2^2$$

- No closed form solution
- Specific algorithms taking the structure of the problem into account (Levenberg-Marquardt)

Example

Given a set of points (x_i, y_i), $i = 1, \ldots, N$, approximate the set of points using a parametric curve $y = \varphi(x; \boldsymbol{\theta})$.

Example: $\varphi(x; \alpha_1, \alpha_2, \tau_1, \tau_2) = \alpha_1 \exp(x/\tau_1) + \alpha_2 \exp(x/\tau_2)$

with $\boldsymbol{\theta} = \{\alpha_1, \alpha_2, \tau_1, \tau_2\}$.

1. Formulate the problem as an optimization problem.
2. Call the appropriate Matlab function.

cf. Matlab simulations.

2. Unconstrained local optimization

Working assumptions

$f : \mathbb{R}^n \to \mathbb{R}$ is $\mathcal{C}^1$ or $\mathcal{C}^2$.

Definitions.

- Gradient of criterion $f(\boldsymbol{x})$: $\nabla f(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \dots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} \in \mathbb{R}^n$

- Hessian of criterion $f(\boldsymbol{x})$:

$$\boldsymbol{H}(\boldsymbol{x}) = \nabla^2 f(\boldsymbol{x}) = \begin{bmatrix} \dots & \dots & \dots \\ \dots & \frac{\partial^2 f}{\partial x_i \, \partial x_j} & \dots \\ \dots & \dots & \dots \end{bmatrix} \in \mathbb{R}^{n \times n}$$

Existence of a local minimizer

Theorem. Given $\mathcal{U}$ a closed subset of $\mathbb{R}^n$, and a continuous function $f : \mathcal{U} \subset \mathbb{R}^n \to \mathbb{R}$.

- If $\mathcal{U}$ is bounded,
- or if $\lim\limits_{\substack{\|\boldsymbol{x}\| \to \infty \\ \boldsymbol{x} \in \mathcal{U}}} f(\boldsymbol{x}) = +\infty$ [coercivity],

then f admits a local minimum on $\mathcal{U}$.

Line search strategy

An iterative algorithm generates a sequence $\boldsymbol{x}_0$, $\boldsymbol{x}_1$, ..., $\boldsymbol{x}_n$.

How to move from $\boldsymbol{x}_k$ to $\boldsymbol{x}_{k+1}$?

- Use the information about function f at $\boldsymbol{x}_k$.
- Possibly use information from earlier iterates $\boldsymbol{x}_{k-1}$, $\boldsymbol{x}_{k-2}$, *etc.*

Line search.

- Choose a direction $\boldsymbol{d}_k$.
- Solve (approximately) the 1D minimization problem:

$$\min_{t>0} f(\boldsymbol{x}_k + t\boldsymbol{d}_k)$$

- Set $\boxed{\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + t\boldsymbol{d}_k}$

Feasible direction

$\boldsymbol{d}$ is a feasible direction at $\boldsymbol{x} \in \mathcal{D}$ if

$$\exists A > 0, \forall t \in [0, A], \boldsymbol{x} + t\boldsymbol{d} \in \mathcal{D}$$

Descent direction

- $\boldsymbol{d}$ is a descent direction of f at $\boldsymbol{x} \in \mathcal{D}$ if

$$\exists A > 0,\ \forall t \in [0, A],\ f(\boldsymbol{x} + t\boldsymbol{d}) \leq f(\boldsymbol{x})$$

- First order Taylor expansion:

$$f(\boldsymbol{x} + t\boldsymbol{d}) = f(\boldsymbol{x}) + t\boldsymbol{d}^T \nabla f(\boldsymbol{x}) + o(t)$$

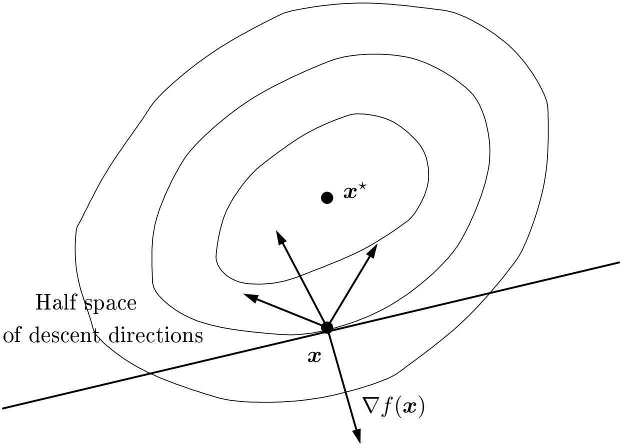
- $\boldsymbol{d}$ is a descent direction $\Longleftrightarrow \boldsymbol{d}^T \nabla f(\boldsymbol{x}) < 0$

  $\Longrightarrow$ Half space of descent directions.

Descent directions and level sets

Level sets: $\mathcal{L}_z = \{\boldsymbol{x} \in \mathbb{R}^n,\ f(\boldsymbol{x}) = z\}$.

Descent directions:



Derivatives

First order Taylor expansion:

$$f(\boldsymbol{x} + t\boldsymbol{d}) = f(\boldsymbol{x}) + t\boldsymbol{d}^T \nabla f(\boldsymbol{x}) + o(t)$$

Second order Taylor expansion:

$$f(\boldsymbol{x} + \boldsymbol{h}) = f(\boldsymbol{x}) + \boldsymbol{h}^T \nabla f(\boldsymbol{x}) + \frac{1}{2}\boldsymbol{h}^T \boldsymbol{H}(\boldsymbol{x})\boldsymbol{h} + o(\|\boldsymbol{h}\|^2)$$

$$f(\boldsymbol{x} + t\boldsymbol{d}) = f(\boldsymbol{x}) + t\boldsymbol{d}^T \nabla f(\boldsymbol{x}) + \frac{t^2}{2}\boldsymbol{d}^T \boldsymbol{H}(\boldsymbol{x})\boldsymbol{d} + o(t^2)$$

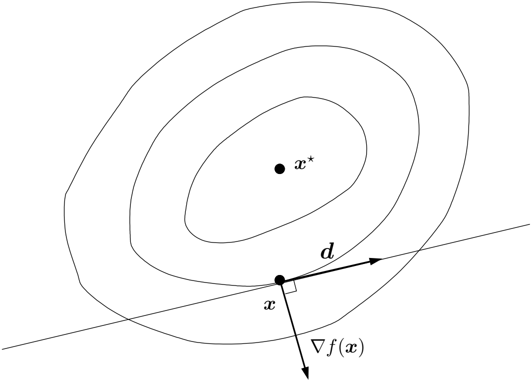
Directional derivative in the direction $\boldsymbol{d}$:

$$\begin{aligned} f'(\boldsymbol{x};\boldsymbol{d}) &= \lim_{t\to 0} \frac{f(\boldsymbol{x}+t\boldsymbol{d}) - f(\boldsymbol{x})}{t} \\ &= \boldsymbol{d}^T \nabla f(\boldsymbol{x}) \end{aligned}$$

Derivatives

If $\boldsymbol{d}$ is tangent to the level set, then $f'(\boldsymbol{x};\boldsymbol{d}) = \boldsymbol{d}^T \nabla f(\boldsymbol{x}) = 0$.

$\nabla f(\boldsymbol{x})$ is thus orthogonal to the level set.



Necessary condition for a local minimizer

Problem. Find $\boldsymbol{x}^\star \in \arg\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x})$

Note. $\max_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}) = -\min_{\boldsymbol{x}\in\mathbb{R}^n}(-f(\boldsymbol{x}))$

Theorem. If f admits a local minimum at $\boldsymbol{x}^\star$, then $\nabla f(\boldsymbol{x}^\star) = 0$.

Proof. Show that $f'(\boldsymbol{x}^\star; \boldsymbol{d}) = \boldsymbol{d}^T \nabla f(\boldsymbol{x}^\star) = 0 \ \forall \boldsymbol{d}$.

Remark. The condition is necessary but not sufficient.

Sufficient condition for a local minimizer

Theorem. If $\nabla f(\boldsymbol{x}^\star) = 0$ and $\boldsymbol{H}(\boldsymbol{x}^\star) > 0$, then f admits a local minimum at $\boldsymbol{x}^\star$.

Positive definiteness. $\boldsymbol{H}(\boldsymbol{x}^\star) > 0 \Leftrightarrow \forall \boldsymbol{d} \neq \mathbf{0},\ \boldsymbol{d}^T \boldsymbol{H}(\boldsymbol{x}^\star)\boldsymbol{d} > 0$.

Proof. Show that $\forall \boldsymbol{d},\ f(\boldsymbol{x}^\star + t\boldsymbol{d}) > f(\boldsymbol{x}^\star)$ for $|t|$ sufficiently small.

Descent algorithms

Choose $\boldsymbol{x}_0$.

For $k \geq 0$,

- Find a descent direction $\boldsymbol{d}_k$
- Find a step $t_k > 0$.
- Compute $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + t_k \boldsymbol{d}_k$.

Choice 1. Constant step $t_k = t$, $\forall k$.

Choice 2. 1D minimisation: $t_k = \arg\min_{t>0} f(\boldsymbol{x}_k + t\boldsymbol{d}_k)$.

Descent algorithms

Descent: $\forall k,\ f(\boldsymbol{x}_{k+1}) < f(\boldsymbol{x}_k)$.

Stopping conditions:

- $f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1}) < \varepsilon_1$.
- $\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|_2 < \varepsilon_2$.
- $\|\nabla f(\boldsymbol{x}_k)\|_2 < \varepsilon_3$.
- $k = K_{\text{max}}$.

Choice of the initial solution

- Unimodal criterion: any value of $\boldsymbol{x}_0$ should work (theoretically...)

Gradient based algorithms

The gradient algorithm.

- Choose $\boldsymbol{x}_0$.
- For $k \geq 0$, do $\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k - t_k \nabla f(\boldsymbol{x}_k)$.

  $\Rightarrow$ Constant step: $t_k = t$, $\forall k$.
  $\Rightarrow$ 1D minimization: $t_k \approx \arg\min_{t>0} f(\boldsymbol{x}_k - t\nabla f(\boldsymbol{x}_k))$.

The conjugate gradient algorithm.

- Choose $\boldsymbol{x}_0$.
- For $k \geq 0$, do $\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k - t_k \nabla f(\boldsymbol{x}_k) + \beta_k(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})$.

Convergence rate is much faster!

Gradient algorithm

Remark.

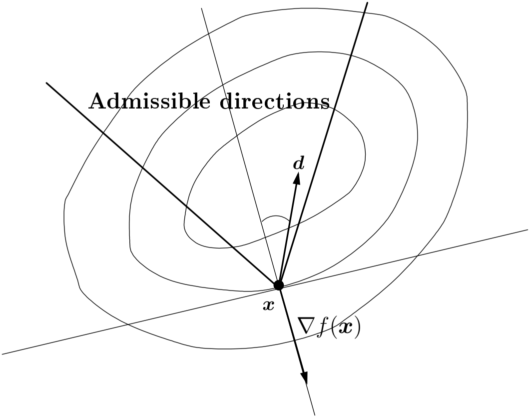
If the step is chosen as

$$t_k = \arg\min_{t>0} f(\boldsymbol{x}_k - t\nabla f(\boldsymbol{x}_k))$$

then forall k, $\nabla f(\boldsymbol{x}_k)$ and $\nabla f(\boldsymbol{x}_{k+1})$ are orthogonal (slow convergence, zigzag phenomenon).

$\Rightarrow$ Linear convergence.

Choice of descent direction



Impose that $|\mathrm{Angle}(\boldsymbol{d}_k, -\nabla f(\boldsymbol{x}_k))| \leq \frac{\pi}{2} - \mu$.

Line minimization

How to compute $\quad t_k \approx \arg\min_{t>0} f(\boldsymbol{x}_k - t\nabla f(\boldsymbol{x}_k)) \quad$?

- Steepest descent: exact minimization.
- Constant step $t_k = t$.
  - Risk of small steps.
  - Does not guarantee that $f(\boldsymbol{x}_k + t\boldsymbol{d}_k) < f(\boldsymbol{x}_k)\ \forall k$.
    $\Rightarrow$ Reajust $t = t/2$.
- Dichotomy, golden section.
- Quadratic approximation of $\phi : t \to f(\boldsymbol{x}_k + t\boldsymbol{d}_k)$.
- Quadratic or cubic approximation of ϕ from the knowledge of $\phi'(0)$.
- Armijo and Wolfe's rules: ensure that t is "acceptable" (sufficient decrease of f, sufficiently large step t).

Line minimization: example

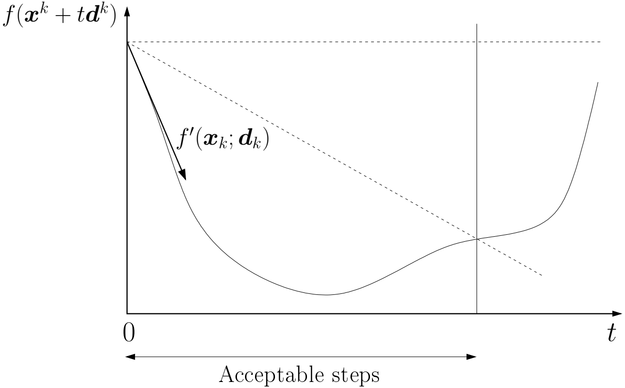
Quadratic interpolator.

Let $\phi(t) := f(\boldsymbol{x}_k + t\boldsymbol{d}_k)$.

Approximate $\phi(t) \approx q(t) = at^2 + bt + c$.

1. Compute $\phi(0) = f(\boldsymbol{x}_k)$.
2. Compute $\phi'(0) = \boldsymbol{d}_k^T \nabla f(\boldsymbol{x}_k)$.
3. Compute $\phi(t_0) = f(\boldsymbol{x}_k + t_0\boldsymbol{d}_k)$ for some t_0.
4. Find the quadratic interpolator: compute a, b, c.
5. Minimize $q(t) \quad \Rightarrow t_k$.

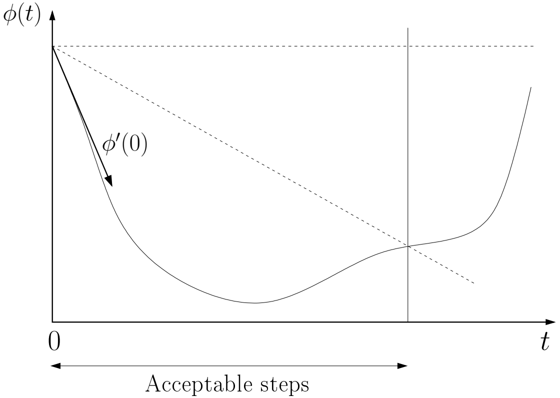
Step selection: Armijo's rule



Choose t such that $\quad f(\boldsymbol{x}_k + t\boldsymbol{d}_k) \leq f(\boldsymbol{x}_k) + c_1\, t\, f'(\boldsymbol{x}_k; \boldsymbol{d}_k)$

with $0 < c_1 < 1$.

Step selection: Armijo's rule

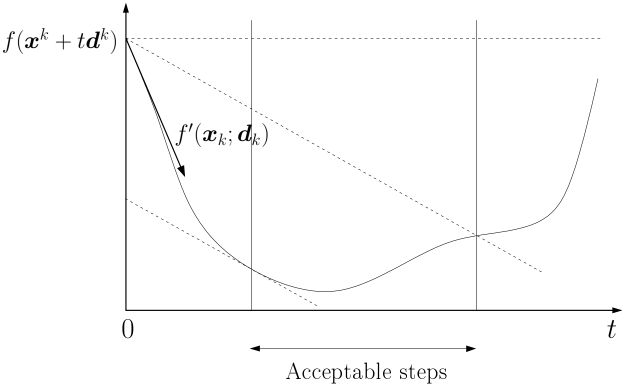


Let $\phi(t) := f(\boldsymbol{x}_k + t\boldsymbol{d}_k)$.

Armijo's condition rewrites $\quad \phi(t) \leq \phi(0) + c_1\, t\, \phi'(0)$

with $\phi'(0) = f'(\boldsymbol{x}_k; \boldsymbol{d}_k) = \nabla f(\boldsymbol{x}_k)^T \boldsymbol{d}_k$.

Step selection: Wolfe's rule



Choose t such that $$\begin{cases} \phi(t) \leq \phi(0) + c_1\, t\, \phi'(0) \\ \phi'(t) \geq c_2\, \phi'(0) \ \text{ with } \ 0 < c_2 < c_1 < 1 \end{cases}$$

Step selection: Wolfe's rule

Choose t such that $\begin{cases} \phi(t) \leq \phi(0) + c_1\, t\, \phi'(0) \\ \phi'(t) \geq c_2\, \phi'(0) \ \text{ with } \ 0 < c_2 < c_1 \end{cases}$

Rewriting:

Choose t such that

$$\begin{cases} f(\boldsymbol{x}_k + t\boldsymbol{d}_k) & \leq \ f(\boldsymbol{x}_k) + c_1\, t\, \nabla f(\boldsymbol{x}_k)^T \boldsymbol{d}_k \\ \nabla f(\boldsymbol{x}_k + t\boldsymbol{d}_k)^T \boldsymbol{d}_k & \geq c_2\, \nabla f(\boldsymbol{x}_k)^T \boldsymbol{d}_k \ \text{ with } \ 0 < c_2 < c_1 \end{cases}$$

Conjugate gradient algorithm

Conjugate gradient.

For $k \geq 0$,

$$\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k + \beta_k \boldsymbol{d}_k \quad \text{with} \quad \boldsymbol{d}_k = -\nabla f(\boldsymbol{x}_k) + \beta_k \boldsymbol{d}_{k-1}.$$

- Fletcher-Reeves: $\beta_k^{\text{FR}} = \dfrac{\|\nabla f(\boldsymbol{x}_k)\|^2}{\|\nabla f(\boldsymbol{x}_{k-1})\|^2}$
- Polak-Ribière: $\beta_k^{\text{PR}} = \dfrac{[\nabla f(\boldsymbol{x}_k)]^T(\nabla f(\boldsymbol{x}_k) - \nabla f(\boldsymbol{x}_{k-1}))}{\|\nabla f(\boldsymbol{x}_{k-1})\|^2}$

Subspace optimization (memory gradient).

$$\boldsymbol{d}_k = -\nabla f(\boldsymbol{x}_k) + \sum_{\ell=1}^{m} \omega_\ell \boldsymbol{d}_{k-\ell} \quad \text{with} \quad \omega_\ell > 0.$$

Use of second order derivatives

Idea.

1. Replace f by its quadratic approximation at point $\boldsymbol{x}_k$:

$$f(\boldsymbol{x}_k + \boldsymbol{d}) \approx f(\boldsymbol{x}_k) + \boldsymbol{d}^T \nabla f(\boldsymbol{x}_k) + \frac{1}{2}\boldsymbol{d}^T \boldsymbol{H}(\boldsymbol{x}_k)\boldsymbol{d}$$

2. Minimize the quadratic approximation: $\boldsymbol{d}_k = -[\boldsymbol{H}(\boldsymbol{x}_k)]^{-1}\boldsymbol{g}_k$

Works only if $\boldsymbol{H}(\boldsymbol{x}_k) > 0$.

Convex optimization

Newton algorithm.

- Choose $\boldsymbol{x}_0$.
- For $k \geq 0$, do $\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k - t_k[\boldsymbol{H}(\boldsymbol{x}_k)]^{-1}\nabla f(\boldsymbol{x}_k)$.

\+ Fast (superlinear) convergence.
– Costly, not suited to large scale applications.

Adaptations

Newton direction: solve $\boldsymbol{H}(\boldsymbol{x}_k)\boldsymbol{d}_k = -\nabla f(\boldsymbol{x}_k)$.

Truncated Newton direction: solve $\boldsymbol{H}(\boldsymbol{x}_k)\boldsymbol{d}_k \approx -\nabla f(\boldsymbol{x}_k)$:

find $\boldsymbol{d}_k$ s.t. $\|\boldsymbol{H}(\boldsymbol{x}_k)\boldsymbol{d}_k + \nabla f(\boldsymbol{x}_k)\| \leq \eta_k \|\nabla f(\boldsymbol{x}_k)\|$ for $0 < \eta_k < 1$.

Modified Newton direction: when H_k is not invertible, solve

$$[\boldsymbol{H}(\boldsymbol{x}_k) + \lambda_k \boldsymbol{I}_n]\boldsymbol{d}_k = -\nabla f(\boldsymbol{x}_k)$$

for $\lambda_k > 0$.

Quasi-Newton methods

Idea: Approximate $\boldsymbol{H}(\boldsymbol{x}_k)$, $\boldsymbol{H}(\boldsymbol{x}_k)^{-1}$, or $\boldsymbol{H}(\boldsymbol{x}_k)^{-1}\nabla f(\boldsymbol{x}_k)$ without computing second order derivatives.

Notations:

- $\boldsymbol{g}_k := \nabla f(\boldsymbol{x}_k)$.
- $\boldsymbol{H}_k$: approximation of $\boldsymbol{H}(\boldsymbol{x}_k)$.
- $\boldsymbol{B}_k$: approximation of $[\boldsymbol{H}(\boldsymbol{x}_k)]^{-1}$.

First order Taylor expansion:

$$\nabla f(\boldsymbol{x}_{k+1} + \boldsymbol{h}) = \nabla f(\boldsymbol{x}_{k+1}) + \boldsymbol{H}_{k+1}\boldsymbol{h} + o(\|\boldsymbol{h}\|)$$

$\Rightarrow$ Choose $\boldsymbol{H}_{k+1}$ such that $\boldsymbol{g}_{k+1} - \boldsymbol{g}_k = \boldsymbol{H}_{k+1}(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k)$.

$\Leftrightarrow$ Choose $\boldsymbol{B}_{k+1}$ such that $\boldsymbol{B}_{k+1}(\boldsymbol{g}_{k+1} - \boldsymbol{g}_k) = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$.

Quasi-Newton algorithm

BFGS algorithm (Broyden, Fletcher, Goldfarb, Shanno):

$\boldsymbol{B}_0 = \boldsymbol{I}_n$ and:

$$\boldsymbol{B}_{k+1} = \boldsymbol{B}_k + \left(1 + \frac{\boldsymbol{p}_k^T \boldsymbol{B}_k \boldsymbol{p}_k}{\boldsymbol{s}_k^T \boldsymbol{p}_k}\right) \frac{\boldsymbol{s}_k \boldsymbol{s}_k^T}{\boldsymbol{s}_k^T \boldsymbol{p}_k} - \frac{\boldsymbol{s}_k \boldsymbol{p}_k^T \boldsymbol{B}_k + \boldsymbol{B}_k \boldsymbol{p}_k \boldsymbol{s}_k^T}{\boldsymbol{s}_k^T \boldsymbol{p}_k}$$

with:

- $\boldsymbol{s}_k := \boldsymbol{x}_{k+1} - \boldsymbol{x}_k$.
- $\boldsymbol{p}_k := \nabla f(\boldsymbol{x}_{k+1}) - \nabla f(\boldsymbol{x}_k)$.

Notes.

- $\boldsymbol{B}_k$ is symmetric.
- For strictly convex f, BFGS preserves positive definiteness of $\boldsymbol{B}_k$.

Quasi-Newton algorithm

DFP algorithm (Davidson, Fletcher, Powell):

$\boldsymbol{B}_0 = \boldsymbol{I}_n$ and:

$$\boldsymbol{B}_{k+1} = \boldsymbol{B}_k + \frac{\boldsymbol{s}_k \boldsymbol{s}_k^T}{\boldsymbol{s}_k^T \boldsymbol{p}_k} - \frac{\boldsymbol{B}_k \boldsymbol{p}_k \boldsymbol{p}_k^T \boldsymbol{B}_k}{\boldsymbol{p}_k^T \boldsymbol{B}_k \boldsymbol{p}_k}$$

Quasi-Newton algorithm

Choose $\boldsymbol{x}_0$.

Set $\boldsymbol{B}_0 = \boldsymbol{I}_n$.

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

- $\boldsymbol{d}_k = -\boldsymbol{B}_k \boldsymbol{g}_k$.
- $t_k = \arg\min_{t>0} f(\boldsymbol{x}_k + t\boldsymbol{d}_k)$.
- $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + t_k \boldsymbol{d}_k$.
- $\boldsymbol{B}_{k+1} = \boldsymbol{B}_k + \ldots$ (BFGS rule).

End.

L-BFGS algorithm

Large scale problems, *e.g.,* image processing.

Avoid to store the Hessian matrix ($n(n+1)/2$ coefficients).

Recursively update $\boldsymbol{B}_k \boldsymbol{g}_k$.

Matlab implementation: run fminunc

1. Write a Matlab function containing the variables x (here, called z) as the first input and at least one ouput (objective value f):

```
function [f,g] = myobj1(z,beta)

x=z(1);
y=z(2);

f = (y-sin(beta*x)-0.1*x*x)^2;
g(1) = -2*(beta*cos(beta*x)+0.2*x)*(y-sin(beta*x)-0.1*x*x);
g(2) = 2*(y-sin(beta*x) -0.1*x*x);
```

- Name of the file: same as the name of the function (here, `myobj1.m`).
- At least one output (`f`), gradient output `g` is optional.
- Recommendation: to speed up the optimization process, provide `g` as output when computation of `g` is possible.

Matlab implementation: run fminunc

2. Run the Matlab solver . This is another Matlab file, *e.g.,* `runfminunc.m`:

```
clear all; % clear memory
close all; % close figures

beta = 2;

options = optimoptions('fminunc');
options = optimoptions(options,'Display','iter',...
   'MaxFunctionEvaluations',1000,'StepTolerance',1e-10,...
   'SpecifyObjectiveGradient',true,'CheckGradients',true);

% Initial solution
x0=[10;10];

% Compute initial objective value
fprintf('x0=(%.2f,%.2f), f=%f\n',x0(1),x0(2),...
        myobj1(x0,beta));
```

```
% Run optimization solver
xsol = fminunc(@(x)myobj1(x,beta),x0,options);

fprintf('Final solution=(%.2f,%.2f), f=%f\n',...
        xsol(1),xsol(2), myobj1(xsol,beta));

% 2D and 3D display

x=0:.1:6;
y=-1:.1:5;

% Pixel coordinates:
[xx,yy]=meshgrid(x,y);

zz=zeros(size(xx));
for i=1:length(xx(:)),
  zz(i)=myobj1([xx(i),yy(i)],beta);
end;
```

```matlab
% 2D image
figure(1);
imagesc(x,y,zz);
colorbar;
axis xy
xlabel('x'); ylabel('y');

% 3D surface
figure(2);
mesh(x,y,zz);
colormap pink
shading flat
xlabel('x'); ylabel('y');
```

3. Least squares problems

Least squares problems

- Goal: $$\min_{\boldsymbol{x}\in\mathbb{R}^n} \left\{ f(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{F}(\boldsymbol{x})\|_2^2 = \frac{1}{2}\sum_{j=1}^{p} F_j^2(\boldsymbol{x}) \right\}$$

- Residual vector $\boldsymbol{F} : \mathbb{R}^n \mapsto \mathbb{R}^p$:

$$\boldsymbol{F}(\boldsymbol{x}) = \begin{bmatrix} F_1(\boldsymbol{x}) \\ \vdots \\ F_p(\boldsymbol{x}) \end{bmatrix}$$

- $\boldsymbol{F}$ linear with respect to $\boldsymbol{x}$?

  Linear least squares *vs* nonlinear least squares.

Example 1

Predict the parameter of the model which best agrees with a time observation:

- y_j: observation at time t_j.
- p observations.
- $\phi(t;\boldsymbol{x}) = x_1 + t x_2 + t^2 x_3$.

$$\Rightarrow \quad \min_{\boldsymbol{x}\in\mathbb{R}^n} \sum_{j=1}^{p} \left[\phi(t_j;\boldsymbol{x}) - y_j\right]^2.$$

- Linear *vs* nonlinear least-squares?
  - $\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{J}\boldsymbol{x} - \boldsymbol{y}$?

Linear least-squares: $\min_{\boldsymbol{x}\in\mathbb{R}^n} \left\{ f(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{F}(\boldsymbol{x})\|_2^2 = \frac{1}{2}\|\boldsymbol{J}\boldsymbol{x} - \boldsymbol{y}\|_2^2 \right\}$

with:

$$\boldsymbol{F}(\boldsymbol{x}) = \begin{bmatrix} x_1 + t_1 x_2 + t_1^2 x_3 - y_1 \\ x_1 + t_2 x_2 + t_2^2 x_3 - y_2 \\ \vdots \\ x_1 + t_p x_2 + t_p^2 x_3 - y_p \end{bmatrix} \in \mathbb{R}^p$$

$$= \begin{bmatrix} 1 & t_1 & t_1^2 \\ 1 & t_2 & t_2^2 \\ \vdots & \vdots & \vdots \\ 1 & t_p & t_p^2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} - \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_p \end{bmatrix}$$

$$= \boldsymbol{J}\boldsymbol{x} - \boldsymbol{y}$$

- Linear least-squares problem: $\min_{x \in \mathbb{R}^n} \left\{ f(x) = \frac{1}{2} \|Jx - y\|_2^2 \right\}$

- Gradient: $\nabla f(x) = J^T(Jx - y)$.

- Solution:

$$\begin{aligned} \nabla f(x^\star) = \mathbf{0} \quad &\Longleftrightarrow \quad J^T(Jx^\star - y) = \mathbf{0} \\ &\Longleftrightarrow \quad (J^T J)x^\star = J^T y \quad \text{(normal equations).} \\ &\Longleftrightarrow \quad x^\star = (J^T J)^{-1} J^T y. \end{aligned}$$

Example 2

Predict the parameter of the model which best agrees with a time observation:

- y_j: observation at time t_j.
- P observations.
- $\phi(t;\boldsymbol{x}) = x_1 + t x_2 + t^2 x_3 + x_4 e^{-x_5 t}$.

$$\Rightarrow \quad \min_{\boldsymbol{x}\in\mathbb{R}^n} \sum_{j=1}^{P} \left[\phi(t_j;\boldsymbol{x}) - y_j\right]^2.$$

- Nonlinear least-squares

Nonlinear least-squares: $\displaystyle \min_{\boldsymbol{x}\in\mathbb{R}^n} \left\{ f(\boldsymbol{x}) = \frac{1}{2}\|F(\boldsymbol{x})\|_2^2 \right\}$

with:

$$\boldsymbol{F}(x_1, x_2, x_3, x_4, x_5) = \begin{bmatrix} F_1(x_1, x_2, x_3, x_4, x_5) \\ F_2(x_1, x_2, x_3, x_4, x_5) \\ \vdots \\ F_p(x_1, x_2, x_3, x_4, x_5) \end{bmatrix}$$

$$= \begin{bmatrix} x_1 + t_1 x_2 + t_1^2 x_3 + x_4 e^{-x_5 t_1} - y_1 \\ x_1 + t_2 x_2 + t_2^2 x_3 + x_4 e^{-x_5 t_2} - y_2 \\ \vdots \\ x_1 + t_p x_2 + t_p^2 x_3 + x_4 e^{-x_5 t_p} - y_p \end{bmatrix} \in \mathbb{R}^p$$

Nonlinear least-squares

- Objective function: $f(\boldsymbol{x}) = \dfrac{1}{2} \displaystyle\sum_{j=1}^{p} F_j^2(\boldsymbol{x})$.
- Partial derivatives: $\dfrac{\partial f}{\partial x_i}(\boldsymbol{x}) = \displaystyle\sum_{j=1}^{p} \dfrac{\partial F_j}{\partial x_i}(\boldsymbol{x}) \, F_j(\boldsymbol{x})$
- Gradient:

$$\nabla f(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial F_1}{\partial x_1}(\boldsymbol{x}) & \frac{\partial F_2}{\partial x_1}(\boldsymbol{x}) & \dots & \frac{\partial F_p}{\partial x_1}(\boldsymbol{x}) \\ \frac{\partial F_1}{\partial x_2}(\boldsymbol{x}) & \frac{\partial F_2}{\partial x_2}(\boldsymbol{x}) & \dots & \frac{\partial F_p}{\partial x_2}(\boldsymbol{x}) \\ \vdots & \vdots & \vdots & \\ \frac{\partial F_1}{\partial x_p}(\boldsymbol{x}) & \frac{\partial F_2}{\partial x_p}(\boldsymbol{x}) & \dots & \frac{\partial F_p}{\partial x_p}(\boldsymbol{x}) \end{bmatrix} \begin{bmatrix} F_1(\boldsymbol{x}) \\ F_2(\boldsymbol{x}) \\ \vdots \\ F_p(\boldsymbol{x}) \end{bmatrix}$$

Gradient and Jacobian

Jacobian $\boldsymbol{J}(\boldsymbol{x})$ is a matrix of size $p \times n$ whose rows identify with $\nabla F_j(\boldsymbol{x})$:

$$\boldsymbol{J}(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial F_1}{\partial x_1}(\boldsymbol{x}) & \frac{\partial F_1}{\partial x_2}(\boldsymbol{x}) & \dots & \frac{\partial F_1}{\partial x_n}(\boldsymbol{x}) \\ \frac{\partial F_2}{\partial x_1}(\boldsymbol{x}) & \frac{\partial F_2}{\partial x_2}(\boldsymbol{x}) & \dots & \frac{\partial F_2}{\partial x_n}(\boldsymbol{x}) \\ \vdots & \vdots & \vdots & \\ \frac{\partial F_p}{\partial x_1}(\boldsymbol{x}) & \frac{\partial F_p}{\partial x_2}(\boldsymbol{x}) & \dots & \frac{\partial F_p}{\partial x_n}(\boldsymbol{x}) \end{bmatrix}$$

$$\boxed{\nabla f(\boldsymbol{x}) = \left[\boldsymbol{J}(\boldsymbol{x})\right]^T \boldsymbol{F}(\boldsymbol{x})}.$$

Second order derivatives

- Partial derivatives: $\dfrac{\partial f}{\partial x_i} = \displaystyle\sum_{k=1}^{p} \dfrac{\partial F_k}{\partial x_i} F_k$

- Second order derivatives:

$$\frac{\partial^2 f}{\partial x_i \, \partial x_j} = \sum_{k=1}^{p} \frac{\partial F_k}{\partial x_i} \frac{\partial F_k}{\partial x_j} + \sum_{k=1}^{p} F_k \frac{\partial^2 F_k}{\partial x_i \, \partial x_j}.$$

- Hessian of f: $\boldsymbol{H}(\boldsymbol{x}) = \nabla^2 f(\boldsymbol{x}) = \begin{bmatrix} \cdots & \cdots & \cdots \\ \cdots & \frac{\partial^2 f}{\partial x_i \, \partial x_j} & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix} \in \mathbb{R}^{n \times n}$

$$\Longrightarrow \boxed{\boldsymbol{H}(\boldsymbol{x}) = \boldsymbol{J}(\boldsymbol{x})^T \boldsymbol{J}(\boldsymbol{x}) + \sum_{k=1}^{p} \boldsymbol{F}_k \boldsymbol{H}_k}$$

Gauss-Newton method

The gradient and Hessian of $f(\boldsymbol{x}) = \dfrac{1}{2}\sum_{j=1}^{p} F_j^2(\boldsymbol{x})$ read:

$$\nabla f(\boldsymbol{x}) = \left[\boldsymbol{J}(\boldsymbol{x})\right]^T \boldsymbol{F}(\boldsymbol{x})$$

$$\boldsymbol{H}(\boldsymbol{x}) = \boldsymbol{J}(\boldsymbol{x})^T \boldsymbol{J}(\boldsymbol{x}) + \sum_{k=1}^{p} \boldsymbol{F}_k \boldsymbol{H}_k$$

1. Approximate $\boldsymbol{H} \approx \boldsymbol{J}(\boldsymbol{x})^T \boldsymbol{J}(\boldsymbol{x})$.
2. Compute descent direction by solving

$$[\boldsymbol{J}(\boldsymbol{x}_k)^T \boldsymbol{J}(\boldsymbol{x}_k)]\boldsymbol{d}_k = -\boldsymbol{J}(\boldsymbol{x}_k)^T \boldsymbol{F}(\boldsymbol{x}_k)$$

3. Line search:

$$\min_{t>0} f(\boldsymbol{x}_k + t\boldsymbol{d}_k)$$

Gauss-Newton method

Validity.

- When residuals are small ($\boldsymbol{F}(\boldsymbol{x}) \approx \mathbf{0}$).
- When $F_j(\boldsymbol{x})$ are nearly linear.

Advantages.

- Numerical cost is cheap (no calculation of second order derivatives).
- When $\boldsymbol{J}(\boldsymbol{x}_k)$ has full rank and $\nabla f(\boldsymbol{x}) \neq \mathbf{0}$, $\boldsymbol{d}_k$ is a descent direction for f.

The Levenberg-Marquardt method

1. Approximate $\boldsymbol{H} \approx \boldsymbol{J}(\boldsymbol{x})^T \boldsymbol{J}(\boldsymbol{x}) + \lambda \boldsymbol{I}_n$ for some $\lambda \geq 0$.
2. Compute descent direction by solving
$$[\boldsymbol{J}(\boldsymbol{x}_k)^T \boldsymbol{J}(\boldsymbol{x}_k) + \lambda_k \boldsymbol{I}_n] \boldsymbol{d}_k = -\boldsymbol{J}(\boldsymbol{x}_k)^T \boldsymbol{F}(\boldsymbol{x}_k)$$
3. Line search:
$$\min_{t>0} f(\boldsymbol{x}_k + t\boldsymbol{d}_k)$$

Comments.

- Case where $J(\boldsymbol{x})$ is rank-deficient, or nearly so.
- $\lambda_k = 0$: Gauss-Newton method.
- $\lambda_k \to \infty$: the gradient algorithm (slower but numerically accurate).
- There are automatic strategies for tuning λ_k.

Matlab implementation: run lsqnonlin

1. Write a Matlab function containing the variables x as the first input and at least one ouput (residual vector $\boldsymbol{F}(\boldsymbol{x})$):

```
function [F,J] = residuals(x,t,y)

% residual vector
F = y-x(1)*exp(-t/x(2)); % same size as y

% Jacobian matrix of size (length(t) x length(x))
% Column 1:  derivative of F_i = y_i - x_1 exp(-t_i/x_2) w.r.t.  x_1
% Column 2:  derivative of F_i = y_i - x_1 exp(-t_i/x_2) w.r.t.  x_2
J = [-exp(-t/x(2)), -x(1)/(x(2)*x(2))*t.*exp(-t./x(2))];
```

- Name of the file: `residuals.m`.
- At least one output (`F`), Jacobian output `J` is optional.
- Recommendation: to speed up the optimization process, provide `J` as output when computation of `J` is easy.

Matlab implementation: run lsqnonlin

2. Run the Matlab solver. This is another Matlab file, *e.g.,* `fit_expo.m`:

```
% load data t and y, defined as column vectors
figure(1), clf, plot(t,y,'o','linewidth',2); grid on;

% Initial parameter values
x_init = [10,10];

opt = optimoptions('lsqnonlin');
opt = optimoptions(opt,'Display','iter');

% If Jacobian is provided as output of function "residuals"
opt = optimoptions(opt,'SpecifyObjectiveGradient',true);

% for debug only, checks that the Jacobian computation
% is correct:
% opt = optimoptions(opt,'CheckGradients',true);
```

```
% Run optimization solver

x = lsqnonlin(@(x)(residuals(x,t,y)),x_init,[],[],opt);

disp('Solution:'); fprintf('%.2f ',x'); disp(' ');

% Plot prediction of data y using model x1 exp(-t/x2)
figure(1), hold on,
plot(t,x(1)*exp(-t/x(2)),'-r','linewidth',2);

% One could also plot the prediction of data y
% using the initial parameters x_init which is less accurate

% plot(t,x_init(1)*exp(-t/x_init(2)),'-g','linewidth',2);
```

4. Constrained local optimization

Constrained minimization

Minimize $f(\boldsymbol{x})$ **subject to** $\begin{cases} g_i(\boldsymbol{x}) \leq 0, & i = 1, \ldots, p \\ h_j(\boldsymbol{x}) = 0, & j = 1, \ldots, q. \end{cases}$

Case of linear constraints:

- Each constraint reads $g_i(\boldsymbol{x}) = \boldsymbol{a}_i^T \boldsymbol{x} - b_i = \sum_{k=1}^{n} a_i(k) x(k) - b_i$.
- Matrix form: $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} \leq \mathbf{0}_p$ with $\boldsymbol{A} \in \mathbb{R}^{p \times n}$.
- Similarly, write $\boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{C}\boldsymbol{x} - \boldsymbol{d}$.

- Matlab: $\boldsymbol{A}$, $\boldsymbol{b}$, $\boldsymbol{C}$, $\boldsymbol{d}$ are provided as inputs of fmincon.
- Case of lower ($x_k \geq \ell_k$) and upper bounds ($x_k \leq u_k$): compute corresponding $\boldsymbol{a}_i$ and b_i.

Constrained minimization

Minimize $f(x)$ **subject to** $\begin{cases} g_i(x) \leq 0, & i = 1, \ldots, p \\ h_j(x) = 0, & j = 1, \ldots, q. \end{cases}$

Case of nonlinear constraints (Matlab):

- Write a function

  ```
  [g,h] = function mycon(x);
  ```

  returning the values of $g_i(x)$ and $h_i(x)$ for all nonlinear constraints.

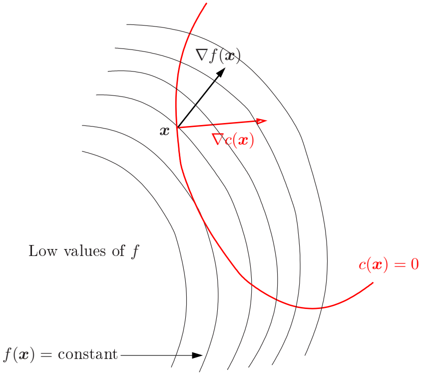
- Provide this function as input of `fmincon`.

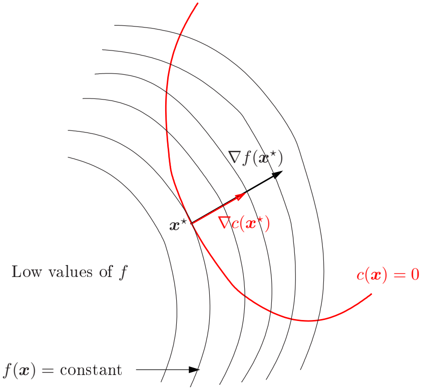
Existence of a minimizer

We will assume that functions f, g_i and h_j are twice differentiable.

If the feasible set $\mathcal{D} = \{\boldsymbol{x} \in \mathbb{R}^n,\, \boldsymbol{g}(\boldsymbol{x}) \leq \mathbf{0}_p,\, \boldsymbol{h}(\boldsymbol{x}) = \mathbf{0}_q\}$ is non-empty and bounded, then there exists a minimizer.

Case of 1 equality constraint $c(\boldsymbol{x}) = 0$





At the optimal point $\boldsymbol{x}^\star$, $\boxed{\nabla f(\boldsymbol{x}^\star) = \lambda \nabla c(\boldsymbol{x}^\star)}$ for some λ.

Generalization for q equality constraints

Problem with equality constraints only:

$$\min_{\boldsymbol{x} \in \mathbb{R}^n} \quad f(\boldsymbol{x}) \quad \text{s.t.} \quad \begin{cases} h_1(\boldsymbol{x}) = 0 \\ h_2(\boldsymbol{x}) = 0 \\ \quad \vdots \\ h_q(\boldsymbol{x}) = 0 \end{cases}$$

Define the Lagrangian function for $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{\lambda} \in \mathbb{R}^q$:

$$\boxed{\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) - \lambda_1 h_1(\boldsymbol{x}) - \ldots - \lambda_q h_q(\boldsymbol{x})}$$

$\boldsymbol{\lambda} = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_q \end{bmatrix}$ are Lagrange multipliers.

First order optimality conditions

Lagrangian: $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) - \sum_{i=1}^{q} \lambda_i h_i(\boldsymbol{x})$

If $\boldsymbol{x}^\star$ is a local minimizer of f, then there exists $\boldsymbol{\lambda}^\star$ such that:

$$\left\{ \begin{array}{l} \nabla_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}^\star, \boldsymbol{\lambda}^\star) = \mathbf{0}_n \\ \\ \nabla_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{x}^\star, \lambda^\star) = \mathbf{0}_q \end{array} \right. \iff \left\{ \begin{array}{l} \nabla f(\boldsymbol{x}^\star) = \sum_{i=1}^{q} \lambda_i^\star \nabla h_i(\boldsymbol{x}^\star) \\ \\ h_1(\boldsymbol{x}^\star) = \ldots = h_q(\boldsymbol{x}^\star) = 0 \end{array} \right.$$

Example: $\min_{\boldsymbol{x} \in \mathbb{R}^2} \; (x_1^2 + x_2^2) \quad \text{s.t.} \quad x_1 + x_2 = 1$

1. Graphical representation.
2. Write the optimality conditions.
3. Solve the system.

Inequality constrained problems

Problem.

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \; f(\boldsymbol{x}) \quad \text{s.t.} \quad \begin{cases} g_1(\boldsymbol{x}) \leq 0 \\ g_2(\boldsymbol{x}) \leq 0 \\ \quad\vdots \\ g_p(\boldsymbol{x}) \leq 0 \end{cases}$$

Definition.

The i-th constraint is active at $\boldsymbol{x}$ if $g_i(\boldsymbol{x}) = 0$.

The i-th constraint is inactive at $\boldsymbol{x}$ if $g_i(\boldsymbol{x}) < 0$.

Lagrangian function

Problem with inequality constraints only:

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \quad f(\boldsymbol{x}) \quad \text{s.t.} \quad \begin{cases} g_1(\boldsymbol{x}) \geq 0 \\ g_2(\boldsymbol{x}) \geq 0 \\ \quad\vdots \\ g_p(\boldsymbol{x}) \geq 0 \end{cases}$$

Define the Lagrangian function for $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{\lambda} \in \mathbb{R}_+^p$:

$$\boxed{\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) - \lambda_1 g_1(\boldsymbol{x}) - \ldots - \lambda_p g_p(\boldsymbol{x})}$$

$$\boldsymbol{\lambda} = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_p \end{bmatrix} \geq \mathbf{0} \qquad \text{are Lagrange multipliers.}$$

Lagrangian function (bis)

Modified Problem:

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \quad f(\boldsymbol{x}) \quad \text{s.t.} \quad \begin{cases} g_1(\boldsymbol{x}) & \leq & 0 \\ g_2(\boldsymbol{x}) & \leq & 0 \\ & \vdots & \\ g_p(\boldsymbol{x}) & \leq & 0 \end{cases}$$

Define the Lagrangian function for $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{\lambda} \in \mathbb{R}_+^p$:

$$\boxed{\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \lambda_1 g_1(\boldsymbol{x}) + \ldots + \lambda_p g_p(\boldsymbol{x})}$$

$$\boldsymbol{\lambda} = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_p \end{bmatrix} \geq \mathbf{0} \quad \text{are Lagrange multipliers.}$$

Karush-Kuhn-Tucker (KKT) conditions

If $\boldsymbol{x}^\star$ is a local solution of

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \text{ s.t. } \boldsymbol{g}(\boldsymbol{x}) \geq \mathbf{0}_p$$

then there exists $\boldsymbol{\lambda}^\star$ such that:

$$\begin{cases} \nabla_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}^\star, \lambda^\star) = \mathbf{0}_n \iff \nabla f(\boldsymbol{x}^\star) = \sum_{i=1}^{p} \lambda_i^\star \nabla g_i(\boldsymbol{x}^\star) \\ \forall i,\ g_i(\boldsymbol{x}^\star) \geq 0 \\ \forall i,\ \lambda_i^\star \geq 0 \\ \forall i,\ \lambda_i^\star g_i(\boldsymbol{x}^\star) = 0 \end{cases}$$

$$\left(\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) - \sum_{i=1}^{p} \lambda_i g_i(\boldsymbol{x}) \right)$$

Karush-Kuhn-Tucker (KKT) conditions

If $\boldsymbol{x}^\star$ is a local solution of

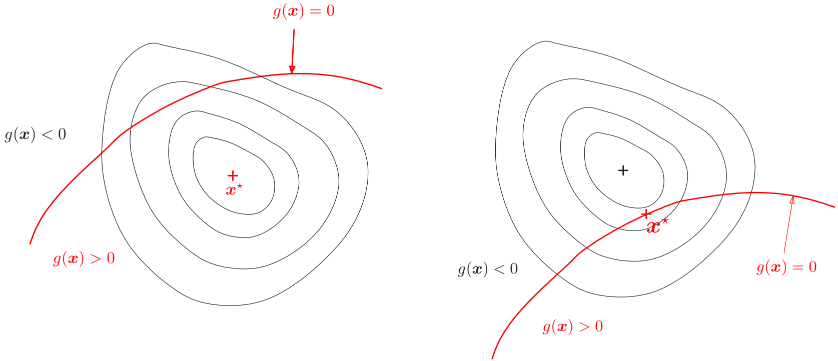
$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \ \text{ s.t. } \ \boldsymbol{g}(\boldsymbol{x}) \leq \mathbf{0}_p$$

then there exists $\boldsymbol{\lambda}^\star$ such that:

$$\begin{cases} \nabla_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}^\star, \boldsymbol{\lambda}^\star) = \mathbf{0}_n \iff \nabla f(\boldsymbol{x}^\star) = -\sum_{i=1}^{p} \lambda_i^\star \nabla g_i(\boldsymbol{x}^\star) \\ \forall i,\ g_i(\boldsymbol{x}^\star) \leq 0 \\ \forall i,\ \lambda_i^\star \geq 0 \\ \forall i,\ \lambda_i^\star g_i(\boldsymbol{x}^\star) = 0 \end{cases}$$

$$\Big(\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \sum_{i=1}^{p} \lambda_i g_i(\boldsymbol{x}) \Big)$$

Case of 1 inequality constraint $g(x) \geq 0$



Inactive constraint: $g(x^\star) > 0$

$\lambda^\star = 0$

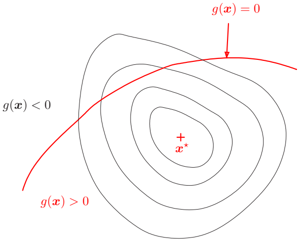
$(\lambda^\star g(x^\star) = 0)$

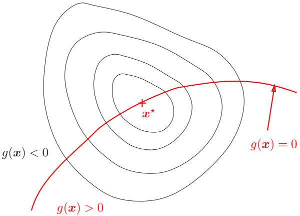
Active constraint: $g(x^\star) = 0$

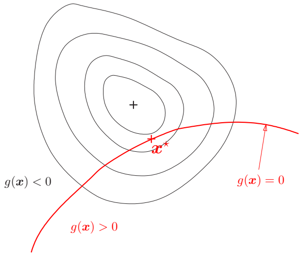
$\lambda^\star > 0$

$(\lambda^\star g(x^\star) = 0)$

Inequality constraints







Inactive	Weakly active	Active
$\lambda^\star = 0$	$\lambda^\star = 0$	$\lambda^\star > 0$
$g(\boldsymbol{x}^\star) > 0$	$g(\boldsymbol{x}^\star) = 0$	$g(\boldsymbol{x}^\star) = 0$

Examples

Example 1: $$\min_{\boldsymbol{x}\in\mathbb{R}^2} \; x_1^2 + x_2^2 \quad \text{s.t.} \quad \begin{cases} x_1 \geq 0 \\ x_2 \geq 0 \\ x_1 + 2x_2 \geq 1. \end{cases}$$

Example 2: [Nocedal & Wright, 2006, p. 329]

$$\min_{\boldsymbol{x}\in\mathbb{R}^2} \; \left(x_1 - \frac{3}{2}\right)^2 + \left(x_2 - \frac{1}{8}\right)^4 \quad \text{s.t.} \quad \begin{cases} x_2 \leq 1 - x_1 \\ x_2 \geq -1 - x_1 \\ x_2 \geq x_1 - 1 \\ x_2 \leq x_1 + 1 \end{cases}$$

1. Graphical representation.
2. Write the optimality conditions.
3. Solve the system.

Necessary condition for a local minimizer

First order optimality condition.

If $\boldsymbol{x}^\star$ is local minimizer, then $\boldsymbol{x}^\star$ satisfies the KKT conditions.

Second order optimality condition.

- Let $\mathcal{A}(\boldsymbol{x})$ denote the set of active constraints at $\boldsymbol{x}$
- Let $\mathcal{F}(\boldsymbol{x})$ denote the set of feasible directions at $\boldsymbol{x}$.
- Let $\mathcal{C}(\boldsymbol{x},\boldsymbol{\lambda}) = \{\boldsymbol{w} \in \mathcal{F}(\boldsymbol{x}) \mid [\nabla g_i(\boldsymbol{x})]^T \boldsymbol{w} = 0 \, \forall i \in \mathcal{A}(\boldsymbol{x}) \text{ with } \lambda_i > 0.\}$

If $\boldsymbol{x}^\star$ satisfies the KKT conditions and

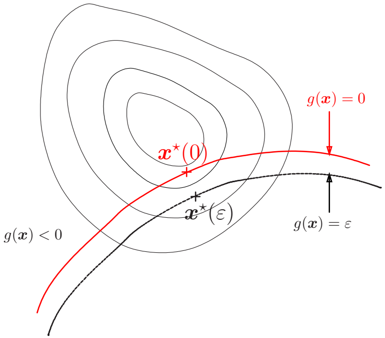
$$\forall \boldsymbol{w} \in \mathcal{C}(\boldsymbol{x}^\star,\boldsymbol{\lambda}^\star)\backslash\{\mathbf{0}\},\ \boldsymbol{w}^T \nabla_{\boldsymbol{xx}} \mathcal{L}(\boldsymbol{x}^\star,\boldsymbol{\lambda}^\star)\boldsymbol{w} > 0$$

then $\boldsymbol{x}^\star$ is a strict local minimizer of the inequality constrained problem.

Interpretation of Lagrange multipliers

For $\varepsilon > 0$, let $\boldsymbol{x}^\star(\varepsilon) = \arg\min_{\boldsymbol{x}} f(\boldsymbol{x}) \ \texttt{s.t.} \ g(\boldsymbol{x}) \geq \varepsilon$.

Let $f^\star(\varepsilon) = f(\boldsymbol{x}^\star(\varepsilon))$. Then, $\boxed{\lambda^\star \|\nabla g(\boldsymbol{x}^\star)\| = \frac{\partial f^\star}{\partial \varepsilon}(\varepsilon = 0)}$.



General case: inequality and equality constraints

Problem: $\min_{\boldsymbol{x}\in\mathbb{R}^n} \; f(\boldsymbol{x}) \;$ s.t. $\begin{cases} g_i(\boldsymbol{x}) \geq 0, & i = 1,\ldots,p \\ h_j(\boldsymbol{x}) = 0, & j = 1,\ldots,q. \end{cases}$

Define the Lagrangian function for $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{\lambda} \in \mathbb{R}_+^p$, $\boldsymbol{\mu} \in \mathbb{R}^q$:

$$\boxed{\mathcal{L}(\boldsymbol{x},\boldsymbol{\lambda},\boldsymbol{\mu}) = f(\boldsymbol{x}) - \sum_{i=1}^{p} \lambda_i g_i(\boldsymbol{x}) - \sum_{i=1}^{q} \mu_j h_j(\boldsymbol{x})}$$

So, $\nabla_{\boldsymbol{x}}\mathcal{L}(\boldsymbol{x},\boldsymbol{\lambda},\boldsymbol{\mu}) = \nabla f(\boldsymbol{x}) - \sum_{i=1}^{p} \lambda_i \nabla g_i(\boldsymbol{x}) - \sum_{j=1}^{q} \mu_j \nabla h_j(\boldsymbol{x})$

KKT conditions

Problem: $\min_{\boldsymbol{x}\in\mathbb{R}^n} \; f(\boldsymbol{x}) \;$ s.t. $\begin{cases} g_i(\boldsymbol{x}) \geq 0, & i = 1, \ldots, p \\ h_j(\boldsymbol{x}) = 0, & j = 1, \ldots, q. \end{cases}$

If $\boldsymbol{x}^\star$ is a local minimizer, then there exists $\boldsymbol{\lambda}^\star$ and $\mu^\star$ such that:

$$\begin{cases} \nabla_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star) = \mathbf{0}_n \\ \forall i, \; g_i(\boldsymbol{x}^\star) \geq 0 \\ \forall j, \; h_j(\boldsymbol{x}^\star) = 0 \\ \forall i, \; \lambda_i^\star \geq 0 \\ \forall i, \; \lambda_i^\star \, g_i(\boldsymbol{x}^\star) = 0 \end{cases}$$

Example

$$\min_{\boldsymbol{x}\in\mathbb{R}^2} \; (4x_1^2 + 2x_2^2) \;\text{ s.t. }\; \begin{cases} 3x_1 + x_2 = 8 \\ 2x_1 + 4x_2 \leq 15 \end{cases}$$

- $\mathcal{L}(x_1, x_2, \lambda, \mu) = 4x_1^2 + 2x_2^2 + \lambda(2x_1 + 4x_2 - 15) - \mu(3x_1 + x_2 - 8)$.
- $\frac{\partial \mathcal{L}}{\partial x_1}(x_1, x_2, \lambda, \mu) = 8x_1 + 2\lambda - 3\mu$.
- $\frac{\partial \mathcal{L}}{\partial x_2}(x_1, x_2, \lambda, \mu) = 4x_2 + 4\lambda - \mu$.
- KKT conditions: $$\begin{cases} 8x_1 + 2\lambda - 3\mu = 0 \\ 4x_2 + 4\lambda - \mu = 0 \\ 3x_1 + x_2 = 8 \\ 2x_1 + 4x_2 \leq 15 \\ \lambda \geq 0 \\ \lambda\,(2x_1 + 4x_2 - 15) = 0 \end{cases}$$

$$\left\{\begin{array}{rcl} 8x_1+2\lambda-3\mu &=& 0\\ 4x_2+4\lambda-\mu &=& 0\\ 3x_1+x_2 &=& 8\\ 2x_1+4x_2 &\leq& 15\\ \lambda &\geq& 0\\ \lambda\,(2x_1+4x_2-15) &=& 0 \end{array}\right. \Longrightarrow \left\{\begin{array}{rcl} 8x_1-12x_2 &=& 10\lambda\\ 3x_1+x_2 &=& 8\\ 2x_1+4x_2 &\leq& 15\\ \lambda &\geq& 0\\ \lambda\,(2x_1+4x_2-15) &=& 0 \end{array}\right.$$

- Assume $2x_1+4x_2=15$. $3x_1+x_2=8$ implies that $x_1=\frac{17}{10}$, $x_2=\frac{29}{10}$. So, $\lambda=\frac{8x_1-12x_2}{10}=-\frac{212}{100}<0$. Contradiction!
- So, $\lambda=0$. The KKT conditions become:

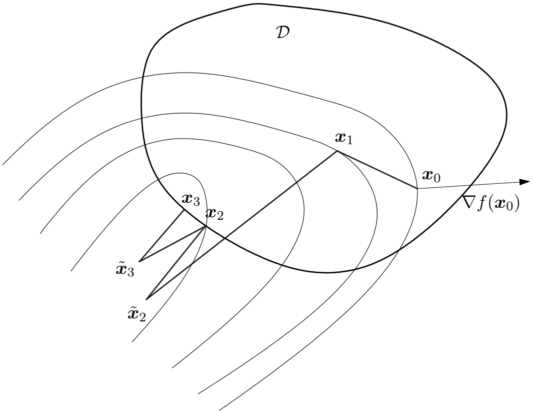
$$\left\{\begin{array}{rcl} \lambda &=& 0\\ \mu &=& \frac{8x_1}{3}\\ \mu &=& 4x_2\\ 3x_1+x_2 &=& 8\\ 2x_1+4x_2 &\leq& 15 \end{array}\right. \Longrightarrow \left\{\begin{array}{rcl} \lambda &=& 0\\ \mu &=& \frac{8x_1}{3}\\ x_1 &=& \frac{3}{2}x_2\\ \frac{11}{2}x_2 &=& 8\\ 2x_1+4x_2 &\leq& 15 \end{array}\right.$$

$$\Longrightarrow \left\{\begin{array}{rcl} \lambda &=& 0\\ \mu &=& \frac{8x_1}{3}\\ x_1 &=& \frac{24}{11}\\ x_2 &=& \frac{16}{11}\\ \text{Constraint}\quad \frac{112}{11} &\leq& 15\ \text{(OK)} \end{array}\right.$$

Gradient projection algorithm

Principle.

1. Line minimization: $\tilde{\boldsymbol{x}}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$.
2. Projection onto convex set: $\boldsymbol{x}_{k+1} = \texttt{proj}(\tilde{\boldsymbol{x}}_{k+1}, \mathcal{D})$.



Other approaches

Interior point approaches.

Example: Replace $\min_{\boldsymbol{x} \geq \mathbf{0}} f(\boldsymbol{x})$ by $\min_{\boldsymbol{x} \in \mathbb{R}^n} \left\{ f(\boldsymbol{x}) - \xi \sum_{i=1}^{n} \log x_i \right\}$.

Constrained minimization is replaced by the unconstrainted minimization of the augmented criterion.

"Exterior" approaches, *e.g.,* quadratic penalty, ADMM.

Example: Replace $\min_{\boldsymbol{x} \geq \mathbf{0}} f(\boldsymbol{x})$ by

$$\min_{\boldsymbol{x}, \boldsymbol{p} \geq \mathbf{0}} \{ \mathcal{K}(\boldsymbol{x}, \boldsymbol{p}; \xi) = f(\boldsymbol{x}) + \xi \|\boldsymbol{x} - \boldsymbol{p}\|^2 \} \text{ with } \xi \to \infty$$

Repeat:

- Minimize $\mathcal{K}$ with respect to $\boldsymbol{x}$ (unconstrained).
- Minimize $\mathcal{K}$ with respect to $\boldsymbol{p}$ (easy): $p_i = \max(x_i, 0)$.

Special case: linear programming

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \ \boldsymbol{c}^T\boldsymbol{x} \ \text{ s.t. } \begin{cases} \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ \boldsymbol{x} \geq \mathbf{0} \end{cases}$$

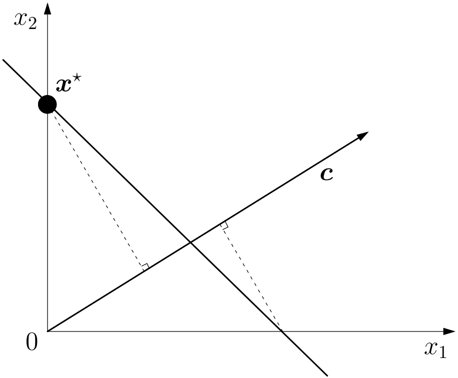
- $\boldsymbol{c} \in \mathbb{R}^n$ with $\boldsymbol{c} \geq \mathbf{0}_n$.
- $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ with $m \leq n$.

Example:

$$\min_{\boldsymbol{x}\in\mathbb{R}^4} \ x_1 + 2x_2 + 3x_3 + 4x_4 \quad \text{s.t.} \quad \begin{cases} x_1 + x_2 + x_3 + x_4 & = & 1 \\ x_1 + x_3 - 3x_4 & = & \frac{1}{2} \\ \boldsymbol{x} & \geq & \mathbf{0}. \end{cases}$$

- Rewrite the problem in matrix form.
- The solution $\boldsymbol{x}$ has two zero coordinates.

Interpretation in terms of orthogonal projections



The solution $x^\star$ has at least ($n - m$) zero coordinates.

Case $n = 2$, $m = 1$: solution has at least one 0 coordinate.

Case $n = 3$, $m = 1$: solution has at least two 0 coordinates.

Case $n = 3$, $m = 2$: solution has at least one 0 coordinate.

Principle of the simplex algorithm

1. Decompose $\boldsymbol{A} = [\boldsymbol{A}_B, \boldsymbol{A}_N]$ with $\boldsymbol{A}_B$ of size $m \times m$, and $\boldsymbol{A}_N$ of size $m \times (n-m)$.

2. Set $\boldsymbol{x}^B = \begin{bmatrix} \mathbf{x}_B \\ \mathbf{0}_{n-m} \end{bmatrix}$ with $\mathbf{x}_B = \boldsymbol{A}_B^{-1}\boldsymbol{b}$ ([1]).

3. $\boldsymbol{x}^B$ is a feasible point if $\mathbf{x}_B \geq \mathbf{0}_m$. The objective function is equal to $f(\boldsymbol{x}^B) = \boldsymbol{c}_B^T \boldsymbol{A}_B^{-1}\boldsymbol{b}$.

4. The simplex algorithm searches for the best decomposition $\boldsymbol{A} = [\boldsymbol{A}_B, \boldsymbol{A}_N]$. The subset B is updated at each iteration in such a way that $f(\boldsymbol{x}^B)$ is decreasing.

---

[1] assuming that the rank of $\boldsymbol{A}$ equals m and that $\boldsymbol{A}_B$ is invertible

Special case: quadratic programming (QP)

$$\min_{\boldsymbol{x} \in \mathbb{R}^n} \quad \mathcal{Q}(\boldsymbol{x}) = \frac{1}{2} \boldsymbol{x}^T \boldsymbol{H} \boldsymbol{x} + \boldsymbol{g}^T \boldsymbol{x} \quad \text{s.t.} \quad \begin{cases} \boldsymbol{a}_i^T \boldsymbol{x} = b_i, \ i \in \mathcal{E} \\ \boldsymbol{a}_i^T \boldsymbol{x} \geq b_i, \ i \in \mathcal{I} \end{cases}$$

- $\boldsymbol{H}$ is symmetric.
- $\boldsymbol{H}$ is symmetric positive semedefinite ($\boldsymbol{x}^T \boldsymbol{H} \boldsymbol{x} \geq 0, \ \forall \boldsymbol{x}$)
  $\Rightarrow$ there exists a minimizer.
- $\boldsymbol{H}$ is symmetric positive definite ($\boldsymbol{x}^T \boldsymbol{H} \boldsymbol{x} > 0, \ \forall \boldsymbol{x} \neq \mathbf{0}$)
  $\Rightarrow$ there exists a unique minimizer.

QP with equality constraints

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \quad \mathcal{Q}(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{\ell}^T\boldsymbol{x} \quad \text{s.t.} \;\; \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

1. Assume([2]) that $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ has rank m. Let $\boldsymbol{A} = [\boldsymbol{A}_1, \boldsymbol{A}_2]$ with $\boldsymbol{A}_1 \in \mathbb{R}^{m\times n}$ invertible, and let $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{bmatrix}$. $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ rereads $\boldsymbol{A}_1\boldsymbol{x}_1 + \boldsymbol{A}_2\boldsymbol{x}_2 = \boldsymbol{b}$, so $\boldsymbol{x}_1 = \boldsymbol{A}_1^{-1}(\boldsymbol{b} - \boldsymbol{A}_2\boldsymbol{x}_2)$.

   $\Rightarrow$ Solve unconstrained problem $\displaystyle\min_{\boldsymbol{x}_2} \;\; \mathcal{Q}(\boldsymbol{A}_1^{-1}(\boldsymbol{b} - \boldsymbol{A}_2\boldsymbol{x}_2), \boldsymbol{x}_2)$.

2. KKT conditions with $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = \dfrac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{\ell}^T\boldsymbol{x} - \boldsymbol{\lambda}^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})$:

   Solve system

$$\begin{cases} \boldsymbol{H}\boldsymbol{x} + \boldsymbol{\ell} - \boldsymbol{A}^T\boldsymbol{\lambda} = \mathbf{0} \\ \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \end{cases} \iff \begin{bmatrix} \boldsymbol{H} & -\boldsymbol{A}^T \\ \boldsymbol{A} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} -\boldsymbol{\ell} \\ \boldsymbol{b} \end{bmatrix}$$

[2] without loss of generality

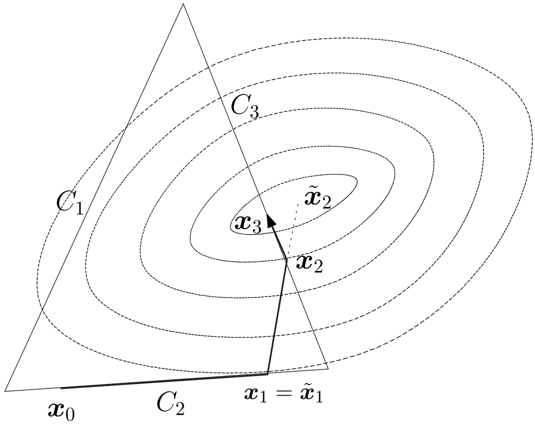
QP with inequality constraints

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \quad \mathcal{Q}(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{\ell}^T\boldsymbol{x} \quad \text{s.t. } \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}$$

Principle of the active-set algorithm.

1. Solve a sequence of QP problems with equality constraints.
2. Use a mechanism to add or remove active constraints.

The active set $\mathcal{A}(\boldsymbol{x}) = \{i : \boldsymbol{a}_i^T\boldsymbol{x} = b_i\}$ is defined as the set of active constraints at $\boldsymbol{x}$.



From a feasible point $\boldsymbol{x}_k$:

1. Solve QP with equality constraints on $\mathcal{A}(\boldsymbol{x}_k)$ only $\Rightarrow \tilde{\boldsymbol{x}}_{k+1}$.
2. If $\tilde{\boldsymbol{x}}_{k+1}$ is feasible, set $\boldsymbol{x}_{k+1} \leftarrow \tilde{\boldsymbol{x}}_{k+1}$. Compute the Lagrange multipliers $\{\lambda_i,\ i \in \mathcal{A}(\boldsymbol{x}_k)\}$. If some $\lambda_i < 0$, set $\mathcal{A}(\boldsymbol{x}_{k+1}) \leftarrow \mathcal{A}(\boldsymbol{x}_k) \backslash \{\arg\min_{i \in \mathcal{A}(\boldsymbol{x}_k)} \lambda_i\}$ and go back to step 1. Else, STOP.
3. Otherwise, set $\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k + \alpha_k(\tilde{\boldsymbol{x}}_{k+1} - \boldsymbol{x}_k)$ with the maximum possible value of $\alpha_k \in]0, 1[$. Compute $\mathcal{A}(\boldsymbol{x}_{k+1})$ and go back to step 1.

Step 1

QP: $\min_{\boldsymbol{x}\in\mathbb{R}^n} \quad \mathcal{Q}(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{\ell}^T\boldsymbol{x} \quad \text{s.t.} \;\; \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}$

QP with equality constraints: here, the inequality constraints are removed.

Define $\boldsymbol{A}_{\mathcal{A}} = \boldsymbol{A}(\mathcal{A}, :)$ and $\delta = \boldsymbol{x} - \boldsymbol{x}_k$ the descent direction.

$$\boxed{\min_{\boldsymbol{x}\in\mathbb{R}^n} \quad \mathcal{Q}(\boldsymbol{x}) \quad \text{s.t.} \;\; \boldsymbol{A}_{\mathcal{A}}\boldsymbol{x} = \boldsymbol{b}_{\mathcal{A}}}$$

$$\Leftrightarrow \quad \min_{\boldsymbol{\delta}\in\mathbb{R}^n} \quad \mathcal{Q}(\boldsymbol{x}_k + \boldsymbol{\delta}) = \mathcal{Q}(\boldsymbol{x}_k) + \mathcal{Q}(\boldsymbol{\delta}) + \boldsymbol{x}_k^T\boldsymbol{H}^T\boldsymbol{\delta} \quad \text{s.t.} \;\; \boldsymbol{A}_{\mathcal{A}}\boldsymbol{\delta} = \mathbf{0}$$

$$\Leftrightarrow \quad \min_{\boldsymbol{\delta}\in\mathbb{R}^n} \quad \frac{1}{2}\boldsymbol{\delta}^t\boldsymbol{H}\boldsymbol{\delta} + (\boldsymbol{H}\boldsymbol{x}_k + \boldsymbol{\ell})^T\boldsymbol{\delta} \quad \text{s.t.} \;\; \boldsymbol{A}_{\mathcal{A}}\boldsymbol{\delta} = \mathbf{0}.$$

Step 2: check KKT conditions for QP

Lagrangian of $\min_{\boldsymbol{x}\in\mathbb{R}^n} \mathcal{Q}(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{\ell}^T\boldsymbol{x} \quad \text{s.t. } \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}$:

$$\mathcal{L}(\boldsymbol{x},\boldsymbol{\lambda}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{\ell}^T\boldsymbol{x} - \boldsymbol{\lambda}^T\boldsymbol{A}\boldsymbol{x} + \boldsymbol{\lambda}^T\boldsymbol{b}$$

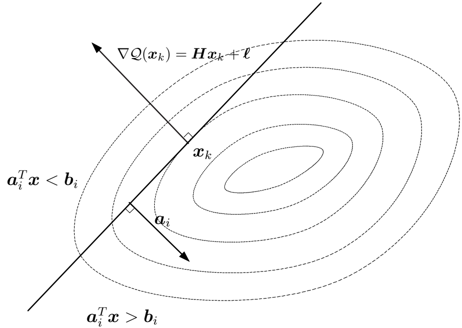
KKT conditions with $\boldsymbol{x} = \boldsymbol{x}_{k+1} \Rightarrow$

$$\left\{ \begin{aligned} \boldsymbol{H}\boldsymbol{x} + \boldsymbol{\ell} - \boldsymbol{A}^T\boldsymbol{\lambda} &= \mathbf{0} \\ \boldsymbol{A}_{\mathcal{A}}\boldsymbol{x} &= \boldsymbol{b}_{\mathcal{A}} \\ \boldsymbol{\lambda}_{\mathcal{A}} &\geq \mathbf{0} \\ \boldsymbol{A}_{\overline{\mathcal{A}}}\boldsymbol{x} &\geq \boldsymbol{b}_{\overline{\mathcal{A}}} \\ \boldsymbol{\lambda}_{\overline{\mathcal{A}}} &= \mathbf{0} \end{aligned} \right.$$

$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_{\mathcal{A}} \\ \boldsymbol{A}_{\overline{\mathcal{A}}} \end{bmatrix}$ implies that $\boldsymbol{A}^T\boldsymbol{\lambda} = \left[\boldsymbol{A}_{\mathcal{A}}^T, \boldsymbol{A}_{\overline{\mathcal{A}}}^T\right] \begin{bmatrix} \boldsymbol{\lambda}_{\mathcal{A}} \\ \mathbf{0} \end{bmatrix} = \boldsymbol{A}_{\mathcal{A}}^T\boldsymbol{\lambda}_{\mathcal{A}}$.

So, the KKT condition becomes $\boxed{\boldsymbol{\lambda}_{\mathcal{A}} = \left[\boldsymbol{A}_{\mathcal{A}}^T\right]^{\dagger}(\boldsymbol{H}\boldsymbol{x} + \boldsymbol{\ell}) \geq \mathbf{0}}$

Case of an active-set with one constraint

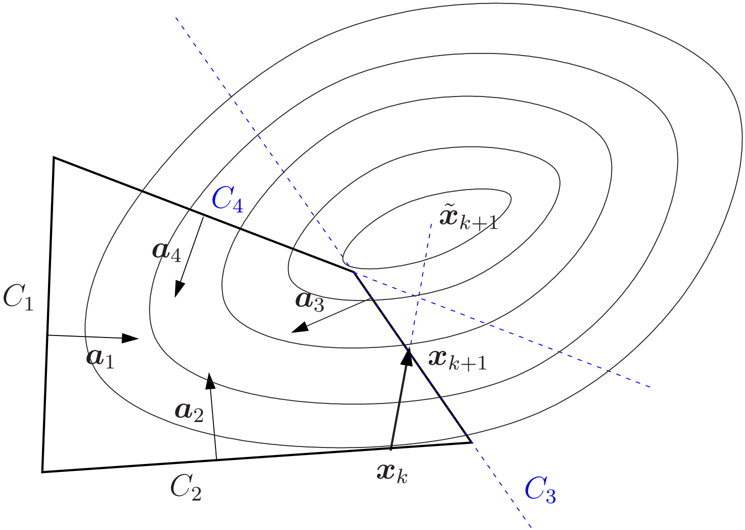


When $\mathcal{A} = \{i\}$ contains only one constraint, $\boldsymbol{a}_i$ is colinear with $\nabla\mathcal{Q}(\boldsymbol{x}_k) = \boldsymbol{H}\boldsymbol{x}_k + \boldsymbol{\ell}$, and

$$\lambda_i = \left[\boldsymbol{A}_{\mathcal{A}}^T\right]^\dagger (\boldsymbol{H}\boldsymbol{x}_k + \boldsymbol{\ell}) = [\boldsymbol{a}_i]^\dagger (\boldsymbol{H}\boldsymbol{x}_k + \boldsymbol{\ell}) = \frac{1}{\|\boldsymbol{a}_i\|^2}\, \boldsymbol{a}_i^T(\boldsymbol{H}\boldsymbol{x}_k + \boldsymbol{\ell}).$$

$\lambda_i < 0$ means that the i-th constraint may be removed

Step 3: illustration



Step 3: compute $\boldsymbol{x}_k + \alpha_k(\tilde{\boldsymbol{x}}_{k+1} - \boldsymbol{x}_k)$

It is easy to check that $\boldsymbol{x}_k + \alpha(\tilde{\boldsymbol{x}}_{k+1} - \boldsymbol{x}_k)$ is feasible when

$$\forall i \notin \mathcal{A}(\boldsymbol{x}_k) \;\text{ such that }\; \boldsymbol{a}_i^T(\tilde{\boldsymbol{x}}_{k+1} - \boldsymbol{x}_k) < 0,\ \alpha \leq \frac{\boldsymbol{b}_i - \boldsymbol{a}_i^T \boldsymbol{x}_k}{\boldsymbol{a}_i^T(\tilde{\boldsymbol{x}}_{k+1} - \boldsymbol{x}_k)}$$

So,

$$\boxed{\alpha_k = \min_{\substack{i \notin \mathcal{A}(\boldsymbol{x}_k) \\ \boldsymbol{a}_i^T(\tilde{\boldsymbol{x}}_{k+1} - \boldsymbol{x}_k) < 0}} \frac{\boldsymbol{b}_i - \boldsymbol{a}_i^T \boldsymbol{x}_k}{\boldsymbol{a}_i^T(\tilde{\boldsymbol{x}}_{k+1} - \boldsymbol{x}_k)}}$$

Sequential Quadratic Programming (SQP)

Solve a constrained problem $\min_{\boldsymbol{x}\in\mathbb{R}^n} \; f(\boldsymbol{x}) \quad \text{s.t.} \quad \begin{cases} g_1(\boldsymbol{x}) \geq 0 \\ \vdots \\ g_p(\boldsymbol{x}) \geq 0 \end{cases}$

Repeat:

1. Replace *f* by its quadratic approximation around $\boldsymbol{x}_k$:

$$f(\boldsymbol{x}_k + \boldsymbol{\delta}) \approx \mathcal{Q}_k(\boldsymbol{\delta}) = f(\boldsymbol{x}_k) + \nabla f(\boldsymbol{x}_k)^T \boldsymbol{\delta} + \frac{1}{2}\boldsymbol{\delta}^T \nabla^2 \mathcal{L}(\boldsymbol{x}_k, \boldsymbol{\lambda}_k)\boldsymbol{\delta}$$

2. Replace g_j by their linear approximation around $\boldsymbol{x}_k$:

$$g_j(\boldsymbol{x}_k + \boldsymbol{\delta}) \approx g_j(\boldsymbol{x}_k) + \nabla g_j(\boldsymbol{x}_k)^T \boldsymbol{\delta}$$

3. Solve the quadratic problem

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \; \mathcal{Q}_k(\boldsymbol{\delta}) \quad \text{s.t.} \quad \begin{cases} g_1(\boldsymbol{x}_k) + \nabla g_j(\boldsymbol{x}_k)^T \boldsymbol{\delta} \geq 0 \\ \vdots \\ g_p(\boldsymbol{x}_k) + \nabla g_p(\boldsymbol{x}_k)^T \boldsymbol{\delta} \geq 0 \end{cases}$$

in a “trust region”, *i.e.,* a neighborhood of $\boldsymbol{x}_k$.

5. Global optimization (recall definitions pp. 17-18)

Goal of global minimization

- Local algorithms: the numerical solution $\hat{x}$ depends on the initial solution x_0.
- Global optimization:
  - less sensitive to initialization.
  - escape from "poor valleys".
  - often relies on the simulation of random variables: stochastic algorithms.

In practice

- First try local solvers with different initial solutions!
- Choose global optimization algorithms when the local optimization output strongly depends on the initial solution.
- Never choose global optimization algorithms when the problem is unimodal (especially for convex problems, including quadratic problems).
- Expectations: find a local minimizer which is not too “poor”. Generally, global algorithms do not guarantee to find the global minimizer in a finite number of iterations.

Definitions

An algorithm is deterministic if the rules do not depend on random draws. For a given input $\boldsymbol{x}_0$, the algorithm will always produce the same iterates $\boldsymbol{x}_k$, and the same output $\hat{\boldsymbol{x}}$.

An algorithm is stochastic if random variables are generated during the iterations. Run the algotihm twice with the same input $\boldsymbol{x}_0$: the algorithm will not produce the same iterates $\boldsymbol{x}_k$, nor the same output $\hat{\boldsymbol{x}}$.

- Idea: explore different valleys where $f(\boldsymbol{x})$ is low as effectively as possible.
- Iterates are realizations of a random process.
- Define a probalility density function.

Choice of a global optimization solver

Depends on:

- the size of the problem (number of variables).
- the number of local valleys.
- the time available for numerical computation.

By increasing level of complexity and computation time:

1. Random exploration with independent draws.
2. Interval optimization (deterministic).
3. Particle Swarm Optimization (PSO), Genetic Algorithm (GA).
4. Simulated Annealing.

Independent drawing of random variables

- Generate random points $\boldsymbol{x}_k \in \mathcal{D}$.
- Evaluate $f(\boldsymbol{x}_k)$.
- Select the point corresponding to the lowest value of $f(\boldsymbol{x}_k)$.

$\hat{f} = +\infty$.

For $k = 1, \ldots, K$,

  Randomly generate $\boldsymbol{x}_k \in \mathcal{D}$

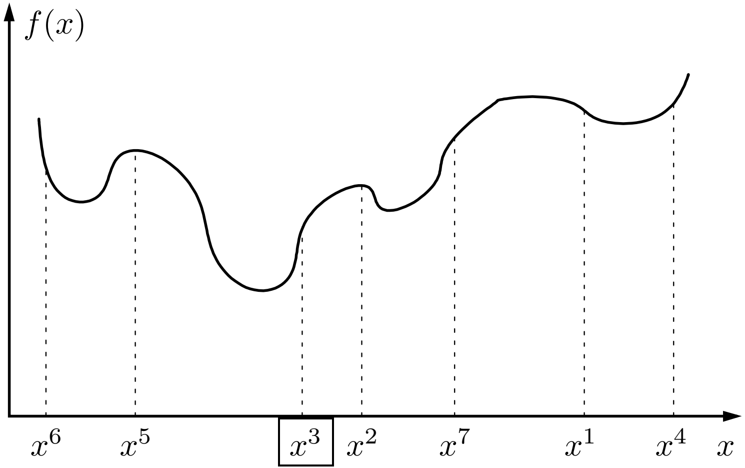
  If $f(\boldsymbol{x}_k) < \hat{f}$,

    $\hat{\boldsymbol{x}} = \boldsymbol{x}_k$ and $\hat{f} = f(\boldsymbol{x}_k)$

  End

End

Independent drawing of random variables



Hybrid approach: independent drawing + local optimization

$\hat{f} = +\infty$.

For $k = 1, \ldots, K$,

  Randomly generate $\boldsymbol{x}_k \in \mathcal{D}$

  Compute $\boldsymbol{y}_k = \text{Local\_Optim}(f; \boldsymbol{x}_k)$

  If $f(\boldsymbol{y}_k) < \hat{f}$,

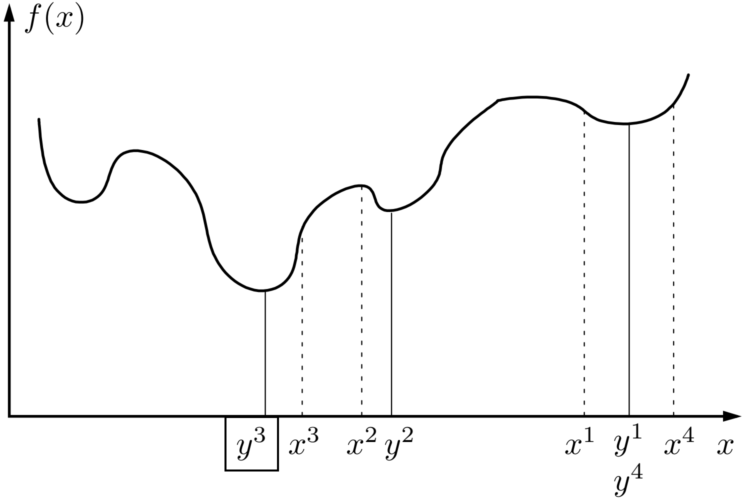
    $\hat{\boldsymbol{x}} = \boldsymbol{y}_k$ and $\hat{f} = f(\boldsymbol{y}_k)$

  End

End

Note: $\boldsymbol{y}_k$ is the a local minimizer in the valley containing $\boldsymbol{x}_k$.

Hybrid approach



Random drawing: which probability distribution?

$\hat{f} = +\infty$.

For $k = 1, \ldots, K$,

  Randomly generate $\boldsymbol{x}_k \in \mathcal{D}$

  If $f(\boldsymbol{x}_k) < \hat{f}$,

    $\hat{\boldsymbol{x}} = \boldsymbol{x}_k$ and $\hat{f} = f(\boldsymbol{x}_k)$

  End

End For

- Effectiveness strongly depends on the choice of the probability distribution.
- Uniform distribution on $\mathcal{D}$? (easy to simulate).
- More specific distribution if prior knowledge is available on the global minimizer $x^\star$?

Interval optimization

- A real number is represented by an interval, *e.g.,* π is replaced by the interval [3.14159, 3.14160].
- In 2D, an interval is a rectangle $[x_1^-, x_1^+] \times [x_2^-, x_2^+]$ with two lower bounds and two upper bounds.
- In nD, an interval is a cube $[x_1^-, x_1^+] \times [x_2^-, x_2^+] \times \cdots \times [x_n^-, x_n^+]$ of $\mathbb{R}^n$.
- Evaluate the objective function on an interval, that is, evaluate the image of an interval by function f. Examples:

$$
\begin{aligned}
[-2, 3] + [5, 7] &= [3, 10] \\
[-3, 2] * [-3, 2] &= [-6, 9] \\
[-3, 2]^2 &= [0, 9] \\
\exp([-2, 3]) &= [\exp(-2), \exp(3)] \\
\sin([0, 2\pi/3]) &= [0, 1] \\
[-3, 4] - [-3, 4] &= [-7, 7] \\
\ln([-2, -1]) &= \emptyset
\end{aligned}
$$

Evaluation of a function on an interval

- Decompose the function as a combination of simple fonctions, and evaluate each simple function.

- Example with $f(x) = x^2 + 2x + 4$:

$$\begin{aligned} f([-3, 4]) &= [-3, 4]^2 + 2 * [-3, 4] + [4, 4] \\ &= [0, 16] + [-6, 8] + [4, 4] = [-2, 28]. \end{aligned}$$

  means that $\forall x \in [-3, 4]$, $f(x) \in [-2, 28]$.

- Remark: the bounds -2 and 28 are not necessarily reached!

- Interval bisection: cut the interval in 2 sub-intervals, $[-3, 0.5]$ and $[0.5, 4]$ and evaluate each sub-interval.

Bisection: 2D example

We evaluate $f([x_1^-, x_1^+] \times [x_2^-, x_2^+]) = [-20, 200]$.
Then, the rectangle is split into 4 rectangles:

- $\mathcal{P}_{11} = \left[x_1^-, \frac{x_1^- + x_1^+}{2}\right] \times \left[x_2^-, \frac{x_2^- + x_2^+}{2}\right]$,
- $\mathcal{P}_{12} = \left[x_1^-, \frac{x_1^- + x_1^+}{2}\right] \times \left[\frac{x_2^- + x_2^+}{2}, x_2^+\right]$,
- $\mathcal{P}_{21} = \left[\frac{x_1^- + x_1^+}{2}, x_1^+\right] \times \left[x_2^-, \frac{x_2^- + x_2^+}{2}\right]$;
- $\mathcal{P}_{22} = \left[\frac{x_1^- + x_1^+}{2}, x_1^+\right] \times \left[\frac{x_2^- + x_2^+}{2}, x_2^+\right]$

and f is re-evaluated on each rectangle:

	$\left[x_2^-, \frac{x_2^- + x_2^+}{2}\right]$	$\left[\frac{x_2^- + x_2^+}{2}, x_2^+\right]$
$\left[x_1^-, \frac{x_1^- + x_1^+}{2}\right]$	$f(\mathcal{P}_{11}) = [32, 100]$	$f(\mathcal{P}_{12}) = [-14, 28]$
$\left[\frac{x_1^- + x_1^+}{2}, x_1^+\right]$	$f(\mathcal{P}_{21}) = [-10, 2]$	$f(\mathcal{P}_{22}) = [2, 3]$

Bisection: 2D example

	$\left[x_2^-, \frac{x_2^-+x_2^+}{2}\right]$	$\left[\frac{x_2^-+x_2^+}{2}, x_2^+\right]$
$\left[x_1^-, \frac{x_1^-+x_1^+}{2}\right]$	X	$f(\mathcal{P}_{12}) = [-14, 28]$
$\left[\frac{x_1^-+x_1^+}{2}, x_1^+\right]$	$f(\mathcal{P}_{21}) = [-10, 2]$	$f(\mathcal{P}_{22}) = [2, 3]$

Conclusion:

- $\mathcal{P}_{11}$ cannot contain a global minimizer.
- $\mathcal{P}_{12}$, $\mathcal{P}_{21}$ and $\mathcal{P}_{22}$ are candidates to contain a global minimizer.
- Make bisection of $\mathcal{P}_{12}$, $\mathcal{P}_{21}$ and $\mathcal{P}_{22}$ and re-evaluate f on the new (smaller) intervals.

Interval optimization

Principle.

- Handle of list of candidate intervals $\{\mathcal{P}_i\}$.
- Evaluate $[f_i^-, f_i^+] = f(\mathcal{P}_i)$.
- Update $\bar{f} = \min_i f_i^+$.
- Simplification of the list: remove $\mathcal{P}_i$ from the list when $f_i^- > \bar{f}$.

Branch and bound philosophy.

- All intervals are tested (bisection + evaluation of f), but some which cannot lead to the solution are deleted.
- The algorithm works for problems of dimension $n \leq 100$.
- The algorithm output is a list of candidate intervals and an upper bound $\bar{f}$ of $\min f(\boldsymbol{x})$.
- The global minimizers are guaranteed to be in one of the intervals in the list.

Evolutionary algorithms

- Particle Swarm Optimization.
- Genetic Algorithm.

Principle.

Evolutionary algorithms make an exploration of the feasible domain $\mathcal{D}$ using several variables in parallel.

Local optimization algorithm.

- 1 variable $\boldsymbol{x}$.
- Iteration k: $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + t_k \boldsymbol{d}_k$.

Evolutionary algorithms.

- N variables $\boldsymbol{x}^1, \dots, \boldsymbol{x}^N \in \mathbb{R}^n$.
- Iteration k: jointly update $\{\boldsymbol{x}_{k+1}^1, \dots, \boldsymbol{x}_{k+1}^N\}$ from $\{\boldsymbol{x}_k^1, \dots, \boldsymbol{x}_k^N\}$.

Evolutionary algorithms

Find a strategy with several variables (individuals, agents, particles) to explore “interesting valleys”. The number of variables N is fixed.

The update of the $\boldsymbol{x}_k^i$ depends on:

1. a local descent move (knowledge of $\nabla f(\boldsymbol{x}_k^i)$).
2. the introduction of random moves to avoid convergence towards a local minimizer.
3. the objective values for other variables $\boldsymbol{x}_k^j$ ($j \neq i$).

Heuristic algorithms, no guarantee of convergence towards a global minimizer.

Simple to implement, several parameters to tune, can work for large dimension n.

Genetic algorithms

Artificial intelligence: simulate the evolution of a population of N individuals according to Darwin's theory:

1. Natural selection: the most adapted individuals tend to live longer and to reproduce more easily.
2. Evolution: some novel species randomly appear.

Sketch of algorithm:

Set $k \leftarrow 0$.
Initialize $\boldsymbol{x}_0^1, \boldsymbol{x}_0^2, \ldots, \boldsymbol{x}_0^N \in \mathbb{R}^n$
For $k = 0, \ldots, K$,
    Select $N/2$ variables (the others are removed);
    Reproduction (mutations, crossbreed);
    Evaluate children;
    Replace some parents by their children.
End For

Genetic algorithms

Initialization.

x_0^i: uniform distribution on $\mathcal{D}$?

Selection:

1. *$N/2$-elitism*: keep the $N/2$ best variables (corresponding to the lowest values of $f(x)$).
2. *Tournament selection*:

   Repeat:
   - Choose q variables randomly;
   - Select the best as parent.

   until the number of parents is equal to $N/2$.

Remark.

- $N/2$-elitism may induce premature convergence towards a local minimizer.
- Tournament selection will tend to explore unlikely valleys.

Genetic algorithms

Crossbreed between variables x_i^k and x_j^k:

Generate a new position

$$x = \alpha_{ij} x_k^i + (1 - \alpha_{ij}) x_k^j$$

where α_{ij} is uniformly drawn on the line joining x_i^k and x_j^k. For instance, choose $\alpha_{ij} \in [-\varepsilon,\ 1 + \varepsilon]$.

Mutation of x_k^i:

$$x = x_k^i + \varepsilon_i \text{ with } \varepsilon_i \sim \mathcal{N}(\mathbf{0}, \sigma^2 I)$$

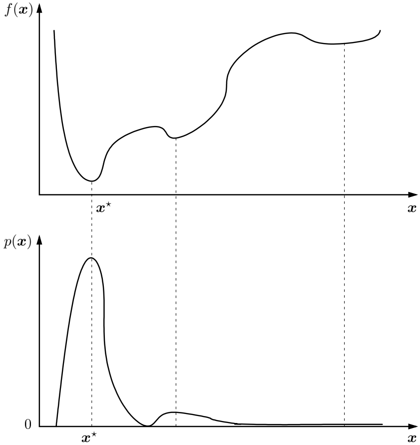
Overall parameters.

- N, σ, q (tournament selection), initial conditions.
- Tradeoff exploration (low q, large σ, large N) *vs* local convergence.

Monte Carlo based methods (simulated annealing...)

Minimize $f(\boldsymbol{x}) \iff$ Maximize $p(\boldsymbol{x}) = \frac{1}{Z}\exp(-f(\boldsymbol{x}))$.

$Z = \int_{\mathcal{D}} \exp(-f(\boldsymbol{x}))\, d\boldsymbol{x}$ is chosen to have a probability distribution.



Monte Carlo based methods

Minimize $f(\boldsymbol{x}) \iff$ Maximize $p(\boldsymbol{x}) = \frac{1}{Z} \exp(-f(\boldsymbol{x}))$.

- Draw samples $\boldsymbol{x}_k$ of the probability distribution $p(\boldsymbol{x})$.
- For large dimension problems, and when Z is unknown, we can make use of Markov chains.
- MCMC methods (*Markov chain Monte Carlo*) create a long Markov chain $\{\ldots, \boldsymbol{X}^k, \ldots\}$ whose samples are asymptotically distributed according to pdf $p(\boldsymbol{x})$.

Bibliography

J. Nocedal and J. S. Wright,
Numerical optimization, 2nd edition, Springer Verlag, New York, Jul. 2006.

R. Fletcher,
Practical Methods of Optimization, 2nd edition, Wiley, 2000.

A. Björck,
Numerical Methods for Least Squares Problems, *Society for Industrial and Applied Mathematics*, Philadephia, Apr. 1996.

R. Horst, P. M. Pardalos, and V. T. Nguyen,
Introduction to Global Optimization, 2nd edition, Springer, Dec. 2000.

P. M. Pardalos, D.-Z. Du, and R. L. Graham,
Handbook of Combinatorial Optimization, Springer Verlag, New York, 2013.

Acknowledgements

Many thanks to:

- Pierre Riedinger (University of Lorraine).
- Said Moussaoui (Ecole Centrale Nantes).
- Jérôme Idier (CNRS).