Computational Maths 2

Introduction to Numerical Optimization

Beniamin Bogosel

CMAP
École Polytechnique

What this course is about?

- theoretical aspects in optimization
- algorithms for numerical optimization
- implementation of optimization algorithms

Objectives

After this course you should:

1. know the basic optimization algorithms: gradient descent, Newton, etc.
2. implement optimization algorithms for problems of reasonable size
3. translate the contents of a problem into an optimization algorithm
4. know how to use existing libraries in order to solve particular classes of optimization problems

Grading

1. 50%: evaluation of your work during practical sessions
   - activity points: at the end of each session you should provide a working Python code related to the current Exercise Sheet and upload it on Moodle
   - solving Challenge or Supplementary exercises (in addition to the main exercises) will give you bonus points
2. 50%: final test during the last practical session
   - work on a given problem: answer some theoretical questions and solve some implementation tasks
   - you are allowed to use all resources available (course notes, personal notes, etc.)

What is optimization?

$\star$ given an objective function $x \mapsto f(x)$, find the value(s) of x which give the smallest value of f!
$\star$ x may be subjected to some constraints
$\star$ often the minimizer x^* may not be found explicitly: numerical simulations are needed in this context

$\star$ numerical optimization algorithms produce a sequence (x_n) defined iteratively using the values of f and possibly its derivatives.
$\star$ various questions arise concerning

- the convergence of the sequence (x_n) to a minimizer of f
- the speed of convergence

Examples

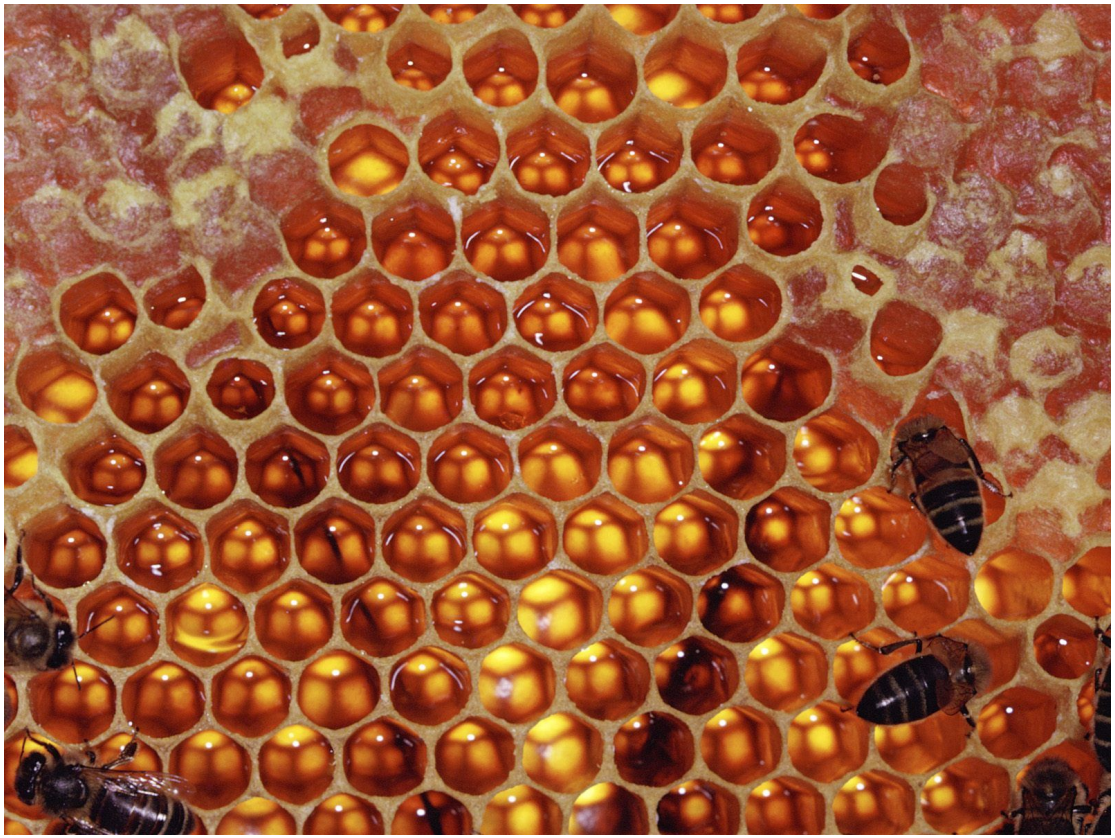
1. Minimize $\|Ax - b\|_2$ where $A \in \mathcal{M}_{m\times n}, x \in \mathbb{R}^n, b \in \mathbb{R}^m$ with $m > n$.

2. Minimize $c \cdot x$ where $c, x \in \mathbb{R}^n$, $x \geq 0$, $Ax \leq b$ (linear programming problem)

3. Model fitting: Given a set of data points (x_i, y_i), $1 \leq i \leq N$ find a function F such that $F(x_i) \approx y_i$.

Examples in Nature

- Honeycombs are optimal in terms of construction cost (mathematical understanding came only recently: Thomas C. Hales (1999))

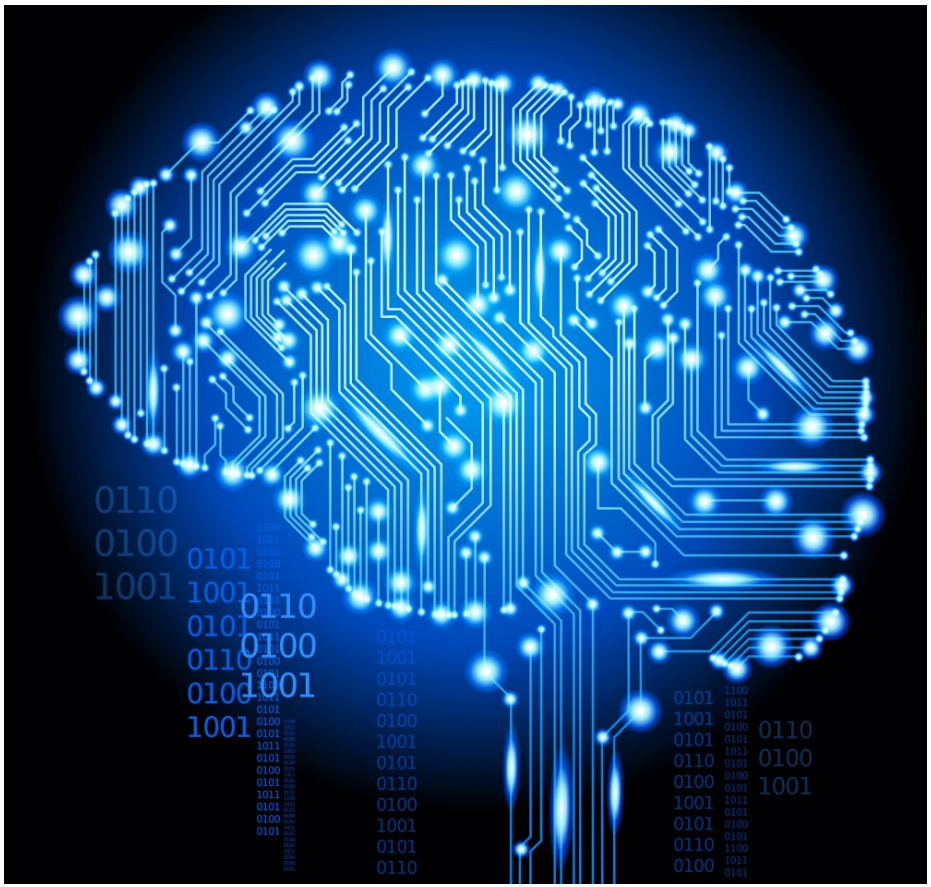


Examples in Nature

- Soap bubbles tend to minimize the surface area while keeping a fixed volume

Applications

- finance, deep learning: process existing information in order to take the best decisions (photo `rostigrabench.ch`)



Applications in industry

- Optimal design of structures: reduce the weight while maintaining the desired mechanical properties

Motivation...

- for practical applications, optimization algorithms are used
- the user should formulate an optimization problem starting from the given data or models
- once a function which associates a real value to a certain set of parameters is known, optimization algorithms can be used to search for the minimum
- the methods of optimization are vast
  - gradient-free vs gradient based methods
  - higher order methods (Newton)
- the choice of the method depends on the objective function: unimodal functions (nice), highly oscillating functions, non-smooth functions, etc.
- often some constraints need to be enforced, which complicate the theoretical and numerical aspects of optimization problems

Contents of the course

1. General aspects in optimization
2. Optimization in dimension 1
   - Methods of order zero (without derivatives)
   - Methods of order one and two (using derivatives)
3. Optimization in higher dimensions
   - Gradient descent methods
   - Newton methods
   - quasi-Newton methods
4. Constrained optimization
   - Lagrange multipliers
   - a quick glimpse of linear programming (emphasis on practical issues)

Optimization: general aspects

- The discrete case
- Continuous optimization

General optimization problem

In the following: $\mathcal{A}$ is a non-void set, J is a real function defined on $\mathcal{A}$.

Canonical formulation

Let $J : \mathcal{A} \to \mathbb{R}$ be a real function. We wish to solve the problem

$$\min_{x \in \mathcal{A}} J(x)$$

Question: what about maximization problems?

General optimization problem

In the following: $\mathcal{A}$ is a non-void set, J is a real function defined on $\mathcal{A}$.

Canonical formulation

Let $J : \mathcal{A} \to \mathbb{R}$ be a real function. We wish to solve the problem

$$\min_{x \in \mathcal{A}} J(x)$$

Remark: Note that maximization problems are also included in this framework

$$\max_{x \in \mathcal{A}} J(x) = - \left(\min_{x \in \mathcal{A}} -J(x) \right).$$

Remark2: The rigorous way is to write inf instead of min when we don't know that a solution exists in $\mathcal{A}$.

General optimization problem

In the following: $\mathcal{A}$ is a non-void set, J is a real function defined on $\mathcal{A}$.

Canonical formulation

Let $J : \mathcal{A} \to \mathbb{R}$ be a real function. We wish to solve the problem

$$\min_{x \in \mathcal{A}} J(x)$$

Remark: Note that maximization problems are also included in this framework

$$\max_{x \in \mathcal{A}} J(x) = - \left(\min_{x \in \mathcal{A}} -J(x) \right).$$

Remark2: The rigorous way is to write inf instead of min when we don't know that a solution exists in $\mathcal{A}$. Questions:

- how do we deal with optimization problems in terms of $\mathcal{A}$? (discrete vs continuous case)
- when do we have a solution? what are the conditions for $\mathcal{A}$ and J?

Optimization: general aspects

- The discrete case
- Continuous optimization

$\mathcal{A}$ is finite

$\mathcal{A} = \{x_1, x_2, ..., x_N\}$ so J takes the values

$$\{J(x_1), J(x_2), ..., J(x_N)\}.$$

Questions:

- what about existence of solutions?
- if a solution exists, how do you find it?

$\mathcal{A}$ is finite

$\mathcal{A} = \{x_1, x_2, ..., x_N\}$ so J takes the values

$$\{J(x_1), J(x_2), ..., J(x_N)\}.$$

- if $\mathcal{A}$ is finite, we always have existence of solutions!
- the difficulty of finding the optimal value among $J(x_i)$ depends on multiple factors:
  - how big is N?
  - how fast can you compute $J(x_i)$?
  - is there some underlying structure which can help us get to the solution faster?

Example 1: Optimal assignment problem

Let's say we have the following situation:

	Person 1	Person 2	Person 3
Job 1	100€	120€	80€
Job 2	150€	110€	120€
Job 3	90€	80€	110€

Questions:

1. What is the optimal assignment: Job $i \longrightarrow$ Person j?

Example 1: Optimal assignment problem

Let's say we have the following situation:

	Person 1	Person 2	Person 3
Job 1	100€	120€	80€
Job 2	150€	110€	120€
Job 3	90€	80€	110€

Questions:

1. What is the optimal assignment: Job $i \longrightarrow$ Person j?
2. What is the cost of the naïve implementation in terms of the number of persons?

Example 1: Optimal assignment problem

Let's say we have the following situation:

	Person 1	Person 2	Person 3
Job 1	100€	120€	80€
Job 2	150€	110€	120€
Job 3	90€	80€	110€

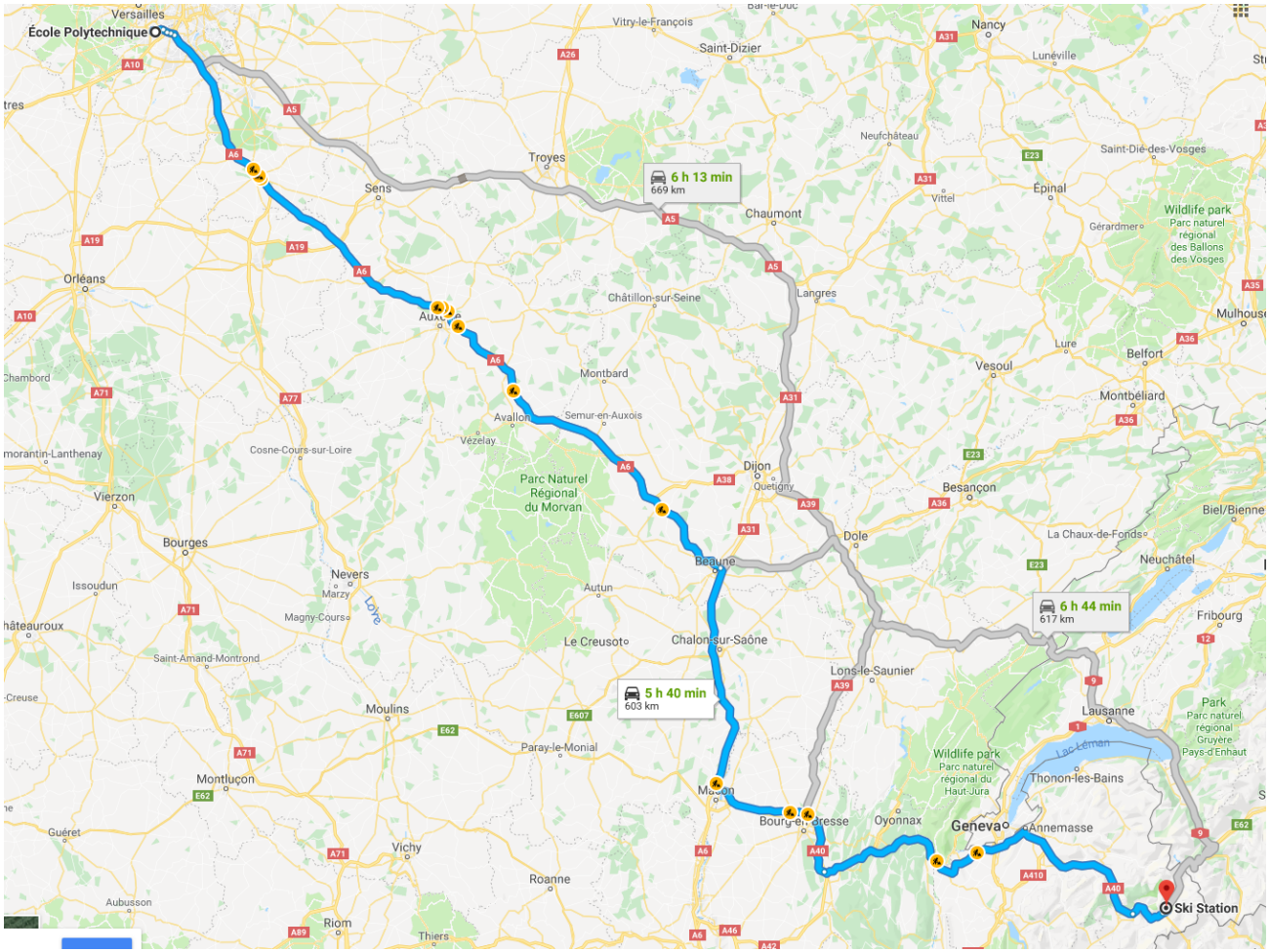
Questions:

1. What is the optimal assignment: Job $i \longrightarrow$ Person j?
2. What is the cost of the naïve implementation in terms of the number of persons? Answer: $O(n!)$
3. Is there a better algorithm? Yes: Hungarian algorithm with complexity $O(n^3)$.

   Reference: link

Example 2: Minimal path through a graph

Dijkstra's algorithm: intelligently find the optimal path going through the branches of your graph



Reference: link

Conclusion on the discrete part

- Discrete optimization problem: finite number of configurations $\longrightarrow$ existence of solutions
- That does not mean that we can always find the optimal solution in reasonable computation time
- We will not talk about discrete optimization in the rest of the course.

Optimization: general aspects

- The discrete case
- Continuous optimization

$\mathcal{A}$ is an infinite subset of $\mathbb{R}^n$

Again, we wish to study the problem

$$\inf_{x \in \mathcal{A}} J(x)$$

Question: Under what classical hypotheses on $\mathcal{A}$ and J can we conclude that the above problem has a solution?

$\mathcal{A}$ is an infinite subset of $\mathbb{R}^n$

Again, we wish to study the problem

$$\inf_{x \in \mathcal{A}} J(x)$$

Answer

If $\mathcal{A}$ is compact and J is continuous then the infimum is reached for some $x_0 \in \mathcal{A}$:

$$\text{there exists } x_0 \in \mathcal{A} \text{ such that } J(x_0) = \min_{x \in \mathcal{A}} J(x)$$

Examples and counterexamples

1. $\mathcal{A} = \{\frac{1}{n} : n \in \mathbb{N}^*\}$, $J(x) = x$
   Issue: If $\mathcal{A}$ is disconnected, how do we choose between its different connected components???
   In the rest of the course, in the one dimensional and higher dimensional case, we always assume $\mathcal{A}$ is connected
2. $\mathcal{A} = (0, 1]$, $f(x) = x^2$
3. $\mathcal{A} = [0, 1]$, $f(x) = \begin{cases} -1/x & x > 0 \\ 0 & x = 0 \end{cases}$

Assumptions

In the following we assume that the function we minimize J is regular of class C^k $(k \geq 1)$ and the set $\mathcal{A}$ is the closure of an open and connected set (unless otherwise stated)

$\star$ **Advantage w.r.t. discrete case:** we use information given by the values of the function J and its derivatives in order to decide how to improve the value of $J(x)$.

$\star$ We can advance with increments which are arbitrarily small in order to decrease J: this is not possible if $\mathcal{A}$ is not open and connected

Optimization in dimension 1

- Methods of order zero (without derivatives)
- Methods of order one and above (with derivatives)

Some basic definitions

Let $f : K \to \mathbb{R}$ be a regular function and K be an interval.

1. x^* is a local minimum of f on K if there exists $\varepsilon > 0$ such that $f(x^*) \leq f(x)$ for every $x \in (x^* - \varepsilon, x^* + \varepsilon)$
2. x^* is a local maximum of f on K if there exists $\varepsilon > 0$ such that $f(x^*) \geq f(x)$ for every $x \in (x^* - \varepsilon, x^* + \varepsilon)$
3. x^* is a global minimum of f on K if $f(x^*) \leq f(x)$ for every $x \in K$
4. x^* is a global maximum of f on K if $f(x^*) \geq f(x)$ for every $x \in K$
5. x^* is an local/global extremum of f on K if it is a local/global minimum or maximum of f

Existence of a minimizer

Compact interval

Let $f : [a, b] \to \mathbb{R}$ be a continuous function. Then f is bounded and it attains its upper and lower bounds on $[a, b]$, i.e. f admits global minima and maxima.

$\star$ a classical condition to recover existence on the whole space is what we call "infinite at infinity"

Existence on $\mathbb{R}$

Let $f : \mathbb{R} \to \mathbb{R}$ be a continuous function such that $f(x) \to +\infty$ when $|x| \to +\infty$ then f admits global minimizers on $\mathbb{R}$.

$\star$ Uniqueness is not guaranteed, in general.

Necessary conditions of optimality

Suppose that f is a C^1 function defined on an interval $K \subset \mathbb{R}$ and that f has a local extremum at x^* which is an interior point of K. Then $f'(x^*) = 0$.

Proof: Classical. Just write $f'(x^*) = \lim_{x \to x^*} \dfrac{f(x) - f(x^*)}{x - x^*}$.

$\star$ points x such that $f'(x) = 0$ are called critical points.

$\star$ what happens if the extremum is attained at the end of the interval?

Euler inequality

Let $f : [a, b] \to \mathbb{R}$ be a C^1 function on an open set containing $[a, b]$. Then

- if a is a local minimum then $f'(a) \geq 0$
- if b is a local minimum then $f'(b) \leq 0$
- if a is a local maximum then $f'(a) \leq 0$
- if b is a local maximum then $f'(b) \geq 0$

Proof: the same idea.

Before going further...

$\star$ Recall the Taylor expansion formula around a: suppose that f is smooth and x is "close to a". Then

$$f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \dots$$

Before going further...

Proposition 1 (Taylor theorem with remainder)

Suppose that $f : \mathbb{R} \to \mathbb{R}$ is of class C^k at a. Then

$$f(x) = \sum_{i=0}^{k} \frac{f^{(i)}(a)}{i!}(x-a)^i + R_k(x)$$

where the remainder $R_k(x)$ is equal to one of the following:

- *$R_k(x) = h_k(x)(x-a)^k$ with $\lim_{x\to a} h_k(x) = 0$. In other words $R_k(x) = o(|x-a|^k)$ as $x \to a$.*
- *if f is of class C^{k+1} then*

$$R_k(x) = \frac{f^{(k+1)}(\xi_L)}{(k+1)!}(x-a)^{k+1}$$

with ξ_L between a and x. This is the Lagrange form of the remainder.

$\star$ Recall the Little-o and Big-O notations:

$$|O(x)| \leq C|x| \text{ and } \frac{o(x)}{|x|} \to 0 \text{ as } |x| \to 0$$

What about sufficient conditions?

$\star$ in general, we may have critical points which are not local extrema
Example: $f(x) = x^3$ has a unique critical point $x = 0$, but $x = 0$ is not a local minimizer.
$\star$ the first option is to look at second order conditions

Second order necessary and sufficient conditions

1. Suppose $f : \mathbb{R} \to \mathbb{R}$ is of class C^2 and $x^* \in \mathbb{R}$. Then

$$x^* \text{ is a local minimum of } f \Longrightarrow f'(x^*) = 0 \text{ and } f''(x^*) \geq 0$$

$$x^* \text{ is a local maximum of } f \Longrightarrow f'(x^*) = 0 \text{ and } f''(x^*) \leq 0$$

2. Suppose $f : \mathbb{R} \to \mathbb{R}$ is of class C^2 and $x^* \in \mathbb{R}$. Then

$$f'(x^*) = 0 \text{ and } f'' \geq 0 \text{ on } (x^* - \varepsilon, x^* + \varepsilon) \Longrightarrow x^* \text{ is a local minimum of } f.$$

This implies the following weaker sufficient condition:

$$f'(x^*) = 0 \text{ and } f''(x^*) > 0 \Longrightarrow x^* \text{ is a local minimum of } f.$$

Important particular case

⋆ the class of convex functions is important from the optimization point of view
⋆ we can have results of existence and uniqueness of minimizers
⋆ first order optimality conditions are necessary and sufficient

Definition 2 (Convex functions)

Let $f : \mathbb{R} \to \mathbb{R}$ be a function.
f is convex if $\forall t \in [0,1],\ \forall x, y \in \mathbb{R}$ we have

$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y)$$

Equivalent definitions:
⋆ f is below its secants
⋆ f is above its tangents (where f is regular)

⋆ if we replace the inequality above with a strict one, we obtain the class of strictly convex functions

Existence and uniqueness: convex case

Proposition 3

Let $f : \mathbb{R} \to \mathbb{R}$ be a convex function. If f is convex then any local minimum of f is a global minimum.

Proposition 4 (Uniqueness)

Let $f : \mathbb{R} \to \mathbb{R}$ be a convex function. If f is strictly convex then there exists at most one minimum of f on $\mathbb{R}$.

$\star$ We cannot say more with strict convexity alone! In particular, strict convexity does not guarantee existence. Consider $f(x) = \exp(x)$.

Proposition 5 (Existence and Uniqueness)

Let $f : \mathbb{R} \to \mathbb{R}$ be a function. Then if

- *$f(x) \to +\infty$ when $|x| \to \infty$*
- *f is strictly convex*

then there exists a unique minimizer x^* of f on $\mathbb{R}$.

Exercise: Prove that a real valued convex function is continuous!

Optimality conditions: convex case

Proposition 6

Suppose that $f : \mathbb{R} \to \mathbb{R}$ is a convex function of class C^1 and $x^* \in \mathbb{R}$. Then the following statements are equivalent:

- *x^* is a global minimum of f*
- *x^* is a local minimum of f*
- *$f'(x^*) = 0$*

$\star$ convexity gives convenient tools for proving convergence results regarding numerical algorithms

$\star$ it is one of the rare hypotheses which can guarantee the convergence of an algorithm to the global minimum

$\star$ numerical algorithms will be applied to general functions, but in general we can only hope to converge to a local minimum

Importance of the 1D case

⋆ It gives an initial framework, to be extended to higher dimensions
⋆ most efficient optimization algorithms use a line-search routine

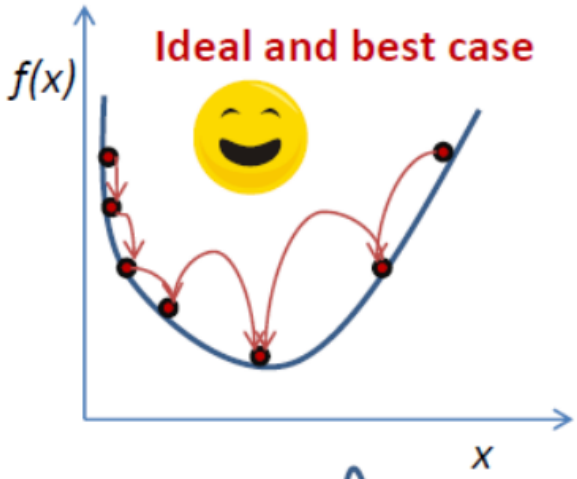
Example of optimization algorithm

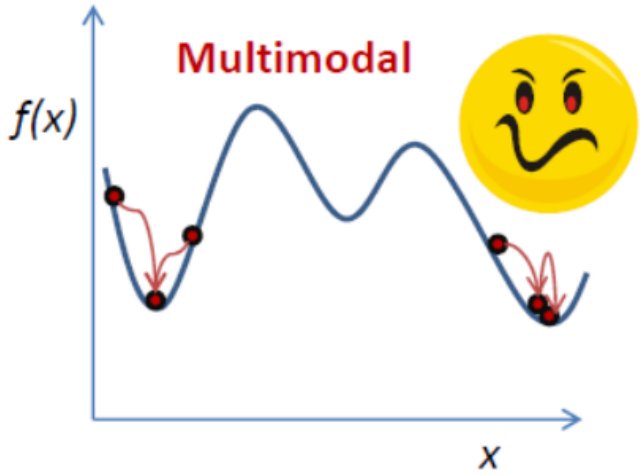
Optimization of a function $f : \mathbb{R}^n \to \mathbb{R}$ starting from an initial point $\mathbf{x}_0$
At iteration i

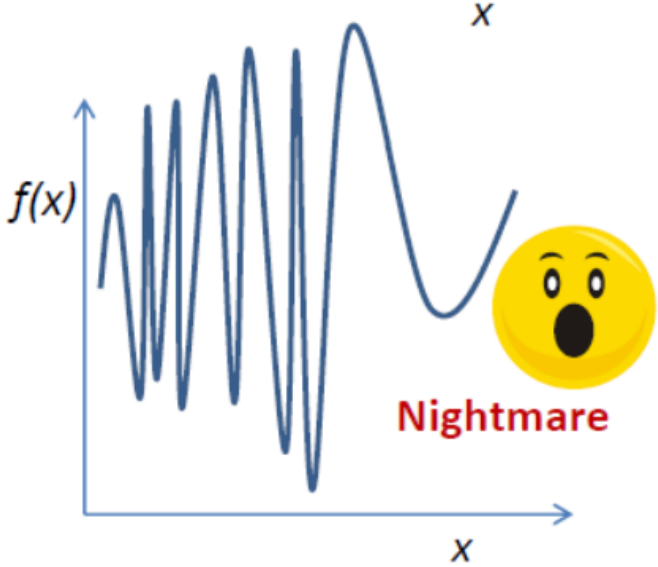
- Point $\mathbf{x}_n$: find a descent direction $\mathbf{d}_n$
- Find a reasonable step size such that $f(\mathbf{x}_n + \gamma \mathbf{d}_n)$ is significantly smaller than $f(\mathbf{x}_n)$

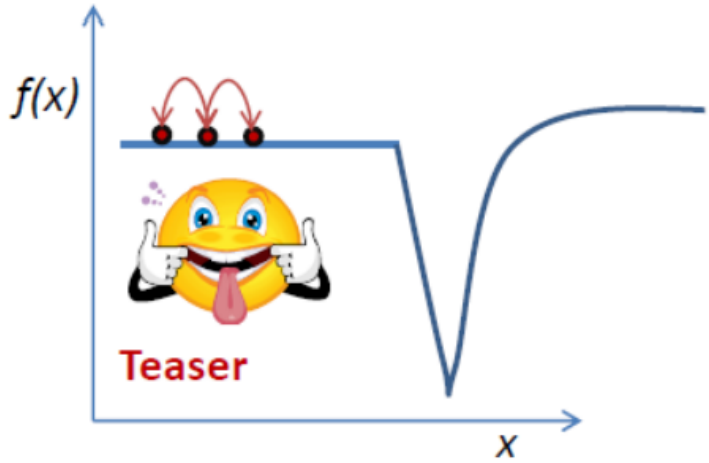
⋆ The second step is essentially a one dimensional optimization routine
⋆ Often it is not reasonable to solve an optimization problem at every iteration

What to expect?









[photo from Ziv Bar-Joseph, used with permision]

Assumption: the function f is unimodal on the segment $[a, b]$, i.e. it possesses a unique local minimum on $[a, b]$

Optimization in dimension 1

- Methods of order zero (without derivatives)
- Methods of order one and above (with derivatives)

Strategy

$\star$ f is unimodal on $[a, b]$: it possesses a unique local minimum $x^* \in [a, b]$

Proposition 7

If f is unimodal on $[a, b]$ with minimum x^* then:
$\star$ *f is strictly decreasing on $[a, x^*]$ and strictly increasing on $[x^*, b]$.*
$\star$ *f is unimodal on every sub-interval $[a', b'] \subset [a, b]$*

$\star$ We wish to reduce the size of the interval $[a, b]$ by computing the value of f at some intermediary points
$\star$ Without the use of derivatives, one intermediary point is not enough. Are two intermediary points enough?

Consider two points $x^+, x^- \in (a, b)$ such that $a < x^- < x^+ < b$.
Case 1: $f(x^-) \leq f(x^+) \Rightarrow \ldots$
Case 2: $f(x^-) \geq f(x^+) \Rightarrow \ldots$

Strategy

$\star$ f is unimodal on $[a, b]$: it possesses a unique local minimum $x^* \in [a, b]$

Proposition 7

If f is unimodal on $[a, b]$ with minimum x^* then:
$\star$ *f is strictly decreasing on $[a, x^*]$ and strictly increasing on $[x^*, b]$.*
$\star$ *f is unimodal on every sub-interval $[a', b'] \subset [a, b]$*

$\star$ We wish to reduce the size of the interval $[a, b]$ by computing the value of f at some intermediary points
$\star$ Without the use of derivatives, one intermediary point is not enough. Are two intermediary points enough?

Consider two points $x^+, x^- \in (a, b)$ such that $a < x^- < x^+ < b$.
Case 1: $f(x^-) \leq f(x^+) \Rightarrow$ x^* is to the left of x^+
Case 2: $f(x^-) \geq f(x^+) \Rightarrow$ x^* is to the right of x^-

Strategy

$\star$ f is unimodal on $[a, b]$: it possesses a unique local minimum $x^* \in [a, b]$

Proposition 7

If f is unimodal on $[a, b]$ with minimum x^* then:
$\star$ *f is strictly decreasing on $[a, x^*]$ and strictly increasing on $[x^*, b]$.*
$\star$ *f is unimodal on every sub-interval $[a', b'] \subset [a, b]$*

$\star$ We wish to reduce the size of the interval $[a, b]$ by computing the value of f at some intermediary points
$\star$ Without the use of derivatives, one intermediary point is not enough. Are two intermediary points enough?

Consider two points $x^+, x^- \in (a, b)$ such that $a < x^- < x^+ < b$.
Case 1: $f(x^-) \leq f(x^+) \Rightarrow$ x^* is to the left of $x^+ \Rightarrow$ replace $[a, b]$ with $[a, x^+]$
Case 2: $f(x^-) \geq f(x^+) \Rightarrow$ x^* is to the right of $x^- \Rightarrow$ replace $[a, b]$ with $[x^-, b]$

Generic Algorithm

Algorithm 1 (Zero-order minimization of a unimodal function)

Initialization: *Initial segment* $S_0 = [a, b]$, *iteration number* $i = 1$

Step i*: Given previous segment* $S_{i-1} = [a_{i-1}, b_{i-1}]$

- *choose points* x_i^-, x_i^+*:* $a_{i-1} < x_i^- < x_i^+ < b_{i-1}$
- *compute* $f(x_i^-)$ *and* $f(x_i^+)$
- *define the new segment as follows*
  - *if* $f(x_i^-) \leq f(x_i^+)$ *then* $S_i = [a_{i-1}, x_i^+]$
  - *if* $f(x_i^-) \geq f(x_i^+)$ *then* $S_i = [x_i^-, b_{i-1}]$
- *go to step* $i + 1$

$\star$ Why does the algorithm work?

- at each step we guarantee that x^* belongs to S_i
- the length of S_i is diminished at each iteration

$\star$ Stopping criterion: the length of the segment S_i is smaller than a tolerance $\varepsilon > 0$

Rate of convergence

⋆ measure the speed of convergence of the iterates to the optimum
⋆ define an error function $\text{err}(x_i)$: for example $\text{err}(x_i) = |x_i - x^*|$
⋆ in the following, denote $r_i = \text{err}(x_i)$

Standard classification

- linear convergence: there exists $C > 0$ such that $r_i \leq Cq^i$
  ⋆ the constant $q \in (0, 1)$ is called the convergence ratio
  ⋆ sufficient condition: $\limsup_{i \to \infty} (r_{i+1}/r_i) < q$
- sublinear convergence: $r_i \to 0$ but is not linearly converging
- superlinear convergence: $r_i \to 0$ with any positive convergence ratio
  ⋆ sufficient condition: $\lim_{i \to \infty} (r_{i+1}/r_i) = 0$
- convergence of order $p > 1$: there exists $C > 0$ such that for i large enough

$$r_{i+1} \leq Cr_i^p$$

  ⋆ p is called the order of convergence
  ⋆ $p = 2$ has a special name: quadratically convergent
  ⋆ every convergence of order $p > 1$ is super-linear, but the reverse implication is not valid

Rates of convergence - Examples

Let $\gamma \in (0, 1)$. Then:

- (γ^n) converges linearly to zero, but not superlinearly
- (γ^{n^2}) converges superlinearly to zero, but not quadratically
- (γ^{2^n}) converges to zero quadratically

Quadratic convergence is much faster than linear convergence

Plotting the order of convergence

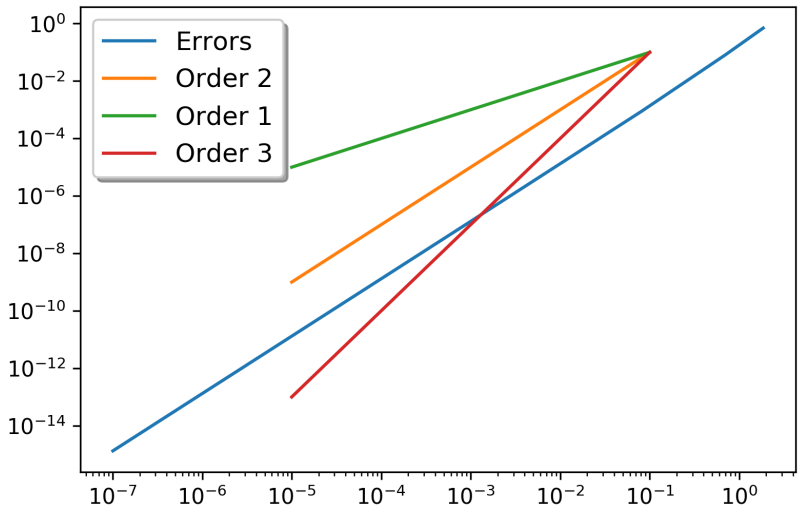
For the convergence of order p we have $r_{i+1} \approx C r_i^p$.

$\star$ representing this directly does not illustrate clearly the power p

$\star$ taking logarithms we get $\log \text{err}(x_{i+1}) \approx \log C + p \log \text{err}(x_i)$

$\star$ therefore, plotting the next error in terms of the previous error in a log-log scale gives the line $y = \log C + px$

$\star$ the slope of the line shows the order of the method!



Back to the zero-order algorithm

$\star$ the interval S_i gives an approximation of x^* with error at most $|S_i|$
$\star$ Trisection algorithm: we can achieve linear convergence

$$x_i^- = \frac{2}{3}a_{i-1} + \frac{1}{3}b_{i-1} \quad x_i^+ = \frac{1}{3}a_{i-1} + \frac{2}{3}b_{i-1}$$

implies $|S_i| = 2/3|S_{i-1}|$.
$\star$ if x_i is an arbitrary point in S_i then

$$|x^* - x_i| \leq \left(\frac{2}{3}\right)^i |b - a|.$$

$\star$ if x_i is an approximation of x^* after k function evaluations then

$$|x^* - x_i| \leq \left(\frac{2}{3}\right)^{\lfloor k/2 \rfloor} |b - a|.$$

$\star$ it is possible to be more efficient by doing one function evaluation when changing from S_{i-1} to S_i

Fibonacci search

⋆ the Fibonacci sequence is defined by

$$F_0 = 1,\ F_1 = 1,\ F_{n+1} = F_n + F_{n-1}.$$

⋆ first few terms are: $1, 1, 2, 3, 5, 8, 13, 21, 34, 55...$

⋆ Fibonacci search: when you know from advance the number of function evaluations N you want to make

Algorithm 2 (Fibonacci search)

Initialization: *Start with* $S_0 = [a_0, b_0]$ *and perform* N *steps as follows:* **For** $i = 1, ..., N-1$

- *choose* x_i^- *and* x_i^+ *such that*

$$|a_{i-1} - x_i^+| = |b_{i-1} - x_i^-| = \frac{F_{N-i}}{F_{N-i+1}}|a_{i-1} - b_{i-1}|$$

- *compute* $f(x_i^-)$ *or* $f(x_i^+)$ *(which one was not computed before)*
- *define the new segment as follows*
  - *if* $f(x_i^-) \leq f(x_i^+)$ *then* $S_i = [a_{i-1}, x_i^+]$
  - *if* $f(x_i^-) \geq f(x_i^+)$ *then* $S_i = [x_i^-, b_{i-1}]$
- *go to step* $i + 1$

Why is this choice ok?

Proposition 8

We need to do only one function evaluation per iteration.

$\star$ $|b_i - a_i| = \frac{F_{N-i}}{F_{N-i+1}} \dots \frac{F_{N-1}}{F_N} |b_0 - a_0| = \frac{F_{N-i}}{F_N} |b_0 - a_0|$

$\star$ in the end $|x^* - x_N| = |b_N - a_N| = \frac{|b_0 - a_0|}{F_N}$

$\star$ Formula: $F_n = \frac{1}{\lambda+2} \left[(\lambda+1)\lambda^n + (-1)^n \lambda^{-n} \right], \ \lambda = \frac{1+\sqrt{5}}{2}$

$\star$ In the end: $|x^* - x_N| \leq C\lambda^{-N} |b_0 - a_0| (1 + o(1))$ which gives a linear convergence rate with ratio $\lambda^{-1} = \frac{2}{1+\sqrt{5}} = 0.61803...$

$\star$ the previous method gave a rate of convergence of $\sqrt{2/3} = 0.81649...$ in terms of the number of evaluations

$\star$ this is the best we can do in a given number of iterations

[J. Kiefer, *Sequential minimax search for a maximum*]

Fun fact - computing Fibonacci numbers

Question

What algorithm do you use to compute F_n given n?

Fun fact - computing Fibonacci numbers

Question

What algorithm do you use to compute F_n given n?

Trivial algorithm

Initialize $F_0 = 1, F_1 = 1$, at each step compute $F_i = F_{i-1} + F_{i-2}$.
Complexity:

Don't store all values F_i if they are not needed: diminish memory consumption

Fun fact - computing Fibonacci numbers

Question

What algorithm do you use to compute F_n given n?

Trivial algorithm

Initialize $F_0 = 1, F_1 = 1$, at each step compute $F_i = F_{i-1} + F_{i-2}$.
Complexity: O(n)

Don't store all values F_i if they are not needed: diminish memory consumption

Fun fact - computing Fibonacci numbers

Question

What algorithm do you use to compute F_n given n?

Trivial algorithm

Initialize $F_0 = 1, F_1 = 1$, at each step compute $F_i = F_{i-1} + F_{i-2}$.
Complexity: O(n)

Don't store all values F_i if they are not needed: diminish memory consumption

Smart algorithm

If $M = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$ then $M^n = \begin{pmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{pmatrix}$.
Complexity:

Fun fact - computing Fibonacci numbers

Question

What algorithm do you use to compute F_n given n?

Trivial algorithm

Initialize $F_0 = 1, F_1 = 1$, at each step compute $F_i = F_{i-1} + F_{i-2}$.
Complexity: O(n)

Don't store all values F_i if they are not needed: diminish memory consumption

Smart algorithm

If $M = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$ then $M^n = \begin{pmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{pmatrix}$.
Complexity: $O(\log n)$

$\star$ Exponentiation is very fast if done properly: search for "exponentiation by squaring" or "fast exponentiation" if you are interested
$\star$ If you want other tricky problems where maths can significantly reduce the complexity of the problem try Project Euler

Another way of computing Fibonacci numbers

Use the following recursion formulas:

$$F_{2n} = F_n(2F_{n+1} - F_n)$$
$$F_{2n+1} = F_{n+1}^2 + F_n^2$$

⋆ This will again give you a $O(\log n)$ algorithm since you can always go from n to $2n$ or $2n+1$: the number of steps is the length of the binary expansion of n
⋆ All this is nice, but be aware that Fibonacci numbers grow exponentially fast:

$$F_n = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^{n+1} - \left(\frac{1-\sqrt{5}}{2}\right)^{n+1}\right]$$

⋆ Note that $F_n \approx \frac{1}{\sqrt{5}}\lambda^{n+1}$
⋆ in NumPy you will quickly go beyond the 16 digit precision

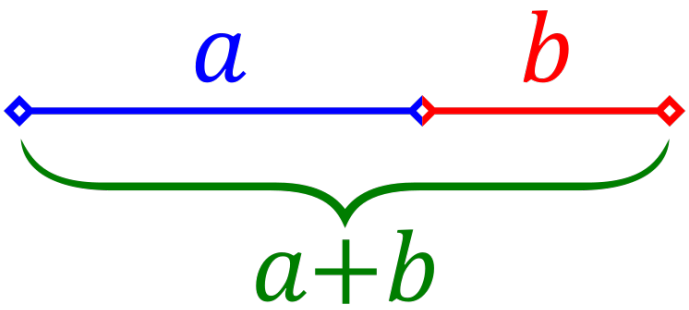
Golden search

⋆ Inconvenient - Fibonacci search: one needs to know in advance the number of function evaluations N

⋆ For large N this can be avoided

⋆ Golden ratio: $\lambda = \frac{1+\sqrt{5}}{2}$

⋆ Essential property:



$a+b$ is to a as a is to b

Algorithm 3 (Golden search)

Initialization: *Start with* $S_0 = [a_0, b_0]$ *and define* $\lambda = \dfrac{\sqrt{5}+1}{2}$

Iterate

- *choose* x_i^- *and* x_i^+ *such that*

$$x_i^- = \frac{\lambda}{\lambda+1} a_{i-1} + \frac{1}{\lambda+1} b_{i-1} \qquad x_i^+ = \frac{1}{\lambda+1} a_{i-1} + \frac{\lambda}{\lambda+1} b_{i-1}$$

- *compute* $f(x_i^-)$ *or* $f(x_i^+)$ *(which one was not computed before)*
- *define the new segment as follows*
  - *if* $f(x_i^-) \leq f(x_i^+)$ *then* $S_i = [a_{i-1}, x_i^+]$
  - *if* $f(x_i^-) \geq f(x_i^+)$ *then* $S_i = [x_i^-, b_{i-1}]$
- *go to step* $i+1$

Until $|S_i|$ *is small enough*

$\star$ Consequence: One of $f(x_i^-)$ and $f(x_i^+)$ was computed previously. Only one evaluation per iteration is needed

$\star$ $|S_N| = \lambda^{-N}|b_0 - a_0|$: same ratio as Fibonacci search

Other methods...

Parabolic approximation knowing the values of f at points a, b, c approximate f by a parabola and choose the next point as

$$x = b - \frac{1}{2}\frac{(b-a)^2(f(b)-f(c)) - (b-c)^2(f(b)-f(a))}{(b-a)(f(b)-f(c)) - (b-c)(f(b)-f(a))}$$

$\star$ this method converges fast if f is close to being quadratic
$\star$ in general, faster methods are combined with robust methods: if the fast method gives an aberrant result at the current iterate, run the robust method instead

Important drawback

$\star$ when using zero-order methods we compare values of the function for different arguments: up to which precision can we detect such differences?
$\star$ near the optimum x^* we have

$$f(x) \approx f(x^*) + \frac{1}{2} f''(x^*)(x - x^*)^2$$

$\star$ if $\frac{1}{2} f''(x^*)(x - x^*)^2 < \varepsilon f(x^*)$ where ε is the machine epsilon (typically around 10^{-16} for double precision) then numerically we don't see any difference between $f(x)$ and $f(x^*)$
$\star$ in conclusion, the algorithm will not be able to tell the difference between $f(x)$ and $f(x^*)$ if

$$|x - x^*| \leq \sqrt{\varepsilon} |x^*| \sqrt{\frac{2|f''(x^*)|}{(x^*)^2 |f(x^*)|}}$$

$\star$ in these cases (in practice, most of the time!), zero-order methods will not be able to obtain precision higher than $\sqrt{\varepsilon}$!!!

Conclusion - zero-order methods

- we may achieve linear convergence rate even with the simple trisection method
- it is important to minimize the number of function evaluations in order to minimize the computational cost of the methods
- with Fibonacci or Golden search we arrive at the best possible convergence ratio of $\lambda^{-1} = 0.61803...$
- if the number of function evaluations is known: use Fibonacci search
- else use Golden search

All of this is to be used when you can't compute the derivatives of f.
!!! As soon as you have access to the derivative, even the most basic algorithm is better than Fibonacci and Golden search, as we will see in the next section !!!

Optimization in dimension 1

- Methods of order zero (without derivatives)
- Methods of order one and above (with derivatives)

Using derivatives...

Assumptions: f is unimodal on $[a, b]$ and is smooth (admits as many derivatives as we want)
Suppose that x^* is a local minimum of f on $[a, b]$

Proposition 9 (Classical result - optimality conditions)

- *If* $x^* \in (a, b)$ *then* $f'(x^*) = 0$ *(x^* is a critical point)*
- *If* $x^* = a$ *then* $f'(x^*) \geq 0$
- *If* $x^* = b$ *then* $f'(x^*) \leq 0$

$\star$ The second and third conditions are called Euler inequalities

Towards an algorithm...

$\star$ Direct consequence of unimodality: if $a < x^* < b$ is the minimizer of f on $[a, b]$ then

$$f'(x) < 0 \text{ for } x \in [a, x^*) \quad \text{and} \quad f'(x) > 0 \text{ for } x \in (x^*, b]$$

$\star$ Therefore, if we choose one intermediary point $a < x_n < b$ then we know the position of x^* w.r.t. x_n by looking at $f'(x_n)$

$\star$ Note that, compared to zero-order methods, one intermediary point is enough in order to reduce the size of the search interval

Simplest algorithm

Algorithm 4 (Bisection)

Initialization: $S_0 = [a_0, b_0]$, $i = 1$

Loop:

- *choose* $x_i = 0.5(a_{i-1} + b_{i-1})$
- *compute* $f'(x_i)$
  - **if** $f'(x_i) < 0$ *then* $S_i = [x_i, b]$
  - **if** $f'(x_i) > 0$ *then* $S_i = [a, x_i]$
  - **if** $f'(x_i) = 0$ *then* $x^* = x_i$ *and* **stop**
- *replace* i *with* $i + 1$ *and continue until the desired precision is reached*

$\star$ the third option ($f'(x_i) = 0$ can (almost) never be verified numerically) when working with fixed machine precision for general functions f

Simplest algorithm

Algorithm 4 (Bisection)

Initialization: $S_0 = [a_0, b_0]$, $i = 1$

Loop:

- *choose* $x_i = 0.5(a_{i-1} + b_{i-1})$
- *compute* $f'(x_i)$
  - **if** $f'(x_i) \leq 0$ *then* $S_i = [x_i, b]$
  - **if** $f'(x_i) > 0$ *then* $S_i = [a, x_i]$
  - ~~**if** $f'(x_i) = 0$ *then* $x^* = x_i$ *and* **stop**~~
- *replace* i *with* $i + 1$ *and continue until the desired precision is reached*

$\star$ the third option ($f'(x_i) = 0$ can (almost) never be verified numerically) when working with fixed machine precision for general functions f

Convergence rate

Proposition 10

The Bisection algorithm converges linearly with ratio 0.5.

Proof: $|S_i| = 0.5|S_{i-1}|$ therefore

$$|x^* - x_N| \leq 0.5^N (b - a).$$

$\star$ Already better than the Fibonacci/Golden search algorithms.
$\star$ Is there a contradiction between the optimality of their claimed optimal rate/ratio of convergence and the result stated above?

Convergence rate

Proposition 10

The Bisection algorithm converges linearly with ratio 0.5.

Proof: $|S_i| = 0.5|S_{i-1}|$ therefore

$$|x^* - x_N| \leq 0.5^N (b - a).$$

$\star$ Already better than the Fibonacci/Golden search algorithms.
$\star$ Is there a contradiction between the optimality of their claimed optimal rate/ratio of convergence and the result stated above?

Answer: No, since the Bisection algorithm uses information about derivatives $f'(x_i)$ of the function f while Fibonacci/Golden search algorithms use only the values of f.

Convergence rate

Proposition 10

The Bisection algorithm converges linearly with ratio 0.5.

Proof: $|S_i| = 0.5|S_{i-1}|$ therefore

$$|x^* - x_N| \leq 0.5^N (b - a).$$

$\star$ Already better than the Fibonacci/Golden search algorithms.
$\star$ Is there a contradiction between the optimality of their claimed optimal rate/ratio of convergence and the result stated above?

Answer: No, since the Bisection algorithm uses information about derivatives $f'(x_i)$ of the function f while Fibonacci/Golden search algorithms use only the values of f.
$\star$ Bisection method can be seen as a search for a zero of f'. For a general function f such that $f'(a)f'(b) \leq 0$ it will converge to a critical point of f

Convergence rate

Proposition 10

The Bisection algorithm converges linearly with ratio 0.5.

Proof: $|S_i| = 0.5|S_{i-1}|$ therefore

$$|x^* - x_N| \leq 0.5^N (b - a).$$

$\star$ Already better than the Fibonacci/Golden search algorithms.
$\star$ Is there a contradiction between the optimality of their claimed optimal rate/ratio of convergence and the result stated above?

Answer: No, since the Bisection algorithm uses information about derivatives $f'(x_i)$ of the function f while Fibonacci/Golden search algorithms use only the values of f.
$\star$ Bisection method can be seen as a search for a zero of f'. For a general function f such that $f'(a)f'(b) \leq 0$ it will converge to a critical point of f
$\star$ Can we reach machine precision using the bisection method? The answer is yes: we compare the values of f' with 0!

Further improvements...

$\star$ all methods presented so far possess global linear convergence assuming that f is unimodal.

$\star$ Can we hope for something better?

Further improvements...

$\star$ all methods presented so far possess global linear convergence assuming that f is unimodal.
$\star$ Can we hope for something better?

Use curve fitting: approximate f locally by a simple function with analytically computable minimum.

Basic ideas:

- for each iteration: a set of working points for which we compute the values and (eventually) the derivatives
- construct an approximating polynomial p
- find analytically the minimum of p and update the family of working points

First example: Newton method

$\star$ suppose that given x we can compute $f(x), f'(x), f''(x)$

Algorithm 5 (Newton method in dimension one)

Initialization: *Choose the starting point* x_0

Step i**:**

- *Compute* $f(x_{i-1}), f'(x_{i-1}), f''(x_{i-1})$ *and approximate* f *around* x_{i-1} *by its second-order Taylor expansion*

$$p(x) = f(x_{i-1}) + f'(x_{i-1})(x - x_i) + \frac{1}{2} f''(x_{i-1})(x - x_{i-1})^2.$$

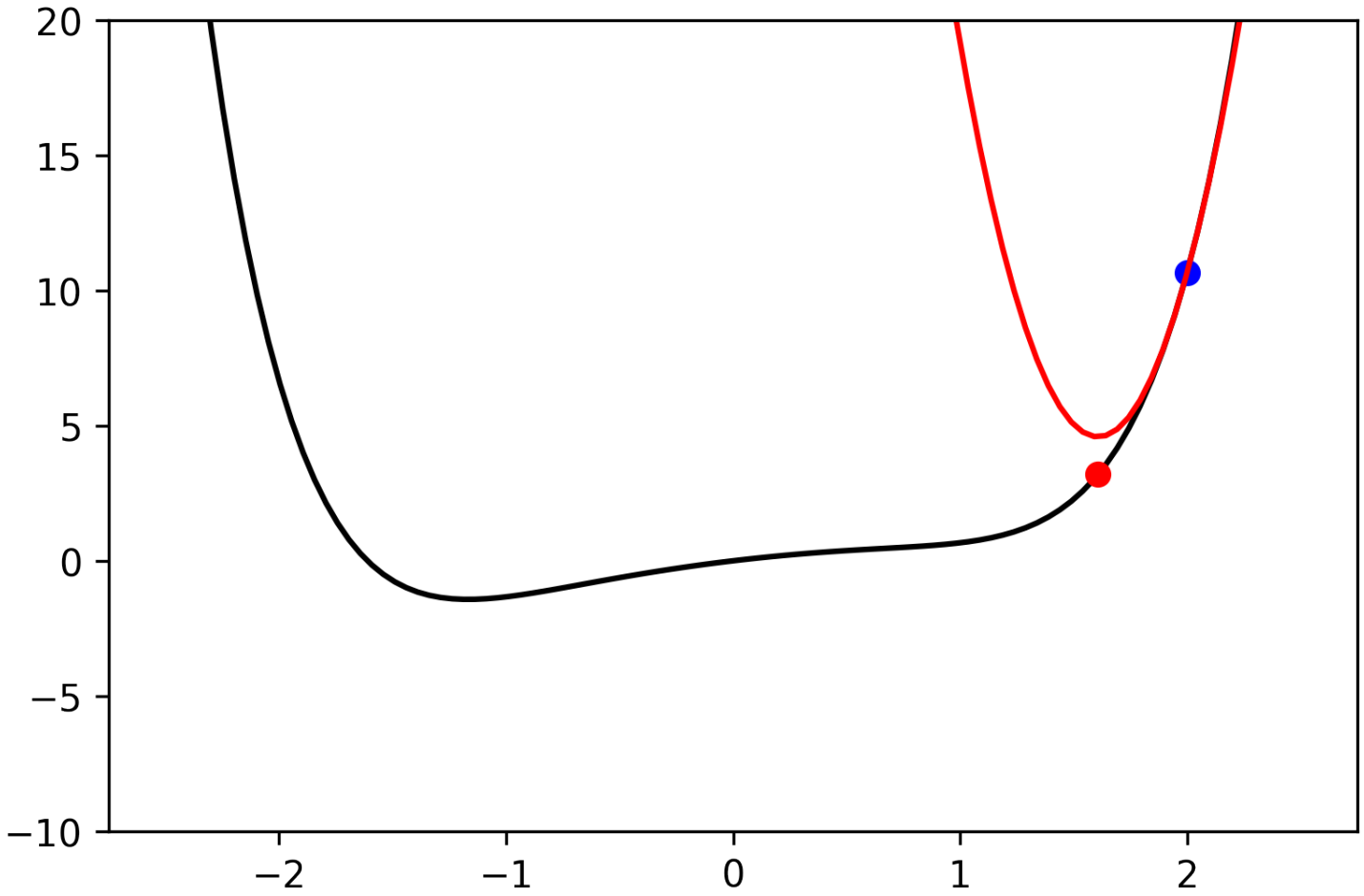
- *choose* x_i *as the critical point of the quadratic function* p*:*

$$x_i = x_{i-1} - \frac{f'(x_{i-1})}{f''(x_{i-1})}.$$

- *replace* i *with* $i + 1$ *and loop*

Example

$f(x) = x^6/6 - x^2/2 + x$ on $[-2.5, 2.5]$, $x_0 = 2$.



Fast convergence...

Proposition 11

Let $x^* \in \mathbb{R}$ be a local minimizer of a smooth function f such that $f'(x^*) = 0$ and $f''(x^*) > 0$. Then the Newton method converges to x^* quadratically, provided that the starting point x_0 is close enough to x^*.

Fast convergence...

Proposition 11

Let $x^* \in \mathbb{R}$ be a local minimizer of a smooth function f such that $f'(x^*) = 0$ and $f''(x^*) > 0$. Then the Newton method converges to x^* quadratically, provided that the starting point x_0 is close enough to x^*.

All the hypotheses are essential!

- What happens for $f(x) = x^4$? Which hypothesis is not verified? Does the algorithm converge for every starting point x_0? What is the observed convergence rate of the algorithm?
- What happens for $f(x) = \sqrt{1 + x^2}$? Does the algorithm converge for every starting point x_0?

Fast convergence...

Proposition 11

Let $x^* \in \mathbb{R}$ be a local minimizer of a smooth function f such that $f'(x^*) = 0$ and $f''(x^*) > 0$. Then the Newton method converges to x^* quadratically, provided that the starting point x_0 is close enough to x^*.

All the hypotheses are essential!

- What happens for $f(x) = x^4$? Which hypothesis is not verified? Does the algorithm converge for every starting point x_0? What is the observed convergence rate of the algorithm?
  Answer: $x^* = 0$, $f''(x^*) = 0$, $x_i = \frac{2}{3}x_{i-1}$. The convergence rate is linear.
- What happens for $f(x) = \sqrt{1 + x^2}$? Does the algorithm converge for every starting point x_0?
  Answer: $x^* = 0$, $f''(x^*) > 0$, $x_i = -x_{i-1}^3$. The convergence rate is cubic when $|x_0| < 1$, but the algorithm does not converge at all for $|x_0| \geq 1$.

Another point of view

⋆ Newton's method can be seen a linearization method for finding the zeros of $g = f'$.
⋆ Indeed, $g(x) = g(x_{i-1}) + g'(x_{i-1})(x - x_{i-1}) + o(|x - x_{i-1}|)$
⋆ Imposing that the linear part is zero amounts to

$$x = -\frac{g(x_{i-1})}{g'(x_{i-1})} + x_{i-1}$$

which is exactly the Newton method

Modified Newton: degenerate case

$\star$ it is possible to show that when $f''(x^*) = 0$ then the rate of convergence is linear

$\star$ if the multiplicity m of the root x^* of f' is known then the following modified Newton method converges quadratically (if it is well defined...)

$$x_{n+1} = x_n - m\frac{f'(x_n)}{f''(x_n)}.$$

$\star$ in practice this does not really help: you don't know the multiplicity *a priori* for a general function f!

A second example: Regula Falsi

⋆ approximate f again by a quadratic polynomial
⋆ we consider two working points with first order information
⋆ given the two last iterates x_{i-1} and x_{i-2} we may approximate $f''(x_{i-1})$ using finite differences

$$f''(x_{i-1}) \approx \frac{f'(x_{i-1}) - f'(x_{i-2})}{x_{i-1} - x_{i-2}}$$

A second example: Regula Falsi

Algorithm 6 (False Position Method)

Initialization: *Choose the starting points* x_0, x_1.
Step $i \geq 2$**:**

- *Compute* $f(x_{i-1}), f'(x_{i-1}), f'(x_{i-2})$ *and approximate* f *around* x_{i-1} *with a second-order polynomial*

$$p(x) = f(x_{i-1}) + f'(x_{i-1})(x - x_i) + \frac{1}{2}\frac{f'(x_{i-1}) - f'(x_{i-2})}{x_{i-1} - x_{i-2}}(x - x_{i-1})^2.$$

- *choose* x_i *as the minimizer of the quadratic function* p*:*

$$x_i = x_{i-1} - f'(x_{i-1})\frac{x_{i-1} - x_{i-2}}{f'(x_{i-1}) - f'(x_{i-2})}.$$

- *replace* i *with* $i + 1$ *and loop*

Remarks

$\star$ The method is symmetric with respect to x_{i-1} and x_{i-2}. It is equivalent to

$$x_i = x_{i-2} - f'(x_{i-2}) \frac{x_{i-1} - x_{i-2}}{f'(x_{i-1}) - f'(x_{i-2})}$$

$\star$ this can be viewed again as a search for a zero of $g = f'$: approximate f' by a straight line through points $(x_{i-1}, f'(x_{i-1}))$ and $(x_{i-2}, f'(x_{i-2}))$.

$\star$ for a non degenerate minimizer x^* of a smooth function f ($f'(x^*) = 0,\ f''(x^*) > 0$) and for x_0, x_1 close enough to x^* the method converges to x^* superlinearly with order of convergence

$$\lambda = (1 + \sqrt{5})/2.$$

$\star$ the Regula Falsi method has a slower convergence rate than Newton's method, but it does not need the knowledge of the second derivative

Cubic fit

⋆ consider two working points x_1 and x_2 with zero and first order information
⋆ define the cubic polynomial such that

$$p(x_1) = f(x_1), p(x_2) = f(x_2), p'(x_1) = f'(x_1), p'(x_2) = f'(x_2)$$

⋆ as the next iterate, choose the local minimizer of p.
⋆ if x^* is non degenerate and the method starts sufficiently close to x^* then the method converges quadratically
⋆ formulas: ... too complicated ... if you are interested, ask for references
⋆ curve fitting is used with polynomials of small degree: we need to be able to compute analytically position of the minima: therefore, there is no point using approximating polynomials of degree higher than four!

Conclusion: curve fitting - towards descent methods

- when it works we achieve superlinear convergence
- What to do when these methods do not work?
  - alternate zero-order or bisection search methods with curve fitting (in cases where curve fitting gives iterates outside the desired search region)
  - at each iteration be sure to decrease the objective function: each method produces a descent direction so we should choose a smaller step size

⋆ Descent direction in 1D:

- if $f'(x) \neq 0$ there are only two options: go left or go right
- choose the direction $d \in \{-1, +1\}$ which decreases f.
- first order Taylor expansion:

$$f(x + \gamma d) = f(x) + \gamma d f'(x) + o(\gamma)$$

- if $df'(x) < 0$ then if γ is small enough then

$$f(x + \gamma d) < f(x)$$

Unexact line search

$\star$ big question: how to choose a descent step?
$\star$ the 1D reasoning will be useful in higher dimensions

Denote $q(t) = f(x + td)$ where d is a descent direction (with $d \in \{\pm 1\}$ in 1D or general in nD), sometimes called merit function.
$\star$ Note that if d is a descent direction, then $q'(0) < 0$

We perform a test for t, with three options

a) t is good

b) t is too big

c) t is too small

We should be able to answer these questions by looking at $q(t)$ and $q'(t)$.
$\star$ perform an iterative process, at each step reducing a confidence interval $[t_l, t_r]$ for t

Generic line-search algorithm

Algorithm 7 (Line-search)

Start with $t_l = 0, t_r = 0$ *and pick an initial* $t > 0$.

Iterate:

Step 1:

If *a) then exit: you found a good t*

If *b) then* $t_r = t$*: you found a new upper bound for t*

If *c) then* $t_l = t$*: you found a new lower bound for t*

Step 2:

If *no valid* t_r *exists we choose a new* $t > t_l$ *(extrapolation step)*

Else *choose a new* $t \in (t_l, t_r)$ *(interpolation step)*

$\star$ a), b), c) should form a partition of $\mathbb{R}_+$

$\star$ if t is big enough c) should be false

$\star$ each interval $[t_l, t_r]$ should contain a sub-interval verifying a)

Armijo's rule

$\star$ $m_1 \in (0,1)$ and $\eta > 1$ are chosen constants.
$\star$ we fix an initial choice of $t = t_0$ (for example $t = 1$)
$\star$ recall that $q'(0) < 0$

a) $\dfrac{q(t) - q(0)}{t} \leq m_1 q'(0)$ (if this is true then t is good)

b) $m_1 q'(0) < \dfrac{q(t) - q(0)}{t}$ (if this is true then t is too big, so $t_r = t$)

c) never

$\star$ if t is too big, then the next t is chosen as t/η (a popular choice is $\eta = 2$).

Proposition 12

Suppose that $q \in C^1$ is bounded from below and $q'(0) < 0$. Then the linear search with Armijo's rule finishes in a finite number of steps.

Convergence may be slow in some cases since we choose once and for all a **maximal step**.

Goldstein-Price rule

$\star$ $m_1 < m_2 \in (0,1)$ are chosen constants
$\star$ recall that $q'(0) < 0$

a) $m_2 q'(0) \leq \frac{q(t)-q(0)}{t} \leq m_1 q'(0)$ (then we have a good t)

b) $m_1 q'(0) < \frac{q(t)-q(0)}{t}$ (then t is too big)

c) $\frac{q(t)-q(0)}{t} < m_2 q'(0)$ (then t is too small)

Proposition 13

Suppose that $q \in C^1$ is bounded from below and $q'(0) < 0$. Then the linear search with the Goldstein-Price rule finishes in a finite number of steps.

$\star$ What about the choice of the constants m_1, m_2?

Wolfe rule

$\star$ $m_1, m_2 \in (0,1)$ are chosen constants
$\star$ recall that $q'(0) < 0$

a) $\frac{q(t)-q(0)}{t} \leq m_1 q'(0)$ and $q'(t) \geq m_2 q'(0)$ (then we have a good t)

b) $\frac{q(t)-q(0)}{t} > m_1 q'(0)$ (then t is too big)

c) $\frac{q(t)-q(0)}{t} \leq m_1 q'(0)$ and $q'(t) < m_2 q'(0)$ (then t is too small)

Proposition 14

Suppose that $q \in C^1$ is bounded from below and $q'(0) < 0$. Then the linear search with the Wolfe rule finishes in a finite number of steps.

$\star$ What about the choice of the constants m_1, m_2?

The quadratic case

Proposition 15

Suppose that q is quadratic with minimum t^*. Then $q(t^*) = q(0) + \frac{1}{2}q'(0)t^*$.

$\star$ we should not refuse the optimal step when q is quadratic!!

$$\frac{q(t^*) - q(0)}{t^*} = \frac{1}{2}q'(0).$$

In conclusion it is recommended to:
$\star$ choose $m_1 < 0.5$ (for Armijo and Goldstein-Price)
$\star$ choose $0.5 < m_2 < 1$ (for Goldstein-Price)

Finally...

Algorithm 8 (Generic gradient descent algorithm)

Initialization: *Choose an initial point x_0 and the eventual parameters for the line-search algorithm*

Step *i:*

- *compute the function value $f(x_{i-1})$ and the derivative $f'(x_{i-1})$*
- *perform the line-search algorithm in order to find a proper descent step t.*
- *choose the next iterate*

$$x_i = x_{i-1} - tf'(x_{i-1}).$$

Stopping criterion: *$|f'(x_i)|$ is small, $|f(x_{i-1}) - f(x_i)|$ is small, the descent step t is too small, maximum number of iterations reached, etc.*

⋆ $f'(x_{i-1})$ can be replaced with any descent direction d.
⋆ various simplified variants exist: fixed descent step, variable descent step
⋆ the generalization to higher dimensions is straightforward

Convergence rate?

$\star$ it is a order 1 algorithm so *a priori* we cannot expect more than linear convergence

$\star$ if $f(x) = x^2$ and we use a fixed step algorithm then the update at each iteration is

$$x_i = x_{i-1} - tf'(x_{i-1}) = (1 - 2t)x_{i-1}.$$

therefore, for $t < 0.5$ we have linear convergence to the optimum.

$\star$ the function $f(x) = x^2$ is strictly convex and quadratic: the ideal case. Therefore we cannot expect something better.

$\star$ locally, around a minimizer x^* the function f is convex. Therefore, if convergence is proved for convex functions, it will follow, that locally, around the minimizer, the convergence is linear

Example of global convergence result

Proposition 16 (Convergence rate for the gradient descent with fixed step)

Suppose that $f : \mathbb{R} \to \mathbb{R}$ is of class C^2 with f' Lipschitz continuous on $\mathbb{R}$: there exists $M > 0$ such that

$$|f'(x) - f'(y)| \leq M|x - y|, \ \forall x, y \in \mathbb{R}.$$

Moreover, suppose that f is α-strictly convex ($f''(x) \geq \alpha > 0$) and that f is ∞ at infinity (so that a minimizer exists).
Then the Gradient Descent algorithm with fixed step t converges to the minimum linearly when t is small enough.

Proof: Define the application $\mathcal{F} : \mathbb{R} \to \mathbb{R}$

$$\mathcal{F}(x) = x - tf'(x)$$

and prove that for t small enough $\mathcal{F}$ is a contraction:

$$|\mathcal{F}(x) - \mathcal{F}(y)| \leq k|x - y|, \quad k \in (0, 1).$$

$\star$ then we know that the fixed point iteration $x_{n+1} = \mathcal{F}(x_n)$ converges to the unique fixed point, which is exactly the optimum.

Example of local result

Proposition 17

Local convergence rate Suppose that $f : [a, b] \to \mathbb{R}$ is unimodal and has a unique minimizer x^* in $[a, b]$. Then if f is of class C^2 and $f''(x^*) > 0$ then the gradient descent algorithm with fixed step t converges linearly to x^* if t is chosen small enough and x_0 is close enough to x^*.

$\star$ use Taylor expansion around x^* to find a recurrence relation for the error!
$\star$ the condition $f''(x^*) > 0$ cannot be ommited: degenerate minimizers will lead to sublinear rate of convergence. Example $f(x) = x^4$.
$\star$ using more involved techniques, it is possible to prove that the gradient descent always converges to a local minimizer, with an eventual sublinear rate of convergence
$\star$ various convergence results can be formulated when using line-search procedures instead of a fixed step: guaranteeing descent is essential for convergence

Conclusions - optimization in dimension one

- there are efficient zero-order algorithms (when derivatives are not available)
- as soon as derivatives can be computed, the convergence is accelerated
- curve-fitting methods give increased convergence rates, but they are sensitive to the initialization
- line-search procedures play an important role even in higher dimensions
- sometimes searching for an optimum is not the main objective but attaining a significant decrease in the objective function is enough
- gradient descent algorithms (almost) always converge to a local minimzer, but the rate of convergence is linear at best