

Numerical Optimization: Introduction and gradient-based methods

Master 2 Recherche LRI

Apprentissage Statistique et Optimisation

Anne Auger

Inria Saclay-Ile-de-France

2012

anne.auger@inria.fr

<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 unconstrained optimization
 - Convexity
- 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:

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

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

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

Exercise

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

$$\mathcal{L}_l = \{\mathbf{x} \in \mathbb{R}^n \mid f(\mathbf{x}) = l\}$$

for all $l \in \mathbb{R}$.

Data fitting - Data calibration

Objective

- Given a sequence of data points $(\mathbf{x}_i, y_i) \in \mathbb{R}^p \times \mathbb{R}, i = 1, \dots, N$, 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): $\{y_i, \underbrace{x_i^1, \dots, x_i^p}_{\mathbf{x}_i^T}\}_{i=1 \dots N}$

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

$$\tilde{\mathbf{X}} \in \mathbb{R}^{N \times (p+1)}, \tilde{\mathbf{w}} \in \mathbb{R}^{p+1}$$

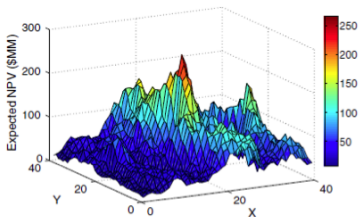
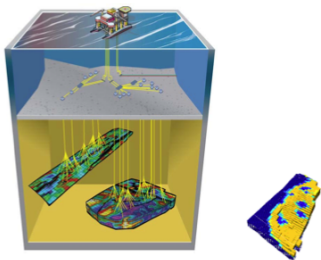
same as data fitting with linear model, i.e. $f_{(\mathbf{w}, \beta)}(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + \beta$,
 $\theta \in \mathbb{R}^{p+1}$

Black-box optimization

Optimization of well placement

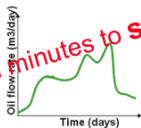
A Real-World Problem in Petroleum Engineering

Well Placement Problem



Onwunalu & Durlofsky (2010)

several minutes to **several hours !!**



Fluid flow simulation

Different notions of optimum

Local versus global minimum:

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

If the previous inequalities are strict we talk about a **strict** local minimum or **strict** global minimum.

Essential infimum of a function:

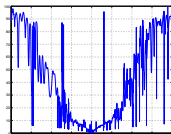
Given a measure μ on \mathbb{R}^n , the essential infimum of f is defined as

$$\text{ess inf } f = \sup\{b \in \mathbb{R} : \mu(\{\mathbf{x} : f(\mathbf{x}) < b\}) = 0\}$$

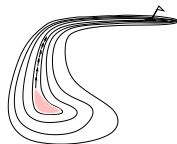
important to keep in mind in the context of stochastic optimization algorithms

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

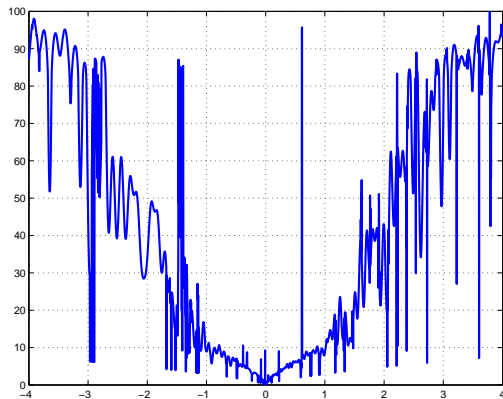


cut from 3-D example, solvable with an evolution strategy



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

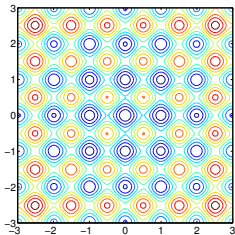
$$\arg \min_{(x_1, \dots, x_n)} f(x_1, \dots, x_n) = \left(\arg \min_{x_1} f(x_1, \dots), \dots, \arg \min_{x_n} f(\dots, x_n) \right)$$

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

Example: Additively decomposable functions

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

Rastrigin function



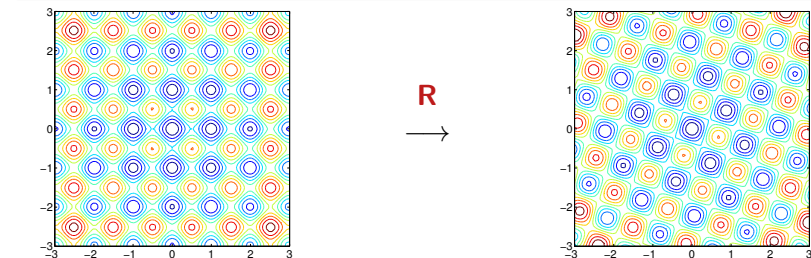
Non-Separable Problems

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

R rotation matrix



¹Hansen, 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

²Salomon (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

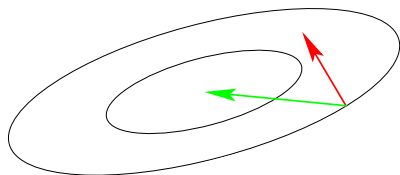
Ill-Conditioned Problems

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

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

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	<p>non-local policy, large sampling width (step-size) as large as possible while preserving a reasonable convergence speed</p> <p>stochastic, non-elitistic, population-based method recombination operator</p> <p>serves as repair mechanism</p>
Dimensionality, Non-Separability	<p>exploiting the problem structure</p> <p>locality, neighborhood, encoding</p>

What Makes a Function Difficult to Solve?

... and what can be done

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

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 without gradients: 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 unconstrained optimization
 - Convexity
- 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 $(E, \|\cdot\|)$ and complete (Banach space), consider $f : U \subset E \rightarrow \mathbb{R}$ with U open set of E .

- f is **differentiable** in $\mathbf{x} \in U$ if there exists a **continuous linear** mapping $Df(\mathbf{x})$ such that

$$f(\mathbf{x} + \mathbf{h}) = f(\mathbf{x}) + Df(\mathbf{x})(\mathbf{h}) + o(\|\mathbf{h}\|) \quad (3)$$

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

Exercise

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 $\|\cdot\|$ 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

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

Riesz representation Theorem

Taylor formula - order one

Replacing the 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

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

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 $(\mathbb{R}^n, \|\cdot\|_2)$ where $\|\mathbf{x}\|_2 = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$ is the euclidian norm

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

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

$f : U \subset E \rightarrow \mathbb{R}$ 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

$f : U \subset E \rightarrow \mathbb{R}$ 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 $D^2f(\mathbf{x})$ such that

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

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

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

where $\nabla^2 f(\mathbf{x}) : E \rightarrow E$ 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

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

Exercise

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 $f : \mathbb{R} \rightarrow \mathbb{R}$

Assume f is derivable

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

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

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

Generalization

If $f : U \subset \mathbb{R}^n \mapsto R$ is differentiable

- Necessary condition: If \mathbf{x}^* is a local extremum of f , then $Df(\mathbf{x}^*) = 0$ and hence $\nabla f(\mathbf{x}^*) = 0$.

proof via Taylor formula

Optimality conditions

Second order necessary and sufficient conditions

If f is twice continuously differentiable

- Necessary condition: if \mathbf{x}^* is a local minimum, then $\nabla f(\mathbf{x}^*) = 0$ and $\nabla^2 f(\mathbf{x}^*)$ is **positive semi-definite**
- Sufficient condition: if $\nabla f(\mathbf{x}^*) = 0$ and $\nabla^2 f(\mathbf{x}^*)$ is **positive definite**, then \mathbf{x}^* is a strict local minimum.

Convex functions

Let U be a convex open of \mathbb{R}^n and $f : U \rightarrow \mathbb{R}$. The function f is said convex if for all $\mathbf{x}, \mathbf{y} \in U$ and for all $t \in [0, 1]$

$$f((1-t)\mathbf{x} + t\mathbf{y}) \leq (1-t)f(\mathbf{x}) + tf(\mathbf{y})$$

Theorem

If f is differentiable, then f is convex if and only if

$$f(\mathbf{y}) - f(\mathbf{x}) \geq Df(\mathbf{x})(\mathbf{y} - \mathbf{x})$$

if $n = 1$, the curve is on top of the tangent

If f is twice continuously differentiable, then f is convex if and only if D^2f satisfy

$$\forall \mathbf{x} \in U, h \in \mathbb{R}^n D^2f(\mathbf{x})(h, h) \geq 0$$

Convex functions

Examples:

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

$$f(\mathbf{x}) = \frac{1}{2} \langle \mathbf{x}, A\mathbf{x} \rangle + \langle a, \mathbf{x} \rangle + b, \quad A \text{ positive definite symmetric}$$

the opposite of the entropy function: $f(\mathbf{x}) = -\sum_{i=1}^n x_i \ln(x_i)$ (the entropy function is then concave)

Exercise

Let $f : U \rightarrow \mathbb{R}$ be a convex and differentiable function defined on a convex open U .

Show that if $Df(\mathbf{x}^*) = 0$, then \mathbf{x}^* is a global minimum of f

Exercise

Show that on a convex function, a descent direction d_k should verify $\nabla f(\mathbf{x}_k) \cdot d_k < 0$

- 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 unconstrained optimization
 - Convexity
- 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.

$$\|\mathbf{x}_n - \mathbf{x}^*\| \xrightarrow{n \rightarrow \infty} 0$$

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

- Equivalently, given a fixed precision ϵ , the algorithms aim at approaching the optimum \mathbf{x}^* 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 $f : \mathbb{R} \rightarrow \mathbb{R}$

- 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: $|x_{n+1} - x^*| \leq \mu |x_n - x^*|^2$

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

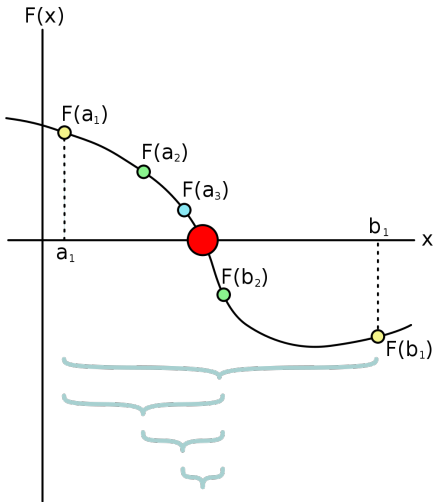
secant method: replaces the derivative computation its the finite difference approximation

Application to the minimization of f

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

Bisection method

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



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

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

linear convergence

Wikipedia image

Relaxation algorithm

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

Algorithm

- successive optimization w.r.t. each variable

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

2 while not happy

for $i = 1, \dots, n$

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

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 initial point $\mathbf{x}^0, k=1$
- 2 while not happy
 - 2.1 choose a **descent direction** $\mathbf{d}_k \neq 0$
 - 2.2 **Line search**: 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

$$f(\mathbf{x} - \sigma \nabla f(\mathbf{x})) = f(\mathbf{x}) - \sigma \|\nabla f(\mathbf{x})\|^2 + o(\sigma \|\nabla f(\mathbf{x})\|) \quad \underbrace{\qquad}_{<} \quad f(\mathbf{x})$$

for σ small enough

Step-size

- optimal step-size

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

- in general precise minimization of (5) is expensive and unnecessary, 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 $Q = \text{cond}(A)$ we have that the gradient descent algorithm with optimal step-size satisfy:

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

linear convergence

- 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.

for $Q \gg 1$, $\frac{Q-1}{Q+1} \approx 1 \Rightarrow$ very slow convergence

i.e. very slow convergence for ill-conditioned problems

Newton algorithm, BFGS

Descent methods

Newton method

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

Newton direction

quadratic convergence

Broyden-Fletcher-Goldfarb-Shanno (BFGS) method

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

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 (shrunk 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 $c_i(\mathbf{x}) = 0, i = 1, \dots, p$

equality constraints

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

inequality constraints