Numerical Optimization: Introduction and gradient-based methods

Master 2 Recherche LRI

Apprentissage Statistique et Optimisation

Anne Auger

Inria Saclay-Ile-de-France

November 2011

http://tao.lri.fr/tiki-index.php?page=Courses

1. Numerical optimization
   - A few examples
     - Data fitting, regression
     - In machine learning
     - Black-box optimization
   - Different notions of optimum
   - Typical difficulties in optimization
   - Deterministic vs stochastic - local versus global methods
2. Mathematical tools and optimality conditions
   - First order differentiability and gradients
   - Second order differentiability and hessian
   - Optimality Conditions for unconstrainted optimization
3. Gradient based optimization algorithms
   - Root finding methods (1-D optimization)
   - Relaxation algorithm
   - Descent methods
     - Gradient descent, Newton descent, BFGS
   - Trust regions methods

Numerical or continuous optimization

Unconstrained optimization

- Optimize a function where parameters to optimize are "continuous" (live in $\mathbb{R}^n$).

$$\min_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x})$$

n: dimension of the problem
corresponds to dimension of euclidian vector space $\mathbb{R}^n$

- Maximization vs Minimization

$$\text{Maximize } f = \text{Minimize } -f$$

Analytical functions

- Convex quadratic function:

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A\mathbf{x} + b^T\mathbf{x} + c \quad (1)$$

$$= \frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^T A(\mathbf{x} - \mathbf{x}_0) + c \quad (2)$$

where $A \in \mathbb{R}^{n\times n}$, symmetric positive definite, $b \in \mathbb{R}^n$, $c \in \mathbb{R}$.

Exercice

Express $\mathbf{x}_0$ in (2) as a function of A and b. Express the minimum of f. For $n = 2$, plot the level sets of a convex-quadratic function where level sets are defined as the sets

$$\mathcal{L}_c = \{\mathbf{x} \in \mathbb{R}^n | f(\mathbf{x}) = c\}.$$

Data fitting - Data calibration

Objective

- Given a sequence of data points $(\mathbf{x}_i, y_i) \in \mathbb{R}^p \times \mathbb{R}, i = 1, \ldots, N$, find a model "$y = f(\mathbf{x})$" that explains the data

  experimental measurements in biology, chemistry, ...
- In general, choice of a parametric model or family of functions $(f_\theta)_{\theta \in \mathbb{R}^n}$ use of expertise for choosing model or simple models only affordable (linear, quadratic)
- Try to find the parameter $\theta \in \mathbb{R}^n$ fitting best to the data

Fitting best to the data

Minimize the quadratic error:

$$\min_{\theta \in \mathbb{R}^n} \sum_{i=1}^{N} |f_\theta(\mathbf{x}_i) - y_i|^2$$

Optimization and machine learning

(Simple) Linear regression

Given a set of data (examples): $\{y_i, \underbrace{x_i^1, \ldots, x_i^p}_{\mathbf{x}_i^T}\}_{i=1\ldots N}$

$$\min_{\mathbf{w}\in\mathbb{R}^p, \beta\in\mathbb{R}} \underbrace{\sum_{i=1}^{N} |\mathbf{w}^T\mathbf{X}_i + \beta - y_i|^2}_{\|\tilde{\mathbf{X}}\mathbf{w}-\mathbf{y}\|^2}$$

same as data fitting with linear model, i.e. $f_{(\mathbf{w},\beta)}(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + \beta$, $\theta \in \mathbb{R}^p$

Optimization and Machine Learning (cont.)

Clustering

k-means

Given a set of observations $(\mathbf{x}_1, \ldots, \mathbf{x}_p)$ where each observation is a n-dimensional real vector, k-means clustering aims to partition the n observations into k sets ($k \leq p$), $S = \{S_1, S_2, \ldots, S_k\}$ so as to minimize the within-cluster sum of squares:

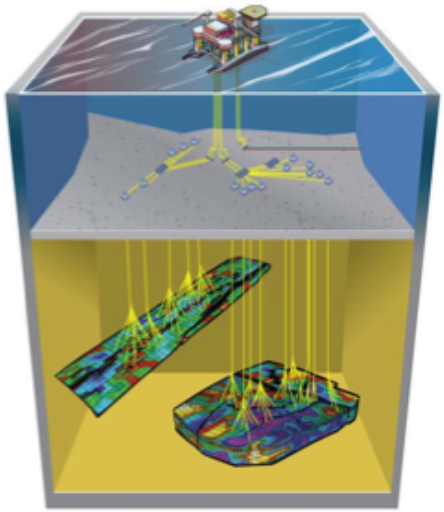
$$\text{minimize}_{\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_k} \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in S_i} \|\mathbf{x}_j - \boldsymbol{\mu}_i\|^2$$

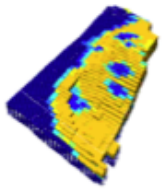
Black-box optimization

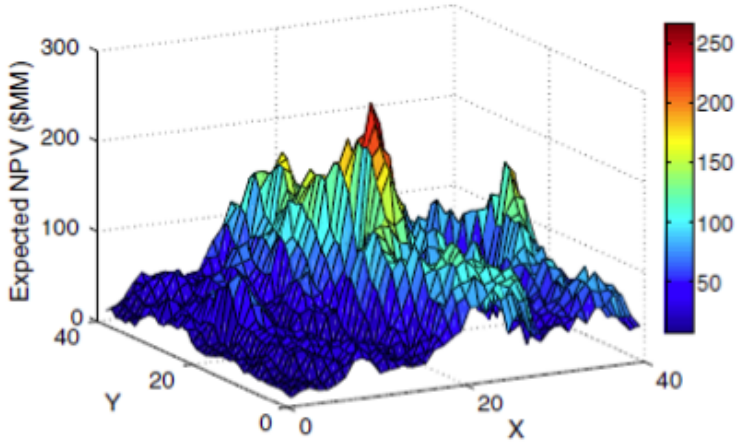
Optimization of well placement

A Real-World Problem in Petroleum Engineering

Well Placement Problem



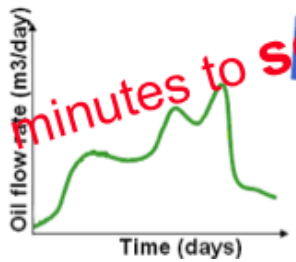




Onwunalu & Durlofsky (2010)

Fluid flow simulation

several minutes to several hours !!



Different notions of optimum

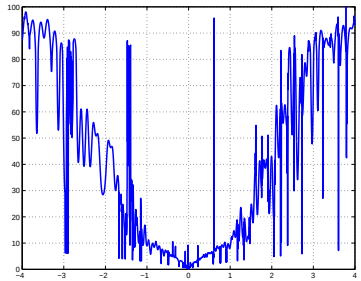
- local versus global minimum
  - local minimum $\mathbf{x}^\star$: for all $\mathbf{x}$ in a neighborhood of $\mathbf{x}^\star$, $f(\mathbf{x}) \geq f(\mathbf{x}^\star)$
  - global minimum: for all $\mathbf{x}$, $f(\mathbf{x}) \geq f(\mathbf{x}^\star)$

- essential infimum of a function:
  Given a measure μ on $\mathbb{R}^n$, the essential infimum of f is defined as
  $\operatorname{ess\,inf} f = \sup\{b \in \mathbb{R} : \mu(\{\mathbf{x} : f(\mathbf{x}) < b\}) = 0\}$

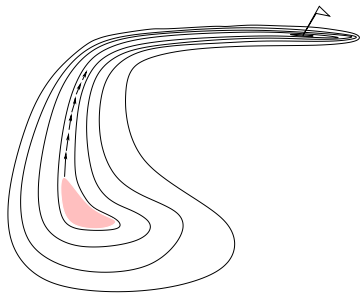
  important to keep in mind in the context of stochastic optimization algorithms

What Makes a Function Difficult to Solve?

- ruggedness
  non-smooth, discontinuous, multimodal, and/or noisy function
- dimensionality
  (considerably) larger than three
- non-separability
  dependencies between the objective variables
- ill-conditioning

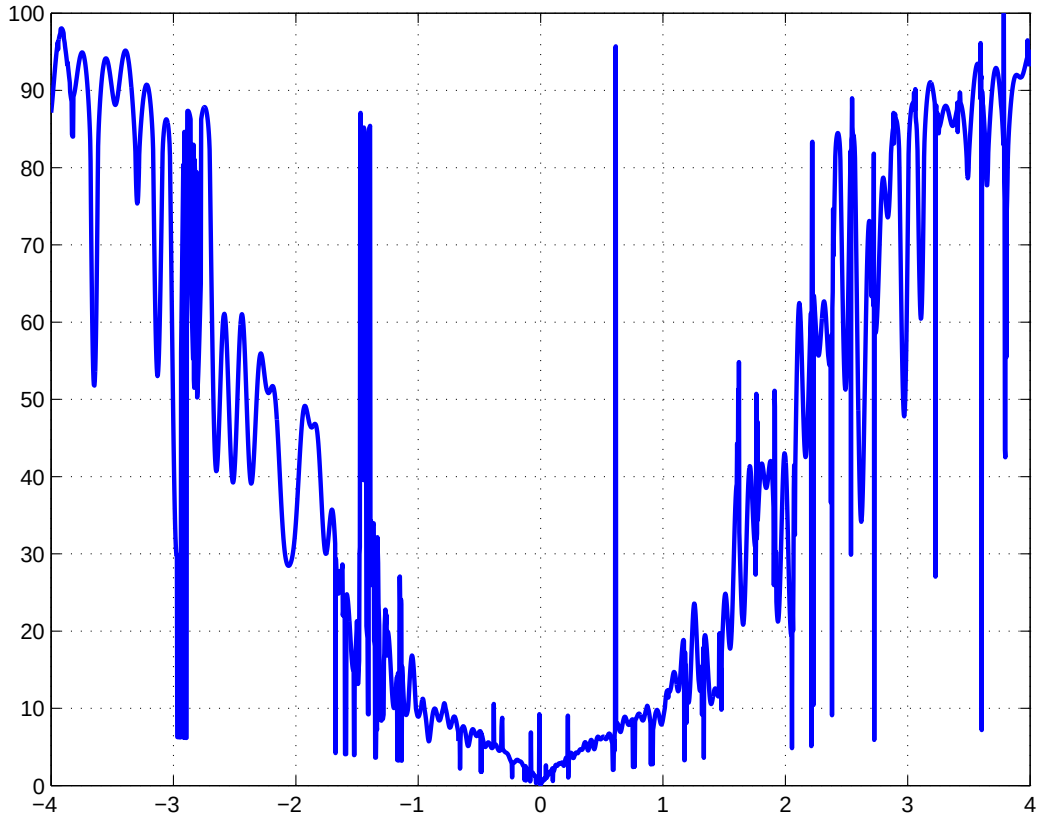


cut from 3-D example, solvable with an evolution strategy



a narrow ridge

What Makes a Function Difficult to Solve?



cut from 3-D example, solvable with an evolution strategy

Curse of Dimensionality

The term *Curse of dimensionality* (Richard Bellman) refers to problems caused by the rapid increase in volume associated with adding extra dimensions to a (mathematical) space.

Curse of Dimensionality

The term *Curse of dimensionality* (Richard Bellman) refers to problems caused by the rapid increase in volume associated with adding extra dimensions to a (mathematical) space.

Example: Consider placing 100 points onto a real interval, say $[0, 1]$. To get similar coverage, in terms of distance between adjacent points, of the 10-dimensional space $[0, 1]^{10}$ would require $100^{10} = 10^{20}$ points. A 100 points appear now as isolated points in a vast empty space.

Curse of Dimensionality

The term *Curse of dimensionality* (Richard Bellman) refers to problems caused by the rapid increase in volume associated with adding extra dimensions to a (mathematical) space.

Example: Consider placing 100 points onto a real interval, say $[0, 1]$. To get similar coverage, in terms of distance between adjacent points, of the 10-dimensional space $[0, 1]^{10}$ would require $100^{10} = 10^{20}$ points. A 100 points appear now as isolated points in a vast empty space.

Consequently, a search policy (e.g. exhaustive search) that is valuable in small dimensions might be useless in moderate or large dimensional search spaces.

Separable Problems

Definition (Separable Problem)

A function f is separable if

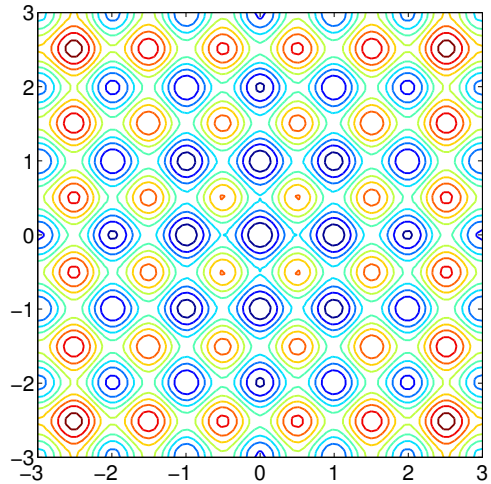
$$\arg\min_{(x_1,\ldots,x_n)} f(x_1,\ldots,x_n) = \left(\arg\min_{x_1} f(x_1,\ldots),\ldots,\arg\min_{x_n} f(\ldots,x_n)\right)$$

$\Rightarrow$ it follows that f can be optimized in a sequence of n independent 1-D optimization processes

Example: Additively decomposable functions

$$f(x_1,\ldots,x_n) = \sum_{i=1}^{n} f_i(x_i)$$

Rastrigin function



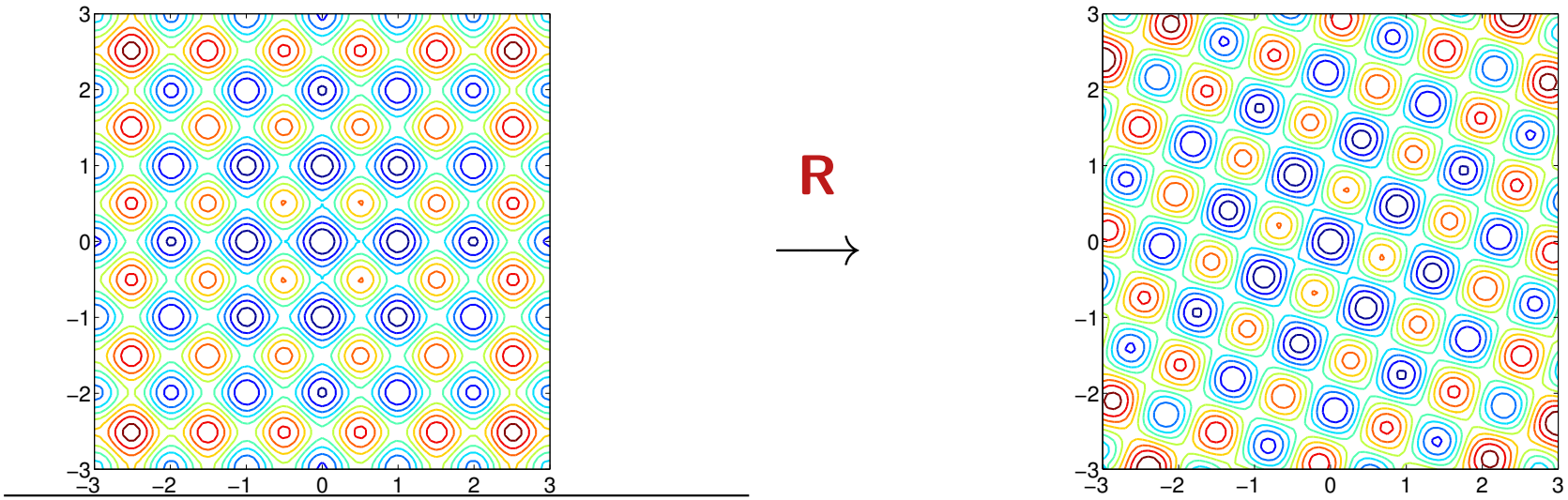
Non-Separable Problems

Building a non-separable problem from a separable one [1,2]

Rotating the coordinate system

- $f : \mathbf{x} \mapsto f(\mathbf{x})$ separable
- $f : \mathbf{x} \mapsto f(\mathbf{R}\mathbf{x})$ non-separable

$\mathbf{R}$ rotation matrix



[1] Hansen, Ostermeier, Gawelczyk (1995). On the adaptation of arbitrary normal mutation distributions in evolution strategies: The generating set adaptation. Sixth ICGA, pp. 57-64, Morgan Kaufmann

[2] Salomon (1996). "Reevaluating Genetic Algorithm Performance under Coordinate Rotation of Benchmark Functions; A survey of some theoretical and practical aspects of genetic algorithms." BioSystems, 39(3):263-278

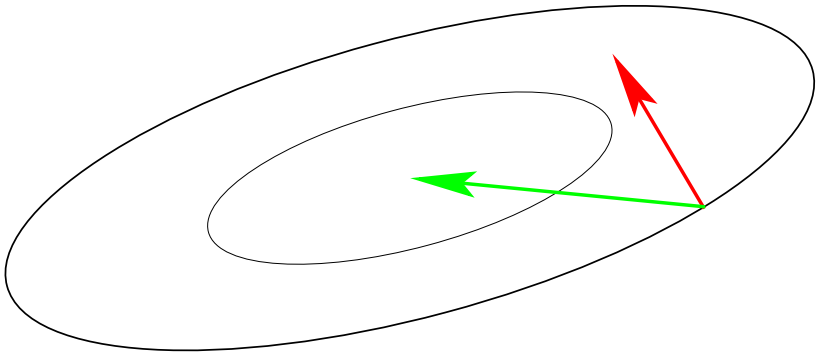
Ill-Conditioned Problems

Curvature of level sets

Consider the convex-quadratic function

$f(\mathbf{x}) = \frac{1}{2}(\mathbf{x} - \mathbf{x}^*)^T \mathbf{H} (\mathbf{x} - \mathbf{x}^*) = \frac{1}{2} \sum_i h_{i,i} x_i^2 + \frac{1}{2} \sum_{i \neq j} h_{i,j} x_i x_j$

$\mathbf{H}$ is Hessian matrix of f and symmetric positive definite



gradient direction $-f'(\mathbf{x})^{\mathrm{T}}$

Newton direction $-\mathbf{H}^{-1} f'(\mathbf{x})^{\mathrm{T}}$

Ill-conditioning means squeezed level sets (high curvature). Condition number equals nine here. Condition numbers up to 10^{10} are not unusual in real world problems.

If $\mathbf{H} \approx \mathbf{I}$ (small condition number of $\mathbf{H}$) first order information (e.g. the gradient) is sufficient. Otherwise second order information (estimation of $\mathbf{H}^{-1}$) is necessary.

What Makes a Function Difficult to Solve?

. . . and what can be done

The Problem	The Approach in ESs and continuous EDAs
Ruggedness	non-local policy, large sampling width (step-size) as large as possible while preserving a reasonable convergence speed
	stochastic, non-elitistic, population-based method
	recombination operator serves as repair mechanism

What Makes a Function Difficult to Solve?

. . . and what can be done

The Problem	The Approach in ESs and continuous EDAs
Ruggedness	non-local policy, large sampling width (step-size) as large as possible while preserving a reasonable convergence speed
	stochastic, non-elitistic, population-based method
	recombination operator serves as repair mechanism
Dimensionality, Non-Separability	exploiting the problem structure locality, neighborhood, encoding

What Makes a Function Difficult to Solve?

. . . and what can be done

The Problem	The Approach in ESs and continuous EDAs
Ruggedness	non-local policy, large sampling width (step-size) as large as possible while preserving a reasonable convergence speed
	stochastic, non-elitistic, population-based method
	recombination operator serves as repair mechanism
Dimensionality, Non-Separability	exploiting the problem structure locality, neighborhood, encoding
Ill-conditioning	second order approach changes the neighborhood metric

Different optimization methods

- Deterministic methods — local methods
  - Convex optimization methods - gradient based methods

    most often require to use gradients of functions

    converge to local optima, fast if function has the right assumptions (smooth enough)
  - deterministic methods that do not require convexity: simplex method, pattern search methods

Different optimization methods (cont.)

- Stochastic methods — global methods

  use of randomness to be able to escape local optima

  - pure random search, simulated annealing (SA)

    not reasonable methods in practice (too slow)

  - genetic algorithms

    not powerful for numerical optimization, originally introduced for binary search spaces $\{0,1\}^n$

  - evolution strategies

    powerful for numerical optimization

Remarks

Impossible to be exhaustive

Classification is not that binary

methods combining deterministic and stochastic do of course exist

1. Numerical optimization
   - A few examples
     - Data fitting, regression
     - In machine learning
     - Black-box optimization
   - Different notions of optimum
   - Typical difficulties in optimization
   - Deterministic vs stochastic - local versus global methods
2. Mathematical tools and optimality conditions
   - First order differentiability and gradients
   - Second order differentiability and hessian
   - Optimality Conditions for unconstrainted optimization
3. Gradient based optimization algorithms
   - Root finding methods (1-D optimization)
   - Relaxation algorithm
   - Descent methods
     - Gradient descent, Newton descent, BFGS
   - Trust regions methods

Differentiability

Generalization of derivability in 1-D

Given a normed vector space $(E, \|.\|)$ and complete (Banach space), consider $f : U \subset E \to \mathbb{R}$ with U open set of E.

- f is differentiable in $\mathbf{x} \in U$ if there exists a continuous linear form $Df_{\mathbf{x}}$ such that

$$f(\mathbf{x}+\mathbf{h}) = f(\mathbf{x}) + Df_{\mathbf{x}}(\mathbf{h}) + o(\|\mathbf{h}\|) \tag{3}$$

$Df_{\mathbf{x}}$ is the differential of f in $\mathbf{x}$

Exercice

Consider $E = \mathbb{R}^n$ with the scalar product $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T \mathbf{y}$. Let $\mathbf{a} \in \mathbb{R}^n$, show that

$$f(\mathbf{x}) = \langle \mathbf{a}, \mathbf{x} \rangle$$

is differentiable and compute its differential.

Gradient

If the norm $\|.\|$ comes from a scalar product, i.e. $\|\mathbf{x}\| = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$ (the Banach space E is then called an Hilbert space), the gradient of f in $\mathbf{x}$ denoted $\nabla f(\mathbf{x})$ is defined as the element of E such that

$$Df_{\mathbf{x}}(\mathbf{h}) = \langle \nabla f(\mathbf{x}), \mathbf{h} \rangle \tag{4}$$

Riesz representation Theorem

Taylor formula - order one

Replacing differential by (4) in (3) we obtain the Taylor formula:

$$f(\mathbf{x} + \mathbf{h}) = f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{h} \rangle + o(\|\mathbf{h}\|)$$

Gradient (cont)

Exercice

Compute the gradient of the function $f(\mathbf{x}) = \langle \mathbf{a}, \mathbf{x} \rangle$.

Exercice

Level sets are sets defined as

$$\mathcal{L}_c = \{\mathbf{x} \in \mathbb{R}^n | f(\mathbf{x}) = c\}.$$

Let $\mathbf{x}_0 \in \mathcal{L}_c \neq \emptyset$. Show that $\nabla f(\mathbf{x}_0)$ is orthogonal to the level set in $\mathbf{x}_0$.

Gradient

Examples

- if $f(\mathbf{x}) = \langle \mathbf{a}, \mathbf{x} \rangle$, $\nabla f(\mathbf{x}) = \mathbf{a}$
- in $\mathbb{R}^n$, if $f(\mathbf{x}) = \mathbf{x}^T A \mathbf{x}$, then $\nabla f(\mathbf{x}) = (A + A^T)\mathbf{x}$
- particular case if $f(\mathbf{x}) = \|\mathbf{x}\|^2$, then $\nabla f(\mathbf{x}) = 2\mathbf{x}$
- in $\mathbb{R}$, $\nabla f(\mathbf{x}) = f'(\mathbf{x})$
- in $(\mathbb{R}^n, \|.\|_2)$ where $\|\mathbf{x}\|_2 = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$ is the euclidian norm

$$\underbrace{\nabla f(\mathbf{x})}_{\text{“natural” gradient (Amari,2000)}} = \underbrace{\left(\frac{\partial f}{\partial x_1}(\mathbf{x}), \ldots, \frac{\partial f}{\partial x_n}(\mathbf{x}) \right)^T}_{\text{“vanilla” gradient}}$$

Attention: this equality will not hold for all norms. Has some importance in machine learning, see the natural gradient topic introduced by Amari.

Second order differentiability

- (first order) differential: gives a linear local approximation
- second order differential: gives a quadratic local approximation

Definition: second order differentiability

$f : U \subset E \to \mathbb{R}$ is differentiable at the second order in $\mathbf{x} \in U$ if it is differentiable in a neighborhood of $\mathbf{x}$ and if $u \mapsto Df_u$ is differentiable in $\mathbf{x}$

Second order differentiability (cont.)

Other definition

$f : U \subset E \to \mathbb{R}$ is differentiable at the second order in $\mathbf{x} \in U$ iff there exists a continuous linear application $Df_{\mathbf{x}}$ and a bilinear symmetric continuous application $D^2 f_{\mathbf{x}}$ such that

$$f(\mathbf{x}+\mathbf{h}) = f(\mathbf{x}) + Df_{\mathbf{x}}(\mathbf{h}) + \frac{1}{2} D^2 f_{\mathbf{x}}(\mathbf{h},\mathbf{h}) + o(\|\mathbf{h}\|^2)$$

In a Hilbert $(E, \langle\,\rangle)$

$$D^2 f_{\mathbf{x}}(\mathbf{h},\mathbf{h}) = \langle \nabla^2 f(\mathbf{x})(\mathbf{h}), \mathbf{h} \rangle$$

where $\nabla^2 f(\mathbf{x}) : E \to E$ is a symmetric continuous operator.

Hessian matrix

In $(\mathbb{R}^n, \langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T \mathbf{y})$, $\nabla^2 f(\mathbf{x})$ is represented by a symmetric matrix called the hessian matrix. It can be computed as

$$\nabla^2(f) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}.$$

Exercice

In $\mathbb{R}^n$, compute the hessian matrix of $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A \mathbf{x}$.

Optimality condition

First order Necessary condition

for 1-D optimization problems $f : \mathbb{R} \to \mathbb{R}$

Assume f is derivable

- $\mathbf{x}^\star$ is a local optimum $\Rightarrow f'(\mathbf{x}^\star) = 0$

not a necessary condition: consider $f(\mathbf{x}) = \mathbf{x}^3$
proof via Taylor formula: $f(\mathbf{x}^\star + \mathbf{h}) = f(\mathbf{x}^\star) + f'(\mathbf{x}^\star)\mathbf{h} + o(\|\mathbf{h}\|)$

- Points $\mathbf{y}$ such that $f'(\mathbf{y}) = 0$ are called critical or stationary points.

Generalization

If $f : E \mapsto R$ with $(E, \langle , \rangle)$ a Hilbert is differentiable

- $\mathbf{x}^\star$ is a local minimum of $f \Rightarrow \nabla f(\mathbf{x}^\star) = 0$.

proof via Taylor formula

Optimality conditions

Second order necessary and sufficient conditions

If f is twice continuously differentiable

- if $\mathbf{x}^\star$ is a local optimum, then $\nabla f(\mathbf{x}^\star) = 0$ and $\nabla^2 f(\mathbf{x}^\star)$ is positive semi-definite
- if $\nabla f(\mathbf{x}^\star) = 0$ and $\nabla^2 f(\mathbf{x}^\star)$ is positive definite, then there is an $\alpha > 0$ such that $f(\mathbf{x}) \geq f(\mathbf{x}^\star) + \alpha \|\mathbf{x} - \mathbf{x}^\star\|^2$

1. Numerical optimization
   - A few examples
     - Data fitting, regression
     - In machine learning
     - Black-box optimization
   - Different notions of optimum
   - Typical difficulties in optimization
   - Deterministic vs stochastic - local versus global methods
2. Mathematical tools and optimality conditions
   - First order differentiability and gradients
   - Second order differentiability and hessian
   - Optimality Conditions for unconstrainted optimization
3. Gradient based optimization algorithms
   - Root finding methods (1-D optimization)
   - Relaxation algorithm
   - Descent methods
     - Gradient descent, Newton descent, BFGS
   - Trust regions methods

Optimization algorithms

- Iterative procedures that generate a sequence $(\mathbf{x}_n)_{n\in\mathbb{N}}$ where at each time step n, $\mathbf{x}_n$ is the estimate of the optimum.

Convergence

- No exact result in general, however the algorithms aim at converging to optima (local or global), i.e.

$$\|\mathbf{x}_n - \mathbf{x}^\star\| \xrightarrow[n\to\infty]{} 0$$

convergence speed of the algorithm = how fast $\|\mathbf{x}_n - \mathbf{x}^\star\|$ goes to zero

- Equivalently, given a fixed precision ϵ, the algorithms aim at approaching the optimum $\mathbf{x}^\star$ with precision ϵ in finite time

running time = how many iterations is needed to reach the precision ϵ

Newton-Raphson method

Method to find the root of a function $f : \mathbb{R} \to \mathbb{R}$

- linear approximation of f: $f(x+h) = f(x) + hf'(x) + o(\|h\|)$
- at the first order, $f(x+h) = 0$ leads to $h = -\frac{f(x)}{f'(x)}$
- Algorithm: $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$
- quadratic convergence: $|x_{n+1} - x^\star| \leq \mu |x_n - x^\star|^2$

provided the initial point is close enough from the root $x^\star$ and there is no other critical point in a neighborhood of $x^\star$

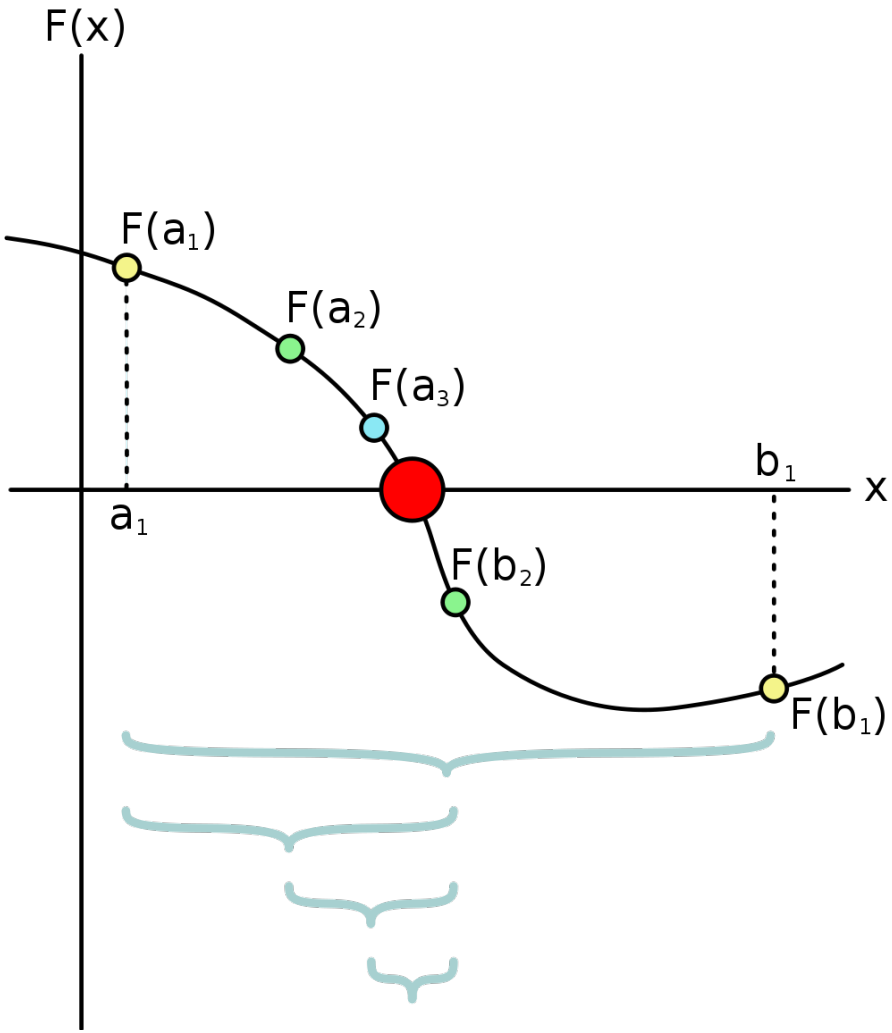
secant method: replaces the derivative computation its the finite difference approximation

Application to the minimization of f

- try to solve the equation $f'(x) = 0$
- algorithm: $x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}$

Bissection method

An other method to find the root of a 1-D function



Wikipedia image

- bisects an interval and then selects a subinterval in which a root must lie
- method guaranteed to converge if f continuous and the intial points bracket the solution

$$\|x_n - x^\star\| \leq \frac{|a_1 - b_1|}{2^n}$$

Relaxation algorithm

Naive approach for optimization of $f : \mathbb{R}^n \to \mathbb{R}$

Algorithm

- successive optimization w.r.t. each variable

1 choose an intial point $\mathbf{x}^0$,k=1

2 while not happy

for $i = 1, \ldots, n$

$$x_i^{k+1} = \arg\min_{x \in \mathbb{R}} f(x_1^{k+1}, \ldots, x_{i-1}^{k+1}, x, x_{i+1}^k, \ldots, x_n^k)$$

use a 1D-minimization procedure

endfor

k=k+1

Critics

- Very bad for non-separable problems
- Not invariant if change of coordinate system

Descent methods

General principle

1. choose an intial point $\mathbf{x}^0$,k=1
2. while not happy
   - 2.1 choose a descent direction $\mathbf{d}_k \neq 0$
   - 2.2 choose a step-size $\sigma_k > 0$
   - 2.3 set $\mathbf{x}_{k+1} = \mathbf{x}_k + \sigma_k \mathbf{d}_k$, $k = k + 1$

Remaining questions

- How to choose $\mathbf{d}_k$?
- How to choose σ_k ?

Gradient descent

- Rationale: $\mathbf{d}_k = -\nabla f(\mathbf{x}_k)$ is a descent direction

Indeed for f differentiable

$$f(\mathbf{x} - \sigma \nabla f(\mathbf{x})) = f(\mathbf{x}) - \sigma \|\nabla f(\mathbf{x})\|^2 + o(\sigma) \underbrace{<}_{\text{for } \sigma \text{ small enough}} f(\mathbf{x})$$

Step-size

- optimal step-size

$$\sigma_k = \arg\min_{\sigma} f(\mathbf{x}_k - \sigma \nabla f(\mathbf{x}_k)) \tag{5}$$

- in general precise minimization of (5) is expensive and unecessary, instead a line search algorithm executes a limited number of trial steps until one loose approximation of the minimum is found

Gradient descent

Convergence rate on convex-quadratic function

- On $f(\mathbf{x}) = -\frac{1}{2}\mathbf{x}^T A\mathbf{x} - b^T\mathbf{x} + c$ with A a symmetric positive definite matrix with $Q = cond(A)$ we have that the gradient descent algorithm with optimal step-size satisfy:

$$f(\mathbf{x}_k) - \min f \leq \left(\frac{Q-1}{Q+1}\right)^{2k} [f(\mathbf{x}_0) - \min f]$$

- The convergence rate depends on the starting point, however $\left(\frac{Q-1}{Q+1}\right)^{2k}$ gives the correct description of the convergence rate for almost all starting points.

Very slow convergence for ill-conditionned problems

Newton algorithm, BFGS

Descent methods

Newton method

- descent direction: $-[\nabla^2 f(\mathbf{x}_k)]^{-1} \nabla f(\mathbf{x}_k)$

Newton direction

- points towards the optimum on $f(\mathbf{x}) = \mathbf{x}^T A \mathbf{x} + b^T \mathbf{x} + c$
- However, hessian matrix is expensive to compute in general and its inversion is also not immediate

Broyden-Fletcher-Goldfarb-Shanno (BFGS) method

- construct in an iterative manner using solely the gradient, an approximation of the Newton direction $-[\nabla^2 f(\mathbf{x}_k)]^{-1} \nabla f(\mathbf{x}_k)$

Trust regions methods

General idea

- instead of choosing a descent direction and minimizing along this direction, trust regions algorithms construct a model m_k of the objective function (usually a quadratic model)
- because the model may not be a good approximation of f far away from the current iterate $\mathbf{x}_k$, the search for a minimizer of the model is restricted to a region (trust region) around the current iterate
- trust region is usually a ball, whose radius is adapted (shrinked if no better solution is found)

Was not handled in this class

but you might want to have a look at it

Constraint optimization

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x})$$

subjected to $c_i(\mathbf{x}) = 0,\ i = 1, \ldots, p$

equality constraints

and $b_j(\mathbf{x}) \geq 0,\ j = 1, \ldots, k$

inequality constraints