

Optimization Methods for Data Analysis

1. Univariate optimization

Wojciech Kotłowski

Institute of Computing Science, PUT

<http://www.cs.put.poznan.pl/wkotlowski/>

3.03.2022

Outline

Optimization of a single-variable (univariate) function ($x \in \mathbb{R}$):

$$\begin{aligned} & \min f(x) \\ \text{s.t. } & x \in \mathcal{X} \subseteq \mathbb{R} \end{aligned}$$

1. Analytical solutions
2. Uniform search
3. Unimodal function optimization: dichotomous search and golden-section search methods.
4. Optimization of differentiable functions: the bisection method
5. Optimization of differentiable functions: the Newton method

Some vocabulary

- Function f has a **global minimum** at x^* if $f(x^*) \leq f(x)$ for all $x \in \mathcal{X}$
- Function f has a **local minimum** at x^* if $f(x^*) \leq f(x)$ for all x within some neighborhood of x^* , i.e. for $x \in (x^* - \epsilon, x^* + \epsilon) \cap \mathcal{X}$ with $\epsilon > 0$
- If f has (local/global) minimum at x^* then x^* is called a (local/global) **minimizer**

Analytical solutions

Analytical solutions

In some cases one can find the minimum of a function without the need to use numerical methods

Assumptions:

- Function f defined over domain $\mathcal{X} = [x_{\min}, x_{\max}]$
- Function f is **differentiable** on \mathcal{X}

A necessary condition for a local minimum

If a differentiable function f has a **local minimum** at x_0 in the interior of \mathcal{X} , then $f'(x_0) = 0$.

A necessary condition for a local minimum

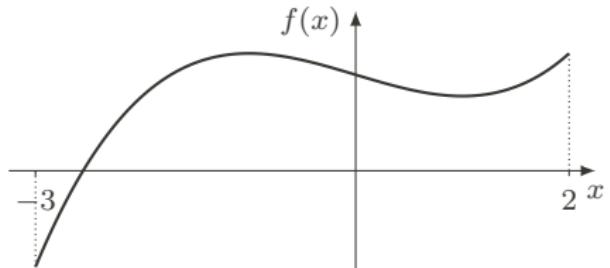
If a differentiable function f has a **local minimum** at x_0 in the interior of \mathcal{X} , then $f'(x_0) = 0$.

Conclusion: A minimum-finding algorithm for a differentiable f over domain $\mathcal{X} = [x_{\min}, x_{\max}]$:

1. Find all points at which the first derivative of f vanishes:
$$\mathcal{X}_0 = \{x \in \mathcal{X}: f'(x) = 0\}$$
2. Check the value of f at each point from \mathcal{X}_0 and on the boundaries of the domain $\{x_{\min}, x_{\max}\}$, and pick the one with the smallest value.

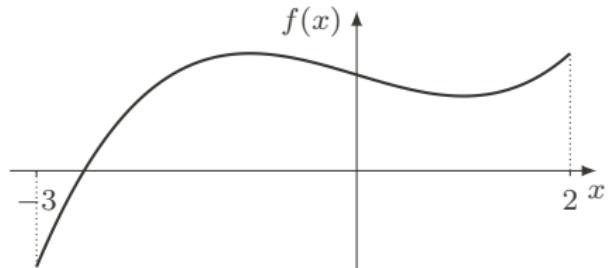
Example

$$\begin{aligned} \min \quad & f(x) = x^3 - 3x + 9 \\ \text{s.t.} \quad & x \in [-3, 2] \end{aligned}$$



Example

$$\begin{aligned} \min \quad & f(x) = x^3 - 3x + 9 \\ \text{s.t.} \quad & x \in [-3, 2] \end{aligned}$$

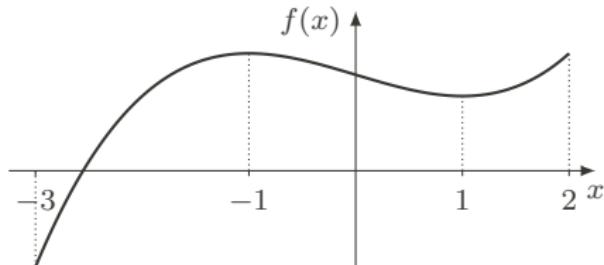


1. Compute the derivative:

$$f'(x) = 3x^2 - 3$$

Example

$$\begin{aligned} \min \quad & f(x) = x^3 - 3x + 9 \\ \text{s.t.} \quad & x \in [-3, 2] \end{aligned}$$



1. Compute the derivative:

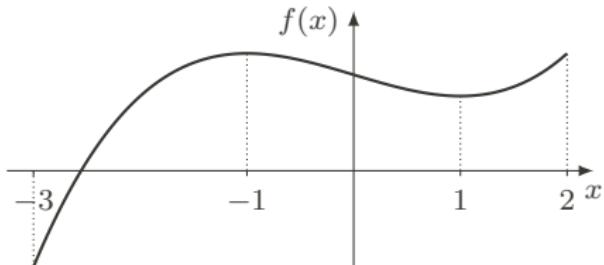
$$f'(x) = 3x^2 - 3$$

2. Find points at which the derivative is zero:

$$f'(x) = 0 \iff 3x^2 - 3 = 0 \iff x = 1 \text{ or } x = -1$$

Example

$$\begin{aligned} \min \quad & f(x) = x^3 - 3x + 9 \\ \text{s.t.} \quad & x \in [-3, 2] \end{aligned}$$



1. Compute the derivative:

$$f'(x) = 3x^2 - 3$$

2. Find points at which the derivative is zero:

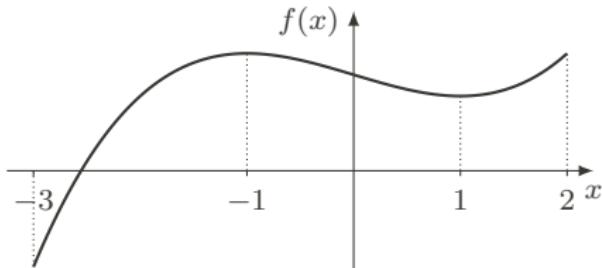
$$f'(x) = 0 \iff 3x^2 - 3 = 0 \iff x = 1 \text{ or } x = -1$$

3. Compute the function value at all candidates for a global minimum:

$$f(-3) = -9, \quad f(-1) = 11, \quad f(1) = 7, \quad f(2) = 11$$

Example

$$\begin{aligned} \min \quad & f(x) = x^3 - 3x + 9 \\ \text{s.t.} \quad & x \in [-3, 2] \end{aligned}$$



1. Compute the derivative:

$$f'(x) = 3x^2 - 3$$

2. Find points at which the derivative is zero:

$$f'(x) = 0 \iff 3x^2 - 3 = 0 \iff x = 1 \text{ or } x = -1$$

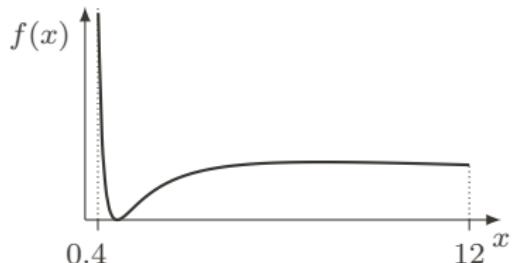
3. Compute the function value at all candidates for a global minimum:

$$f(-3) = -9, \quad f(-1) = 11, \quad f(1) = 7, \quad f(2) = 11$$

4. **Conclusion:** global minimum at $x^* = -3$

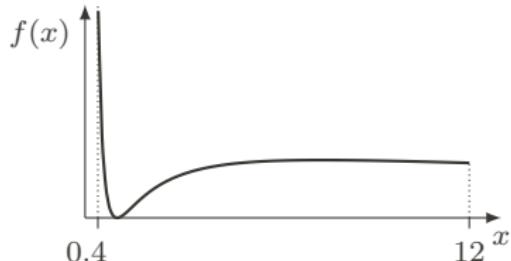
Example

$$\begin{aligned} \min \quad & f(x) = \frac{\ln^2(x)}{x} \\ \text{s.t.} \quad & x \in [0.4, 12] \end{aligned}$$



Example

$$\begin{aligned} \min \quad & f(x) = \frac{\ln^2(x)}{x} \\ \text{s.t.} \quad & x \in [0.4, 12] \end{aligned}$$

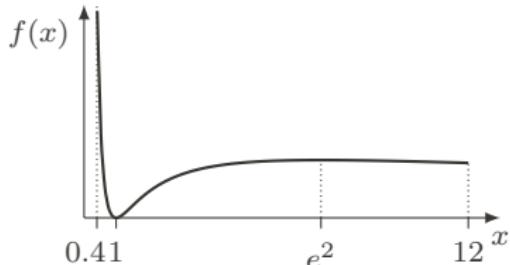


1. Compute the derivative:

$$f'(x) = \frac{2\ln(x) - \ln^2(x)}{x^2} = \frac{\ln(x)(2 - \ln(x))}{x^2}$$

Example

$$\begin{aligned} \min \quad & f(x) = \frac{\ln^2(x)}{x} \\ \text{s.t.} \quad & x \in [0.4, 12] \end{aligned}$$



1. Compute the derivative:

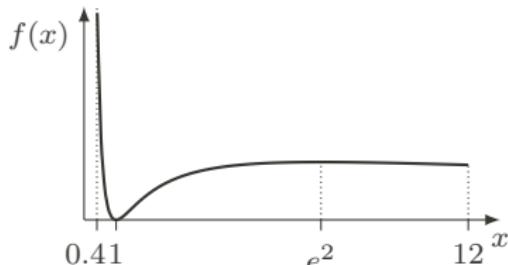
$$f'(x) = \frac{2\ln(x) - \ln^2(x)}{x^2} = \frac{\ln(x)(2 - \ln(x))}{x^2}$$

2. Find points at which the derivative is zero:

$$f'(x) = 0 \iff \ln(x)(2 - \ln(x)) = 0 \iff x = 1 \text{ or } x = e^2$$

Example

$$\begin{aligned} \min \quad & f(x) = \frac{\ln^2(x)}{x} \\ \text{s.t.} \quad & x \in [0.4, 12] \end{aligned}$$



1. Compute the derivative:

$$f'(x) = \frac{2\ln(x) - \ln^2(x)}{x^2} = \frac{\ln(x)(2 - \ln(x))}{x^2}$$

2. Find points at which the derivative is zero:

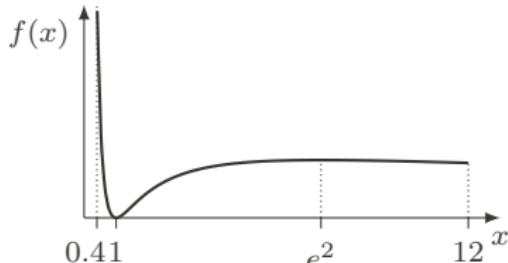
$$f'(x) = 0 \iff \ln(x)(2 - \ln(x)) = 0 \iff x = 1 \text{ or } x = e^2$$

3. Compute the function value at all candidates for global minimum:

$$f(1) = 0, \quad f(e^2) = 0.541, \quad f(0.4) = 2.1, \quad f(12) = 0.515$$

Example

$$\begin{aligned} \min \quad & f(x) = \frac{\ln^2(x)}{x} \\ \text{s.t.} \quad & x \in [0.4, 12] \end{aligned}$$



1. Compute the derivative:

$$f'(x) = \frac{2\ln(x) - \ln^2(x)}{x^2} = \frac{\ln(x)(2 - \ln(x))}{x^2}$$

2. Find points at which the derivative is zero:

$$f'(x) = 0 \iff \ln(x)(2 - \ln(x)) = 0 \iff x = 1 \text{ or } x = e^2$$

3. Compute the function value at all candidates for global minimum:

$$f(1) = 0, \quad f(e^2) = 0.541, \quad f(0.4) = 2.1, \quad f(12) = 0.515$$

4. **Conclusion:** global minimum at $x^* = 1$

Is this all we need in univariate optimization?

Is this all we need in univariate optimization?

Of course, it is not!

- It rarely happens we can analytically derive all points at which the derivative vanishes, i.e. solve $f'(x) = 0$
- The function can be non-differentiable

Is this all we need in univariate optimization?

Of course, it is not!

- It rarely happens we can analytically derive all points at which the derivative vanishes, i.e. solve $f'(x) = 0$
- The function can be non-differentiable

Therefore, we need to resort to **numerical methods**

Uniform search

Problem statement

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & x_{\min} \leq x \leq x_{\max} \end{aligned}$$

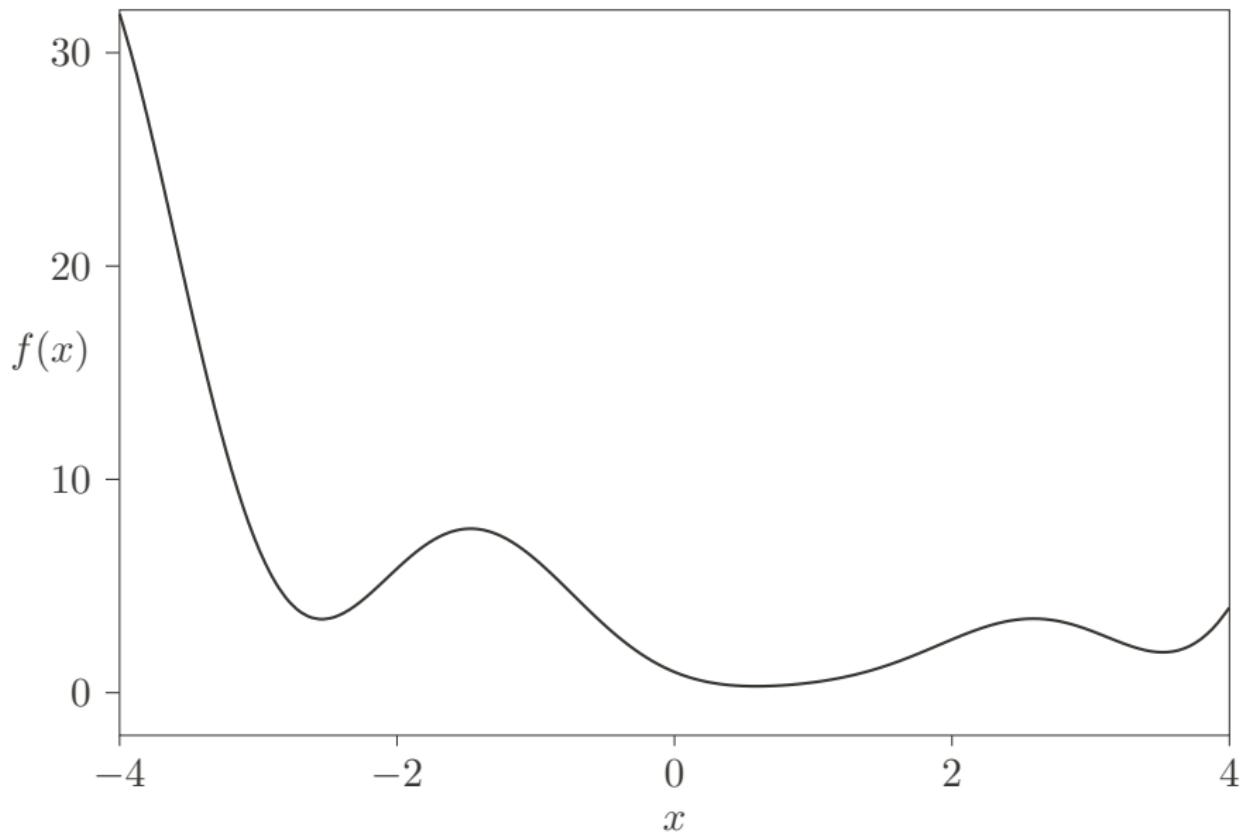
- No assumptions on differentiability
- The function shape unknown

Uniform search method

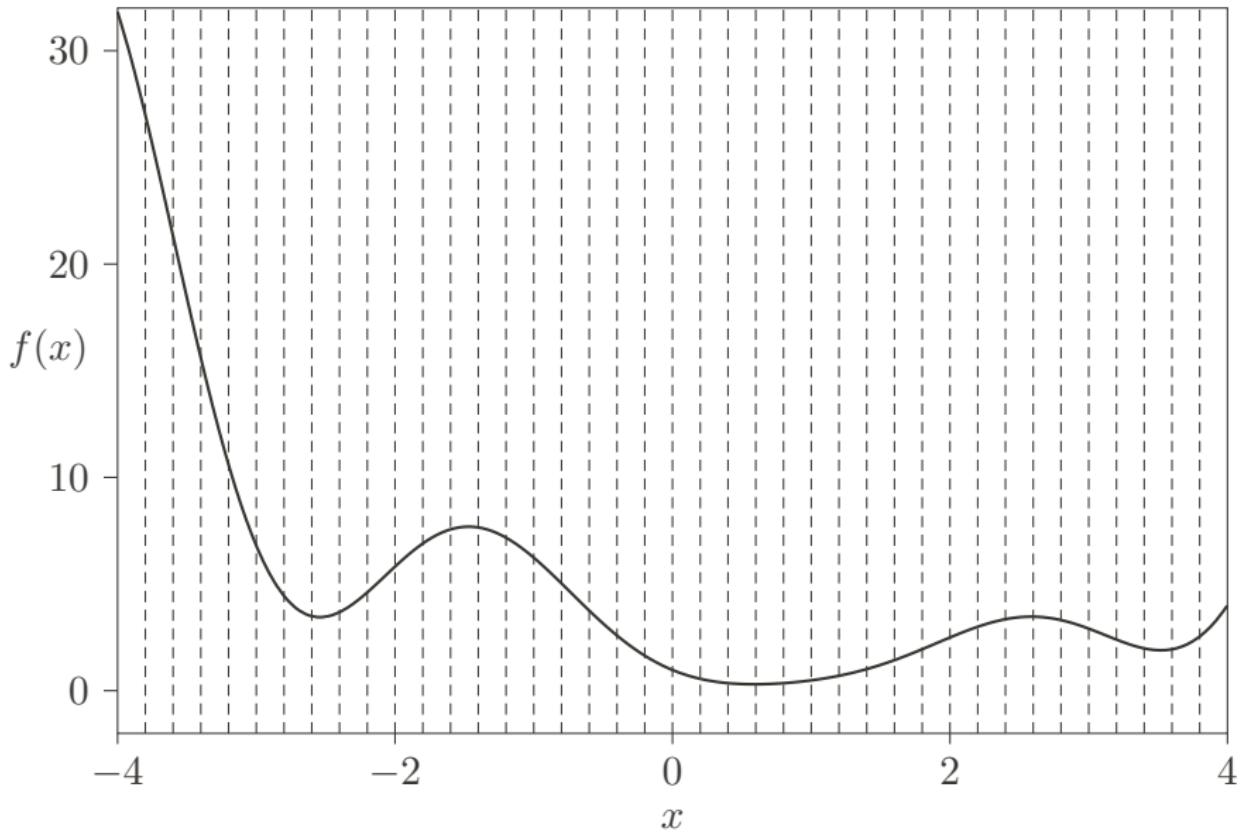
Divide the domain $\mathcal{X} = [x_{\min}, x_{\max}]$ onto a large number of short consecutive intervals, and check the function value in each (e.g., in midpoints of the intervals)

Return point \hat{x} with the smallest value

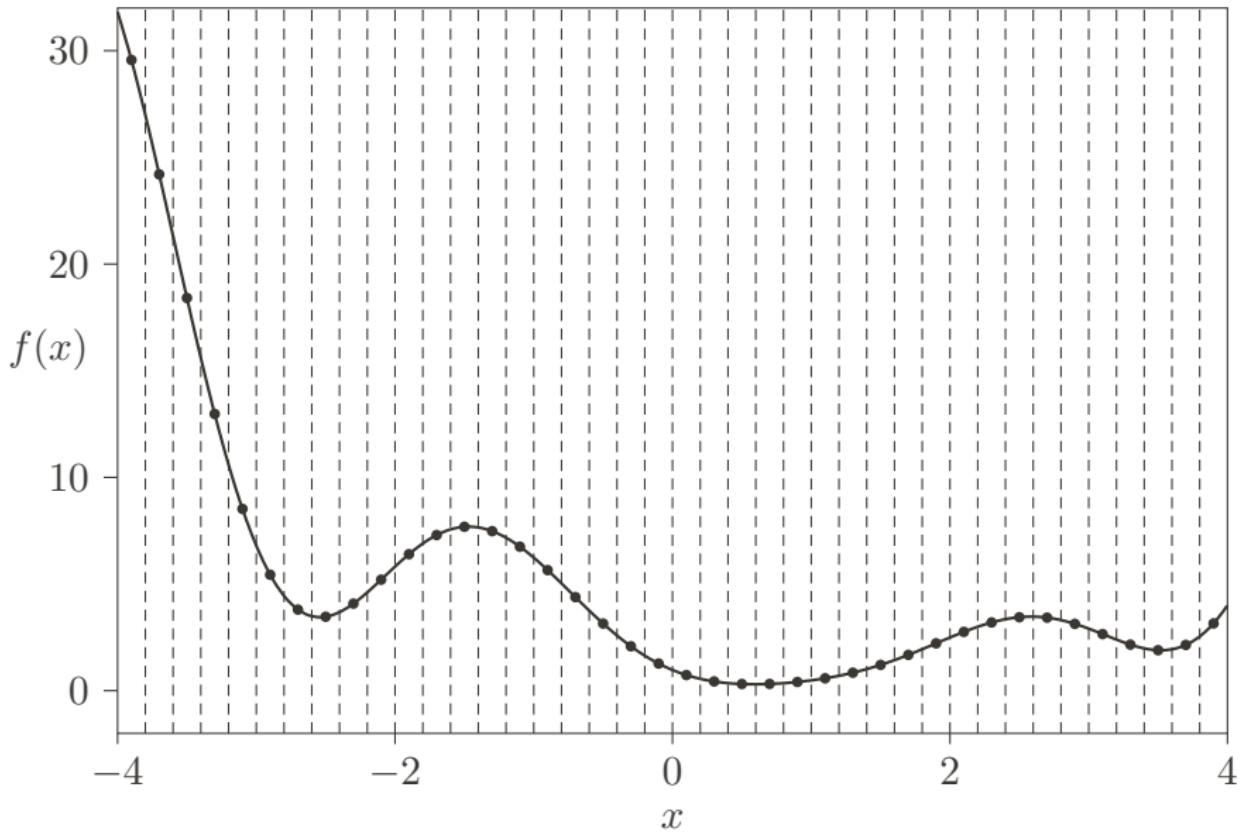
Uniform search – example



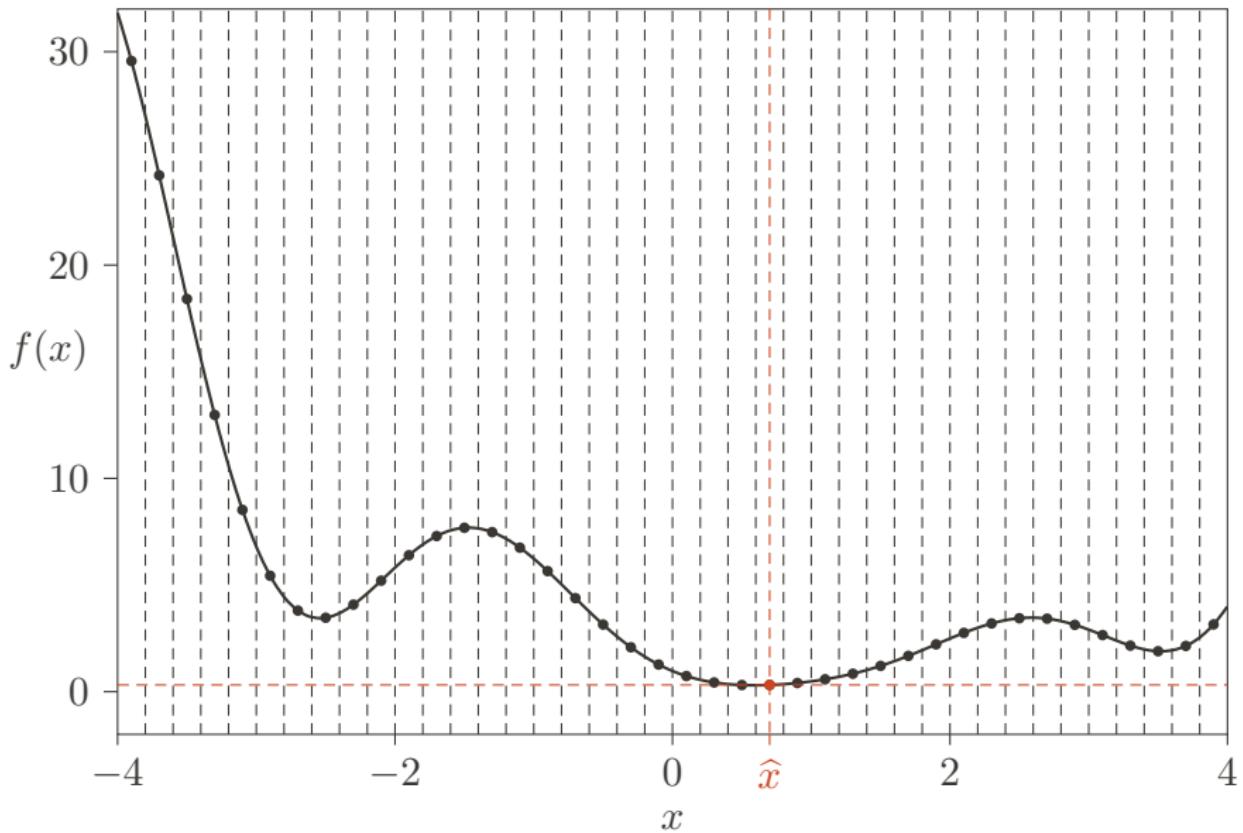
Uniform search – example



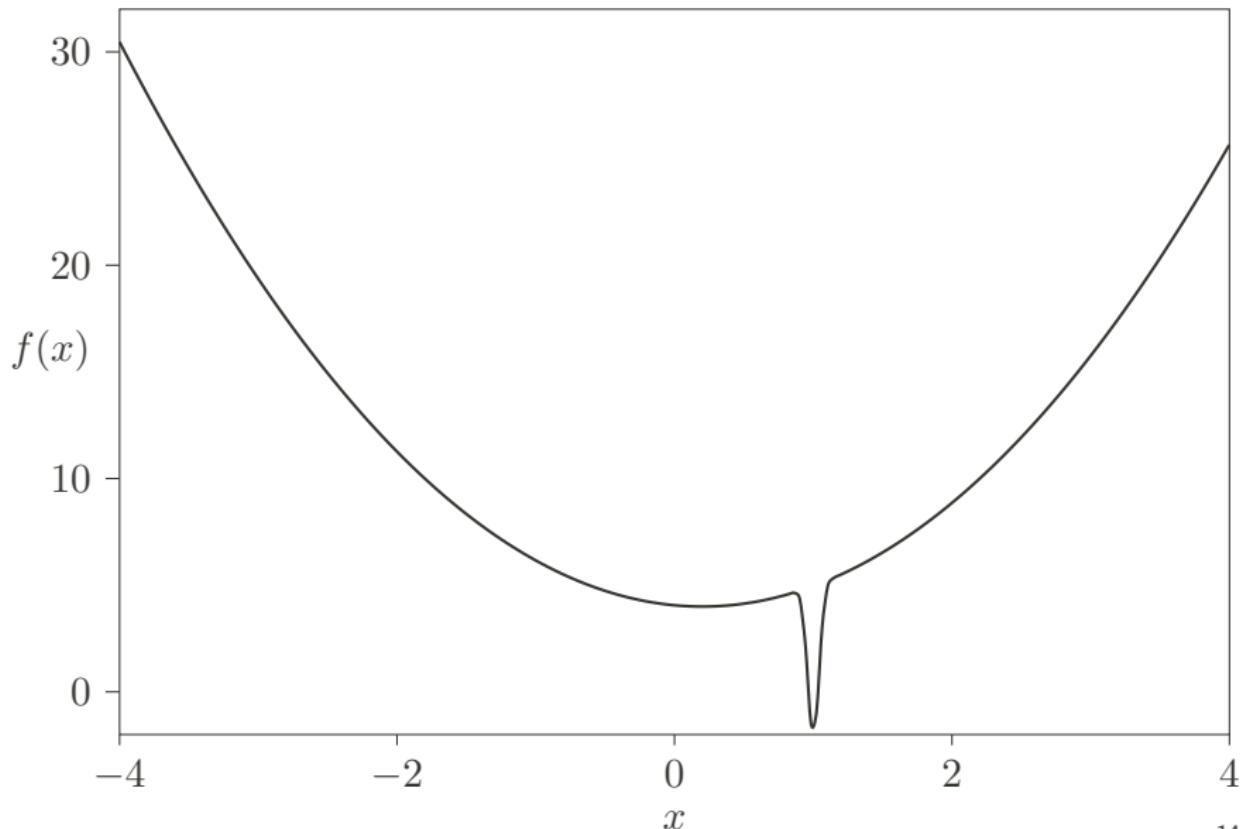
Uniform search – example



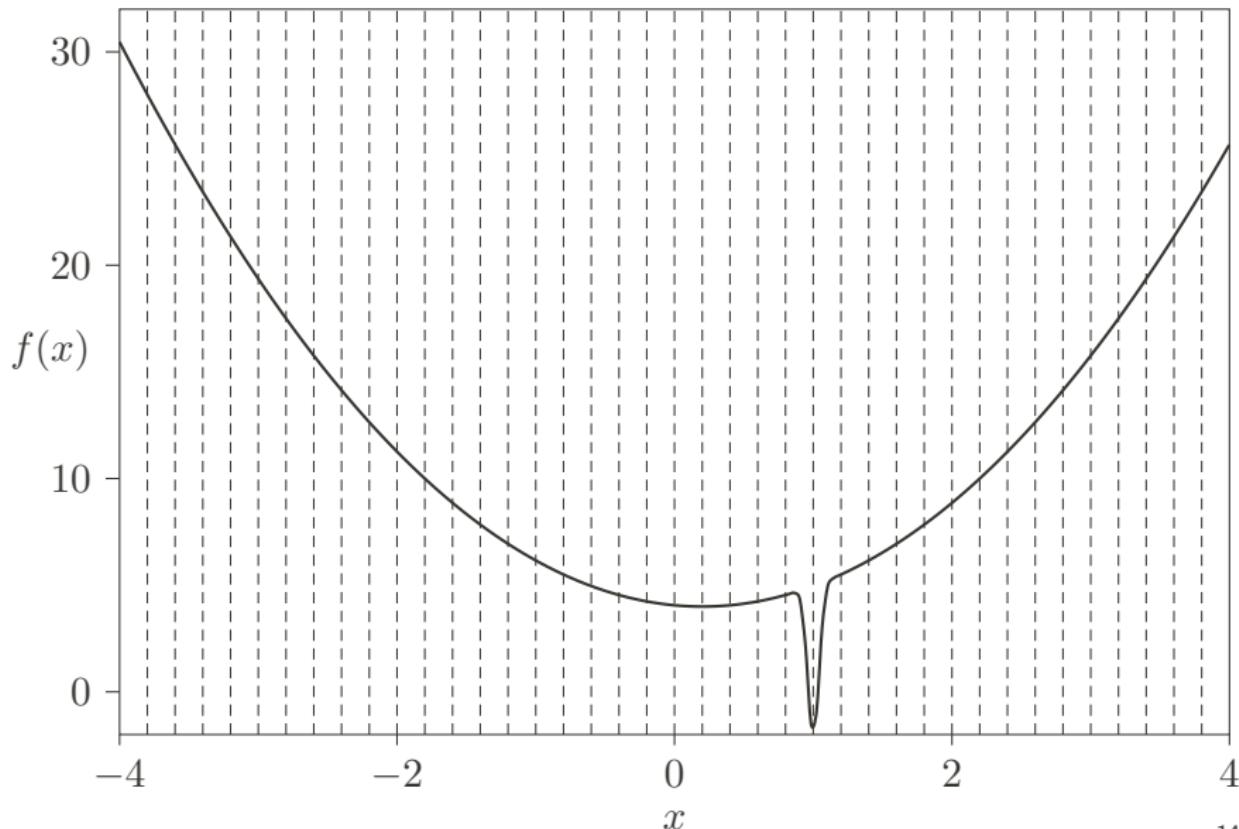
Uniform search – example



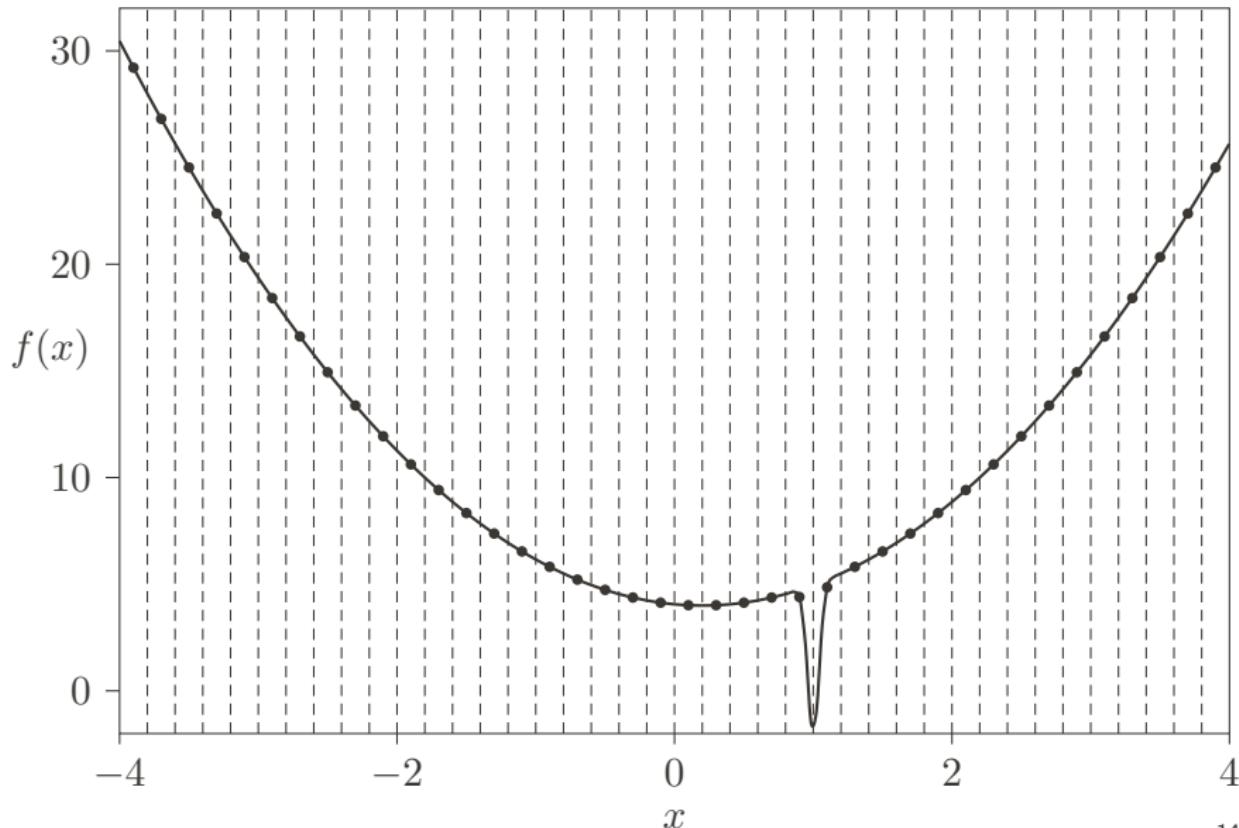
Does uniform search guarantee finding the global minimum?



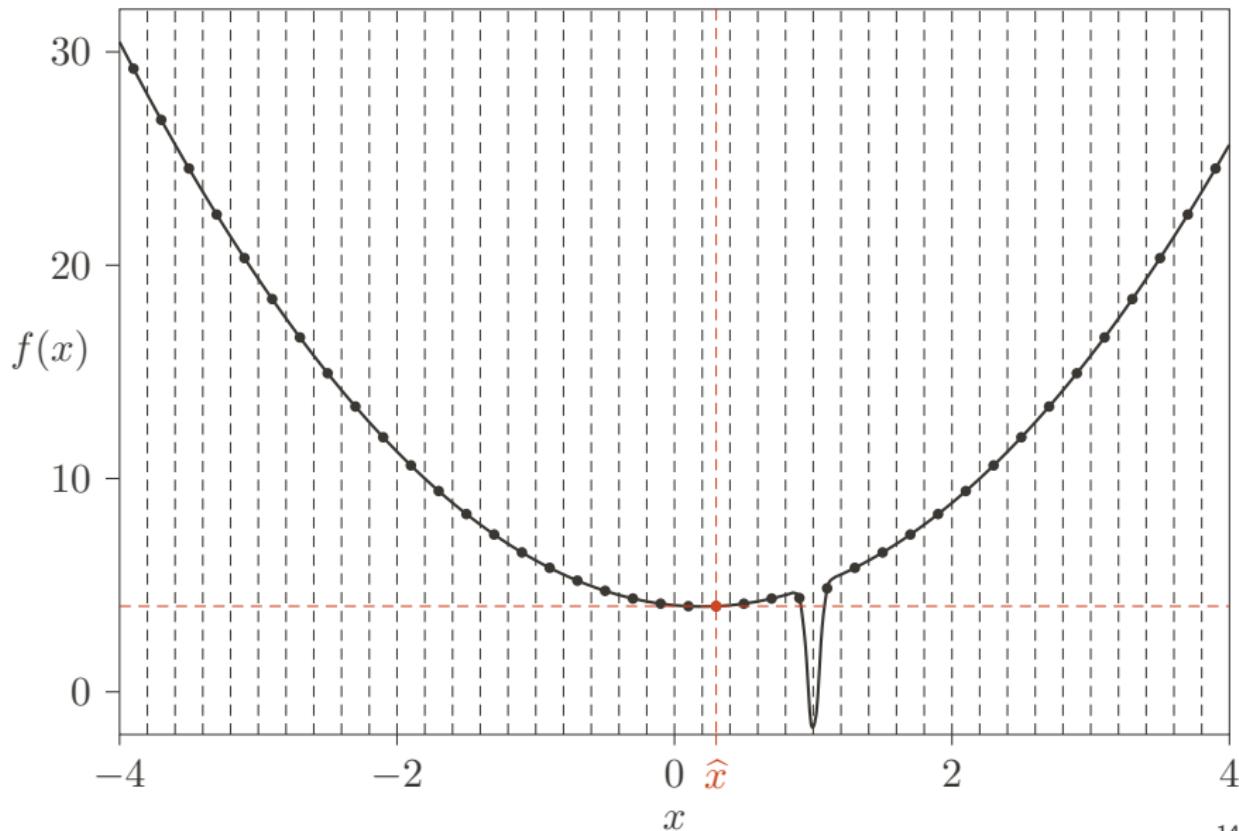
Does uniform search guarantee finding the global minimum?



Does uniform search guarantee finding the global minimum?



Does uniform search guarantee finding the global minimum?



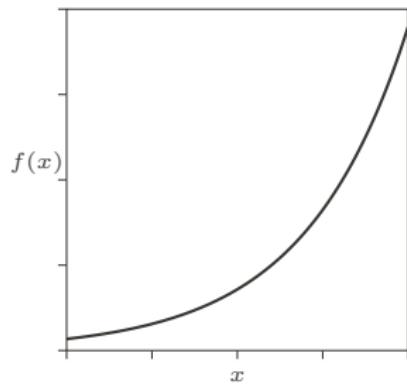
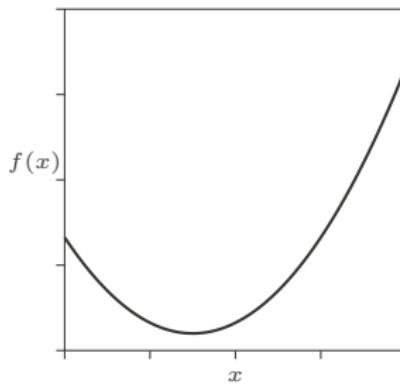
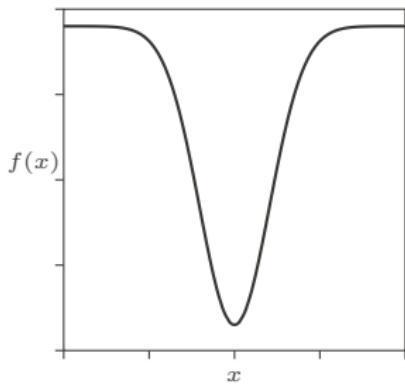
Uniform search

- Scales **linearly** with the number of intervals (very inefficient!)
- Does not guarantee finding the global minimum, not even approximately
(although works well for many practical objective functions)
- The quality of returned solution strongly depends on the regularity of the objective function
- Alternative: **random search** (sample points from the domain and check the function value), has similar drawbacks.
- Hard to do anything better without further assumptions

Optimization of unimodal functions (without access to derivative)

Unimodal function

A **unimodal function** is a function that has only a single (global) minimum (can be on a boundary of the domain)



Properties of a unimodal function

Let a function $f(x)$ has a single (strict) minimum over $[a, b]$.

Pick arbitrary $x_\ell, x_r \in [a, b]$ with $x_\ell < x_r$.

- If $f(x_\ell) \geq f(x_r)$ then f **does not** have a minimum in $[a, x_\ell]$
- If $f(x_\ell) \leq f(x_r)$ then f **does not** have a minimum in $(x_r, b]$

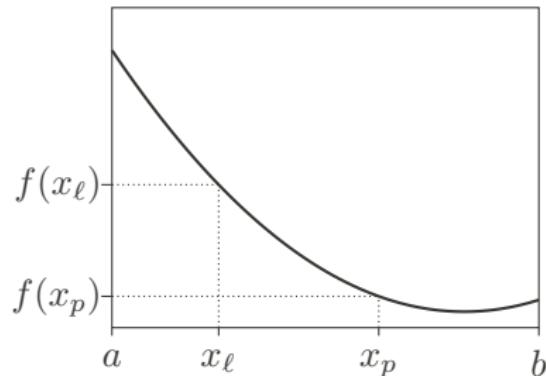
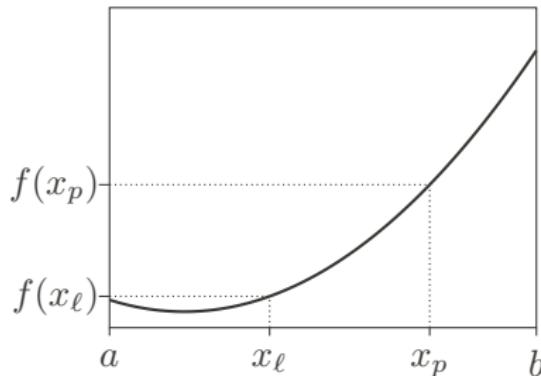
Properties of a unimodal function

Let a function $f(x)$ has a single (strict) minimum over $[a, b]$.

Pick arbitrary $x_\ell, x_r \in [a, b]$ with $x_\ell < x_r$.

- If $f(x_\ell) \geq f(x_r)$ then f **does not** have a minimum in $[a, x_\ell]$
- If $f(x_\ell) \leq f(x_r)$ then f **does not** have a minimum in $(x_r, b]$

Proof: with pictures, by contradiction



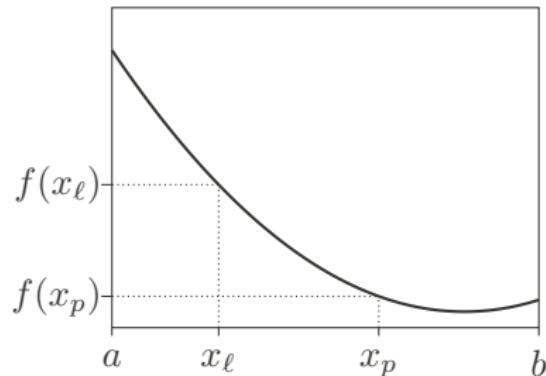
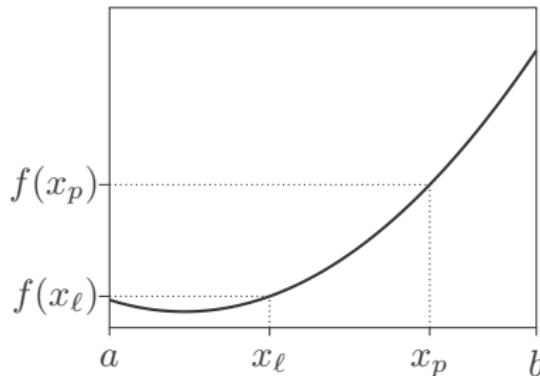
Properties of a unimodal function

Let a function $f(x)$ has a single (strict) minimum over $[a, b]$.

Pick arbitrary $x_\ell, x_r \in [a, b]$ with $x_\ell < x_r$.

- If $f(x_\ell) \geq f(x_r)$ then f **does not** have a minimum in $[a, x_\ell]$
- If $f(x_\ell) \leq f(x_r)$ then f **does not** have a minimum in $(x_r, b]$

Proof: with pictures, by contradiction



Conclusion: evaluating the objective at two points x_ℓ, x_r will always let us discard one of the intervals, $[a, x_\ell]$ or $(x_r, b]$!

Minimum-finding algorithm for unimodal functions

Start with the entire domain $a = x_{\min}$, $b = x_{\max}$

In each iteration:

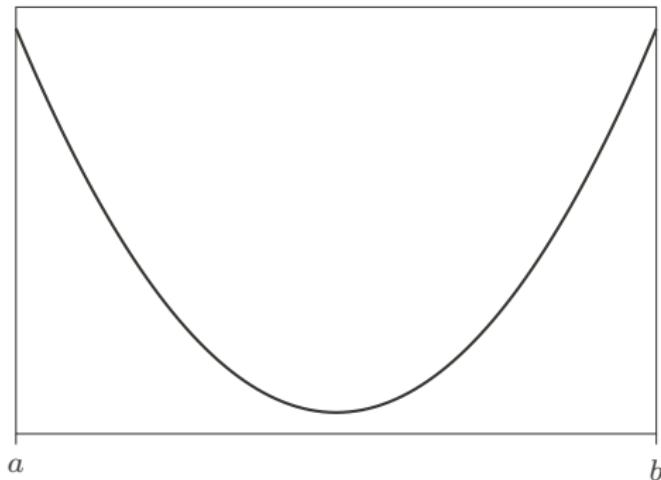
1. Pick $x_\ell, x_r \in [a, b]$ such that $x_\ell < x_r$
2. If $f(x_\ell) \leq f(x_r)$, set $b = x_r$, otherwise set $a = x_\ell$

Minimum-finding algorithm for unimodal functions

Start with the entire domain $a = x_{\min}$, $b = x_{\max}$

In each iteration:

1. Pick $x_\ell, x_r \in [a, b]$ such that $x_\ell < x_r$
2. If $f(x_\ell) \leq f(x_r)$, set $b = x_r$, otherwise set $a = x_\ell$

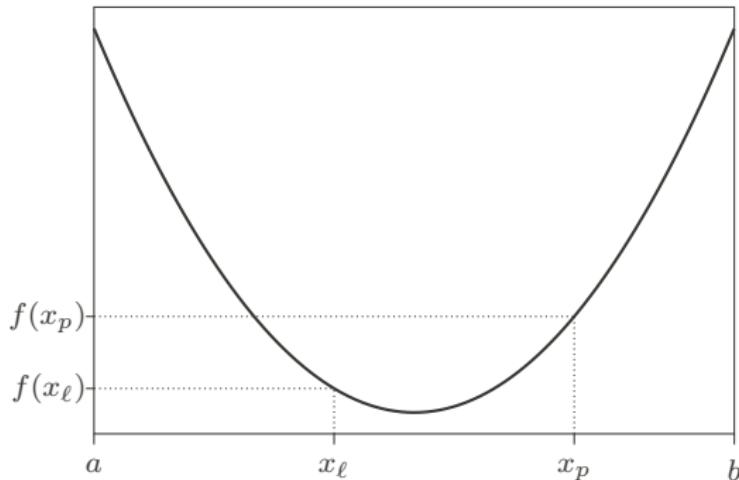


Minimum-finding algorithm for unimodal functions

Start with the entire domain $a = x_{\min}$, $b = x_{\max}$

In each iteration:

1. Pick $x_\ell, x_r \in [a, b]$ such that $x_\ell < x_r$
2. If $f(x_\ell) \leq f(x_r)$, set $b = x_r$, otherwise set $a = x_\ell$

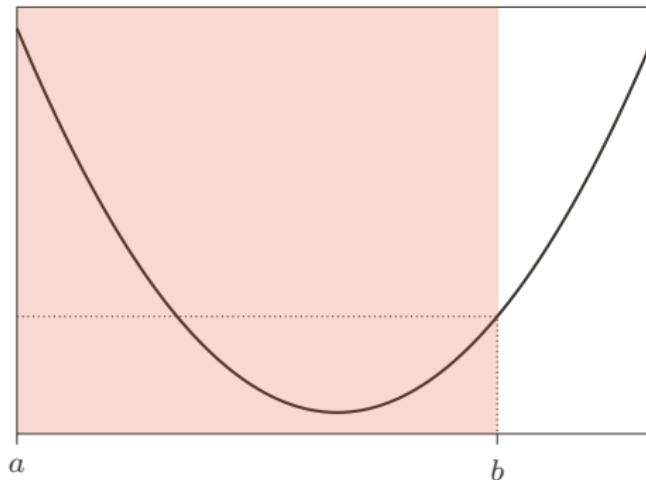


Minimum-finding algorithm for unimodal functions

Start with the entire domain $a = x_{\min}$, $b = x_{\max}$

In each iteration:

1. Pick $x_\ell, x_r \in [a, b]$ such that $x_\ell < x_r$
2. If $f(x_\ell) \leq f(x_r)$, set $b = x_r$, otherwise set $a = x_\ell$

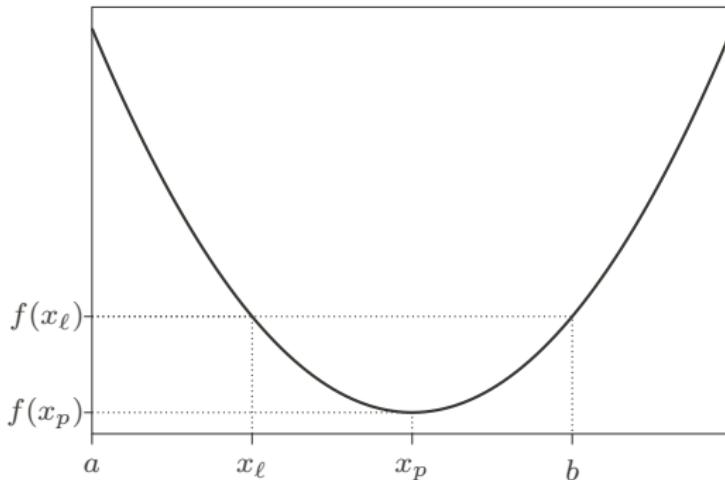


Minimum-finding algorithm for unimodal functions

Start with the entire domain $a = x_{\min}$, $b = x_{\max}$

In each iteration:

1. Pick $x_\ell, x_r \in [a, b]$ such that $x_\ell < x_r$
2. If $f(x_\ell) \leq f(x_r)$, set $b = x_r$, otherwise set $a = x_\ell$

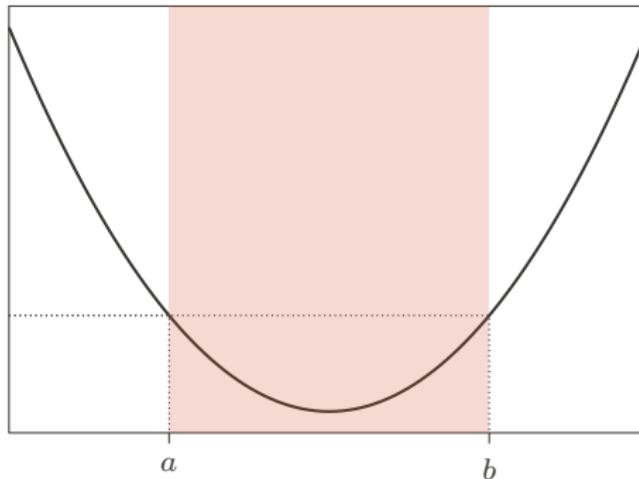


Minimum-finding algorithm for unimodal functions

Start with the entire domain $a = x_{\min}$, $b = x_{\max}$

In each iteration:

1. Pick $x_\ell, x_r \in [a, b]$ such that $x_\ell < x_r$
2. If $f(x_\ell) \leq f(x_r)$, set $b = x_r$, otherwise set $a = x_\ell$

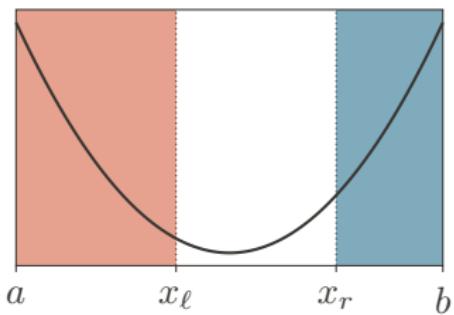


Minimum-finding algorithm for unimodal functions

How to pick points x_ℓ, x_r in order to narrow the search space as much as possible?

Minimum-finding algorithm for unimodal functions

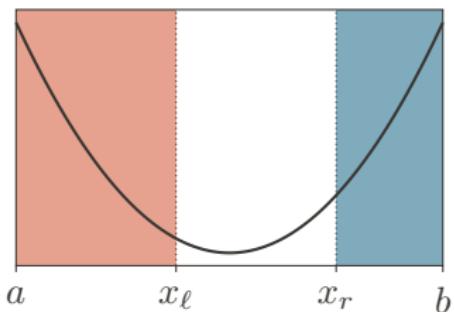
How to pick points x_ℓ, x_r in order to narrow the search space as much as possible?



In each iteration discard one of the colored areas (we do not know which)

Minimum-finding algorithm for unimodal functions

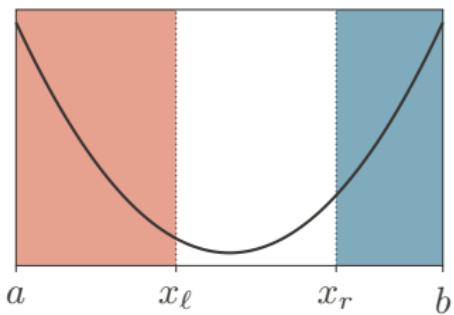
How to pick points x_ℓ, x_r in order to narrow the search space as much as possible?



In each iteration discard one of the colored areas (we do not know which)
Both areas should be **of the same size**

Minimum-finding algorithm for unimodal functions

How to pick points x_ℓ, x_r in order to narrow the search space as much as possible?



In each iteration discard one of the colored areas (we do not know which)
Both areas should be **of the same size**
Points x_ℓ, x_r as close to the center as possible

Dichotomous search

Input: a procedure for evaluating objective $f(x)$ at any point of the domain $[x_{\min}, x_{\max}]$; number of function evaluations n ; a very small constant δ (e.g. $\delta = 10^{-8}$)

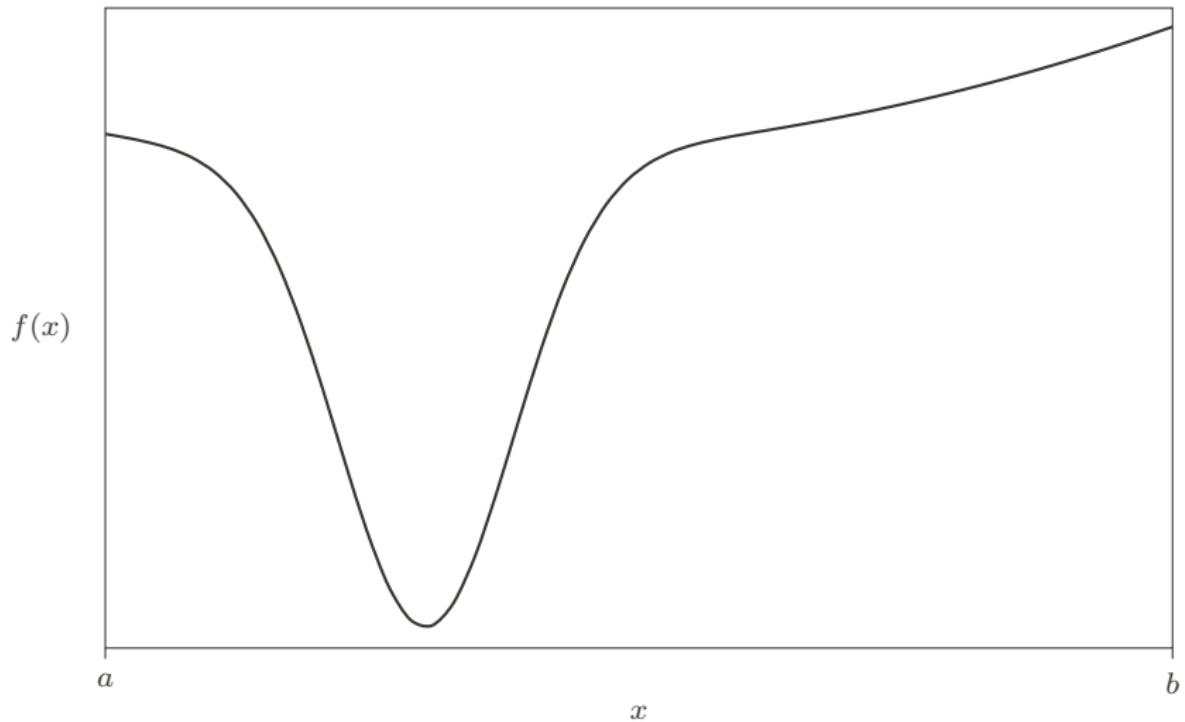
Initialize: $a = x_{\min}, b = x_{\max}$

For $k = 1, 2, \dots, n/2$:

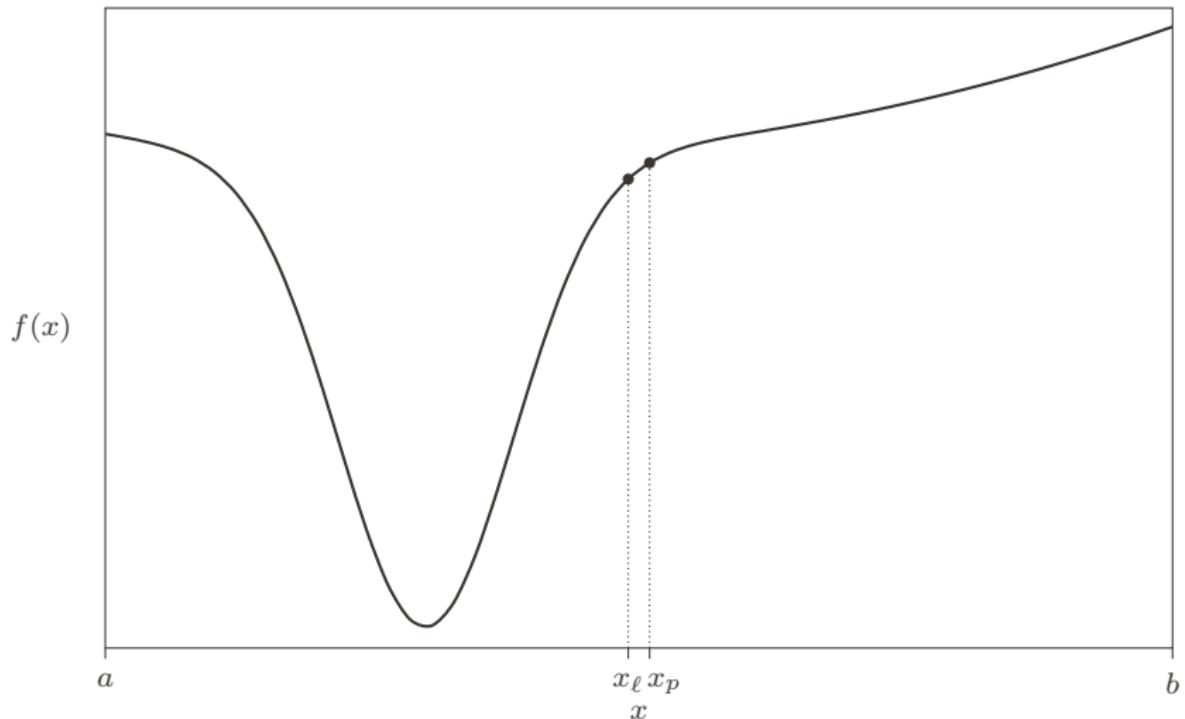
1. Compute the midpoint $m = \frac{a+b}{2}$
2. Set $x_\ell = m - \delta, x_r = m + \delta$
3. Evaluate function values $f(x_\ell)$ i $f(x_r)$
4. If $f(x_\ell) \leq f(x_r)$, set $b = x_r$, otherwise set $a = x_\ell$

Return $x_n = \frac{a+b}{2}$

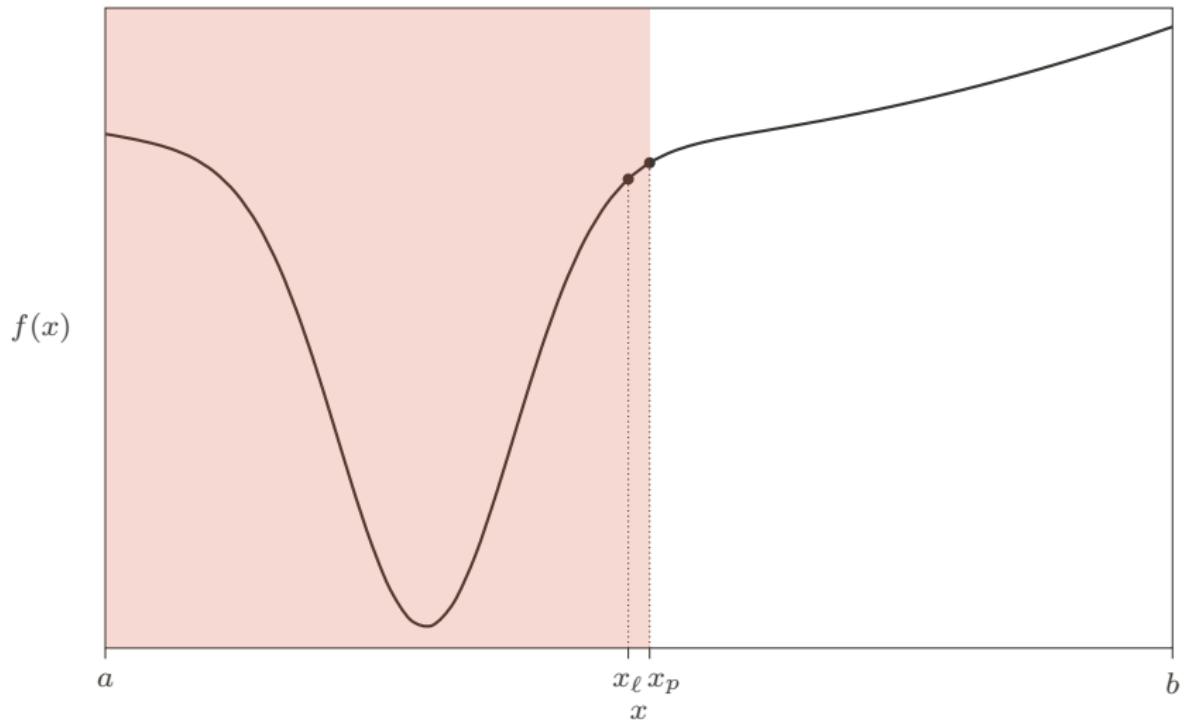
Dichotomous search – example



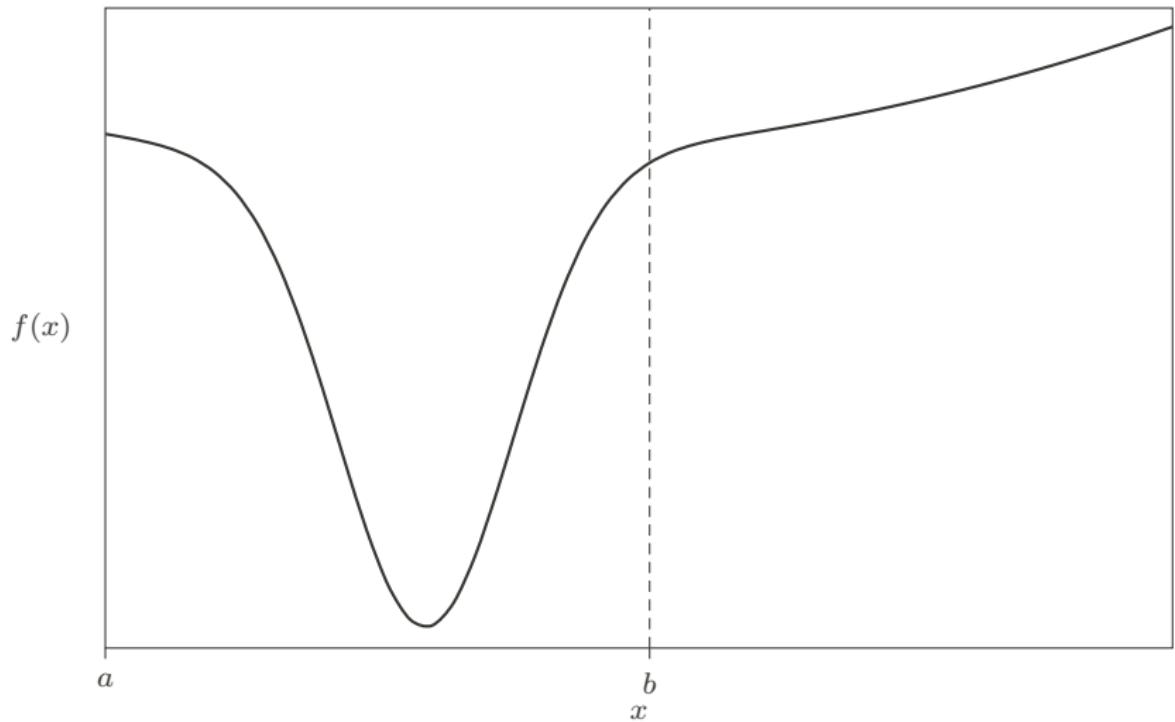
Dichotomous search – example



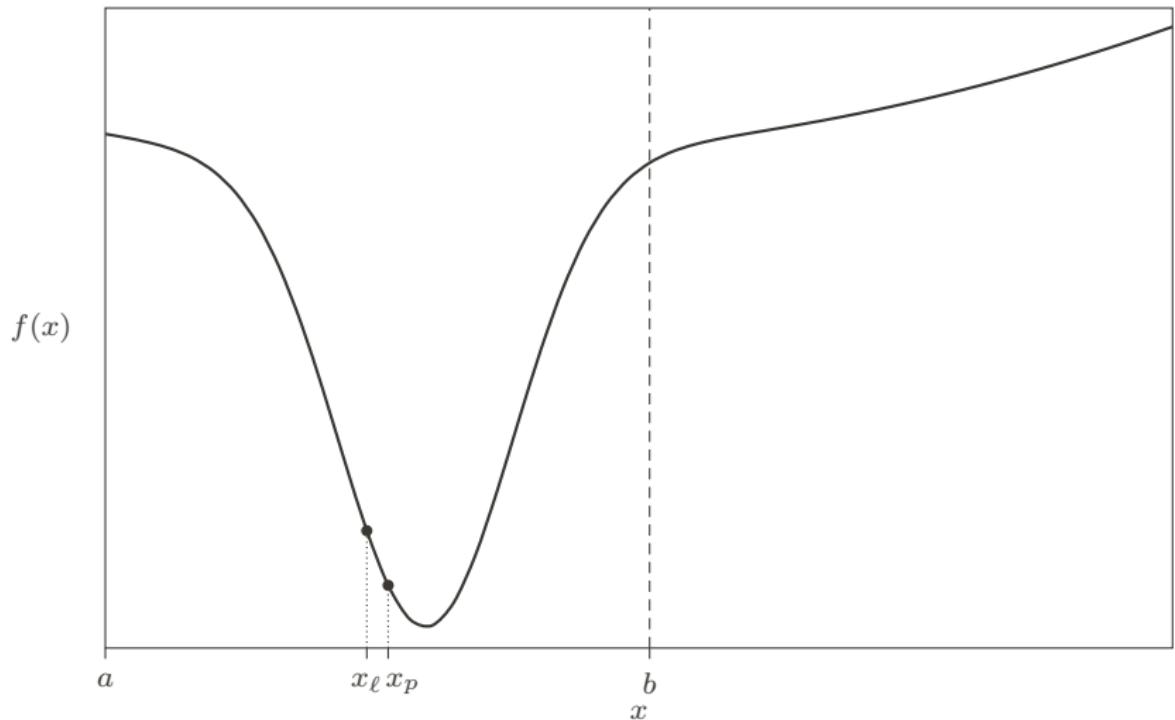
Dichotomous search – example



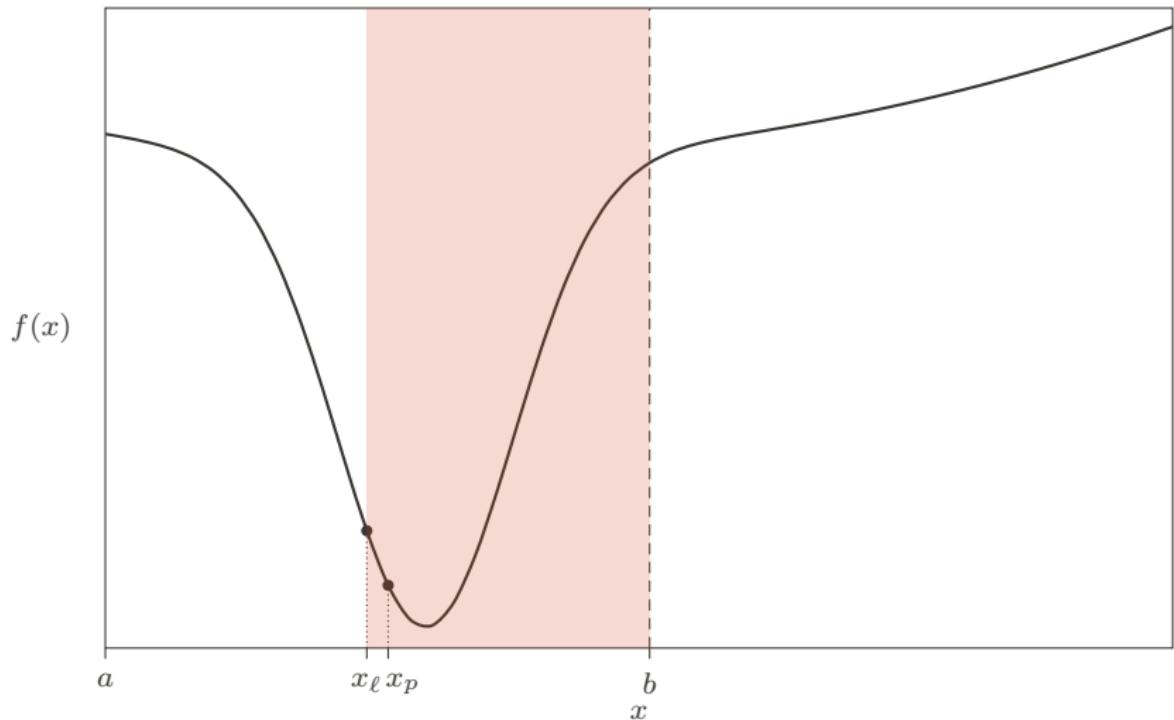
Dichotomous search – example



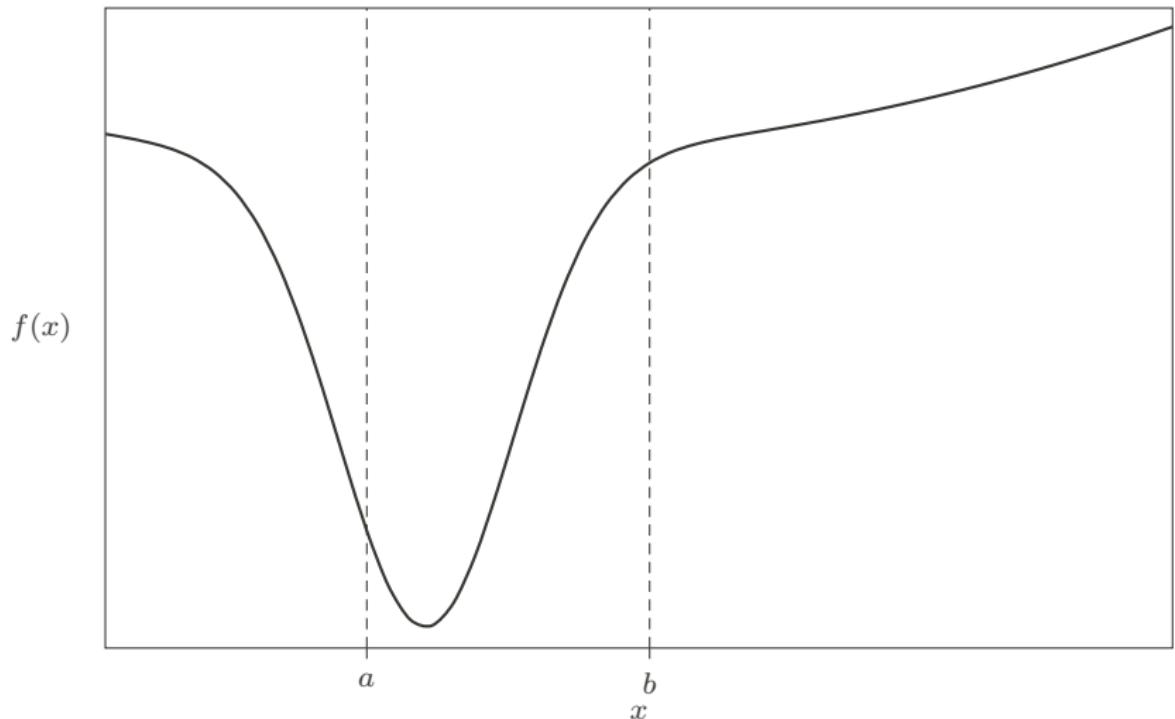
Dichotomous search – example



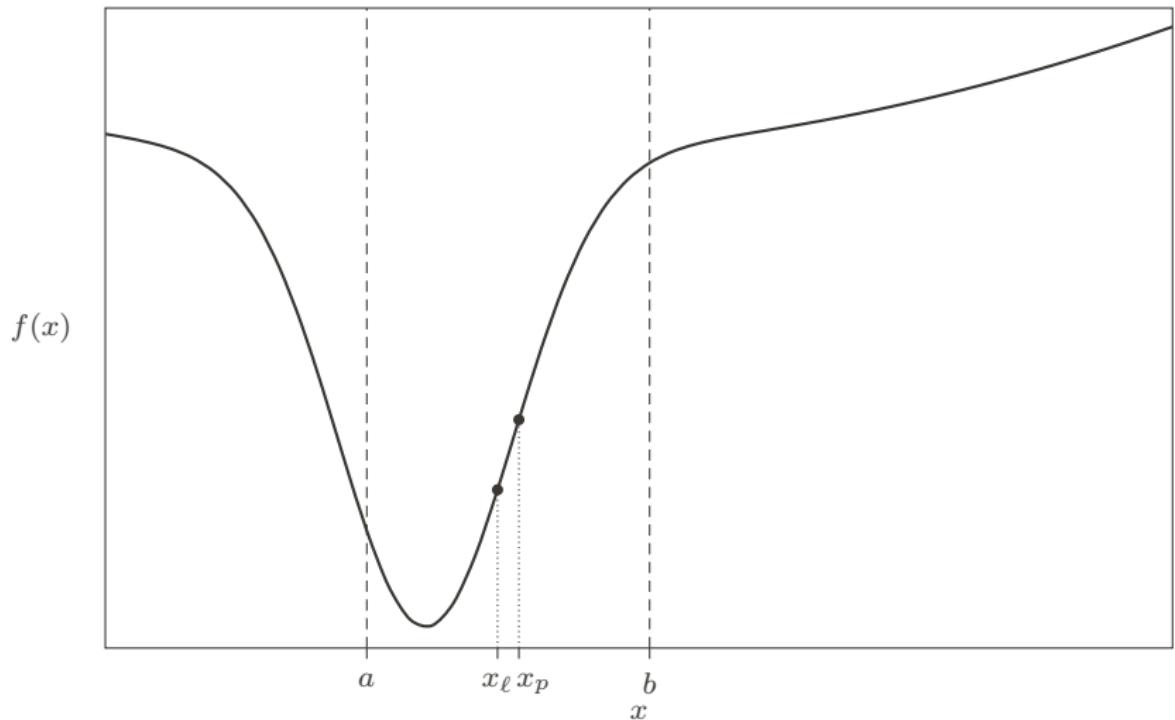
Dichotomous search – example



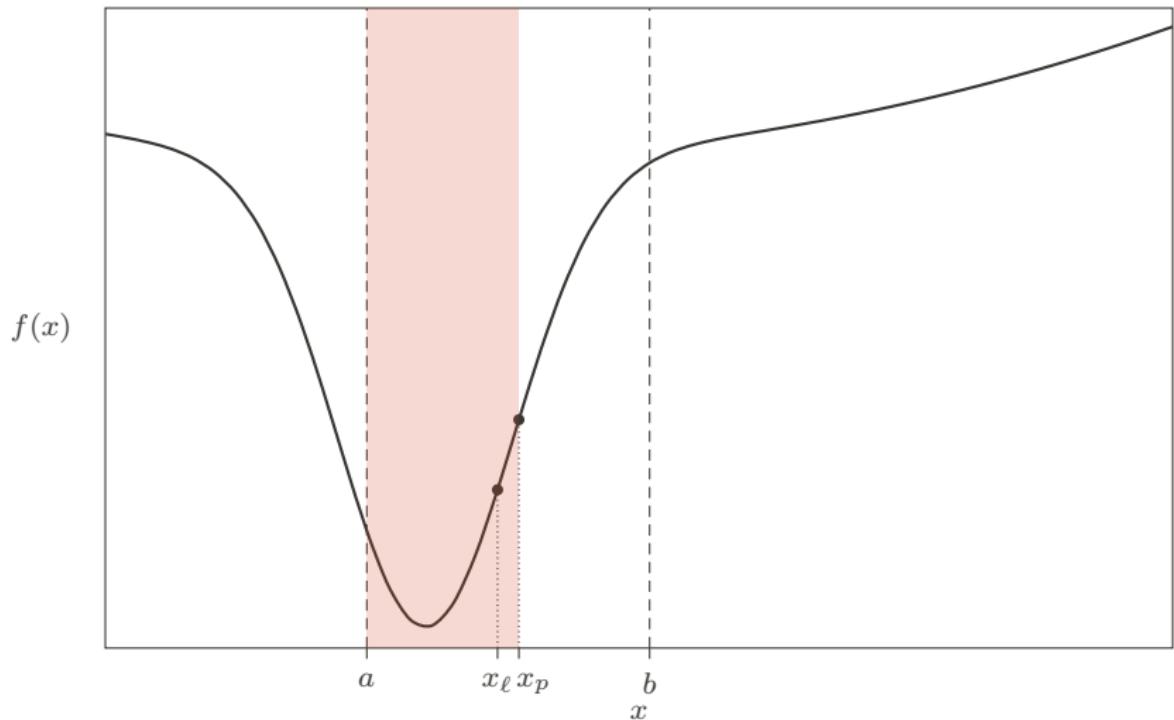
Dichotomous search – example



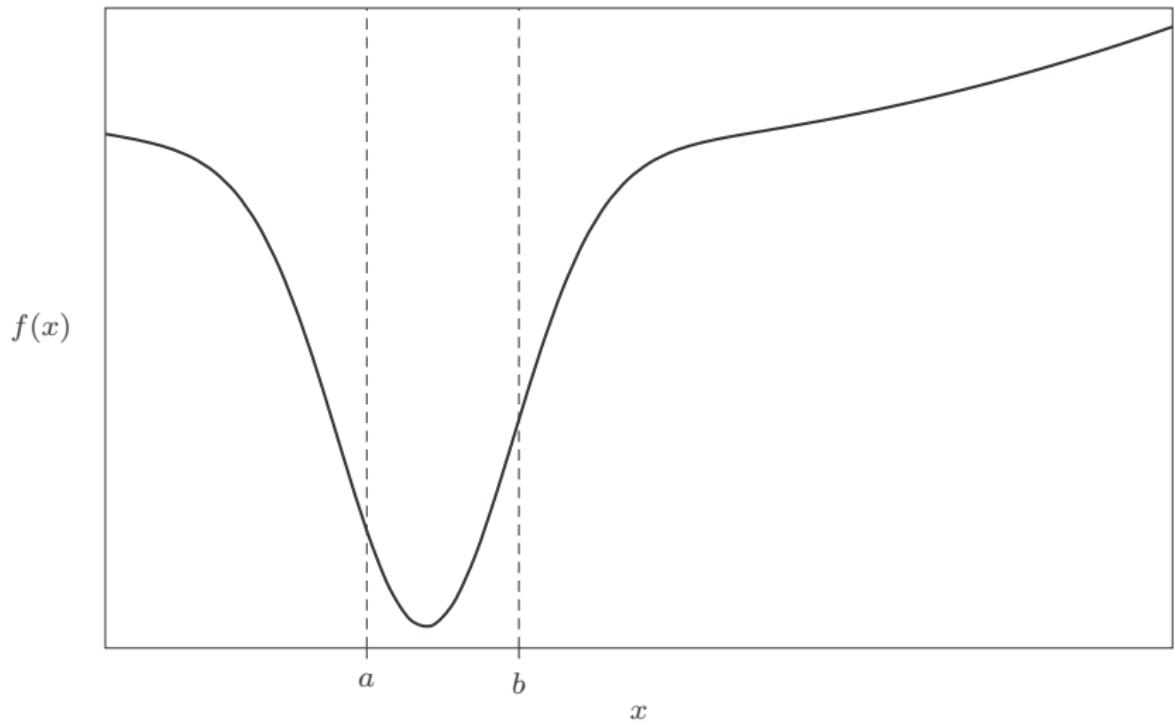
Dichotomous search – example



Dichotomous search – example



Dichotomous search – example



Dichotomous search – convergence

- A search area of length x at the beginning of an iteration, is reduced to $\frac{1}{2}x + \delta \simeq \frac{1}{2}x$ at the end of the iteration

Dichotomous search – convergence

- A search area of length x at the beginning of an iteration, is reduced to $\frac{1}{2}x + \delta \simeq \frac{1}{2}x$ at the end of the iteration
- **Conclusion:** starting with the entire domain of length $|\mathcal{X}|$, after k iterations the minimum is localized up to $\left(\frac{1}{2}\right)^k |\mathcal{X}|$ accuracy

Dichotomous search – convergence

- A search area of length x at the beginning of an iteration, is reduced to $\frac{1}{2}x + \delta \simeq \frac{1}{2}x$ at the end of the iteration
- **Conclusion:** starting with the entire domain of length $|\mathcal{X}|$, after k iterations the minimum is localized up to $\left(\frac{1}{2}\right)^k |\mathcal{X}|$ accuracy
- **Conclusion:** after n function evaluations

$$|\textcolor{brown}{x}_n - \textcolor{teal}{x}^*| \leq \left(\frac{1}{2}\right)^{n/2} |\mathcal{X}| = \left(\frac{1}{\sqrt{2}}\right)^n |\mathcal{X}|,$$

point returned
by the algorithm

the minimizer of f

Dichotomous search – convergence

n	$ x_n - x^* $
0	1
1	0.707
2	0.5
5	0.177
10	0.031
20	0.001
50	$3 \cdot 10^{-8}$

Can we do it faster?

Can we do it faster?

So far, at each iteration we evaluated the objective function twice and reduced the search area by half.

One of the two evaluation points becomes a new boundary of the search space, and we evaluate the function again at two new points.

Can we do it faster?

So far, at each iteration we evaluated the objective function twice and reduced the search area by half.

One of the two evaluation points becomes a new boundary of the search space, and we evaluate the function again at two new points.

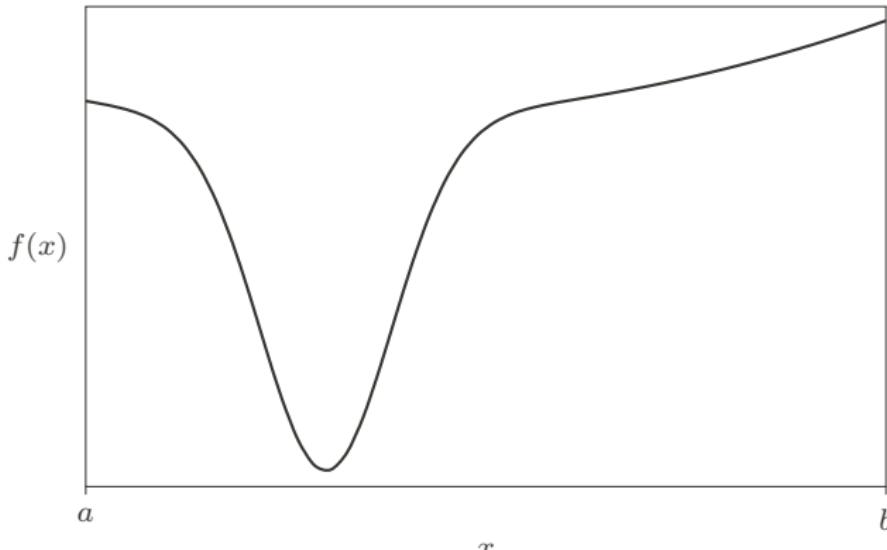
Idea: try to re-use one of the points from the previous iteration to evaluate the function only once per iteration!

Can we do it faster?

So far, at each iteration we evaluated the objective function twice and reduced the search area by half.

One of the two evaluation points becomes a new boundary of the search space, and we evaluate the function again at two new points.

Idea: try to re-use one of the points from the previous iteration to evaluate the function only once per iteration!

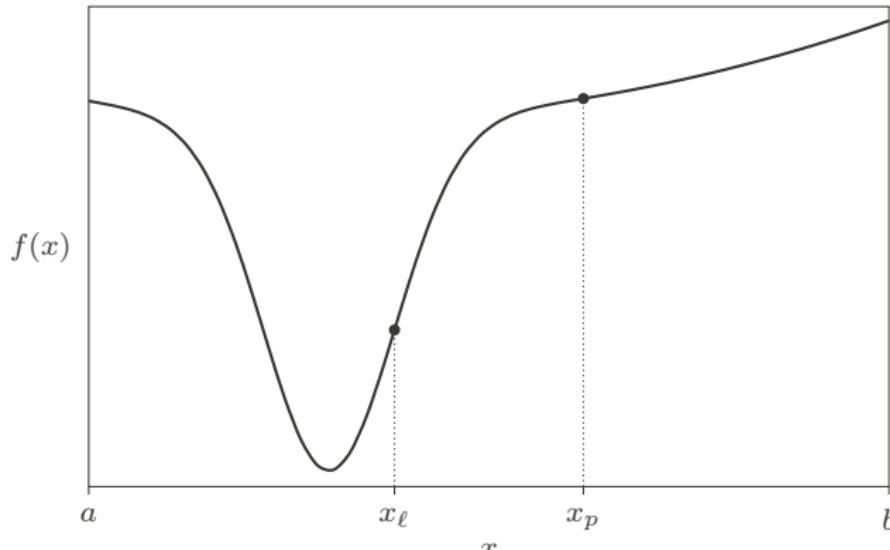


Can we do it faster?

So far, at each iteration we evaluated the objective function twice and reduced the search area by half.

One of the two evaluation points becomes a new boundary of the search space, and we evaluate the function again at two new points.

Idea: try to re-use one of the points from the previous iteration to evaluate the function only once per iteration!

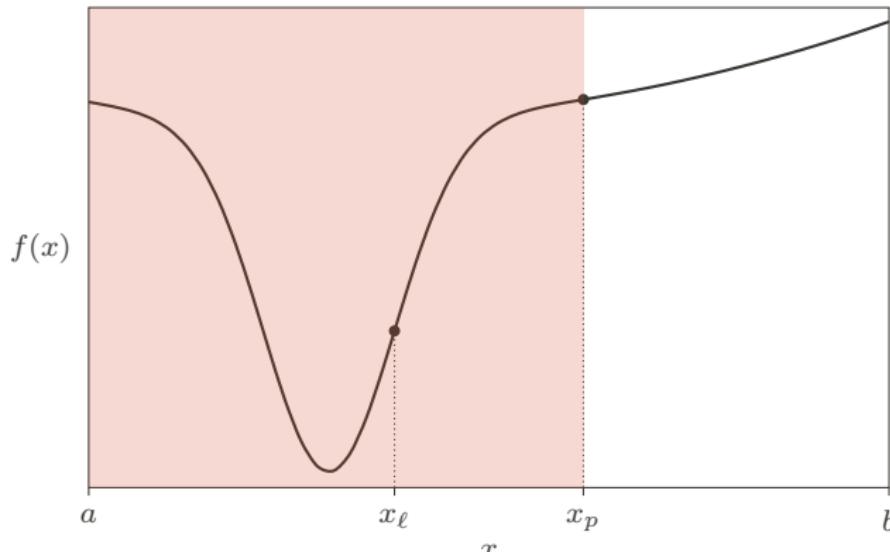


Can we do it faster?

So far, at each iteration we evaluated the objective function twice and reduced the search area by half.

One of the two evaluation points becomes a new boundary of the search space, and we evaluate the function again at two new points.

Idea: try to re-use one of the points from the previous iteration to evaluate the function only once per iteration!

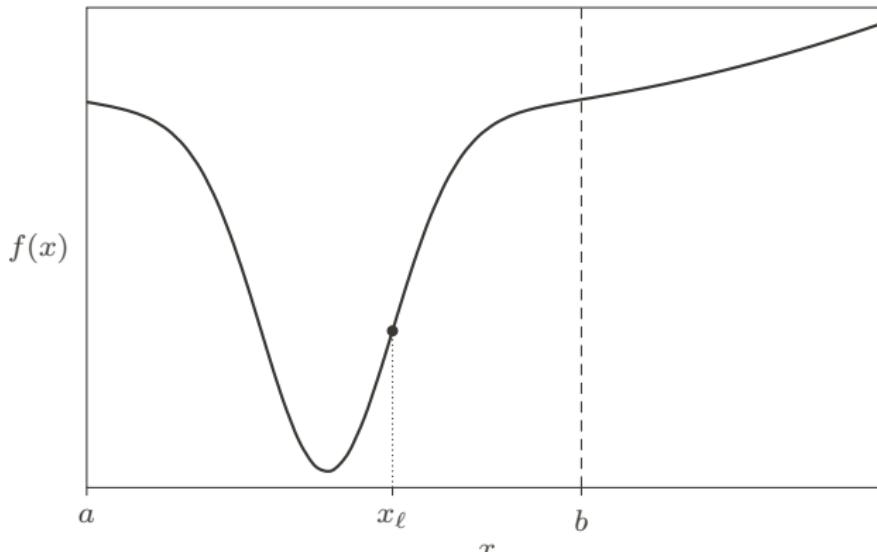


Can we do it faster?

So far, at each iteration we evaluated the objective function twice and reduced the search area by half.

One of the two evaluation points becomes a new boundary of the search space, and we evaluate the function again at two new points.

Idea: try to re-use one of the points from the previous iteration to evaluate the function only once per iteration!

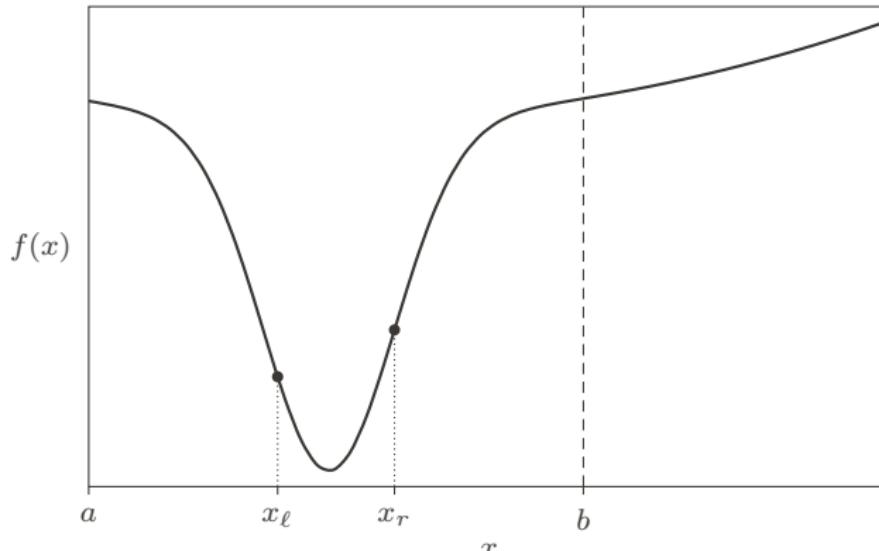


Can we do it faster?

So far, at each iteration we evaluated the objective function twice and reduced the search area by half.

One of the two evaluation points becomes a new boundary of the search space, and we evaluate the function again at two new points.

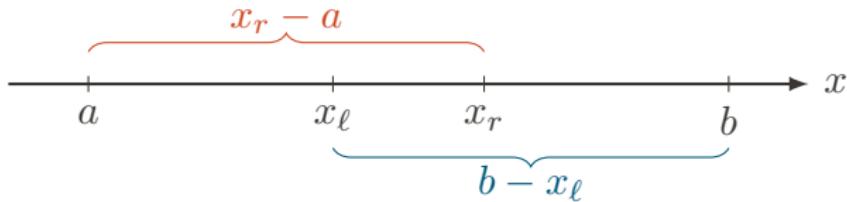
Idea: try to re-use one of the points from the previous iteration to evaluate the function only once per iteration!



Construction of the algorithm



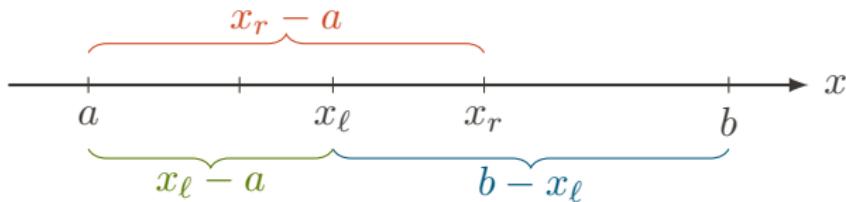
Construction of the algorithm



- In the next iteration, the area shrinks to either $x_r - a$ or $b - x_\ell$. As we do not know which case will happen, we set:

$$x_r - a = b - x_\ell$$

Construction of the algorithm



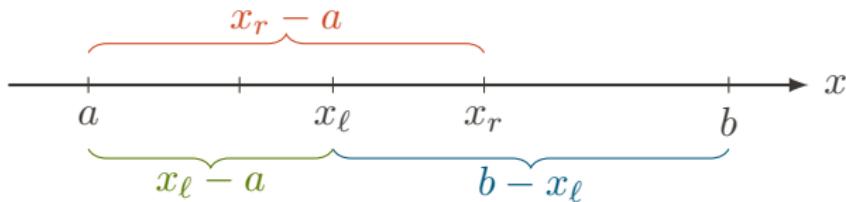
- In the next iteration, the area shrinks to either $x_r - a$ or $b - x_\ell$. As we do not know which case will happen, we set:

$$x_r - a = b - x_\ell$$

- If the updated area is, e.g., $x_r - a$, point x_ℓ will be re-used in the next iteration. To keep the proportions fixed, we set:

$$\frac{x_r - a}{b - a} = \frac{x_\ell - a}{x_r - a}$$

Construction of the algorithm



- In the next iteration, the area shrinks to either $x_r - a$ or $b - x_l$. As we do not know which case will happen, we set:

$$x_r - a = b - x_l \implies x_l - a = b - x_r$$

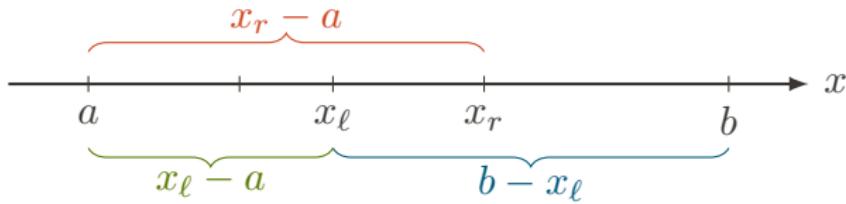
- If the updated area is, e.g., $x_r - a$, point x_l will be re-used in the next iteration. To keep the proportions fixed, we set:

$$\frac{x_r - a}{b - a} = \frac{x_l - a}{x_r - a}$$

- Substituting the first equation to the second one:

$$\frac{x_r - a}{b - a} = \frac{b - x_r}{x_r - a} = \frac{b - a - (x_r - a)}{x_r - a} = \frac{b - a}{x_r - a} - 1$$

Construction of the algorithm



- In the next iteration, the area shrinks to either $x_r - a$ or $b - x_\ell$. As we do not know which case will happen, we set:

$$x_r - a = b - x_\ell \implies x_\ell - a = b - x_r$$

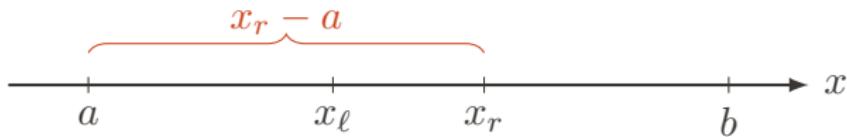
- If the updated area is, e.g., $x_r - a$, point x_ℓ will be re-used in the next iteration. To keep the proportions fixed, we set:

$$\frac{x_r - a}{b - a} = \frac{x_\ell - a}{x_r - a}$$

- Substituting the first equation to the second one:

$$\underbrace{\frac{x_r - a}{b - a}}_q = \frac{b - x_r}{x_r - a} = \frac{b - a - (x_r - a)}{x_r - a} = \underbrace{\frac{b - a}{x_r - a}}_{1/q} - 1$$

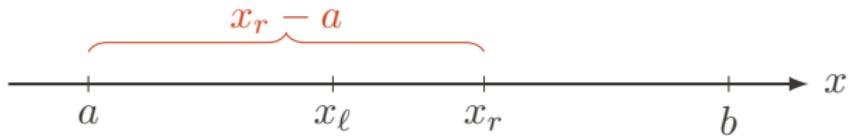
Construction of the algorithm



- We obtained the following equation:

$$q = \frac{1}{q} - 1 \quad \Rightarrow \quad q^2 + q - 1 = 0, \quad \text{for } q = \frac{x_r - a}{b - a}$$

Construction of the algorithm



- We obtained the following equation:

$$q = \frac{1}{q} - 1 \implies q^2 + q - 1 = 0, \quad \text{for } q = \frac{x_r - a}{b - a}$$

- Solution:

$$q = \frac{\sqrt{5} - 1}{2} \simeq 0.618.$$

Golden ratio.

Golden-section search method

Input: a procedure for evaluating objective $f(x)$ at any point of the domain $[x_{\min}, x_{\max}]$; number of function evaluations n ;
 $\alpha := \frac{\sqrt{5}-1}{2} \simeq 0.618$

Initialize: $a = x_{\min}, b = x_{\max}$
 $x_{\ell} = \alpha a + (1 - \alpha)b, x_r = (1 - \alpha)a + \alpha b$

Evaluate $f(x_{\ell}), f(x_r)$

For $k = 2, 3, \dots, n$:

If $f(x_{\ell}) < f(x_r)$

Set $b = x_r, x_r = x_{\ell}, x_{\ell} = \alpha a + (1 - \alpha)b$

If $k \neq n$, evaluate $f(x_{\ell})$.

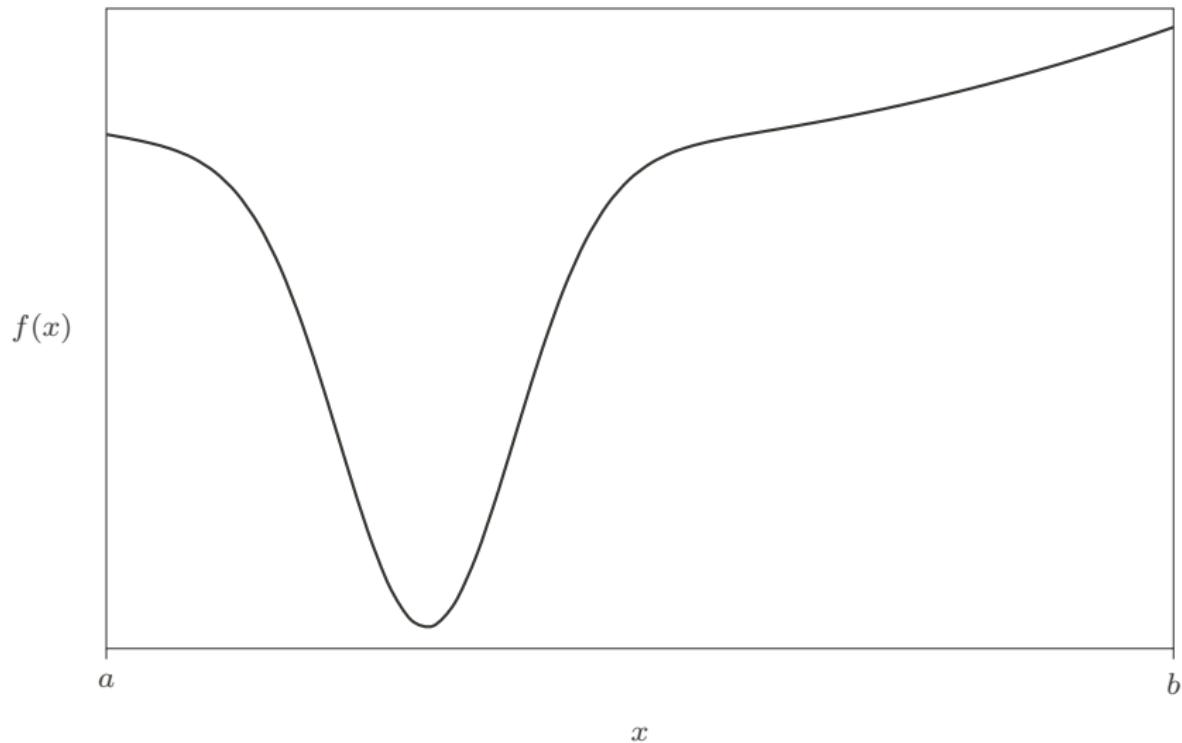
else

Set $a = x_{\ell}, x_{\ell} = x_r, x_r = (1 - \alpha)a + \alpha b$

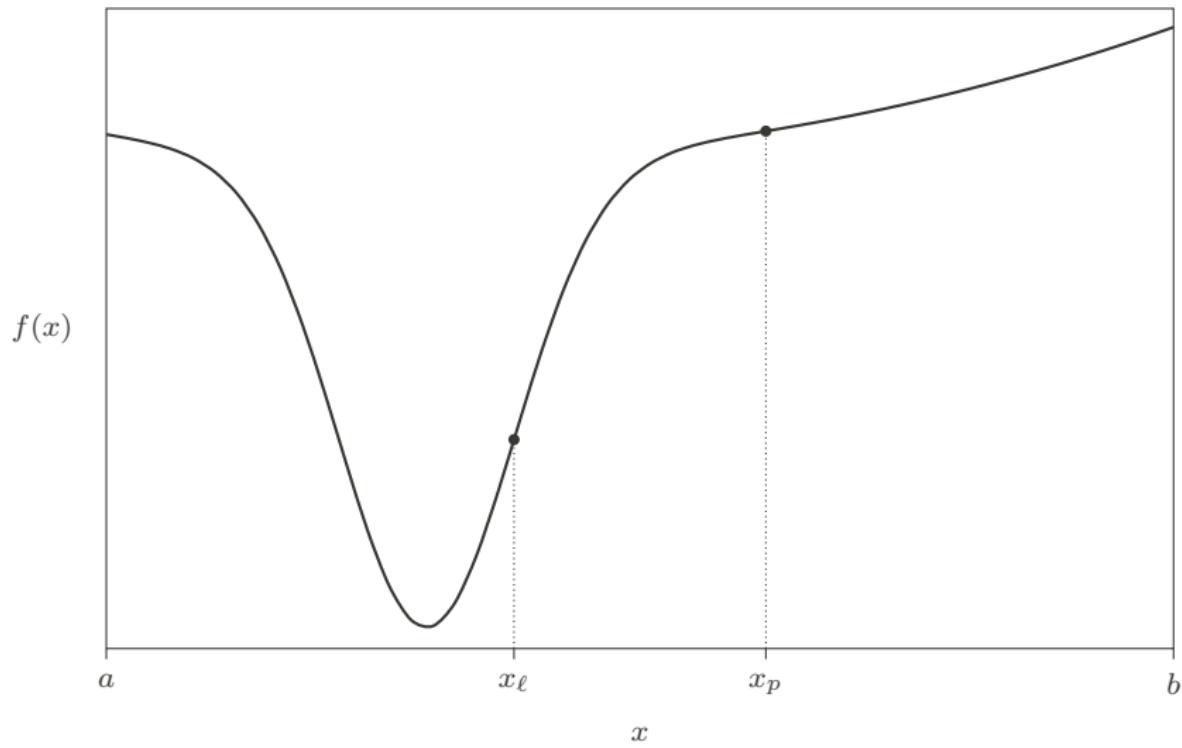
If $k \neq n$, evaluate $f(x_r)$.

Return point $x_n = \frac{a+b}{2}$

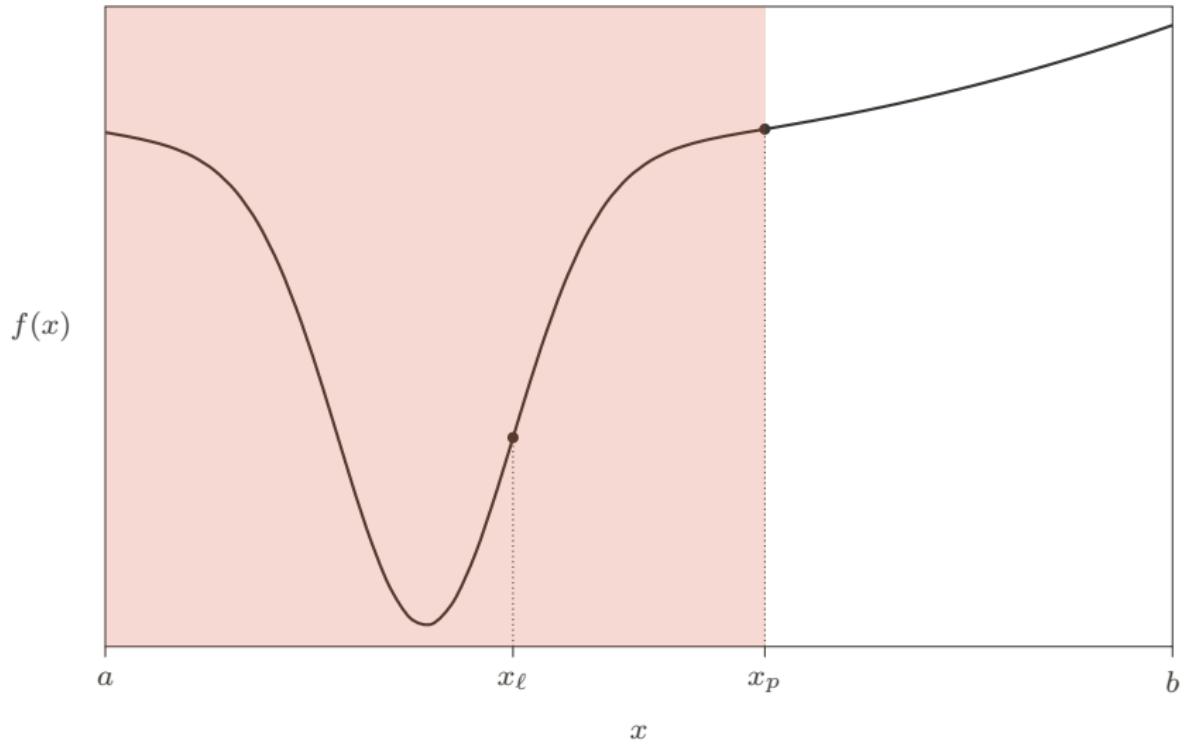
Golden-section search – example



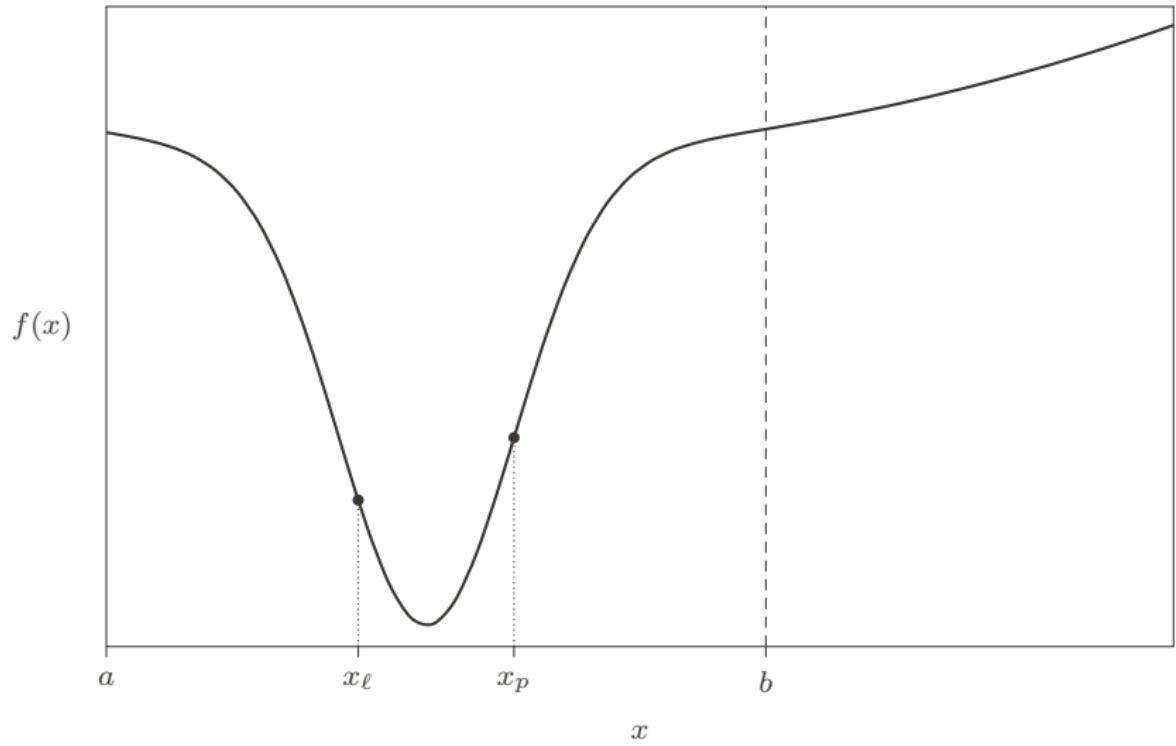
Golden-section search – example



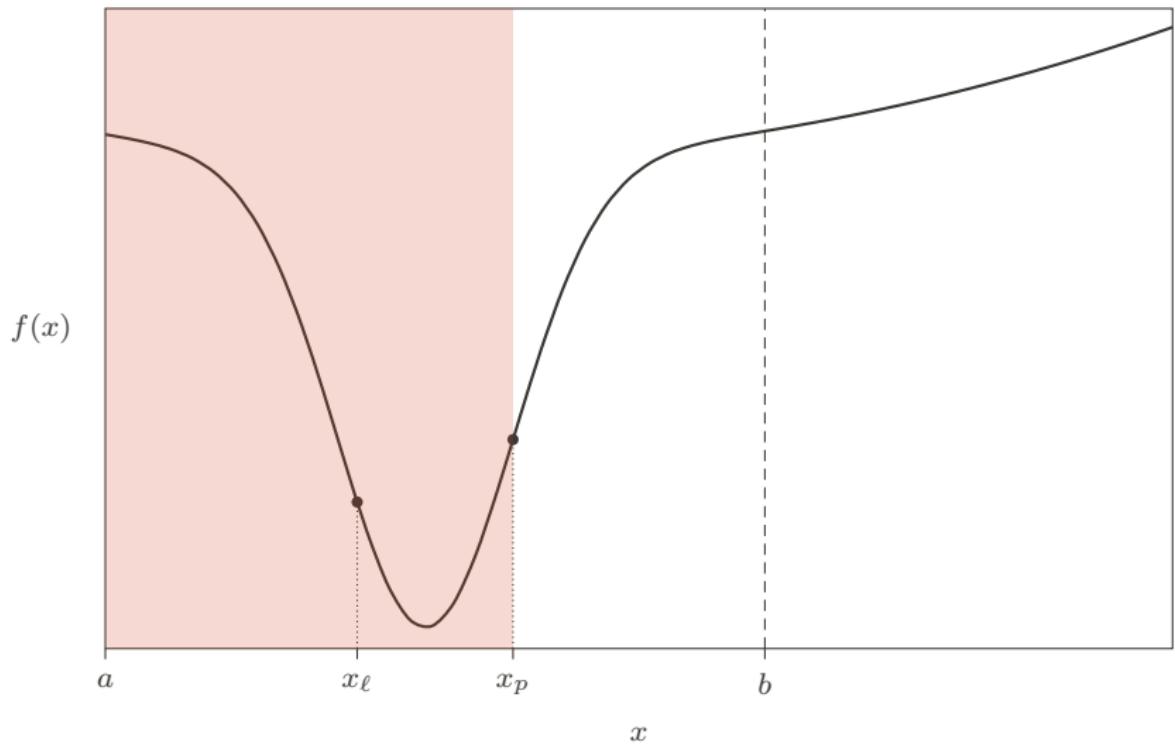
Golden-section search – example



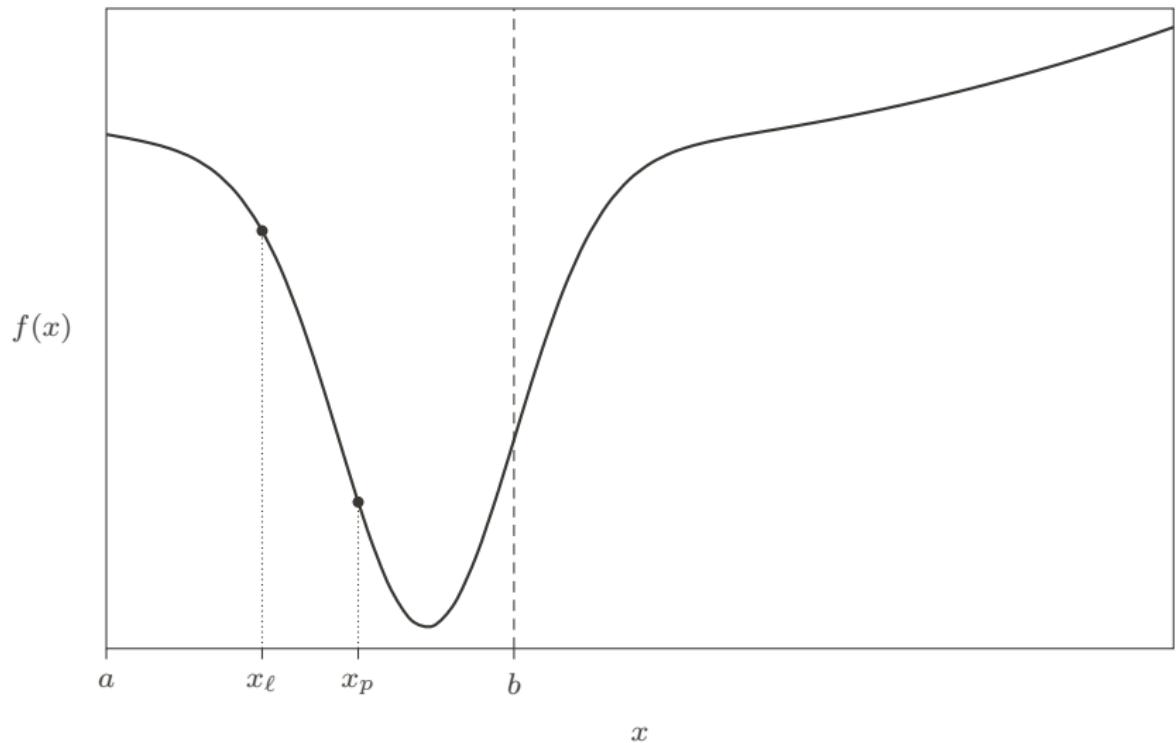
Golden-section search – example



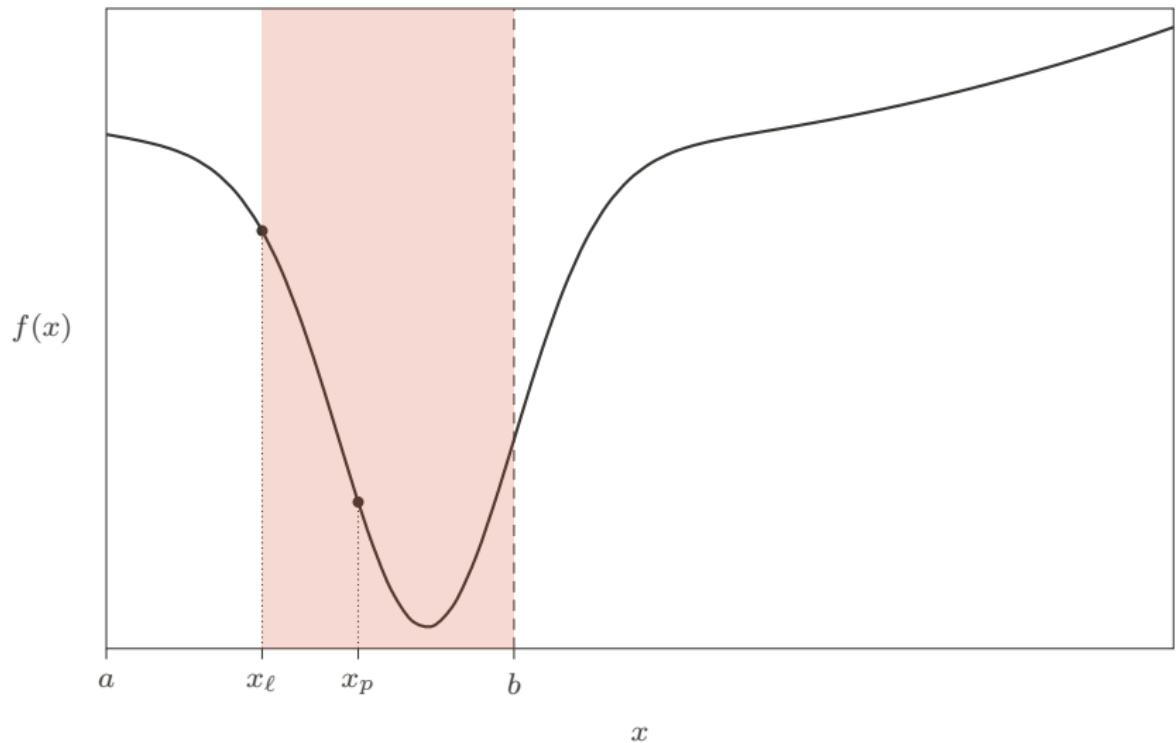
Golden-section search – example



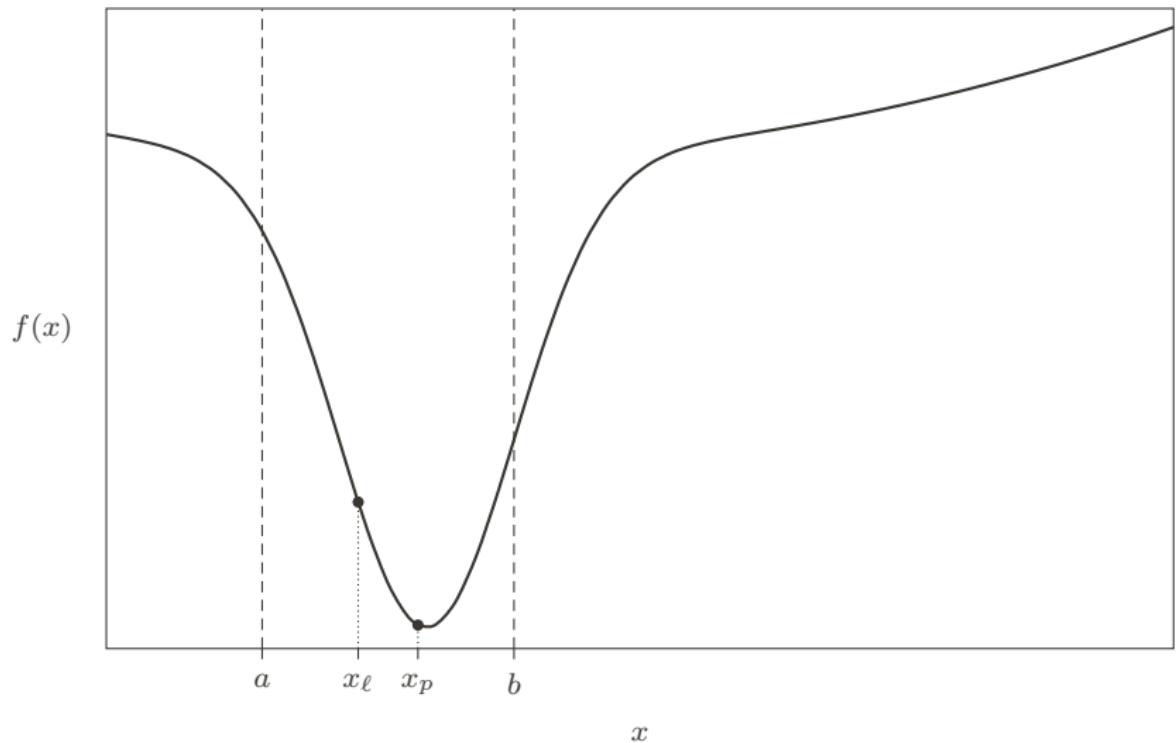
Golden-section search – example



Golden-section search – example



Golden-section search – example



Golden-section search – convergence

- A search area of length x at the beginning of an iteration, is reduced to αx at the end of the iteration, where $\alpha = \frac{\sqrt{5}-1}{2}$

Golden-section search – convergence

- A search area of length x at the beginning of an iteration, is reduced to αx at the end of the iteration, where $\alpha = \frac{\sqrt{5}-1}{2}$
- **Conclusion:** starting with the entire domain of length $|\mathcal{X}|$, after $n - 1$ iterations the minimum is localized up to $\alpha^{n-1}|\mathcal{X}|$ accuracy

Golden-section search – convergence

- A search area of length x at the beginning of an iteration, is reduced to αx at the end of the iteration, where $\alpha = \frac{\sqrt{5}-1}{2}$
- **Conclusion:** starting with the entire domain of length $|\mathcal{X}|$, after $n - 1$ iterations the minimum is localized up to $\alpha^{n-1}|\mathcal{X}|$ accuracy
- **Conclusion:** after n **function evaluations**

$$|x_n - x^*| \leq \alpha^{n-1} |\mathcal{X}| \simeq (0.618)^n |\mathcal{X}|,$$

Comparison: dichotomous vs. golden-section search

algorithm	convergence
dichotomous	$\left(\frac{1}{\sqrt{2}}\right)^n \simeq (0.707)^n$
golden-section	$\left(\frac{\sqrt{5}-1}{2}\right)^{n-1} \simeq (0.618)^n$

Comparison: dichotomous vs. golden-section search

algorithm	convergence
dichotomous	$\left(\frac{1}{\sqrt{2}}\right)^n \simeq (0.707)^n$
golden-section	$\left(\frac{\sqrt{5}-1}{2}\right)^{n-1} \simeq (0.618)^n$

- Convergence rate $(0.618)^n$ is **asymptotically optimal**.
- The optimal algorithm is the **Fibonacci search** method:
 - ▶ Convergence similar to the golden-section search (asymptotically identical)
 - ▶ More complicated and therefore somewhat less practical

Optimization of differentiable functions: The bisection algorithm

Assumptions

Assumptions:

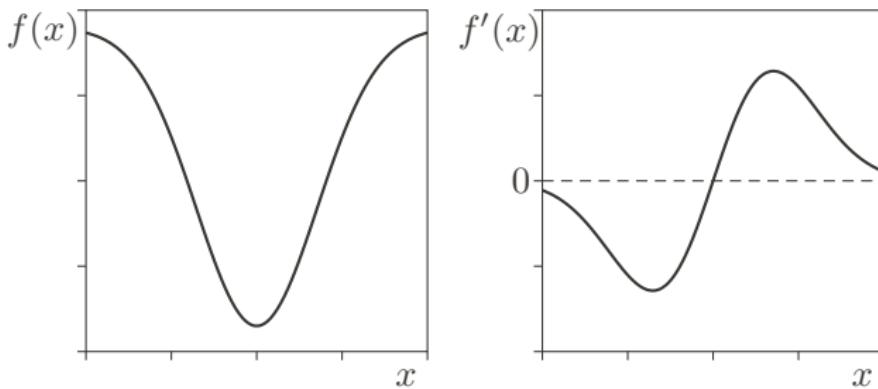
1. The objective function f has a **continuous derivative** on $\mathcal{X} = [x_{\min}, x_{\max}]$
2. The derivative \mathcal{X} switches its sign once (from negative to positive)

Assumptions

Assumptions:

1. The objective function f has a **continuous derivative** on $\mathcal{X} = [x_{\min}, x_{\max}]$
2. The derivative \mathcal{X} switches its sign once (from negative to positive)

Conclusion: the function is **unimodal**: first strictly decreasing, then strictly increasing, with a minimum at a point of **zero derivative**

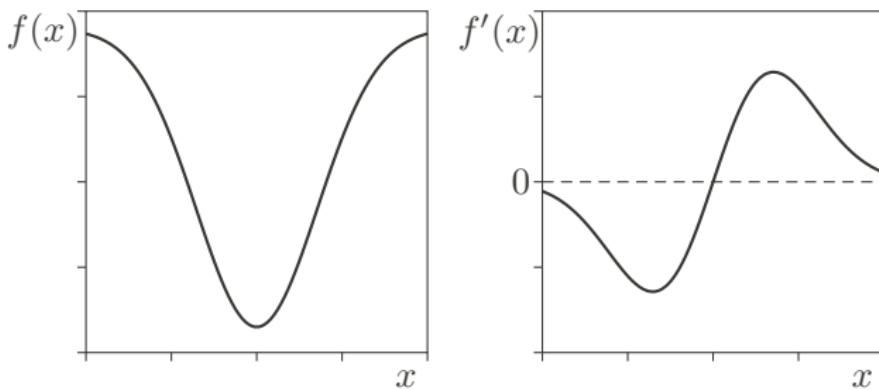


Assumptions

Assumptions:

1. The objective function f has a **continuous derivative** on $\mathcal{X} = [x_{\min}, x_{\max}]$
2. The derivative \mathcal{X} switches its sign once (from negative to positive)

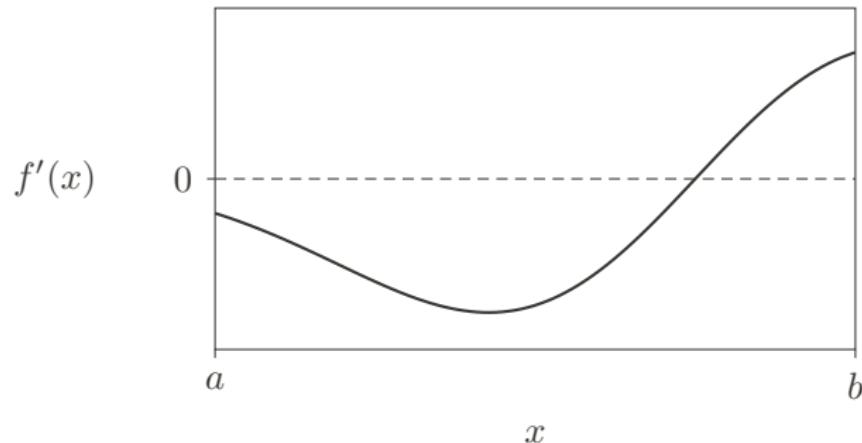
Conclusion: the function is **unimodal**: first strictly decreasing, then strictly increasing, with a minimum at a point of **zero derivative**



Conclusion: We reduce the problem to finding a single **root** (zero) of the derivative function

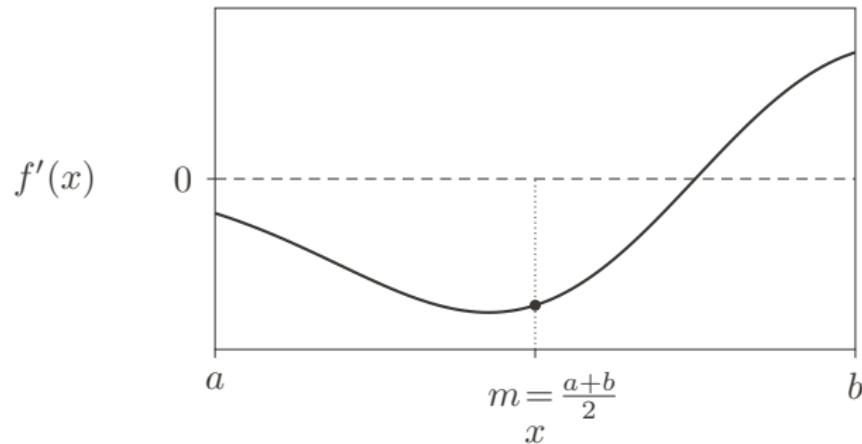
Bisection method

Take an interval $[a, b]$ with $f'(a) < 0$ i $f'(b) > 0$, in which f' has a single root.



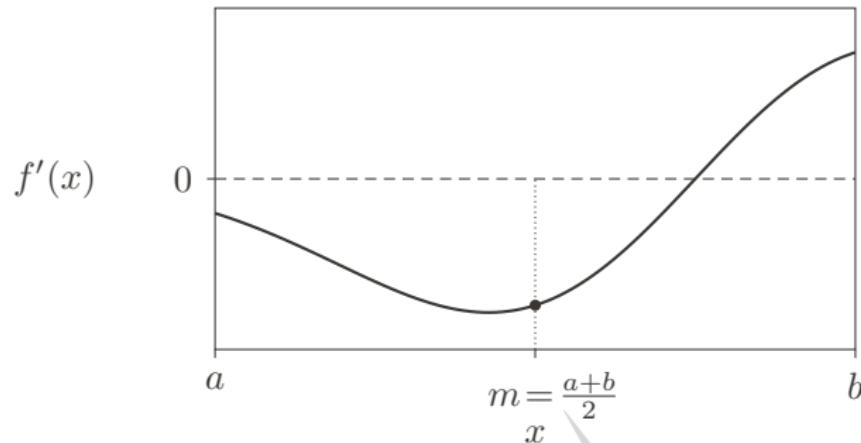
Bisection method

Take an interval $[a, b]$ with $f'(a) < 0$ i $f'(b) > 0$, in which f' has a single root.



Bisection method

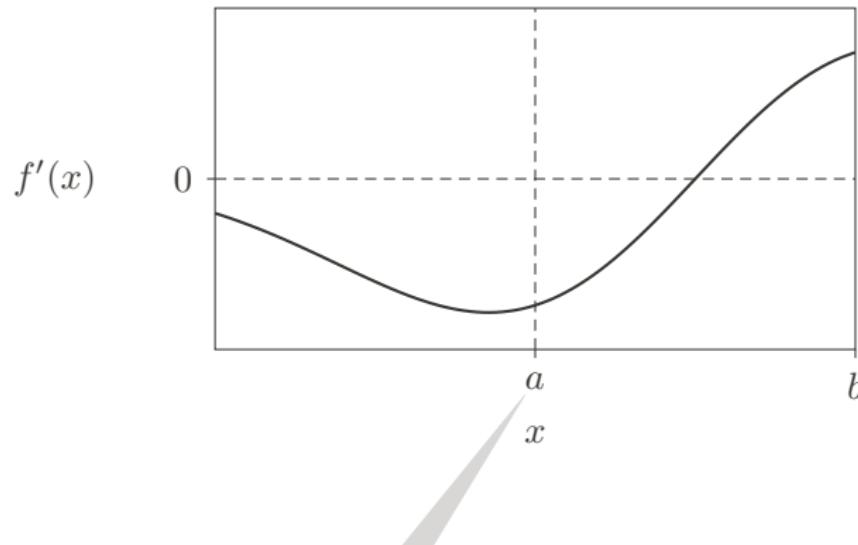
Take an interval $[a, b]$ with $f'(a) < 0$ i $f'(b) > 0$, in which f' has a single root.



As $f'(m) < 0$, the root must be to the **right** of m !

Bisection method

Take an interval $[a, b]$ with $f'(a) < 0$ i $f'(b) > 0$, in which f' has a single root.



We reduce the search area by setting $a = m$ and we again have $f'(a) < 0$, $f'(b) > 0$

Bisection method

Input: a procedure for evaluating the derivative $f'(x)$ at any point of the domain $[x_{\min}, x_{\max}]$; number of derivative evaluations n ;

Initialize: $a = x_{\min}, b = x_{\max}$

For $k = 1, 2, \dots, n$:

 Evaluate $f'(m)$ at $m = \frac{a+b}{2}$

 If $f'(m) = 0$, stop the algorithm (minimum found)

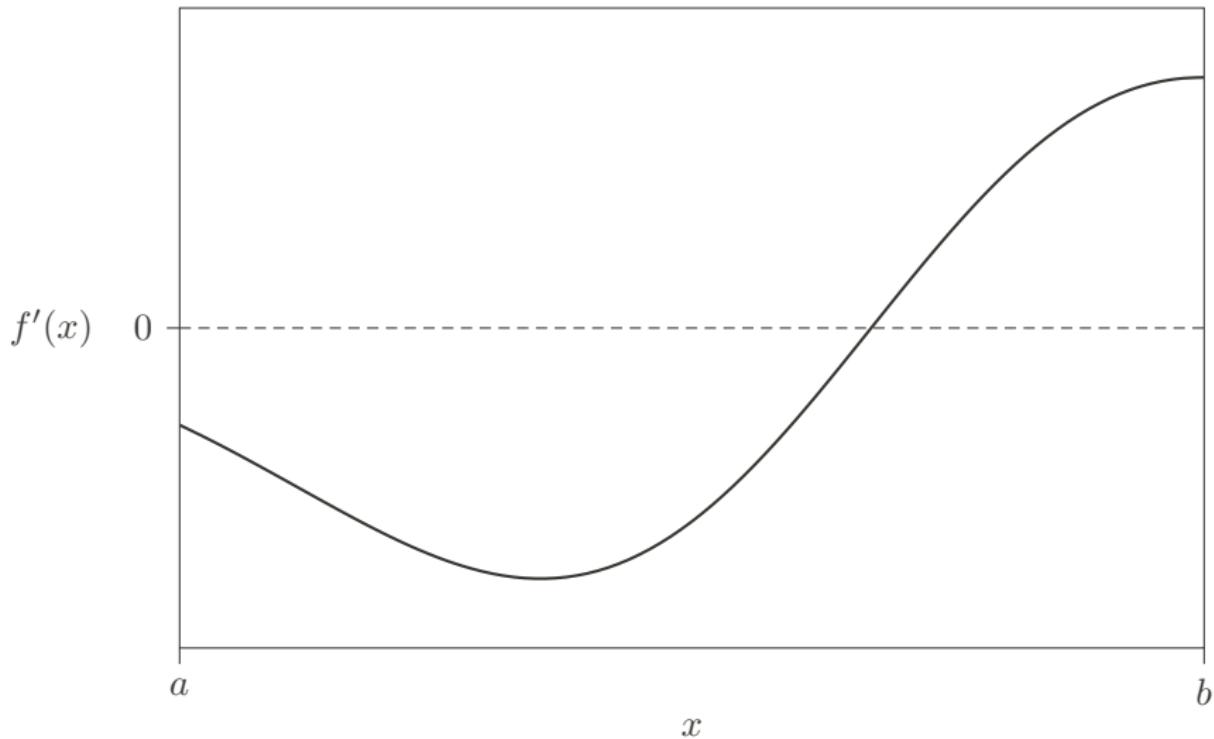
 Else:

 If $f'(m) > 0$ set $b = m$

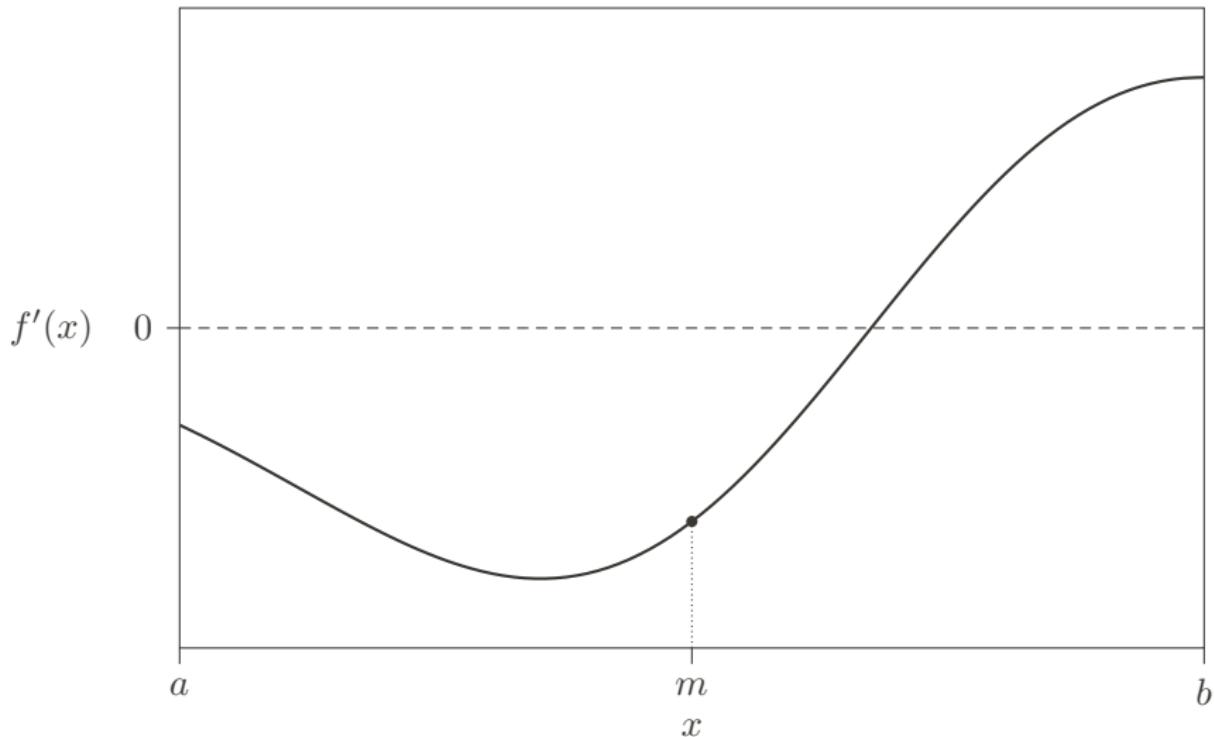
 If $f'(m) < 0$ set $a = m$

Return $x_n = \frac{a+b}{2}$

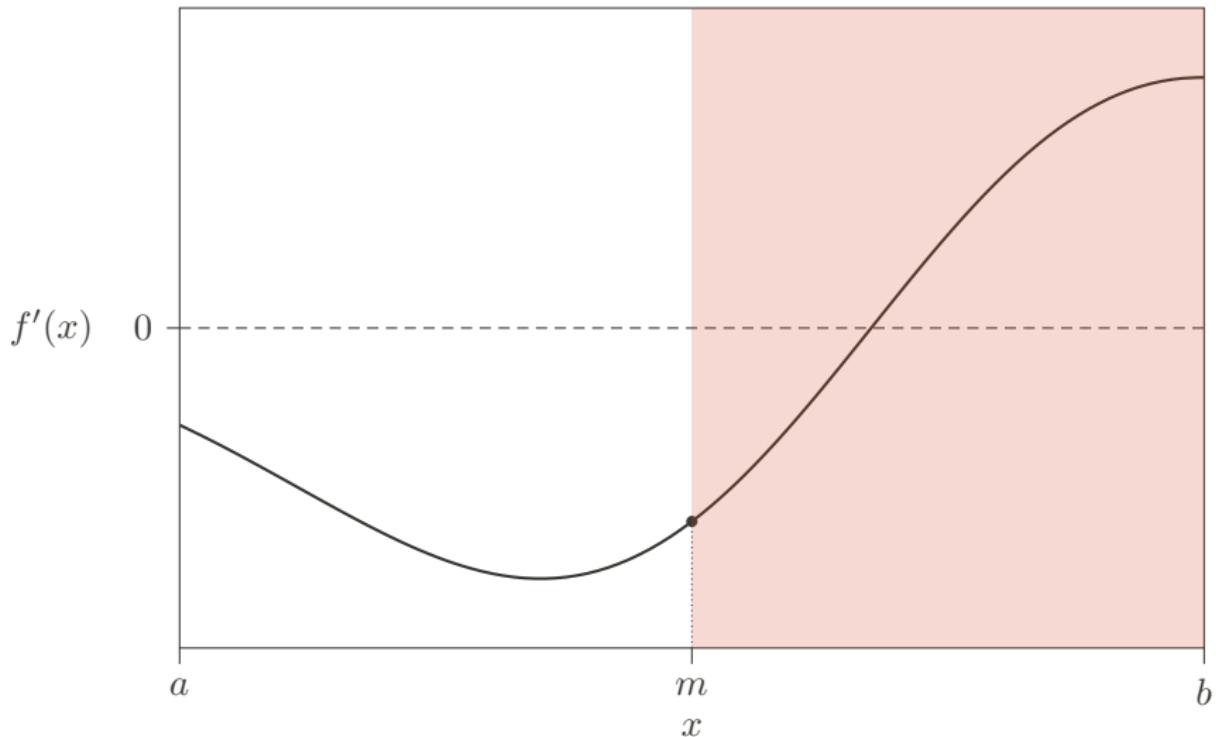
Bisection method – example



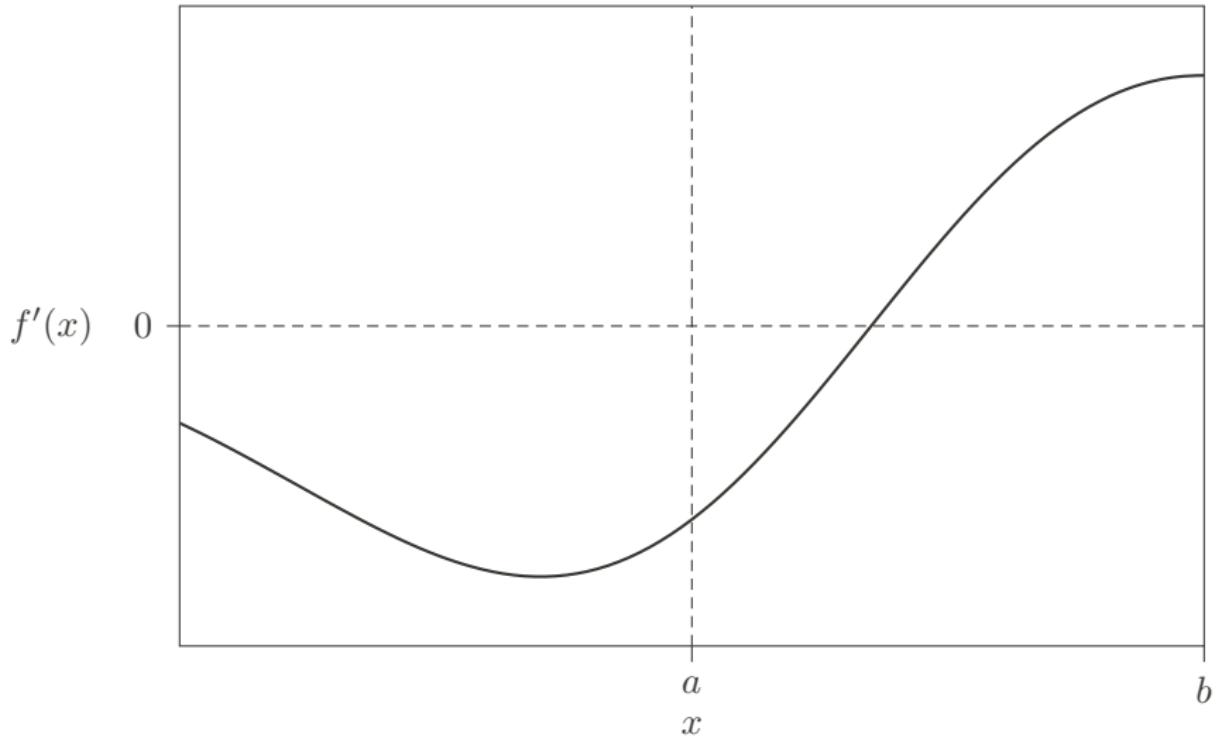
Bisection method – example



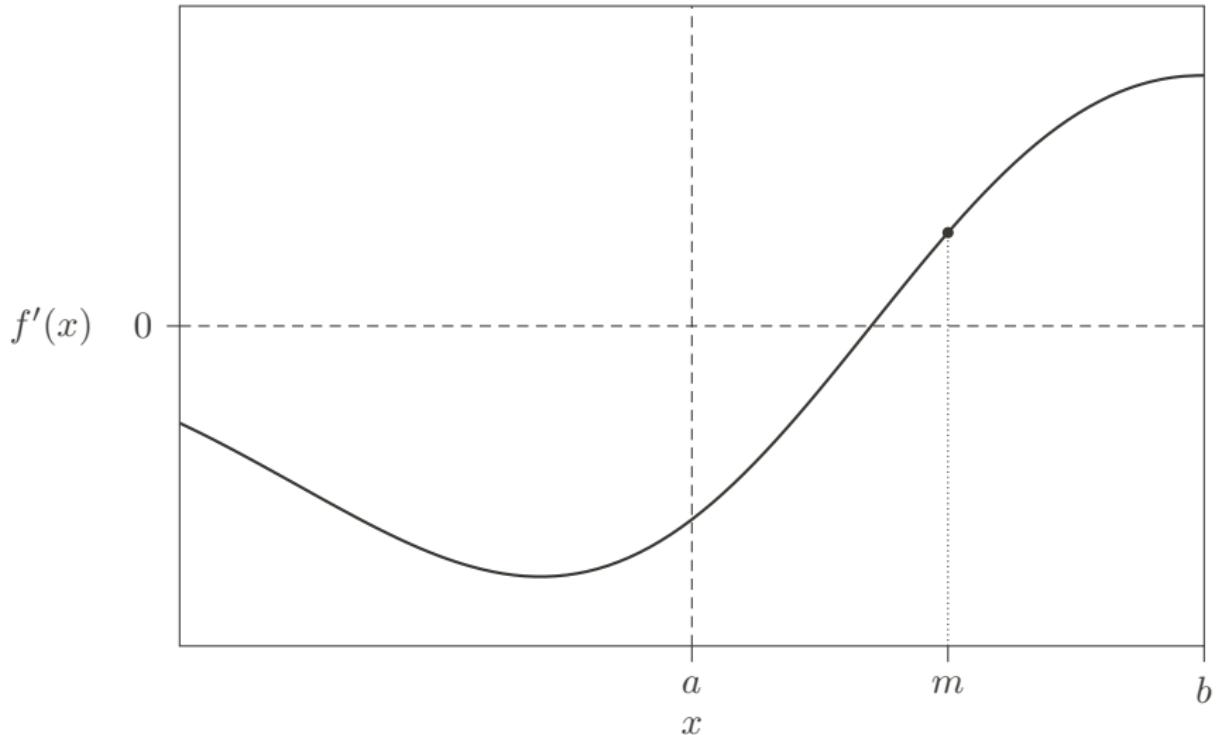
Bisection method – example



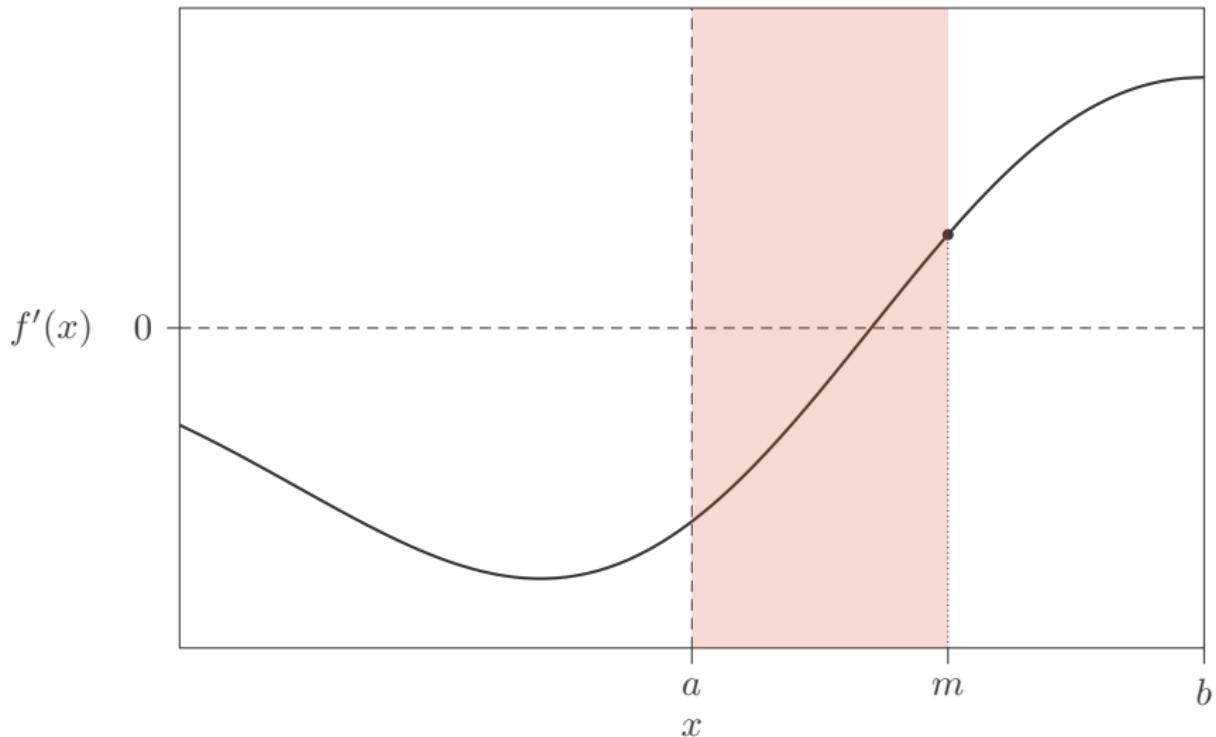
Bisection method – example



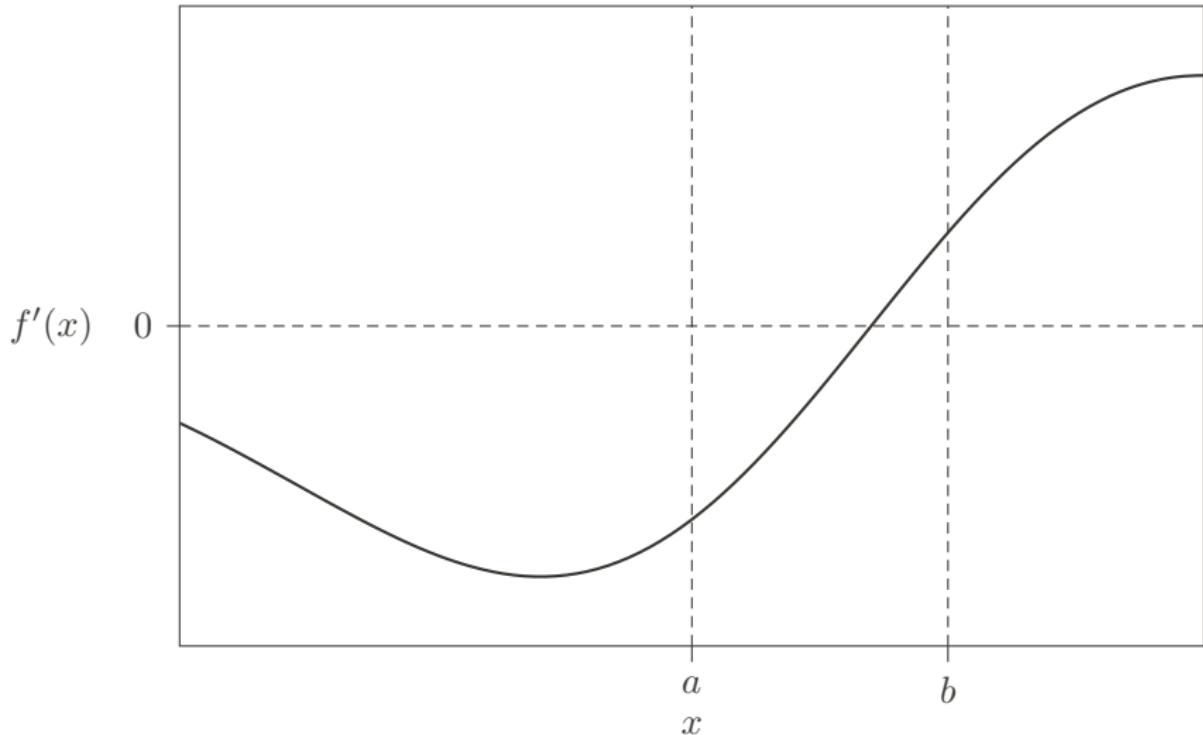
Bisection method – example



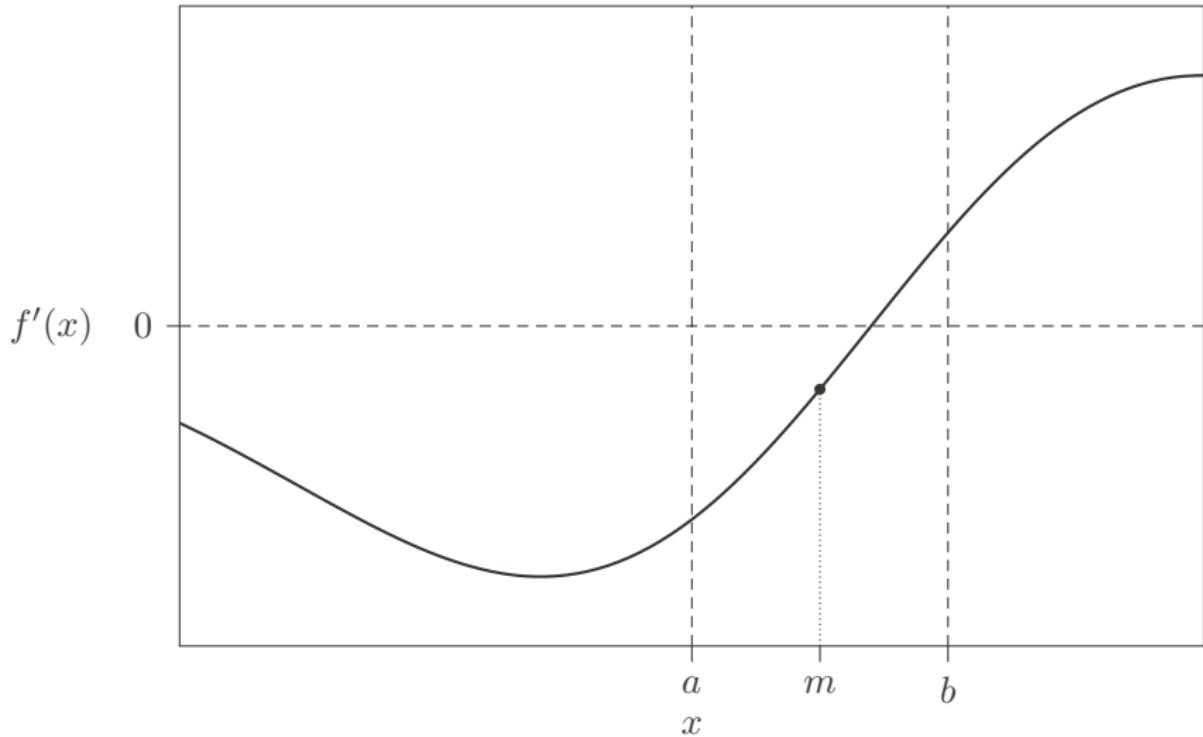
Bisection method – example



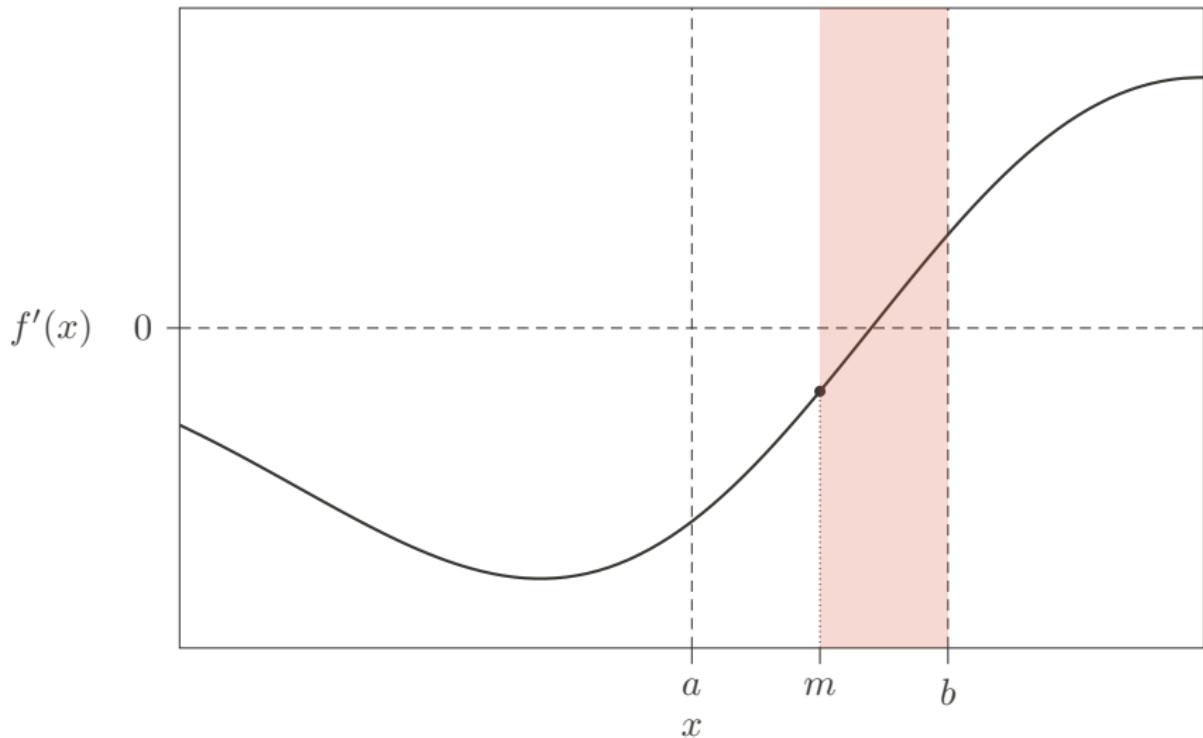
Bisection method – example



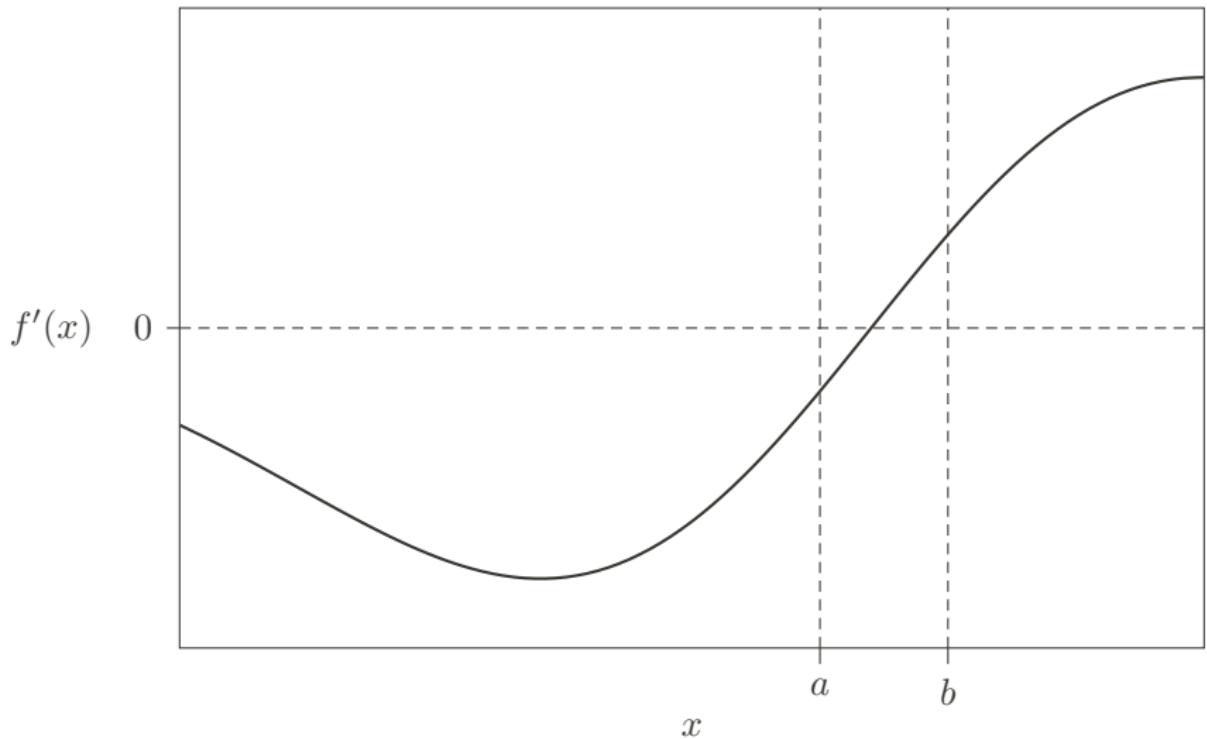
Bisection method – example



Bisection method – example



Bisection method – example



Bisection method – convergence

- A search area of length x at the beginning of an iteration, is reduced to $\frac{1}{2}x$ at the end of the iteration

Bisection method – convergence

- A search area of length x at the beginning of an iteration, is reduced to $\frac{1}{2}x$ at the end of the iteration
- **Conclusion:** after n derivative evaluations:

$$|x_n - x^*| \leq \left(\frac{1}{2}\right)^n |\mathcal{X}|$$

Bisection method – convergence

- A search area of length x at the beginning of an iteration, is reduced to $\frac{1}{2}x$ at the end of the iteration
- **Conclusion:** after n derivative evaluations:

$$|x_n - x^*| \leq \left(\frac{1}{2}\right)^n |\mathcal{X}|$$

algorithm	convergence
dichotomous	$(0.707)^n$
golden-section	$(0.618)^n$
bisection	$(0.5)^n$

Bisection method – convergence

- A search area of length x at the beginning of an iteration, is reduced to $\frac{1}{2}x$ at the end of the iteration
- **Conclusion:** after n derivative evaluations:

$$|x_n - x^*| \leq \left(\frac{1}{2}\right)^n |\mathcal{X}|$$

algorithm	convergence
dichotomous	$(0.707)^n$
golden-section	$(0.618)^n$
bisection	$(0.5)^n$

An access to the derivative speeds up the algorithm!

The dichotomous search algorithm is an approximated bisection method where the sign of the derivative at m is estimated by $f(m + \delta) - f(m - \delta)$

Optimization of differentiable functions: the Newton algorithm

Taylor polynomials and Taylor's theorem

A polynomial:

$$P_m(x; x_0) = f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \dots + \frac{f^{(m)}(x_0)}{m!}(x - x_0)^m$$

is called the m -th order **Taylor polynomial** of function f at point x_0 .

Taylor polynomials approximate the shape of f around x_0 , with the accuracy increasing with their order m .

Taylor polynomials and Taylor's theorem

A polynomial:

$$P_m(x; x_0) = f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \dots + \frac{f^{(m)}(x_0)}{m!}(x - x_0)^m$$

is called the m -th order **Taylor polynomial** of function f at point x_0 .

Taylor polynomials approximate the shape of f around x_0 , with the accuracy increasing with their order m .

Taylor's theorem: The error of approximating $f(x)$ with the Taylor polynomial $P_m(x; x_0)$ around x_0 :

$$f(x) = P_m(x; x_0) + R_m(x; x_0), \quad R_m(x; x_0) = \underbrace{\frac{f^{(m+1)}(\xi)}{(m+1)!}(x - x_0)^{m+1}}_{\text{Lagrange remainder}},$$

where ξ is some point in between x and x_0

Taylor polynomials – example

Expanding $f(x) = e^x$ into Taylor polynomials at point $x_0 = 0$.

Taylor polynomials – example

Expanding $f(x) = e^x$ into Taylor polynomials at point $x_0 = 0$.

$$f(x) = e^x \quad f(x_0) = e^0 = 1 \quad P_0(x; 0) = 1$$

Taylor polynomials – example

Expanding $f(x) = e^x$ into Taylor polynomials at point $x_0 = 0$.

$$f(x) = e^x \quad f(x_0) = e^0 = 1 \quad P_0(x; 0) = 1$$

$$f'(x) = e^x \quad f'(x_0) = e^0 = 1 \quad P_1(x; 0) = 1 + \frac{1}{1!}x = 1 + x$$

Taylor polynomials – example

Expanding $f(x) = e^x$ into Taylor polynomials at point $x_0 = 0$.

$$f(x) = e^x \quad f(x_0) = e^0 = 1 \quad P_0(x; 0) = 1$$

$$f'(x) = e^x \quad f'(x_0) = e^0 = 1 \quad P_1(x; 0) = 1 + \frac{1}{1!}x = 1 + x$$

$$f''(x) = e^x \quad f''(x_0) = e^0 = 1 \quad P_2(x; 0) = 1 + x + \frac{1}{2!}x^2 = 1 + x + \frac{x^2}{2}$$

Taylor polynomials – example

Expanding $f(x) = e^x$ into Taylor polynomials at point $x_0 = 0$.

$$f(x) = e^x \quad f(x_0) = e^0 = 1 \quad P_0(x; 0) = 1$$

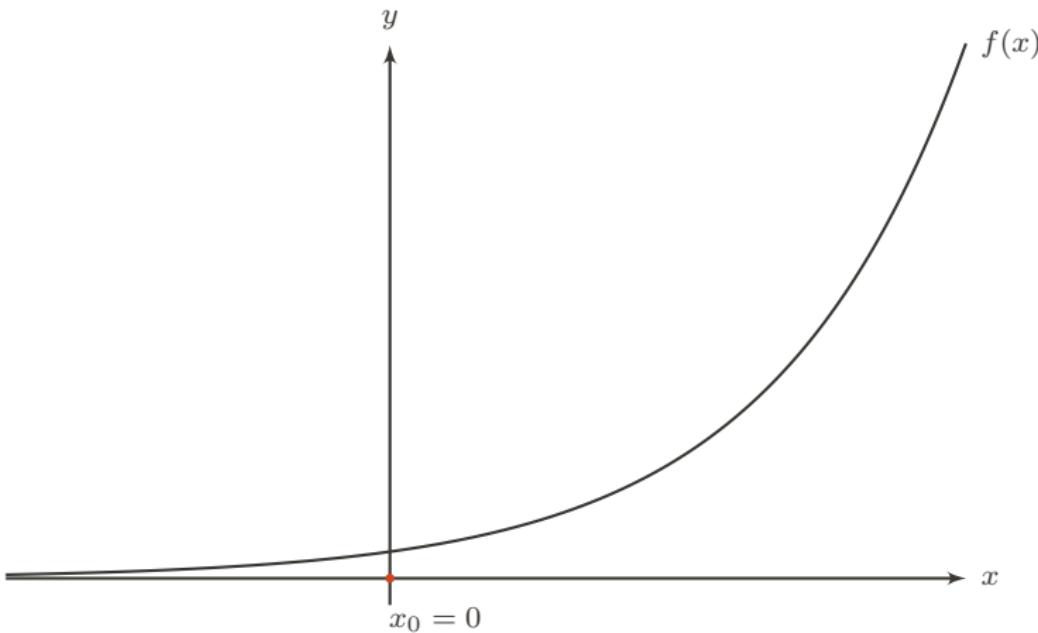
$$f'(x) = e^x \quad f'(x_0) = e^0 = 1 \quad P_1(x; 0) = 1 + \frac{1}{1!}x = 1 + x$$

$$f''(x) = e^x \quad f''(x_0) = e^0 = 1 \quad P_2(x; 0) = 1 + x + \frac{1}{2!}x^2 = 1 + x + \frac{x^2}{2}$$

$$f^{(3)}(x) = e^x \quad f^{(3)}(x_0) = e^0 = 1 \quad P_3(x; 0) = 1 + x + \frac{x^2}{2} + \frac{x^3}{6}$$

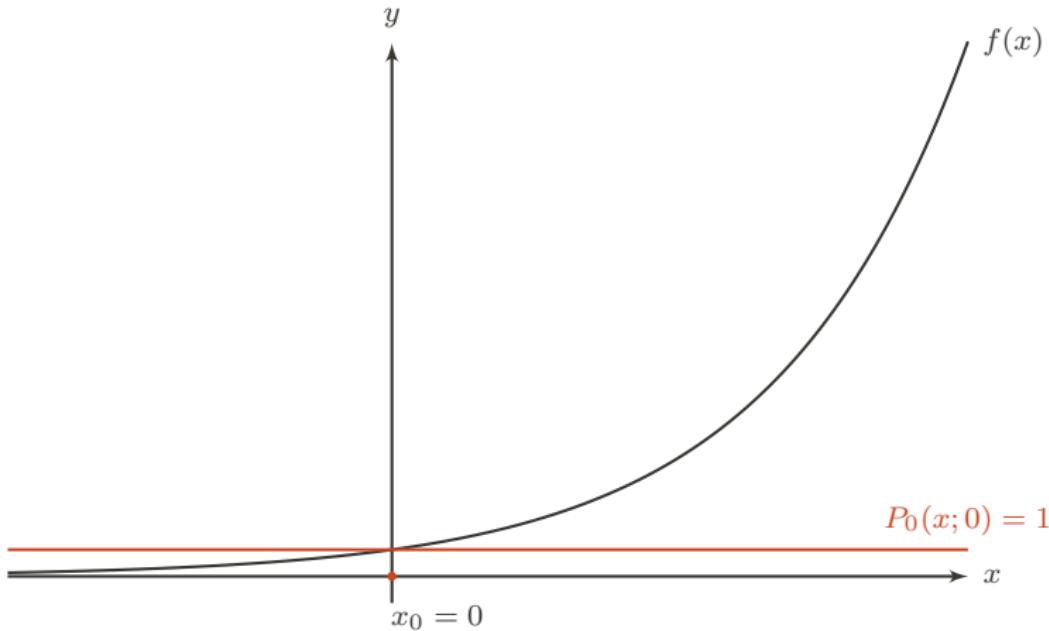
Taylor polynomials – example

$$f(x) = e^x, \quad x_0 = 0.$$



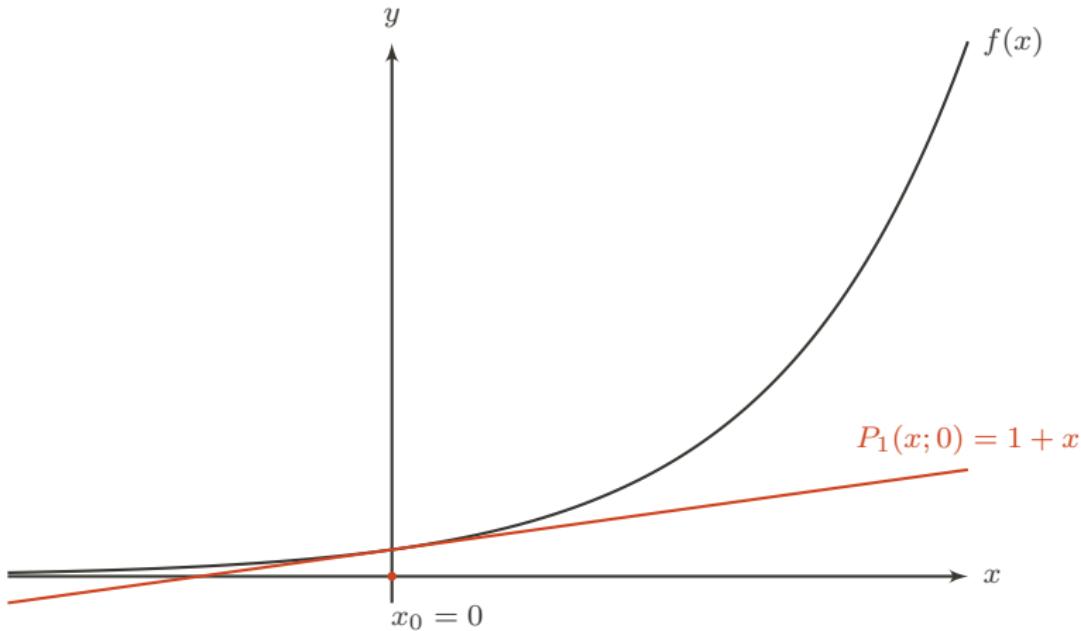
Taylor polynomials – example

$$f(x) = e^x, \quad x_0 = 0.$$



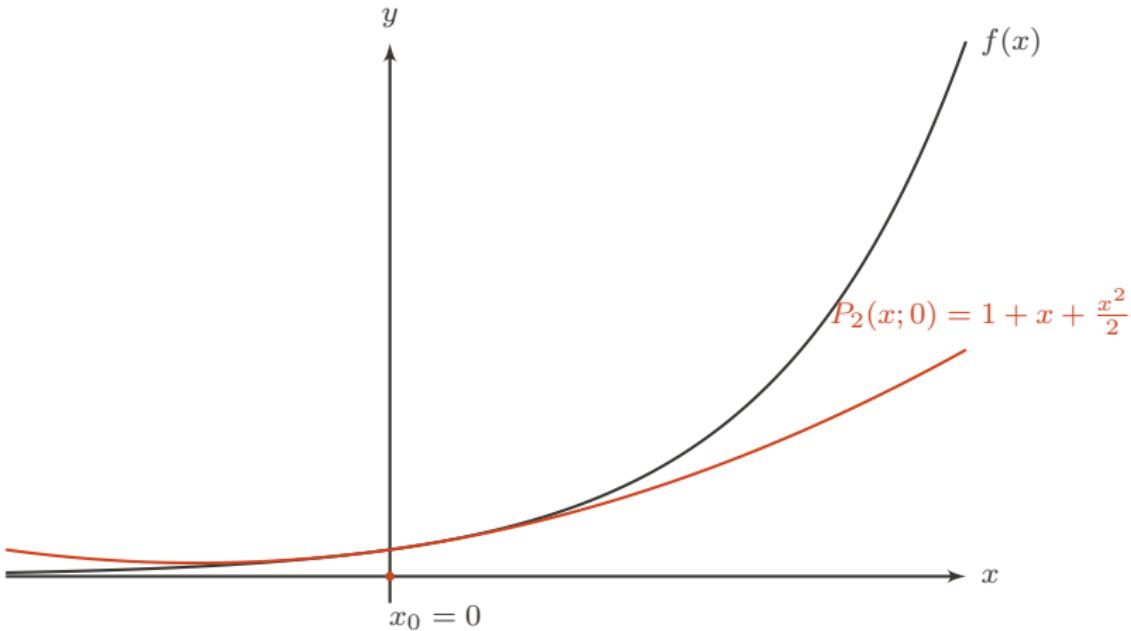
Taylor polynomials – example

$$f(x) = e^x, \quad x_0 = 0.$$



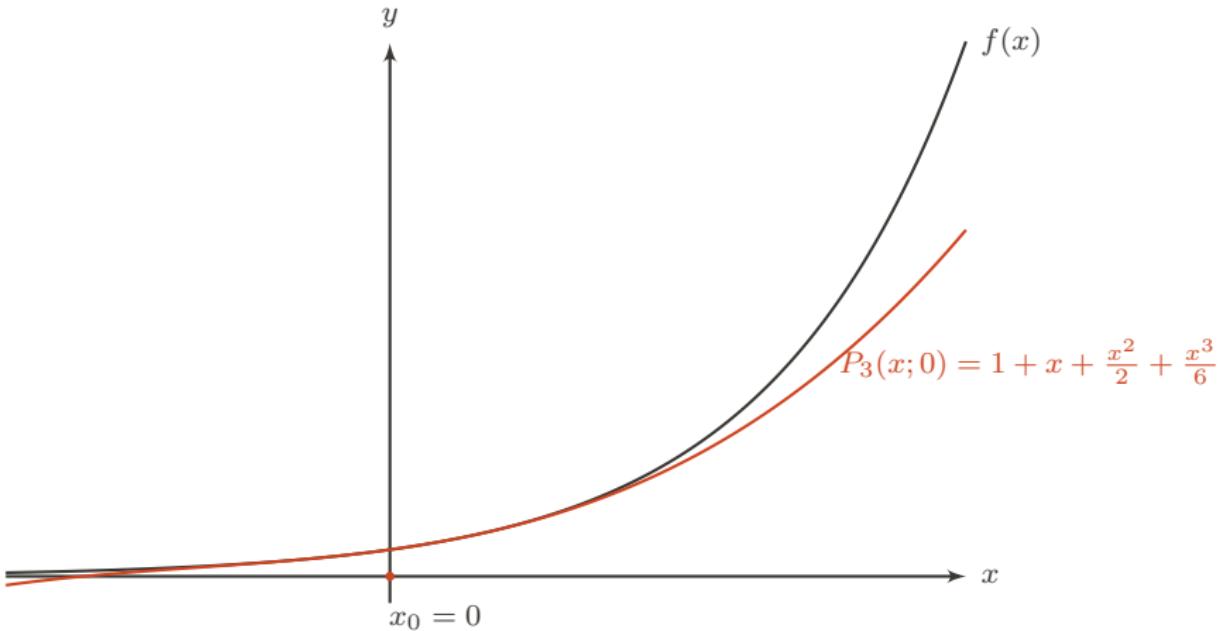
Taylor polynomials – example

$$f(x) = e^x, \quad x_0 = 0.$$



Taylor polynomials – example

$$f(x) = e^x, \quad x_0 = 0.$$



The assumptions of the Newton algorithm

Sufficient condition for a minimum: If $f'(x^*) = 0$ and $f''(x^*) > 0$ at some $x^* \in \mathcal{X}$, then the function $f(x)$ has a **local minimum** at x^* .

The assumptions of the Newton algorithm

Sufficient condition for a minimum: If $f'(x^*) = 0$ and $f''(x^*) > 0$ at some $x^* \in \mathcal{X}$, then the function $f(x)$ has a **local minimum** at x^* .

Inspired by this property, we will assume that $f''(x^*) > 0$ at a minimizer x^* of $f(x)$

The assumptions of the Newton algorithm

Sufficient condition for a minimum: If $f'(x^*) = 0$ and $f''(x^*) > 0$ at some $x^* \in \mathcal{X}$, then the function $f(x)$ has a **local minimum** at x^* .

Inspired by this property, we will assume that $f''(x^*) > 0$ at a minimizer x^* of $f(x)$

We further assume that the second derivative is **continuous**, which implies $f''(x) > 0$ in some **neighborhood** of x^* .

The Newton method

- The Newton method is an iterative **unconstrained** minimization algorithm for a function with continuous first and second derivatives

The Newton method

- The Newton method is an iterative **unconstrained** minimization algorithm for a function with continuous first and second derivatives
- The outcome of the method is a sequence of points x_1, x_2, \dots converging to a local minimizer x^* of $f(x)$

The Newton method

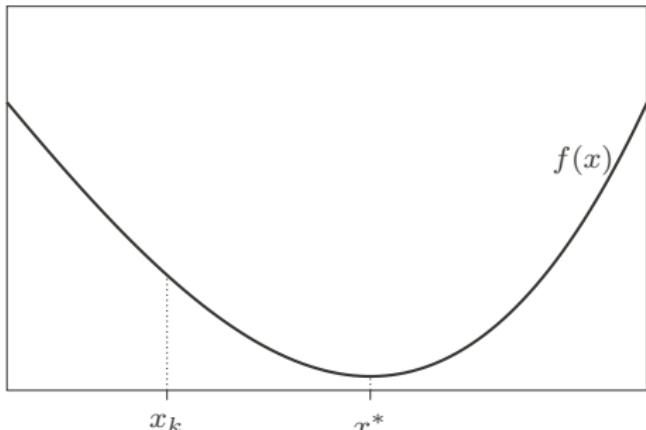
- The Newton method is an iterative **unconstrained** minimization algorithm for a function with continuous first and second derivatives
- The outcome of the method is a sequence of points x_1, x_2, \dots converging to a local minimizer x^* of $f(x)$
- At each iteration $k = 1, 2, \dots$ we **approximate** function $f(x)$ with a **second order Taylor polynomial** at point x_k :

$$f(x) \simeq P_2(x; x_k) = f(x_k) + f'(x_k)(x - x_k) + \frac{1}{2}f''(x_k)(x - x_k)^2$$

The Newton method

- The Newton method is an iterative **unconstrained** minimization algorithm for a function with continuous first and second derivatives
- The outcome of the method is a sequence of points x_1, x_2, \dots converging to a local minimizer x^* of $f(x)$
- At each iteration $k = 1, 2, \dots$ we **approximate** function $f(x)$ with a **second order Taylor polynomial** at point x_k :

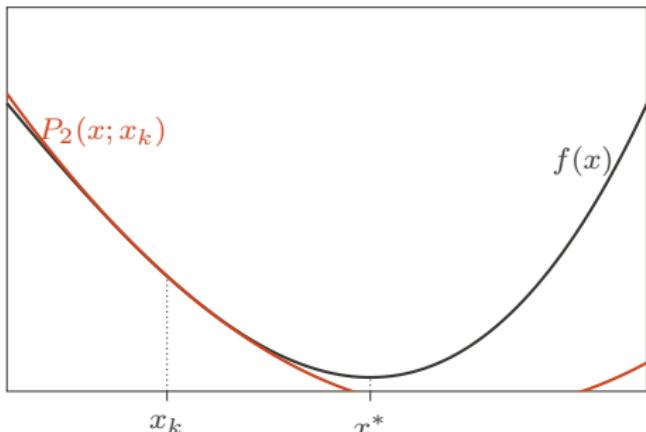
$$f(x) \simeq P_2(x; x_k) = f(x_k) + f'(x_k)(x - x_k) + \frac{1}{2}f''(x_k)(x - x_k)^2$$



The Newton method

- The Newton method is an iterative **unconstrained** minimization algorithm for a function with continuous first and second derivatives
- The outcome of the method is a sequence of points x_1, x_2, \dots converging to a local minimizer x^* of $f(x)$
- At each iteration $k = 1, 2, \dots$ we **approximate** function $f(x)$ with a **second order Taylor polynomial** at point x_k :

$$f(x) \simeq P_2(x; x_k) = f(x_k) + f'(x_k)(x - x_k) + \frac{1}{2}f''(x_k)(x - x_k)^2$$



Approximation of $f(x)$ with a **quadratic function** around x_k

Explanation: The minimum of quadratic function can easily be found **analytically**!

Newton method

From assumption, $f''(x) > 0$ around x^* , so for x_k sufficiently close to x^*

$$P_2(x; x_k) = f(x_k) + f'(x_k)(x - x_k) + \frac{1}{2} \underbrace{f''(x_k)}_{>0} (x - x_k)^2$$

is a **convex parabola** with a single minimum

Newton method

From assumption, $f''(x) > 0$ around x^* , so for x_k sufficiently close to x^*

$$P_2(x; x_k) = f(x_k) + f'(x_k)(x - x_k) + \frac{1}{2} \underbrace{f''(x_k)}_{>0} (x - x_k)^2$$

is a **convex parabola** with a single minimum

The next point x_{k+1} is chosen as the **minimum** of $P_2(x; x_k)$ obtained by setting the derivative of $P_2(x; x_k)$ to zero:

$$P'_2(x; x_k) = 0 \iff f'(x_k) + f''(x_k)(x - x_k) = 0$$

Newton method

From assumption, $f''(x) > 0$ around x^* , so for x_k sufficiently close to x^*

$$P_2(x; x_k) = f(x_k) + f'(x_k)(x - x_k) + \frac{1}{2} \underbrace{f''(x_k)}_{>0} (x - x_k)^2$$

is a **convex parabola** with a single minimum

The next point x_{k+1} is chosen as the **minimum** of $P_2(x; x_k)$ obtained by setting the derivative of $P_2(x; x_k)$ to zero:

$$P_2'(x; x_k) = 0 \iff f'(x_k) + f''(x_k)(x - x_k) = 0$$

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

Newton method

Input: procedures for evaluating $f'(x)$ and f''
at any point of the domain; number of iterations n ;

Initialize: x_1 sufficiently close to x^*

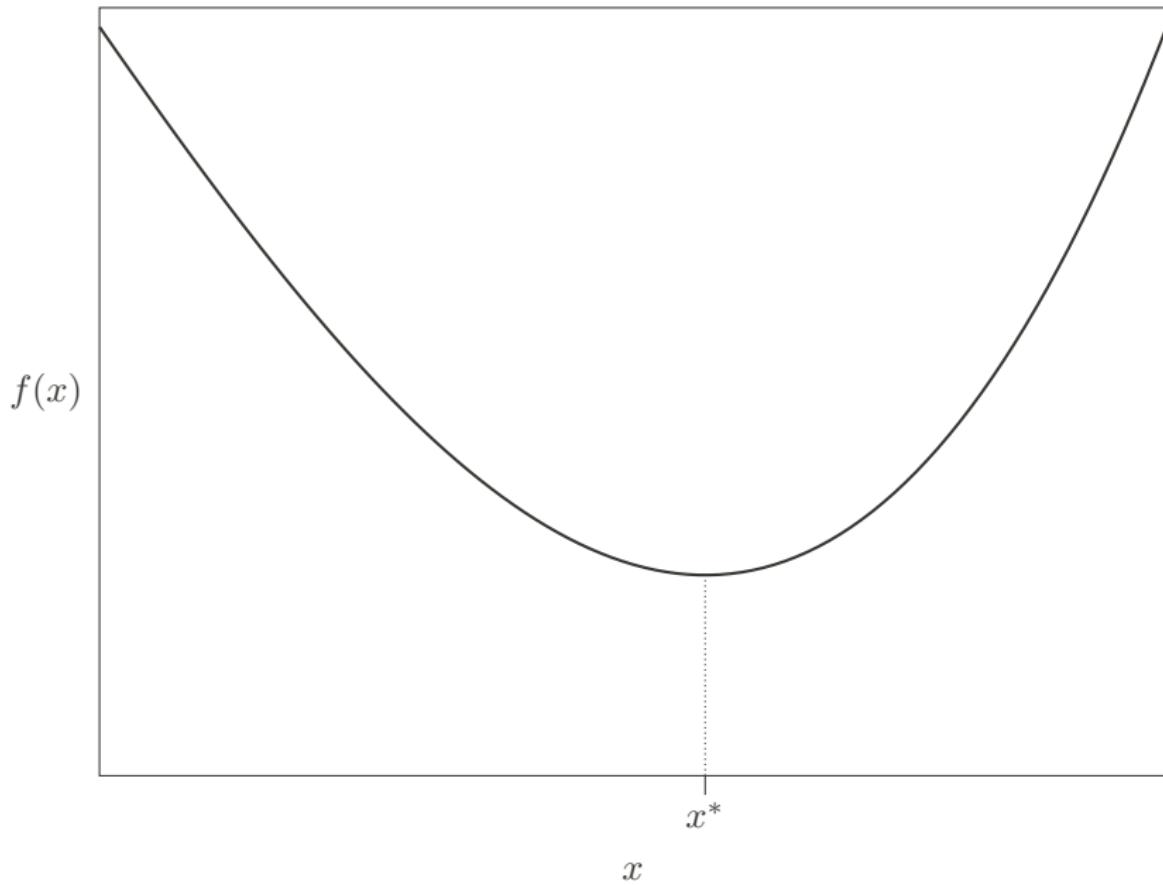
For $k = 1, 2, \dots, n$:

 Evaluate the 1st and 2nd derivatives at x_k : $f'(x_k)$, $f''(x_k)$

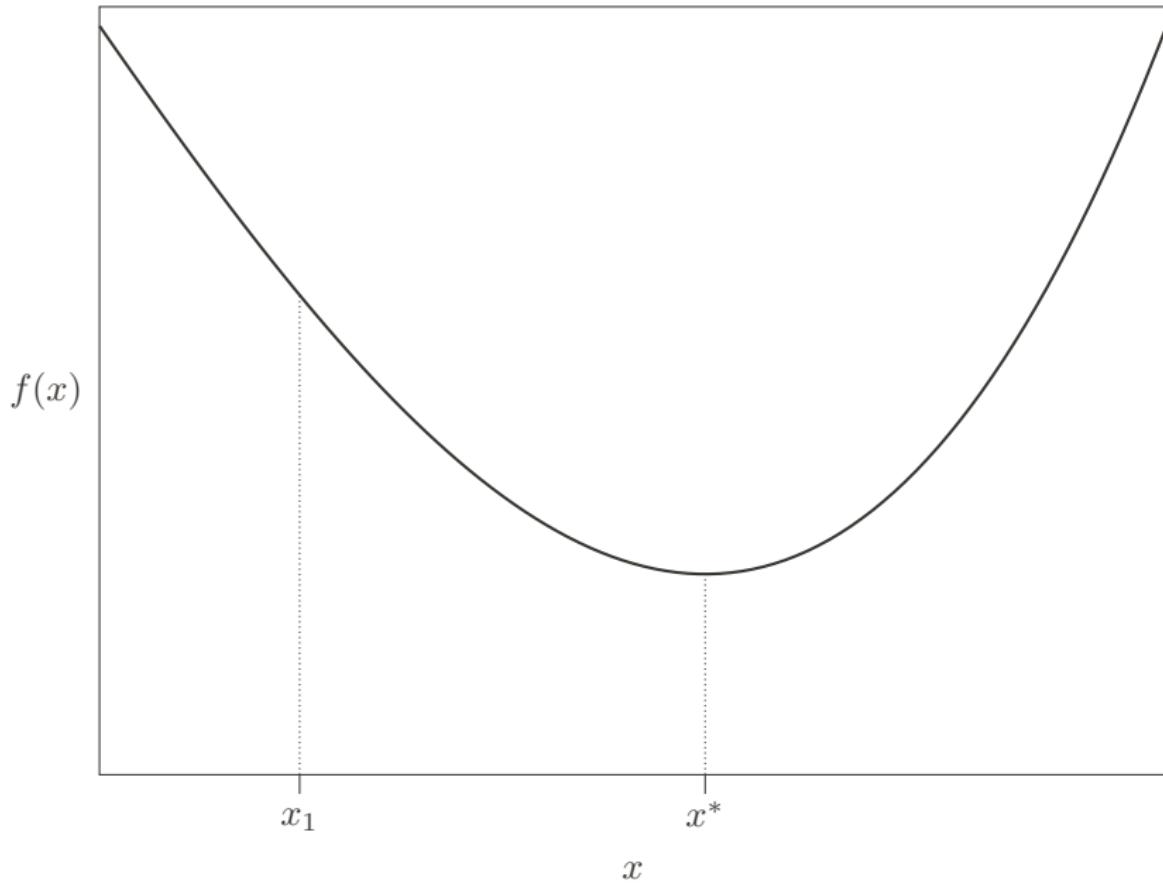
 Set $x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$

Return x_{n+1}

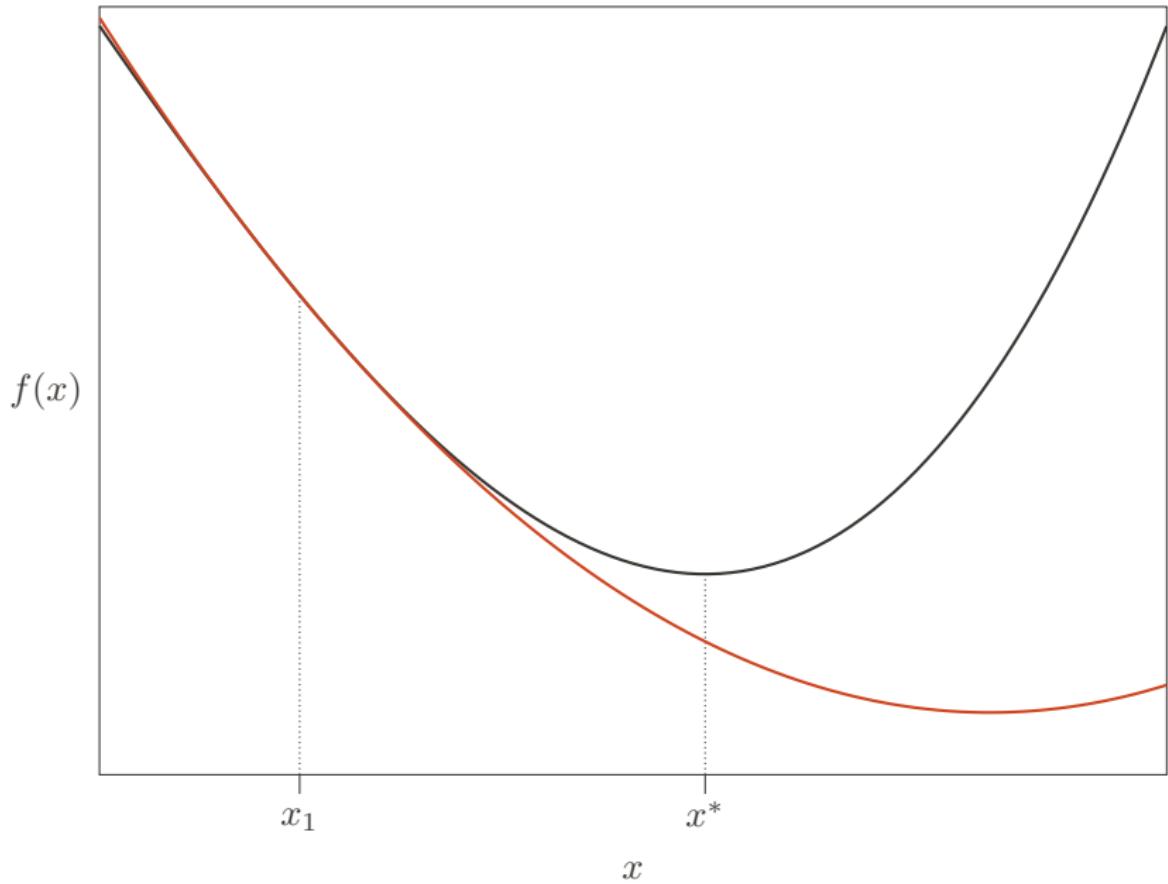
Newton method – example



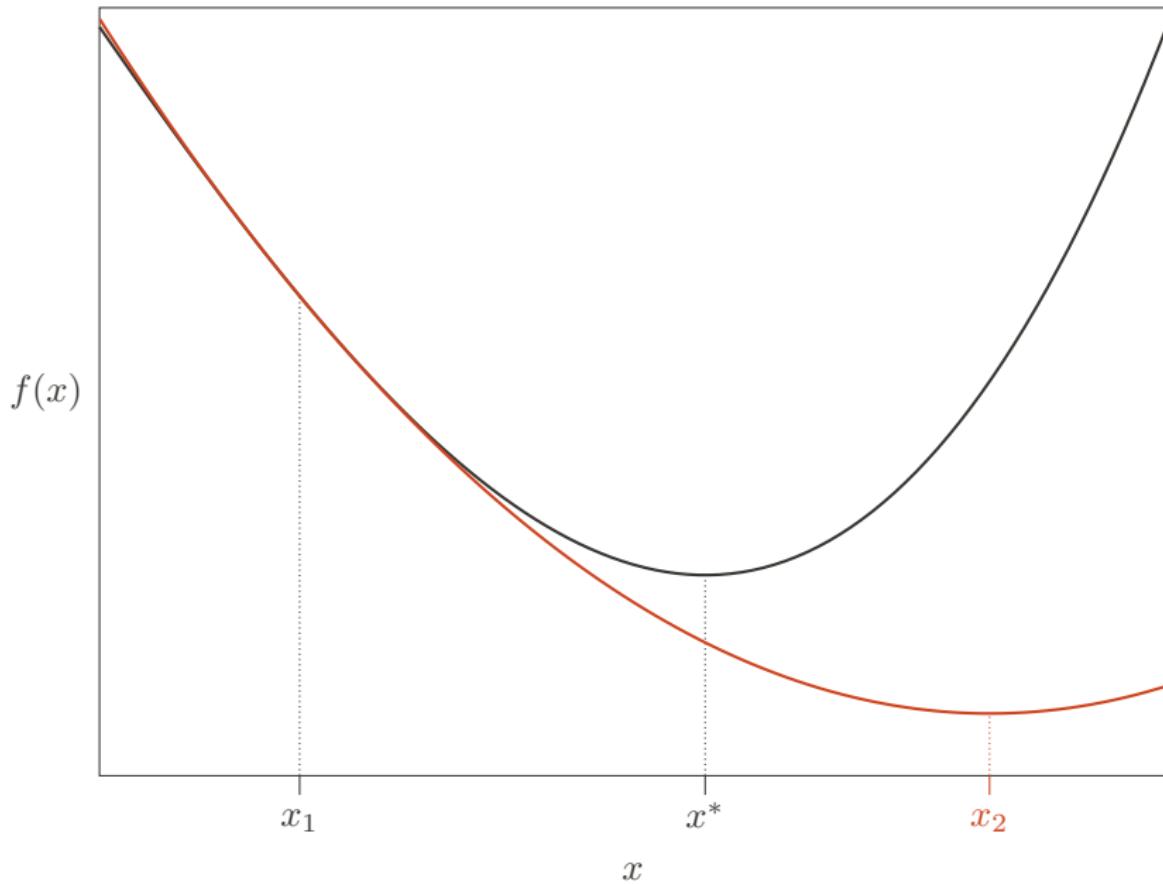
Newton method – example



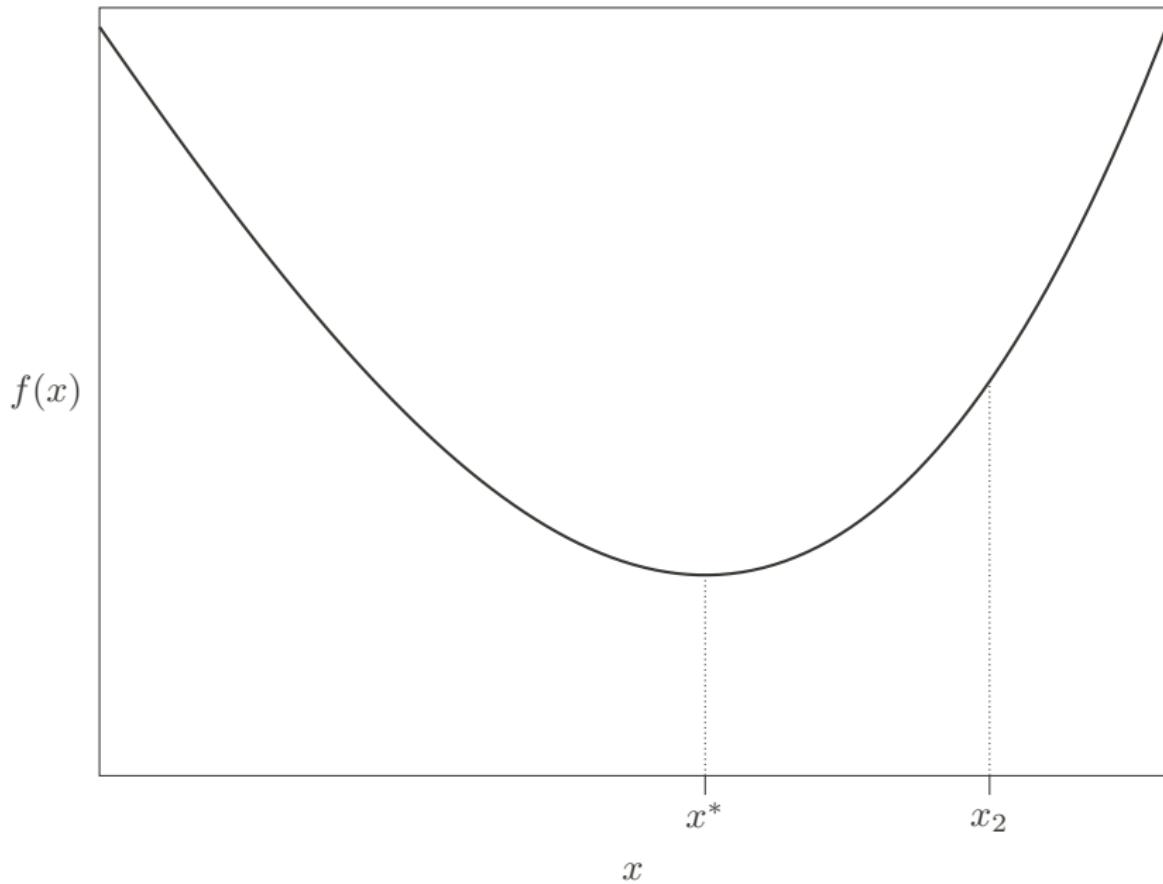
Newton method – example



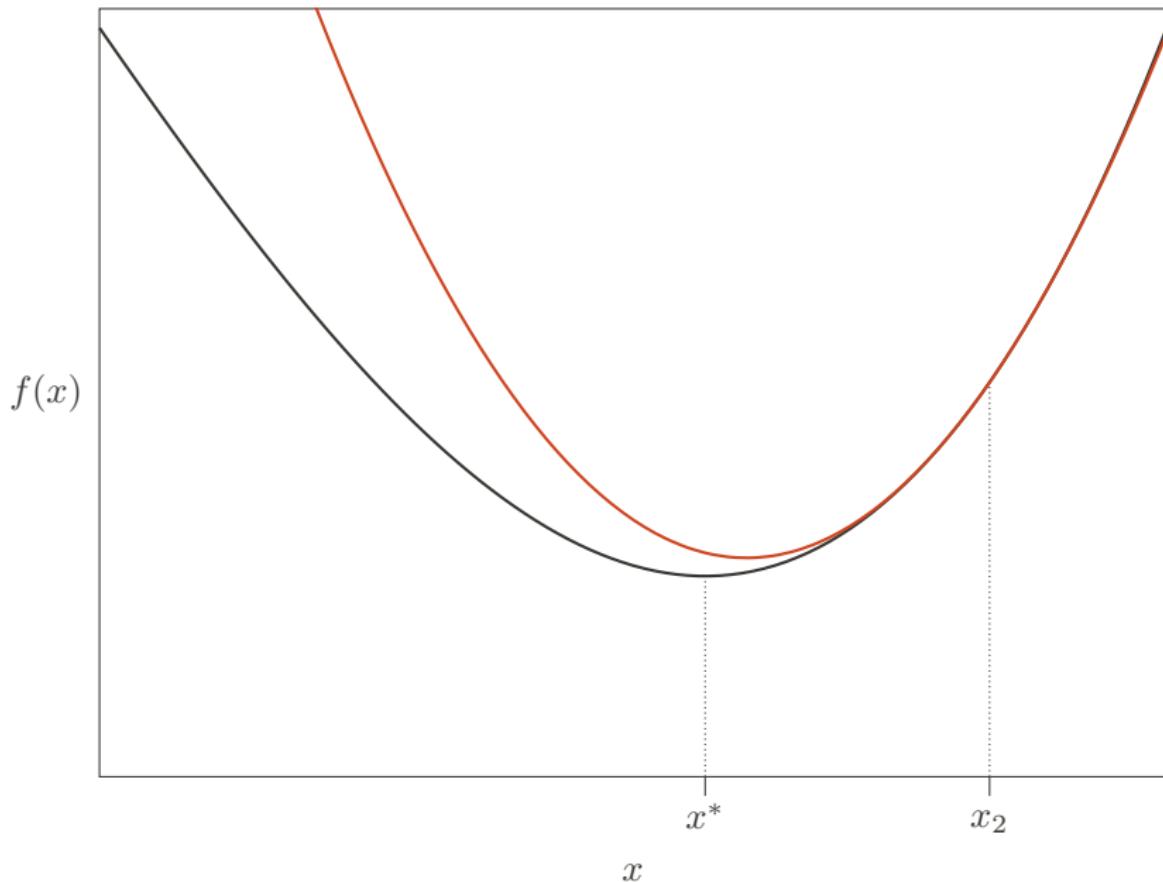
Newton method – example



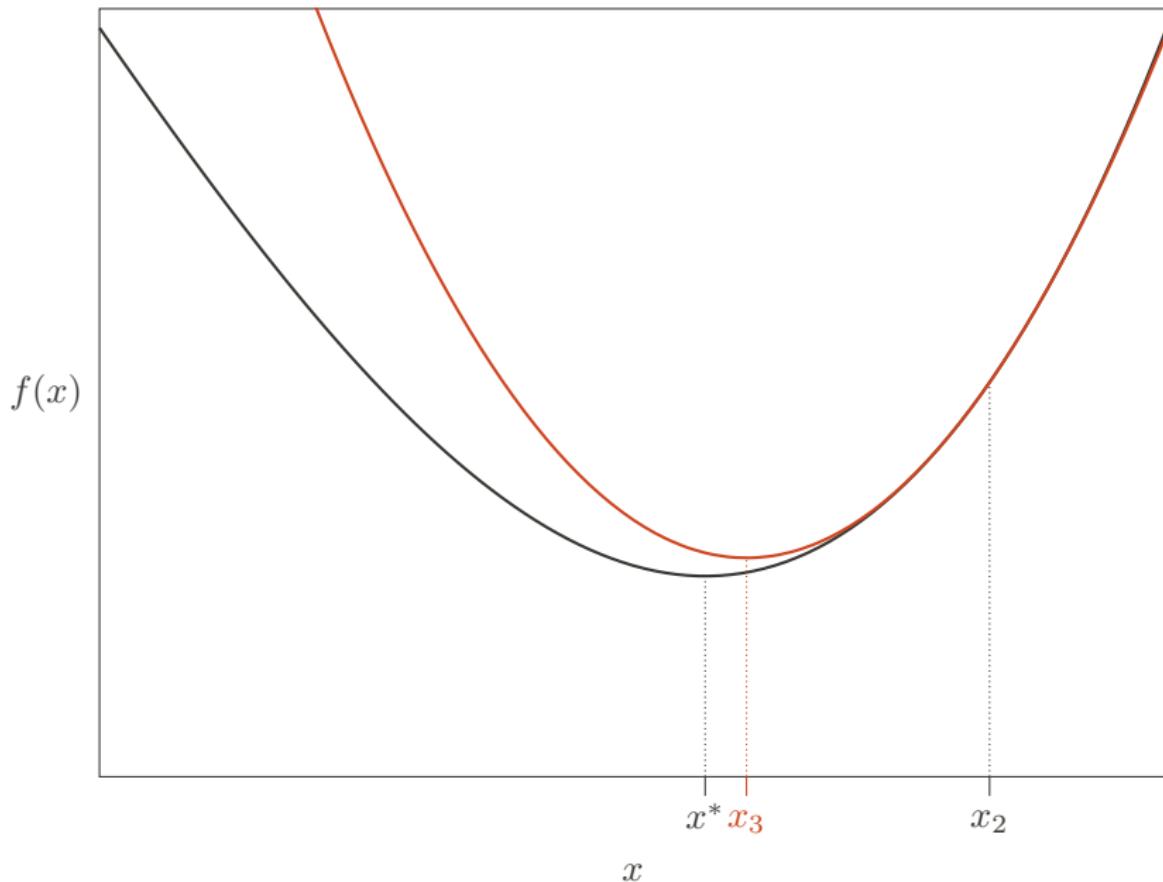
Newton method – example



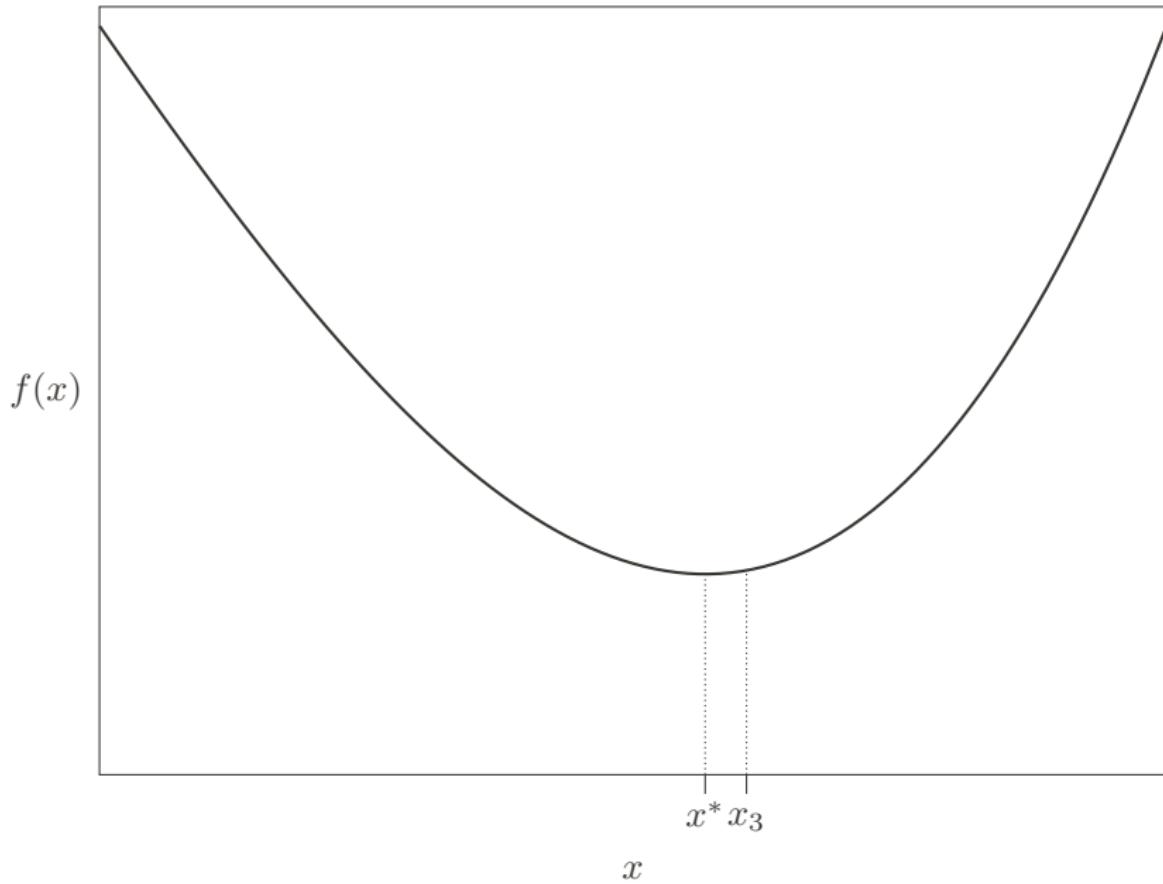
Newton method – example



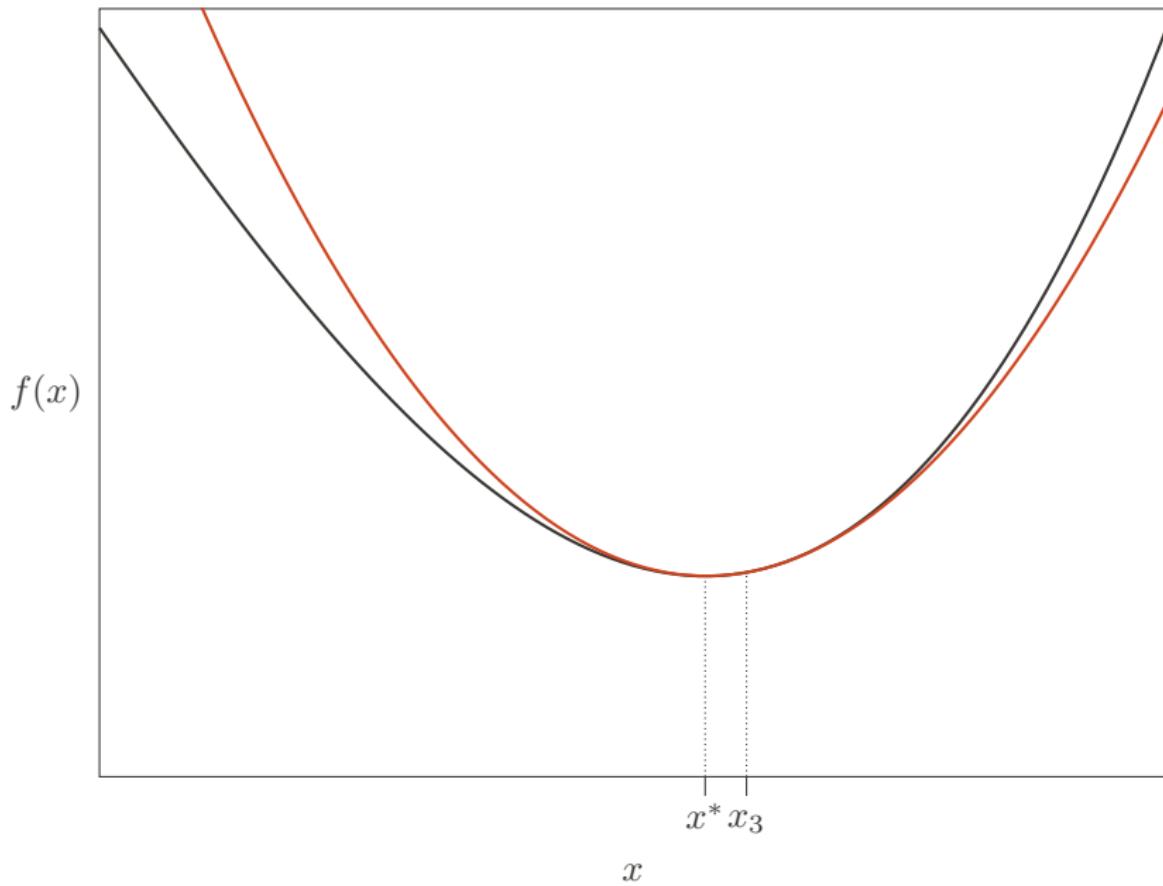
Newton method – example



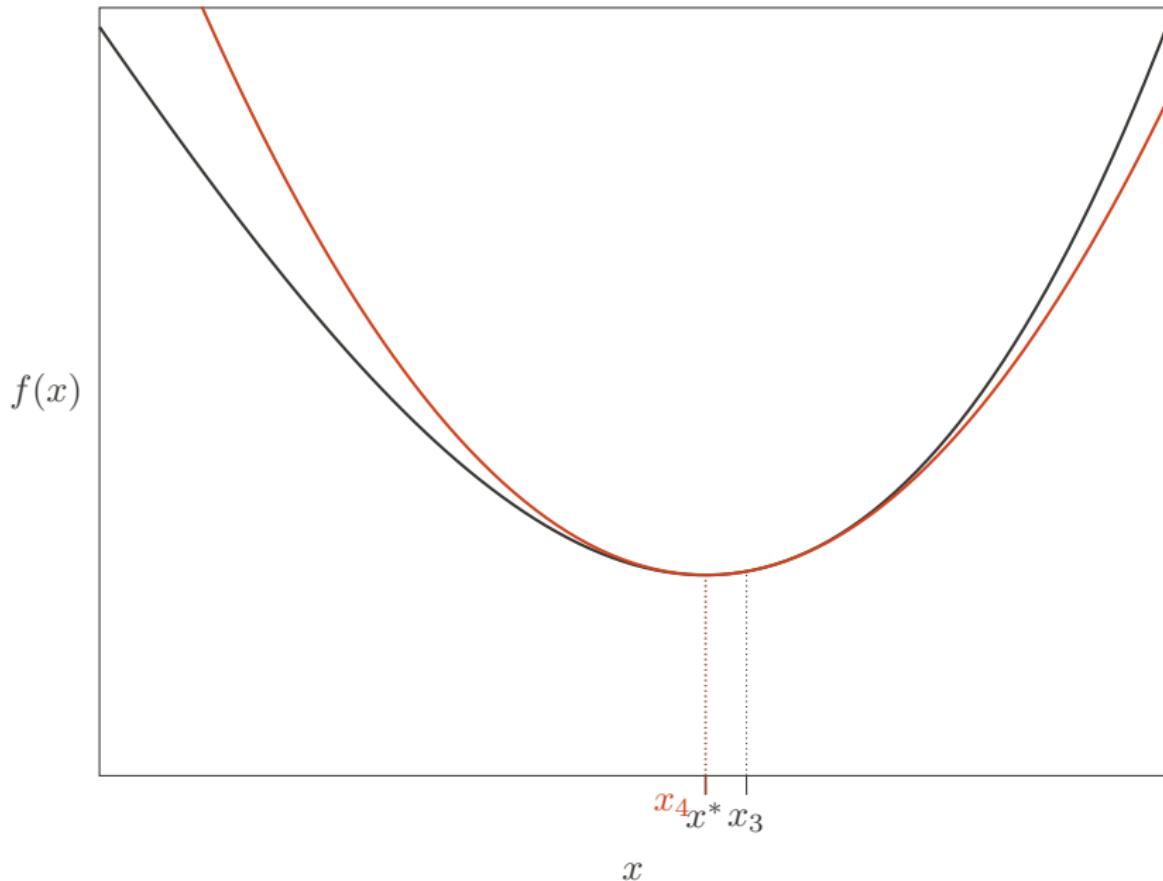
Newton method – example



Newton method – example



Newton method – example



Newton method – example

Find the minimum of $f(x) = e^x + e^{-x}$

$$f'(x) = e^x - e^{-x}, \quad f''(x) = e^x + e^{-x}$$

The global minimum easy to find analytically: $x^* = 0$.

Newton method – example

Find the minimum of $f(x) = e^x + e^{-x}$

$$f'(x) = e^x - e^{-x}, \quad f''(x) = e^x + e^{-x}$$

The global minimum easy to find analytically: $x^* = 0$.

Start from $x_1 = 1$

Newton method – example

Find the minimum of $f(x) = e^x + e^{-x}$

$$f'(x) = e^x - e^{-x}, \quad f''(x) = e^x + e^{-x}$$

The global minimum easy to find analytically: $x^* = 0$.

Start from $x_1 = 1$

k	x_k	$ x_k - x^* $
1	1	1
2	0.238	0.238
3	0.004	0.004
4	$2.9 \cdot 10^{-8}$	$2.9 \cdot 10^{-8}$
5	$-2.51 \cdot 10^{-17}$	$2.51 \cdot 10^{-17}$

Newton method – example

Find the minimum of $f(x) = e^x + e^{-x}$

$$f'(x) = e^x - e^{-x}, \quad f''(x) = e^x + e^{-x}$$

The global minimum easy to find analytically: $x^* = 0$.

Start from $x_1 = 10$

Newton method – example

Find the minimum of $f(x) = e^x + e^{-x}$

$$f'(x) = e^x - e^{-x}, \quad f''(x) = e^x + e^{-x}$$

The global minimum easy to find analytically: $x^* = 0$.

Start from $x_1 = 10$

k	x_k	$ x_k - x^* $
1	10	10
2	9	9
3	8	8
4	7	7
5	6	6
6	5	5
7	4	4
8	3	3
9	2.005	2.005
10	1.041	1.041
11	0.263	0.263
12	0.006	0.006
13	$6.8 \cdot 10^{-8}$	$6.8 \cdot 10^{-8}$
14	$1.5 \cdot 10^{-17}$	$1.5 \cdot 10^{-17}$

Newton method – example

Find the minimum of $f(x) = x^2 + 2x + 1$

$$f'(x) = 2x + 2 \quad f''(x) = 2$$

The global minimum easy to find analytically: $x^* = -1$.

Newton method – example

Find the minimum of $f(x) = x^2 + 2x + 1$

$$f'(x) = 2x + 2 \quad f''(x) = 2$$

The global minimum easy to find analytically: $x^* = -1$.

Start from $x_1 = 5$

k	x_k	$ x_k - x^* $
1	5	6
2	-1	0

Newton method – example

Find the minimum of $f(x) = x^2 + 2x + 1$

$$f'(x) = 2x + 2 \quad f''(x) = 2$$

The global minimum easy to find analytically: $x^* = -1$.

Start from $x_1 = 5$

k	x_k	$ x_k - x^* $
1	5	6
2	-1	0

Newton method finds the minimum of **quadratic function** in just a **single step!**

Why?

Newton method – example

Find the minimum of $f(x) = x^2 + 2x + 1$

$$f'(x) = 2x + 2 \quad f''(x) = 2$$

The global minimum easy to find analytically: $x^* = -1$.

Start from $x_1 = 5$

k	x_k	$ x_k - x^* $
1	5	6
2	-1	0

Newton method finds the minimum of **quadratic function** in just a **single step!**

Why? Because approximation of the quadratic function by a quadratic polynomial is exact!

Newton method – example

Find the minimum of $f(x) = x - \log(x)$

$$f'(x) = 1 - \frac{1}{x}, \quad f''(x) = \frac{1}{x^2}$$

The global minimum easy to find analytically: $x^* = 1$.

Newton method – example

Find the minimum of $f(x) = x - \log(x)$

$$f'(x) = 1 - \frac{1}{x}, \quad f''(x) = \frac{1}{x^2}$$

The global minimum easy to find analytically: $x^* = 1$.

Start from $x_1 = 1.5$

Newton method – example

Find the minimum of $f(x) = x - \log(x)$

$$f'(x) = 1 - \frac{1}{x}, \quad f''(x) = \frac{1}{x^2}$$

The global minimum easy to find analytically: $x^* = 1$.

Start from $x_1 = 1.5$

k	x_k	$ x_k - x^* $
1	1.5	0.5
2	0.75	0.25
3	0.9375	0.0625
4	0.996	0.004
5	0.999	$1.53 \cdot 10^{-5}$
6	0.999	$2.33 \cdot 10^{-10}$

Newton method – example

Find the minimum of $f(x) = x - \log(x)$

$$f'(x) = 1 - \frac{1}{x}, \quad f''(x) = \frac{1}{x^2}$$

The global minimum easy to find analytically: $x^* = 1$.

Start from $x_1 = 3$

Newton method – example

Find the minimum of $f(x) = x - \log(x)$

$$f'(x) = 1 - \frac{1}{x}, \quad f''(x) = \frac{1}{x^2}$$

The global minimum easy to find analytically: $x^* = 1$.

Start from $x_1 = 3$

k	x_k	$ x_k - x^* $
1	3	2
2	-3	4
3	-4	5
4	-15	16
5	-16	17
6	-255	256
...
10	-4294967295	4294967296

The algorithm **diverged** to (minus) infinity!

Newton method

- Needs access to the first two derivatives
- For unconstrained optimization only (in its basic form)
- Needs to be started **sufficiently close** to the minimum, otherwise can even diverge!
- Started sufficiently close to the minimum, the algorithm converges **rapidly**, usually within just a few iterations! (therefore used very often in practice)

Convergence of the Newton method

Remainder (Taylor's theorem):

$$f(x) = P_m(x; x_0) + \frac{f^{(m+1)}(\xi)}{(m+1)!} (x - x_0)^{m+1}$$

where ξ is a point in between x and x_0 . In particular for $m = 1$:

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2} f''(\xi)(x - x_0)^2$$

Convergence of the Newton method

Remainder (Taylor's theorem):

$$f(x) = P_m(x; x_0) + \frac{f^{(m+1)}(\xi)}{(m+1)!} (x - x_0)^{m+1}$$

where ξ is a point in between x and x_0 . In particular for $m = 1$:

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2} f''(\xi)(x - x_0)^2$$

Conclusion: Using the theorem to the **derivative** $f'(x)$ (with $m = 1$) for $x = x^*$ and $x_0 = x_k$, we have:

$$\underbrace{f'(x^*)}_{=0} = f'(x_k) + f''(x_k)(x^* - x_k) + \frac{1}{2} f'''(\xi)(x^* - x_k)^2,$$

(assuming the function is three times differentiable)

Convergence of the Newton method

Remainder (Taylor's theorem):

$$f(x) = P_m(x; x_0) + \frac{f^{(m+1)}(\xi)}{(m+1)!} (x - x_0)^{m+1}$$

where ξ is a point in between x and x_0 . In particular for $m = 1$:

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2} f''(\xi)(x - x_0)^2$$

Conclusion: Using the theorem to the **derivative** $f'(x)$ (with $m = 1$) for $x = x^*$ and $x_0 = x_k$, we have:

$$\underbrace{f'(x^*)}_{=0} = f'(x_k) + f''(x_k)(x^* - x_k) + \frac{1}{2} f'''(\xi)(x^* - x_k)^2,$$

(assuming the function is three times differentiable)

$$\text{which gives: } -f'(x_k) = f''(x_k)(x^* - x_k) + \frac{1}{2} f'''(\xi)(x^* - x_k)^2$$

Convergence of the Newton method

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

From the definition of Newton method:

$$x_{k+1} - x^* = x_k - x^* - \frac{f'(x_k)}{f''(x_k)}$$

Convergence of the Newton method

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

From the definition of Newton method:

$$x_{k+1} - x^* = x_k - x^* - \frac{f'(x_k)}{f''(x_k)}$$

Using the expression in the box above:

$$x_{k+1} - x^* = x_k - x^* + \frac{f''(x_k)(x^* - x_k) + \frac{1}{2}f'''(\xi)(x^* - x_k)^2}{f''(x_k)}$$

Convergence of the Newton method

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

From the definition of Newton method:

$$x_{k+1} - x^* = x_k - x^* - \frac{f'(x_k)}{f''(x_k)}$$

Using the expression in the box above:

$$\begin{aligned} x_{k+1} - x^* &= x_k - x^* + \frac{f''(x_k)(x^* - x_k) + \frac{1}{2}f'''(\xi)(x^* - x_k)^2}{f''(x_k)} \\ &= \frac{1}{2} \frac{f'''(\xi)}{f''(x_k)} (x^* - x_k)^2 \end{aligned}$$

Convergence of the Newton method

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

From the definition of Newton method:

$$x_{k+1} - x^* = x_k - x^* - \frac{f'(x_k)}{f''(x_k)}$$

Using the expression in the box above:

$$\begin{aligned} x_{k+1} - x^* &= x_k - x^* + \frac{f''(x_k)(x^* - x_k) + \frac{1}{2}f'''(\xi)(x^* - x_k)^2}{f''(x_k)} \\ &= \frac{1}{2} \frac{f'''(\xi)}{f''(x_k)} (x^* - x_k)^2 \end{aligned}$$

The convergence is **quadratic**:

$$|x_{k+1} - x^*| = \beta_k (x_k - x^*)^2, \quad \beta_k = \left| \frac{1}{2} \frac{f'''(\xi)}{f''(x_k)} \right|$$

The order of convergence

Let $p \geq 1$ be the largest number for which the following limit exists:

$$\beta = \lim_{k \rightarrow \infty} \beta_k, \quad \text{where } \beta_k = \frac{|x_{k+1} - x^*|}{|x_k - x^*|^p}.$$

Then p is called **the order of convergence** of a sequence x_k to x^* .

The order of convergence

Let $p \geq 1$ be the largest number for which the following limit exists:

$$\beta = \lim_{k \rightarrow \infty} \beta_k, \quad \text{where } \beta_k = \frac{|x_{k+1} - x^*|}{|x_k - x^*|^p}.$$

Then p is called **the order of convergence** of a sequence x_k to x^* .

- If $p = 1$ and $\beta \in (0, 1)$ then the sequence **converges linearly** (dichotomous search, golden-section search, bisection)
- If $p = 2$ the the sequence **converges quadratically** (Newton method)

The order of convergence – comparison

For simplicity assume $|x_{k+1} - x^*| = \beta|x_k - x^*|^p$ for all n . We will compare the orders of convergence for $\beta = 0.5$ and $p = 1, 2$, i.e.:

$$|x_{k+1} - x^*| = 0.5|x_k - x^*|, \quad |x_{k+1} - x^*| = 0.5|x_k - x^*|^2$$

The order of convergence – comparison

For simplicity assume $|x_{k+1} - x^*| = \beta|x_k - x^*|^p$ for all n . We will compare the orders of convergence for $\beta = 0.5$ and $p = 1, 2$, i.e.:

$$|x_{k+1} - x^*| = 0.5|x_k - x^*|, \quad |x_{k+1} - x^*| = 0.5|x_k - x^*|^2$$

Start from $|x_1 - x^*| = 1$.

k	$ x_k - x^* $	
	$p = 1$	$p = 2$
1	1	1
2	0.5	0.5
3	0.25	0.125
4	0.125	0.00781
5	0.0625	$3.05 \cdot 10^{-5}$
6	0.03125	$4.66 \cdot 10^{-10}$
7	0.01563	$1.08 \cdot 10^{-19}$

The order of convergence – comparison

For simplicity assume $|x_{k+1} - x^*| = \beta|x_k - x^*|^p$ for all n . We will compare the orders of convergence for $\beta = 0.5$ and $p = 1, 2$, i.e.:

$$|x_{k+1} - x^*| = 0.5|x_k - x^*|, \quad |x_{k+1} - x^*| = 0.5|x_k - x^*|^2$$

Start from $|x_1 - x^*| = 4$.

k	$ x_k - x^* $	
	$p = 1$	$p = 2$
1	4	4
2	2	8
3	1	32
4	0.5	512
5	0.25	131072
6	0.125	8589934592

The secant method

- The second derivative used in the Newton method can sometimes be hard (or impossible) to access.

The secant method

- The second derivative used in the Newton method can sometimes be hard (or impossible) to access.
- **Solution:** using the definition of the second derivative:

$$f''(x) = \lim_{\Delta \rightarrow 0} \frac{f'(x + \Delta) - f'(x)}{\Delta}$$

we approximate:

$$f''(x_k) \simeq \frac{f'(x_k) - f'(x_{k-1})}{x_k - x_{k-1}}$$

The secant method

- The second derivative used in the Newton method can sometimes be hard (or impossible) to access.
- **Solution:** using the definition of the second derivative:

$$f''(x) = \lim_{\Delta \rightarrow 0} \frac{f'(x + \Delta) - f'(x)}{\Delta}$$

we approximate:

$$f''(x_k) \simeq \frac{f'(x_k) - f'(x_{k-1})}{x_k - x_{k-1}}$$

- As a result, we obtained the **secant method**:

$$x_{k+1} = x_k - \frac{f'(x_k)(x_k - x_{k-1})}{f'(x_k) - f'(x_{k-1})}$$

The secant method

- The second derivative used in the Newton method can sometimes be hard (or impossible) to access.
- **Solution:** using the definition of the second derivative:

$$f''(x) = \lim_{\Delta \rightarrow 0} \frac{f'(x + \Delta) - f'(x)}{\Delta}$$

we approximate:

$$f''(x_k) \simeq \frac{f'(x_k) - f'(x_{k-1})}{x_k - x_{k-1}}$$

- As a result, we obtained the **secant method**:

$$x_{k+1} = x_k - \frac{f'(x_k)(x_k - x_{k-1})}{f'(x_k) - f'(x_{k-1})}$$

- It can be shown that the secant method has order of convergence
 $p = \frac{1+\sqrt{5}}{2} \simeq 1.618$

Stopping rules for optimization algorithms

All presented algorithms can in principle work indefinitely long, improving the approximation of the minimum at each new iteration, but will usually never reach the minimum precisely.

Thus we need a **stopping rule**

Stopping rules for optimization algorithms

All presented algorithms can in principle work indefinitely long, improving the approximation of the minimum at each new iteration, but will usually never reach the minimum precisely.

Thus we need a **stopping rule**

For simplicity, we so far used a stopping rule based on a **computational budget**, where a number of iterations (function evaluations) n is given in advance.

Stopping rules for optimization algorithms

All presented algorithms can in principle work indefinitely long, improving the approximation of the minimum at each new iteration, but will usually never reach the minimum precisely.

Thus we need a **stopping rule**

For simplicity, we so far used a stopping rule based on a **computational budget**, where a number of iterations (function evaluations) n is given in advance.

Alternatively, we can consider stopping when the **approximation error** of x_k drops below some small number ϵ (np. $\epsilon = 10^{-8}$). For example, stop the algorithm:

- when $|x_k - x^*| \leq \epsilon$ (from a theoretical analysis)

Stopping rules for optimization algorithms

All presented algorithms can in principle work indefinitely long, improving the approximation of the minimum at each new iteration, but will usually never reach the minimum precisely.

Thus we need a **stopping rule**

For simplicity, we so far used a stopping rule based on a **computational budget**, where a number of iterations (function evaluations) n is given in advance.

Alternatively, we can consider stopping when the **approximation error** of x_k drops below some small number ϵ (np. $\epsilon = 10^{-8}$). For example, stop the algorithm:

- when $|x_k - x^*| \leq \epsilon$ (from a theoretical analysis)
- when $|f'(x_k)| \leq \epsilon$ (using $f'(x^*) = 0$)

Stopping rules for optimization algorithms

All presented algorithms can in principle work indefinitely long, improving the approximation of the minimum at each new iteration, but will usually never reach the minimum precisely.

Thus we need a **stopping rule**

For simplicity, we so far used a stopping rule based on a **computational budget**, where a number of iterations (function evaluations) n is given in advance.

Alternatively, we can consider stopping when the **approximation error** of x_k drops below some small number ϵ (np. $\epsilon = 10^{-8}$). For example, stop the algorithm:

- when $|x_k - x^*| \leq \epsilon$ (from a theoretical analysis)
- when $|f'(x_k)| \leq \epsilon$ (using $f'(x^*) = 0$)
- when $\frac{|x_k - x_{k-1}|}{|x_{k-1}|} \leq \epsilon$ (relative change in the solution small)

Stopping rules for optimization algorithms

All presented algorithms can in principle work indefinitely long, improving the approximation of the minimum at each new iteration, but will usually never reach the minimum precisely.

Thus we need a **stopping rule**

For simplicity, we so far used a stopping rule based on a **computational budget**, where a number of iterations (function evaluations) n is given in advance.

Alternatively, we can consider stopping when the **approximation error** of x_k drops below some small number ϵ (np. $\epsilon = 10^{-8}$). For example, stop the algorithm:

- when $|x_k - x^*| \leq \epsilon$ (from a theoretical analysis)
- when $|f'(x_k)| \leq \epsilon$ (using $f'(x^*) = 0$)
- when $\frac{|x_k - x_{k-1}|}{|x_{k-1}|} \leq \epsilon$ (relative change in the solution small)
- when $\frac{|f(x_k) - f(x_{k-1})|}{|f(x_{k-1})|} \leq \epsilon$ (relative change in the function value small)

Optimization Methods for Data Analysis

2. Mathematical basis

Wojciech Kotłowski

Institute of Computing Science, PUT
<http://www.cs.put.poznan.pl/wkotlowski/>

14.03.2022

Outline

1. Linear algebra: vectors and matrices
2. Calculus: partial derivatives, gradient, Hessian.

Vectors and matrices

Vector space \mathbb{R}^n

A space \mathbb{R}^n is defined as an n -fold Cartesian product

$$\mathbb{R}^n = \underbrace{\mathbb{R} \times \dots \times \mathbb{R}}_n$$

The elements of this space $x = (x_1, x_2, \dots, x_n)$ are called **vectors**.

Vector space \mathbb{R}^n

A space \mathbb{R}^n is defined as an n -fold Cartesian product

$$\mathbb{R}^n = \underbrace{\mathbb{R} \times \dots \times \mathbb{R}}_n$$

The elements of this space $\mathbf{x} = (x_1, x_2, \dots, x_n)$ are called **vectors**.

Operations on vectors:

- Addition: for $\mathbf{x} = (x_1, \dots, x_n)$ i $\mathbf{y} = (y_1, \dots, y_n)$

$$\mathbf{x} + \mathbf{y} = (x_1 + y_1, \dots, x_n + y_n)$$

- Scalar multiplication: for $\mathbf{x} = (x_1, \dots, x_n)$ i $\alpha \in \mathbb{R}$

$$\alpha \mathbf{x} = (\alpha x_1, \dots, \alpha x_n)$$

Vector space \mathbb{R}^n

A space \mathbb{R}^n is defined as an n -fold Cartesian product

$$\mathbb{R}^n = \underbrace{\mathbb{R} \times \dots \times \mathbb{R}}_n$$

The elements of this space $\mathbf{x} = (x_1, x_2, \dots, x_n)$ are called **vectors**.

Operations on vectors:

- Addition: for $\mathbf{x} = (x_1, \dots, x_n)$ i $\mathbf{y} = (y_1, \dots, y_n)$

$$\mathbf{x} + \mathbf{y} = (x_1 + y_1, \dots, x_n + y_n)$$

- Scalar multiplication: for $\mathbf{x} = (x_1, \dots, x_n)$ i $\alpha \in \mathbb{R}$

$$\alpha \mathbf{x} = (\alpha x_1, \dots, \alpha x_n)$$

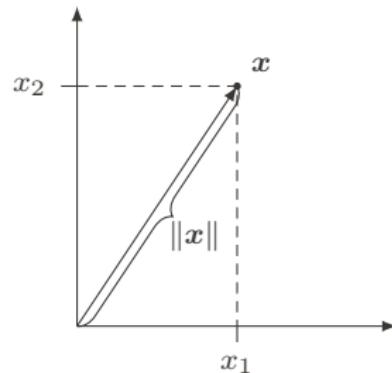
Special symbols:

$$\mathbf{0} = (0, 0, \dots, 0), \quad \mathbf{e}_k = (0, \dots, 0, \underbrace{1}_{k\text{-th entry}}, 0, \dots, 0)$$

Vector norm

A **norm** (length) of a vector:

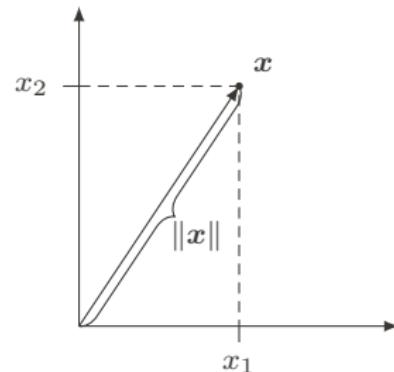
$$\|x\| = \sqrt{x_1^2 + \dots + x_n^2}$$



Vector norm

A **norm** (length) of a vector:

$$\|x\| = \sqrt{x_1^2 + \dots + x_n^2}$$

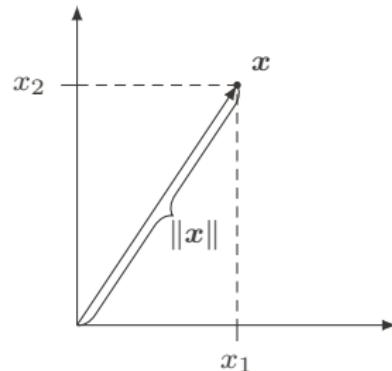


Fact: It holds $\|\alpha x\| = |\alpha| \|x\|$ for any scalar $\alpha \in \mathbb{R}$

Vector norm

A **norm** (length) of a vector:

$$\|x\| = \sqrt{x_1^2 + \dots + x_n^2}$$



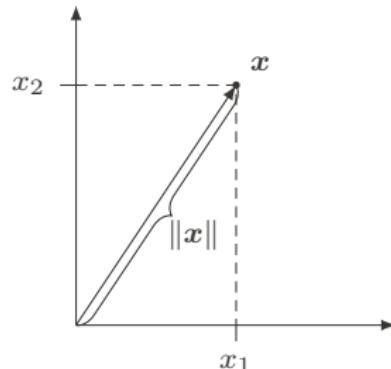
Fact: It holds $\|\alpha x\| = |\alpha| \|x\|$ for any scalar $\alpha \in \mathbb{R}$

A vector v such, that $\|v\| = 1$, is called a **unit vector**

Vector norm

A **norm** (length) of a vector:

$$\|x\| = \sqrt{x_1^2 + \dots + x_n^2}$$



Fact: It holds $\|\alpha x\| = |\alpha| \|x\|$ for any scalar $\alpha \in \mathbb{R}$

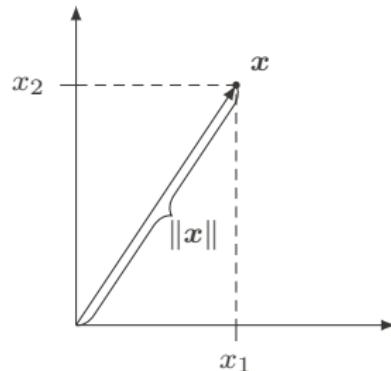
A vector v such, that $\|v\| = 1$, is called a **unit vector**

Fact: A vector $v = \frac{x}{\|x\|}$ is a unit vector

Vector norm

A **norm** (length) of a vector:

$$\|x\| = \sqrt{x_1^2 + \dots + x_n^2}$$



Fact: It holds $\|\alpha x\| = |\alpha| \|x\|$ for any scalar $\alpha \in \mathbb{R}$

A vector v such, that $\|v\| = 1$, is called a **unit vector**

Fact: A vector $v = \frac{x}{\|x\|}$ is a unit vector

Proof:

$$\|v\| = \left\| \frac{x}{\|x\|} \right\| = \frac{1}{\|x\|} \|x\| = 1$$

Digression: ℓ_p norms

A vector norm can be generalized to an ℓ_p -norm, for any $p \in [1, \infty]$

$$\|\mathbf{x}\|_p = \sqrt[p]{|x_1|^p + \dots + |x_n|^p}$$

- $\|\mathbf{x}\|_1 = |x_1| + \dots + |x_n|$
- $\|\mathbf{x}\|_2 = \sqrt{x_1^2 + \dots + x_n^2} \equiv \|\mathbf{x}\|$
- $\|\mathbf{x}\|_\infty = \max_{i=1,\dots,n} |x_i|$

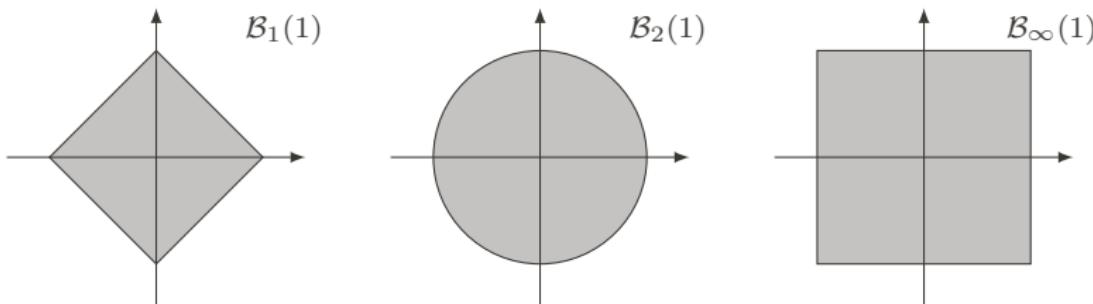
Digression: ℓ_p norms

A vector norm can be generalized to an ℓ_p -norm, for any $p \in [1, \infty]$

$$\|\mathbf{x}\|_p = \sqrt[p]{|x_1|^p + \dots + |x_n|^p}$$

- $\|\mathbf{x}\|_1 = |x_1| + \dots + |x_n|$
- $\|\mathbf{x}\|_2 = \sqrt{x_1^2 + \dots + x_n^2} \equiv \|\mathbf{x}\|$
- $\|\mathbf{x}\|_\infty = \max_{i=1,\dots,n} |x_i|$

A p -norm ball of radius r : $\mathcal{B}_p(r) = \{\mathbf{x}: |x_1|^p + \dots + |x_n|^p \leq r^p\}$

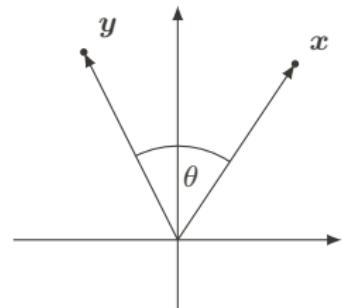


Dot product

A **dot product** (or **scalar product**) between two vectors:

$$\mathbf{x} \cdot \mathbf{y} = \mathbf{y} \cdot \mathbf{x} = x_1y_1 + x_2y_2 + \dots + x_ny_n$$

In particular: $\mathbf{x} \cdot \mathbf{x} = \|\mathbf{x}\|^2$

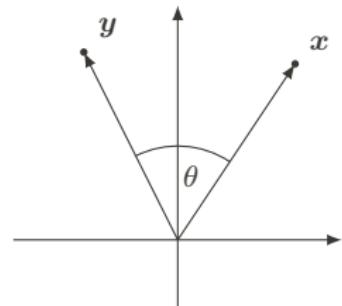


Dot product

A **dot product** (or **scalar product**) between two vectors:

$$\mathbf{x} \cdot \mathbf{y} = \mathbf{y} \cdot \mathbf{x} = x_1 y_1 + x_2 y_2 + \dots + x_n y_n$$

In particular: $\mathbf{x} \cdot \mathbf{x} = \|\mathbf{x}\|^2$



We have:

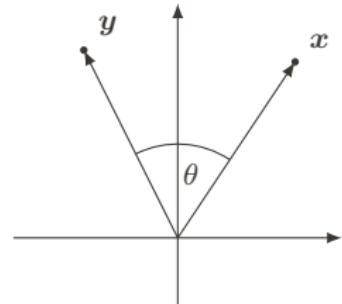
$$\cos \theta = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|} = \frac{\mathbf{x}}{\|\mathbf{x}\|} \cdot \frac{\mathbf{y}}{\|\mathbf{y}\|}$$

Dot product

A **dot product** (or **scalar product**) between two vectors:

$$\mathbf{x} \cdot \mathbf{y} = \mathbf{y} \cdot \mathbf{x} = x_1 y_1 + x_2 y_2 + \dots + x_n y_n$$

In particular: $\mathbf{x} \cdot \mathbf{x} = \|\mathbf{x}\|^2$



We have:

$$\cos \theta = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|} = \frac{\mathbf{x}}{\|\mathbf{x}\|} \cdot \frac{\mathbf{y}}{\|\mathbf{y}\|}$$

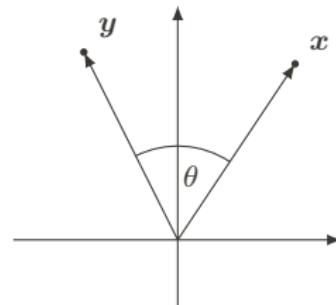
Vector, for which $\mathbf{x} \cdot \mathbf{y} = 0$, are called **orthogonal** (perpendicular)

Dot product

A **dot product** (or **scalar product**) between two vectors:

$$\mathbf{x} \cdot \mathbf{y} = \mathbf{y} \cdot \mathbf{x} = x_1 y_1 + x_2 y_2 + \dots + x_n y_n$$

In particular: $\mathbf{x} \cdot \mathbf{x} = \|\mathbf{x}\|^2$



We have:

$$\cos \theta = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|} = \frac{\mathbf{x}}{\|\mathbf{x}\|} \cdot \frac{\mathbf{y}}{\|\mathbf{y}\|}$$

Vector, for which $\mathbf{x} \cdot \mathbf{y} = 0$, are called **orthogonal** (perpendicular)

Cauchy-Schwarz inequality:

$$|\mathbf{x} \cdot \mathbf{y}| \leq \|\mathbf{x}\| \|\mathbf{y}\|$$

(interpretation: $|\cos \theta| \leq 1$)

Dot product

Exercise: for a given vector x , find a unit vector v , which (i) maximizes (ii) minimizes the dot product $x \cdot v$

Dot product

Exercise: for a given vector x , find a unit vector v , which (i) maximizes (ii) minimizes the dot product $x \cdot v$

Solution: from Cauchy-Schwarz inequality:

$$-\|x\| \underbrace{\|v\|}_{=1} \leq x \cdot v \leq \|x\| \underbrace{\|v\|}_{=1}$$

Dot product

Exercise: for a given vector x , find a unit vector v , which (i) maximizes (ii) minimizes the dot product $x \cdot v$

Solution: from Cauchy-Schwarz inequality:

$$-\|x\| \leq x \cdot v \leq \|x\|$$

Dot product

Exercise: for a given vector x , find a unit vector v , which (i) maximizes (ii) minimizes the dot product $x \cdot v$

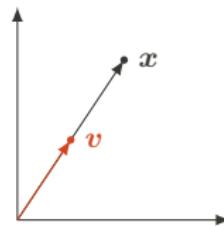
Solution: from Cauchy-Schwarz inequality:

$$-\|x\| \leq x \cdot v \leq \|x\|$$

Taking $v = \frac{x}{\|x\|}$ gives:

$$x \cdot v = \frac{x \cdot x}{\|x\|} = \|x\|$$

Vector $v = \frac{x}{\|x\|}$ **maximizes** $x \cdot v$



Dot product

Exercise: for a given vector x , find a unit vector v , which (i) maximizes (ii) minimizes the dot product $x \cdot v$

Solution: from Cauchy-Schwarz inequality:

$$-\|x\| \leq x \cdot v \leq \|x\|$$

Taking $v = \frac{x}{\|x\|}$ gives:

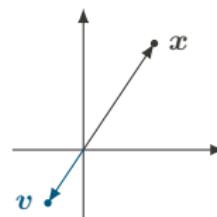
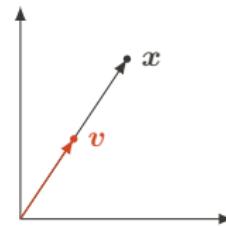
$$x \cdot v = \frac{x \cdot x}{\|x\|} = \|x\|$$

Vector $v = \frac{x}{\|x\|}$ **maximizes** $x \cdot v$

Likewise, taking $v = -\frac{x}{\|x\|}$ gives:

$$x \cdot v = -\|x\|$$

Vector $v = -\frac{x}{\|x\|}$ **minimizes** $x \cdot v$



Matrices

A **matrix of size $m \times n$** is a rectangular array (table) of $m \cdot n$ real numbers arranged in m rows and n columns:

$$\mathbf{A} = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{m1} & A_{m2} & \cdots & A_{mn} \end{bmatrix}$$

The space of all such matrices is denoted by $\mathbb{R}^{m \times n}$

Operations on matrices

- Scalar multiplication: for $A \in \mathbb{R}^{m \times n}$ and $\alpha \in \mathbb{R}$

$$B = \alpha A \in \mathbb{R}^{m \times n} \quad \text{is given by} \quad B_{ij} = \alpha A_{ij}$$

Operations on matrices

- Scalar multiplication: for $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\alpha \in \mathbb{R}$

$$\mathbf{B} = \alpha \mathbf{A} \in \mathbb{R}^{m \times n} \quad \text{is given by} \quad B_{ij} = \alpha A_{ij}$$

- Matrix addition: for $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{B} \in \mathbb{R}^{m \times n}$

$$\mathbf{C} = \mathbf{A} + \mathbf{B} \in \mathbb{R}^{m \times n} \quad \text{is given by} \quad C_{ij} = A_{ij} + B_{ij}$$

Operations on matrices

- Scalar multiplication: for $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\alpha \in \mathbb{R}$

$$\mathbf{B} = \alpha \mathbf{A} \in \mathbb{R}^{m \times n} \quad \text{is given by} \quad B_{ij} = \alpha A_{ij}$$

- Matrix addition: for $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{B} \in \mathbb{R}^{m \times n}$

$$\mathbf{C} = \mathbf{A} + \mathbf{B} \in \mathbb{R}^{m \times n} \quad \text{is given by} \quad C_{ij} = A_{ij} + B_{ij}$$

- Matrix multiplication: for $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{B} \in \mathbb{R}^{n \times p}$

$$\mathbf{C} = \mathbf{AB} \in \mathbb{R}^{m \times p} \quad \text{is given by} \quad C_{ij} = \sum_{k=1}^n A_{ik} B_{kj}$$

Note: matrix multiplication is not commutative!

Operations on matrices

- Scalar multiplication: for $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\alpha \in \mathbb{R}$

$$\mathbf{B} = \alpha \mathbf{A} \in \mathbb{R}^{m \times n} \quad \text{is given by} \quad B_{ij} = \alpha A_{ij}$$

- Matrix addition: for $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{B} \in \mathbb{R}^{m \times n}$

$$\mathbf{C} = \mathbf{A} + \mathbf{B} \in \mathbb{R}^{m \times n} \quad \text{is given by} \quad C_{ij} = A_{ij} + B_{ij}$$

- Matrix multiplication: for $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{B} \in \mathbb{R}^{n \times p}$

$$\mathbf{C} = \mathbf{AB} \in \mathbb{R}^{m \times p} \quad \text{is given by} \quad C_{ij} = \sum_{k=1}^n A_{ik} B_{kj}$$

Note: matrix multiplication is not commutative!

- Transposition: for $\mathbf{A} \in \mathbb{R}^{m \times n}$

$$\mathbf{B} = \mathbf{A}^\top \in \mathbb{R}^{n \times m} \quad \text{is given by} \quad B_{ij} = A_{ji}$$

It holds $(\mathbf{A} + \mathbf{B})^\top = \mathbf{A}^\top + \mathbf{B}^\top$ and $(\mathbf{AB})^\top = \mathbf{B}^\top \mathbf{A}^\top$

Square matrix

A matrix, for which $n = m$, is called a **square matrix of order n**

Square matrix

A matrix, for which $n = m$, is called a **square matrix of order n**

Diagonal matrix

Non-zero elements only on the **diagonal**

$$A = \begin{bmatrix} A_{11} & 0 & 0 & \cdots & 0 \\ 0 & A_{22} & 0 & \cdots & 0 \\ 0 & 0 & A_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & A_{nn} \end{bmatrix}$$

Square matrix

A matrix, for which $n = m$, is called a **square matrix of order n**

Diagonal matrix

Non-zero elements only on the **diagonal**

$$A = \begin{bmatrix} A_{11} & 0 & 0 & \cdots & 0 \\ 0 & A_{22} & 0 & \cdots & 0 \\ 0 & 0 & A_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & A_{nn} \end{bmatrix}$$

Identity matrix

It is an **identity element** with respect to matrix multiplication, i.e. for any

$$A \in \mathbb{R}^{n \times n}, AI = IA = A$$

$$I = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

Square matrix

A matrix, for which $n = m$, is called a **square matrix of order n**

Diagonal matrix

Non-zero elements only on the **diagonal**

$$A = \begin{bmatrix} A_{11} & 0 & 0 & \cdots & 0 \\ 0 & A_{22} & 0 & \cdots & 0 \\ 0 & 0 & A_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & A_{nn} \end{bmatrix}$$

Identity matrix

It is an **identity element** with respect to matrix multiplication, i.e. for any

$$A \in \mathbb{R}^{n \times n}, AI = IA = A$$

$$I = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

Symmetric matrix: $A^\top = A$, i.e., $A_{ij} = A_{ji}$.

Vectors as matrices

We will treat vectors $\mathbf{x} = (x_1, \dots, x_n)$ as $n \times 1$ matrices (**column vectors**), Then \mathbf{x}^\top of size $1 \times n$ is a **row vector**

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{x}^\top = [x_1 \ x_2 \ \cdots \ x_n]$$

Vectors as matrices

Multiplication of matrices and vectors: in the matrix representation, for any square matrix $A \in \mathbb{R}^{n \times n}$ and vectors $x, y \in \mathbb{R}^n$

Vectors as matrices

Multiplication of matrices and vectors: in the matrix representation, for any square matrix $A \in \mathbb{R}^{n \times n}$ and vectors $x, y \in \mathbb{R}^n$

- Ax is a column vector of size $n \times 1$

Vectors as matrices

Multiplication of matrices and vectors: in the matrix representation, for any square matrix $A \in \mathbb{R}^{n \times n}$ and vectors $x, y \in \mathbb{R}^n$

- Ax is a column vector of size $n \times 1$
- $x^\top A$ is a row vector of size $1 \times n$

Vectors as matrices

Multiplication of matrices and vectors: in the matrix representation, for any square matrix $A \in \mathbb{R}^{n \times n}$ and vectors $x, y \in \mathbb{R}^n$

- Ax is a column vector of size $n \times 1$
- $x^\top A$ is a row vector of size $1 \times n$
- Products Ax^\top i xA do not make sense

Vectors as matrices

Multiplication of matrices and vectors: in the matrix representation, for any square matrix $A \in \mathbb{R}^{n \times n}$ and vectors $x, y \in \mathbb{R}^n$

- Ax is a column vector of size $n \times 1$
- $x^\top A$ is a row vector of size $1 \times n$
- Products Ax^\top i xA do not make sense
- $x^\top Ay$ is a scalar (number)

Vectors as matrices

Multiplication of matrices and vectors: in the matrix representation, for any square matrix $A \in \mathbb{R}^{n \times n}$ and vectors $x, y \in \mathbb{R}^n$

- Ax is a column vector of size $n \times 1$
- $x^\top A$ is a row vector of size $1 \times n$
- Products Ax^\top i xA do not make sense
- $x^\top Ay$ is a scalar (number)
- $x^\top Ay = y^\top A^\top x$, since by transposition of product rule $(ABC)^\top = C^\top B^\top A^\top$ it holds:

$$x^\top Ay = \underbrace{(x^\top Ay)^\top}_{\text{because it's a number}} = y^\top A^\top (x^\top)^\top = y^\top A^\top x$$

Vectors as matrices

Multiplication of matrices and vectors: in the matrix representation, for any square matrix $A \in \mathbb{R}^{n \times n}$ and vectors $x, y \in \mathbb{R}^n$

- Ax is a column vector of size $n \times 1$
- $x^\top A$ is a row vector of size $1 \times n$
- Products Ax^\top i xA do not make sense
- $x^\top Ay$ is a scalar (number)
- $x^\top Ay = y^\top A^\top x$, since by transposition of product rule $(ABC)^\top = C^\top B^\top A^\top$ it holds:

$$x^\top Ay = \underbrace{(x^\top Ay)^\top}_{\text{because it's a number}} = y^\top A^\top (x^\top)^\top = y^\top A^\top x$$

- In particular, if A is symmetric: $x^\top Ay = y^\top Ax$

For a symmetric matrix A it holds

$$x^\top Ay = y^\top Ax$$

Dot product and outer product

For any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ in their matrix representation:

$$\mathbf{x}^\top \mathbf{y} = [x_1 \ x_2 \ \cdots \ x_n] \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = x_1 y_1 + \dots x_n y_n = \mathbf{x} \cdot \mathbf{y}$$

From now on, we denote the dot product as $\mathbf{x}^\top \mathbf{y}$ (or $\mathbf{y}^\top \mathbf{x}$).

In particular: $\mathbf{x}^\top \mathbf{x} = \|\mathbf{x}\|^2$

Dot product and outer product

For any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ in their matrix representation:

$$\mathbf{x}^\top \mathbf{y} = [x_1 \ x_2 \ \cdots \ x_n] \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = x_1 y_1 + \dots x_n y_n = \mathbf{x} \cdot \mathbf{y}$$

From now on, we denote the dot product as $\mathbf{x}^\top \mathbf{y}$ (or $\mathbf{y}^\top \mathbf{x}$).

In particular: $\mathbf{x}^\top \mathbf{x} = \|\mathbf{x}\|^2$

The product $\mathbf{x}\mathbf{y}^\top \in \mathbb{R}^{n \times n}$ is called an **outer product** of \mathbf{x} and \mathbf{y} :

$$\mathbf{x}\mathbf{y}^\top = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \begin{bmatrix} y_1 & y_2 & \cdots & y_n \end{bmatrix} = \begin{bmatrix} x_1 y_1 & x_1 y_2 & x_1 y_3 & \cdots & x_1 y_n \\ x_2 y_1 & x_2 y_2 & x_2 y_3 & \cdots & x_2 y_n \\ x_3 y_1 & x_3 y_2 & x_3 y_3 & \cdots & x_3 y_n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_n y_1 & x_n y_2 & x_n y_3 & \cdots & x_n y_n \end{bmatrix}$$

The matrix determinant

A **determinant** of a square matrix \mathbf{A} is a real-valued function $\det \mathbf{A}$ defined inductively as:

1. For $\mathbf{A} \in \mathbb{R}^{1 \times 1}$, $\det \mathbf{A} = A_{11}$
2. For $\mathbf{A} \in \mathbb{R}^{n \times n}$ with $n \geq 2$:

$$\det \mathbf{A} = \sum_{i=1}^n (-1)^{i+j} A_{ij} \det \mathbf{A}_{ij},$$

where i is any row, and $\mathbf{A}_{ij} \in \mathbb{R}^{(n-1) \times (n-1)}$ is a matrix obtained from \mathbf{A} by removing its i -th row and its j -th column

The matrix determinant

A **determinant** of a square matrix \mathbf{A} is a real-valued function $\det \mathbf{A}$ defined inductively as:

1. For $\mathbf{A} \in \mathbb{R}^{1 \times 1}$, $\det \mathbf{A} = A_{11}$
2. For $\mathbf{A} \in \mathbb{R}^{n \times n}$ with $n \geq 2$:

$$\det \mathbf{A} = \sum_{i=1}^n (-1)^{i+j} A_{ij} \det \mathbf{A}_{ij},$$

where i is any row, and $\mathbf{A}_{ij} \in \mathbb{R}^{(n-1) \times (n-1)}$ is a matrix obtained from \mathbf{A} by removing its i -th row and its j -th column

Example for matrix $\mathbf{A} \in \mathbb{R}^{2 \times 2}$:

$$\det \mathbf{A} = \det \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = A_{11}A_{22} - A_{12}A_{21}$$

Matrix inverse

Let A be a square matrix of order n . A **(matrix) inverse** of A , denoted by A^{-1} , is defined by:

$$AA^{-1} = A^{-1}A = I$$

Matrix inverse

Let A be a square matrix of order n . A **(matrix) inverse** of A , denoted by A^{-1} , is defined by:

$$AA^{-1} = A^{-1}A = I$$

Not every matrix has an inverse!

A matrix which has an inverse, is called **invertible**, while a matrix which does not have an inverse, is called **singular** (or **non-invertible**)

Matrix inverse

Let A be a square matrix of order n . A **(matrix) inverse** of A , denoted by A^{-1} , is defined by:

$$AA^{-1} = A^{-1}A = I$$

Not every matrix has an inverse!

A matrix which has an inverse, is called **invertible**, while a matrix which does not have an inverse, is called **singular** (or **non-invertible**)

Fact: a square matrix A is invertible if and only if $\det A \neq 0$.

Computing the matrix inverse

In general, the inverse can be obtained using a **Gaussian elimination method** in time $O(n^3)$

Consider two special cases:

Computing the matrix inverse

In general, the inverse can be obtained using a **Gaussian elimination method** in time $O(n^3)$

Consider two special cases:

- Diagonal matrix:

$$\mathbf{A} = \begin{bmatrix} A_{11} & 0 & \cdots & 0 \\ 0 & A_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A_{nn} \end{bmatrix}, \quad \mathbf{A}^{-1} = \begin{bmatrix} A_{11}^{-1} & 0 & \cdots & 0 \\ 0 & A_{22}^{-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A_{nn}^{-1} \end{bmatrix}$$

Computing the matrix inverse

In general, the inverse can be obtained using a **Gaussian elimination method** in time $O(n^3)$

Consider two special cases:

- Diagonal matrix:

$$\mathbf{A} = \begin{bmatrix} A_{11} & 0 & \cdots & 0 \\ 0 & A_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A_{nn} \end{bmatrix}, \quad \mathbf{A}^{-1} = \begin{bmatrix} A_{11}^{-1} & 0 & \cdots & 0 \\ 0 & A_{22}^{-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A_{nn}^{-1} \end{bmatrix}$$

- A 2×2 matrix:

$$\mathbf{A} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \quad \mathbf{A}^{-1} = \frac{1}{\det \mathbf{A}} \begin{bmatrix} A_{22} & -A_{12} \\ -A_{21} & A_{11} \end{bmatrix}$$

Eigenvalues and eigenvectors of a symmetric matrix

Let $A \in \mathbb{R}^{n \times n}$ be a symmetric matrix ($A^\top = A$).

If for some $\lambda \in \mathbb{R}$ and some unit vector $v \in \mathbb{R}^n$ it holds:

$$Av = \lambda v,$$

then λ and v are called, respectively, an **eigenvalue** and an associated **eigenvector** of A .

Eigenvalues and eigenvectors of a symmetric matrix

Let $A \in \mathbb{R}^{n \times n}$ be a symmetric matrix ($A^\top = A$).

If for some $\lambda \in \mathbb{R}$ and some unit vector $v \in \mathbb{R}^n$ it holds:

$$Av = \lambda v,$$

then λ and v are called, respectively, an **eigenvalue** and an associated **eigenvector** of A .

Fact: A symmetric matrix of order n has n eigenvalues $\lambda_1, \dots, \lambda_n$ (need not be distinct), while the associated eigenvectors v_1, \dots, v_n are **orthogonal**, i.e. $v_i^\top v_j = 0$ for any $i \neq j$.

Eigenvalues and eigenvectors of a symmetric matrix

Let $A \in \mathbb{R}^{n \times n}$ be a symmetric matrix ($A^\top = A$).

If for some $\lambda \in \mathbb{R}$ and some unit vector $v \in \mathbb{R}^n$ it holds:

$$Av = \lambda v,$$

then λ and v are called, respectively, an **eigenvalue** and an associated **eigenvector** of A .

Fact: A symmetric matrix of order n has n eigenvalues $\lambda_1, \dots, \lambda_n$ (need not be distinct), while the associated eigenvectors v_1, \dots, v_n are **orthogonal**, i.e. $v_i^\top v_j = 0$ for any $i \neq j$.

Fact: If a symmetric matrix A has eigenvectors v_1, \dots, v_n and eigenvalues $\lambda_1, \dots, \lambda_n$, then its inverse A^{-1} has the same eigenvectors and eigenvalues $\lambda_1^{-1}, \dots, \lambda_n^{-1}$

Eigenvalues and eigenvectors of a symmetric matrix

Let $A \in \mathbb{R}^{n \times n}$ be a symmetric matrix ($A^\top = A$).

If for some $\lambda \in \mathbb{R}$ and some unit vector $v \in \mathbb{R}^n$ it holds:

$$Av = \lambda v,$$

then λ and v are called, respectively, an **eigenvalue** and an associated **eigenvector** of A .

Fact: A symmetric matrix of order n has n eigenvalues $\lambda_1, \dots, \lambda_n$ (need not be distinct), while the associated eigenvectors v_1, \dots, v_n are **orthogonal**, i.e. $v_i^\top v_j = 0$ for any $i \neq j$.

Fact: If a symmetric matrix A has eigenvectors v_1, \dots, v_n and eigenvalues $\lambda_1, \dots, \lambda_n$, then its inverse A^{-1} has the same eigenvectors and eigenvalues $\lambda_1^{-1}, \dots, \lambda_n^{-1}$

Conclusion: A symmetric matrix is invertible if and only if all its eigenvalues are **non-zero**

Quadratic form

For a symmetric matrix $A \in \mathbb{R}^{n \times n}$, the following expression:

$$x^\top A x = \sum_{i=1}^n \sum_{j=1}^n A_{ij} x_i x_j$$

is called a **quadratic form** with respect to matrix A

Quadratic form

For a symmetric matrix $A \in \mathbb{R}^{n \times n}$, the following expression:

$$x^\top Ax = \sum_{i=1}^n \sum_{j=1}^n A_{ij}x_i x_j$$

is called a **quadratic form** with respect to matrix A

A matrix A is **positive-definite** if for any non-zero vector $v \in \mathbb{R}^n$, $v^\top Av > 0$, and **positive semi-definite**, if $v^\top Av \geq 0$.

Quadratic form

For a symmetric matrix $A \in \mathbb{R}^{n \times n}$, the following expression:

$$x^\top Ax = \sum_{i=1}^n \sum_{j=1}^n A_{ij} x_i x_j$$

is called a **quadratic form** with respect to matrix A

A matrix A is **positive-definite** if for any non-zero vector $v \in \mathbb{R}^n$, $v^\top Av > 0$, and **positive semi-definite**, if $v^\top Av \geq 0$.

Fact: If λ_{\min} and λ_{\max} are, respectively, the smallest and the largest eigenvalues of A , then for any vector v :

$$\lambda_{\min} \|v\|^2 \leq v^\top Av \leq \lambda_{\max} \|v\|^2$$

Quadratic form

For a symmetric matrix $A \in \mathbb{R}^{n \times n}$, the following expression:

$$x^\top Ax = \sum_{i=1}^n \sum_{j=1}^n A_{ij} x_i x_j$$

is called a **quadratic form** with respect to matrix A

A matrix A is **positive-definite** if for any non-zero vector $v \in \mathbb{R}^n$, $v^\top Av > 0$, and **positive semi-definite**, if $v^\top Av \geq 0$.

Fact: If λ_{\min} and λ_{\max} are, respectively, the smallest and the largest eigenvalues of A , then for any vector v :

$$\lambda_{\min} \|v\|^2 \leq v^\top Av \leq \lambda_{\max} \|v\|^2$$

Fact: A matrix A is positive-definite (positive semi-definite), if all its eigenvalues are positive (non-negative)

Positive-definite matrices

Testing whether A is positive-definite is complicated: requires e.g. to compute the smallest eigenvalue

Positive-definite matrices

Testing whether A is positive-definite is complicated: requires e.g. to compute the smallest eigenvalue

For 2×2 matrices there is a much simpler test: a 2×2 matrix is positive-definite if and only if $\det A > 0$ i $A_{11} > 0$

Positive-definite matrices

Testing whether A is positive-definite is complicated: requires e.g. to compute the smallest eigenvalue

For 2×2 matrices there is a much simpler test: a 2×2 matrix is positive-definite if and only if $\det A > 0$ i $A_{11} > 0$

Example: Consider a matrix:

$$A = \begin{bmatrix} 3 & -1 \\ -1 & 2 \end{bmatrix}$$

Positive-definite matrices

Testing whether A is positive-definite is complicated: requires e.g. to compute the smallest eigenvalue

For 2×2 matrices there is a much simpler test: a 2×2 matrix is positive-definite if and only if $\det A > 0$ i $A_{11} > 0$

Example: Consider a matrix:

$$A = \begin{bmatrix} 3 & -1 \\ -1 & 2 \end{bmatrix} \quad \det A = 5, \quad A_{11} = 3$$

Conclusion: A is positive-definite

(indeed, the eigenvalues of A are $\lambda_1 = 3.62, \lambda_2 = 1.38$)

Positive-definite matrices

Testing whether A is positive-definite is complicated: requires e.g. to compute the smallest eigenvalue

For 2×2 matrices there is a much simpler test: a 2×2 matrix is positive-definite if and only if $\det A > 0$ i $A_{11} > 0$

Example: Consider a matrix:

$$A = \begin{bmatrix} 3 & -1 \\ -1 & 2 \end{bmatrix} \quad \det A = 5, \quad A_{11} = 3$$

Conclusion: A is positive-definite

(indeed, the eigenvalues of A are $\lambda_1 = 3.62, \lambda_2 = 1.38$)

Example: Consider a matrix:

$$A = \begin{bmatrix} 1 & -3 \\ -3 & 4 \end{bmatrix}$$

Positive-definite matrices

Testing whether A is positive-definite is complicated: requires e.g. to compute the smallest eigenvalue

For 2×2 matrices there is a much simpler test: a 2×2 matrix is positive-definite if and only if $\det A > 0$ i $A_{11} > 0$

Example: Consider a matrix:

$$A = \begin{bmatrix} 3 & -1 \\ -1 & 2 \end{bmatrix} \quad \det A = 5, \quad A_{11} = 3$$

Conclusion: A is positive-definite

(indeed, the eigenvalues of A are $\lambda_1 = 3.62, \lambda_2 = 1.38$)

Example: Consider a matrix:

$$A = \begin{bmatrix} 1 & -3 \\ -3 & 4 \end{bmatrix} \quad \det A = -5, \quad A_{11} = 1$$

Conclusion: A is not positive-definite

(indeed, its eigenvalues are $\lambda_1 = 5.85, \lambda_2 = -0.85$)

Partial derivatives, gradient, Hessian

Partial derivative

Univariate function. Derivative of function $f: \mathbb{R} \rightarrow \mathbb{R}$ at point x :

$$f'(x) = \frac{df(x)}{dx} = \lim_{\Delta \rightarrow 0} \frac{f(x + \Delta) - f(x)}{\Delta}$$

Partial derivative

Univariate function. Derivative of function $f: \mathbb{R} \rightarrow \mathbb{R}$ at point x :

$$f'(x) = \frac{df(x)}{dx} = \lim_{\Delta \rightarrow 0} \frac{f(x + \Delta) - f(x)}{\Delta}$$

Multivariate function. Partial derivative of $f: \mathbb{R}^n \rightarrow \mathbb{R}$ at \mathbf{x} with respect to variable x_i :

$$\begin{aligned}\frac{\partial f(\mathbf{x})}{\partial x_i} &= \lim_{\Delta \rightarrow 0} \frac{f(x_1, x_2, \dots, x_i + \Delta, \dots, x_n) - f(\mathbf{x})}{\Delta} \\ &= \lim_{\Delta \rightarrow 0} \frac{f(\mathbf{x} + \Delta \mathbf{e}_i) - f(\mathbf{x})}{\Delta}\end{aligned}$$

All variables except x_i are treated as constants

Partial derivative

Univariate function. Derivative of function $f: \mathbb{R} \rightarrow \mathbb{R}$ at point x :

$$f'(x) = \frac{df(x)}{dx} = \lim_{\Delta \rightarrow 0} \frac{f(x + \Delta) - f(x)}{\Delta}$$

Multivariate function. Partial derivative of $f: \mathbb{R}^n \rightarrow \mathbb{R}$ at \mathbf{x} with respect to variable x_i :

$$\begin{aligned}\frac{\partial f(\mathbf{x})}{\partial x_i} &= \lim_{\Delta \rightarrow 0} \frac{f(x_1, x_2, \dots, x_i + \Delta, \dots, x_n) - f(\mathbf{x})}{\Delta} \\ &= \lim_{\Delta \rightarrow 0} \frac{f(\mathbf{x} + \Delta \mathbf{e}_i) - f(\mathbf{x})}{\Delta}\end{aligned}$$

All variables except x_i are treated as constants

Example: for $f(x_1, x_2) = x_1^2 e^{x_2} + x_2$

$$\frac{\partial f(x_1, x_2)}{\partial x_1} = 2x_1 e^{x_2}, \quad \frac{\partial f(x_1, x_2)}{\partial x_2} = x_1^2 e^{x_2} + 1$$

Gradient

The **gradient** of a function $f(\mathbf{x})$ at \mathbf{x} is a vector of partial derivatives:

$$\nabla f(\mathbf{x}) = \left(\frac{\partial f(\mathbf{x})}{\partial x_1}, \dots, \frac{\partial f(\mathbf{x})}{\partial x_n} \right)$$

Gradient – examples

Exercise: calculate the gradient of a linear function

$$f(\mathbf{x}) = c + \mathbf{b}^\top \mathbf{x} = c + \sum_{i=1}^n b_i x_i$$

Gradient – examples

Exercise: calculate the gradient of a linear function

$$f(\mathbf{x}) = c + \mathbf{b}^\top \mathbf{x} = c + \sum_{i=1}^n b_i x_i$$

Solution: using the linearity of the derivative:

$$\frac{\partial f(\mathbf{x})}{\partial x_k} = \frac{\partial}{\partial x_k} \left(c + \sum_{i=1}^n b_i x_i \right) = \underbrace{\frac{\partial c}{\partial x_k}}_0 + \sum_{i=1}^n b_i \underbrace{\frac{\partial x_i}{\partial x_k}}_{\delta_{ik}} = b_k,$$

where δ_{ik} is the so called **Kronecker's delta**: $\delta_{ik} = \begin{cases} 0 & i \neq k \\ 1 & i = k \end{cases}$

Gradient – examples

Exercise: calculate the gradient of a linear function

$$f(\mathbf{x}) = c + \mathbf{b}^\top \mathbf{x} = c + \sum_{i=1}^n b_i x_i$$

Solution: using the linearity of the derivative:

$$\frac{\partial f(\mathbf{x})}{\partial x_k} = \frac{\partial}{\partial x_k} \left(c + \sum_{i=1}^n b_i x_i \right) = \underbrace{\frac{\partial c}{\partial x_k}}_0 + \sum_{i=1}^n b_i \underbrace{\frac{\partial x_i}{\partial x_k}}_{\delta_{ik}} = b_k,$$

where δ_{ik} is the so called **Kronecker's delta**: $\delta_{ik} = \begin{cases} 0 & i \neq k \\ 1 & i = k \end{cases}$

Therefore $\nabla f(\mathbf{x}) = \mathbf{b}$

Gradient – examples

Exercise: calculate the gradient of a linear function

$$f(\mathbf{x}) = c + \mathbf{b}^\top \mathbf{x} = c + \sum_{i=1}^n b_i x_i$$

Solution: using the linearity of the derivative:

$$\frac{\partial f(\mathbf{x})}{\partial x_k} = \frac{\partial}{\partial x_k} \left(c + \sum_{i=1}^n b_i x_i \right) = \underbrace{\frac{\partial c}{\partial x_k}}_0 + \sum_{i=1}^n b_i \underbrace{\frac{\partial x_i}{\partial x_k}}_{\delta_{ik}} = b_k,$$

where δ_{ik} is the so called **Kronecker's delta**: $\delta_{ik} = \begin{cases} 0 & i \neq k \\ 1 & i = k \end{cases}$

Therefore $\nabla f(\mathbf{x}) = \mathbf{b}$

$$\nabla(c + \mathbf{b}^\top \mathbf{x}) = \mathbf{b}$$

Gradient – examples

Exercise: calculate the gradient of a quadratic form with symmetric matrix \mathbf{A} :

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x} = \sum_{i=1}^n \sum_{j=1}^n A_{ij} x_i x_j$$

Gradient – examples

Exercise: calculate the gradient of a quadratic form with symmetric matrix \mathbf{A} :

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x} = \sum_{i=1}^n \sum_{j=1}^n A_{ij} x_i x_j$$

Solution: using the linearity of the derivative:

$$\frac{\partial f(\mathbf{x})}{\partial x_k} = \sum_{i=1}^n \sum_{j=1}^n A_{ij} \frac{\partial(x_i x_j)}{\partial x_k} = \sum_{i=1}^n \sum_{j=1}^n A_{ij} \left(\underbrace{\frac{\partial x_i}{\partial x_k}}_{\delta_{ik}} x_j + x_i \underbrace{\frac{\partial x_j}{\partial x_k}}_{\delta_{jk}} \right)$$

Gradient – examples

Exercise: calculate the gradient of a quadratic form with symmetric matrix \mathbf{A} :

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x} = \sum_{i=1}^n \sum_{j=1}^n A_{ij} x_i x_j$$

Solution: using the linearity of the derivative:

$$\begin{aligned} \frac{\partial f(\mathbf{x})}{\partial x_k} &= \sum_{i=1}^n \sum_{j=1}^n A_{ij} \frac{\partial(x_i x_j)}{\partial x_k} = \sum_{i=1}^n \sum_{j=1}^n A_{ij} \left(\underbrace{\frac{\partial x_i}{\partial x_k} x_j}_{\delta_{ik}} + x_i \underbrace{\frac{\partial x_j}{\partial x_k}}_{\delta_{jk}} \right) \\ &= \sum_{j=1}^n A_{kj} x_j + \sum_{i=1}^n \underbrace{A_{ik}}_{A_{ki}} x_i = 2 \sum_{j=1}^n A_{kj} x_j \end{aligned}$$

Gradient – examples

Exercise: calculate the gradient of a quadratic form with symmetric matrix \mathbf{A} :

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x} = \sum_{i=1}^n \sum_{j=1}^n A_{ij} x_i x_j$$

Solution: using the linearity of the derivative:

$$\begin{aligned}\frac{\partial f(\mathbf{x})}{\partial x_k} &= \sum_{i=1}^n \sum_{j=1}^n A_{ij} \frac{\partial(x_i x_j)}{\partial x_k} = \sum_{i=1}^n \sum_{j=1}^n A_{ij} \left(\underbrace{\frac{\partial x_i}{\partial x_k} x_j}_{\delta_{ik}} + x_i \underbrace{\frac{\partial x_j}{\partial x_k}}_{\delta_{jk}} \right) \\ &= \sum_{j=1}^n A_{kj} x_j + \sum_{i=1}^n \underbrace{A_{ik}}_{A_{ki}} x_i = \underbrace{2 \sum_{j=1}^n A_{kj} x_j}_{\text{matrix} \times \text{vector}}\end{aligned}$$

Gradient – examples

Exercise: calculate the gradient of a quadratic form with symmetric matrix \mathbf{A} :

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x} = \sum_{i=1}^n \sum_{j=1}^n A_{ij} x_i x_j$$

Solution: using the linearity of the derivative:

$$\begin{aligned}\frac{\partial f(\mathbf{x})}{\partial x_k} &= \sum_{i=1}^n \sum_{j=1}^n A_{ij} \frac{\partial(x_i x_j)}{\partial x_k} = \sum_{i=1}^n \sum_{j=1}^n A_{ij} \left(\underbrace{\frac{\partial x_i}{\partial x_k} x_j}_{\delta_{ik}} + x_i \underbrace{\frac{\partial x_j}{\partial x_k}}_{\delta_{jk}} \right) \\ &= \sum_{j=1}^n A_{kj} x_j + \sum_{i=1}^n \underbrace{A_{ik}}_{A_{ki}} x_i = \underbrace{2 \sum_{j=1}^n A_{kj} x_j}_{\text{matrix} \times \text{vector}}\end{aligned}$$

Therefore: $\nabla f(\mathbf{x}) = 2\mathbf{A}\mathbf{x}$

Gradient – examples

Exercise: calculate the gradient of a quadratic form with symmetric matrix \mathbf{A} :

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x} = \sum_{i=1}^n \sum_{j=1}^n A_{ij} x_i x_j$$

Solution: using the linearity of the derivative:

$$\begin{aligned}\frac{\partial f(\mathbf{x})}{\partial x_k} &= \sum_{i=1}^n \sum_{j=1}^n A_{ij} \frac{\partial(x_i x_j)}{\partial x_k} = \sum_{i=1}^n \sum_{j=1}^n A_{ij} \left(\underbrace{\frac{\partial x_i}{\partial x_k} x_j}_{\delta_{ik}} + x_i \underbrace{\frac{\partial x_j}{\partial x_k}}_{\delta_{jk}} \right) \\ &= \sum_{j=1}^n A_{kj} x_j + \sum_{i=1}^n \underbrace{A_{ik}}_{A_{ki}} x_i = \underbrace{2 \sum_{j=1}^n A_{kj} x_j}_{\text{matrix} \times \text{vector}}\end{aligned}$$

Therefore: $\nabla f(\mathbf{x}) = 2\mathbf{A}\mathbf{x}$

$$\boxed{\nabla (\mathbf{x}^\top \mathbf{A} \mathbf{x}) = 2\mathbf{A}\mathbf{x}}$$

The chain rule

Composition of two univariate functions

The derivative of a composite function $g(x) = f(y(x))$ for $x \in \mathbb{R}$

$$g'(x) = f'(y(x)) \cdot y'(x),$$

or in Leibniz's notation:

$$\frac{dg(x)}{dx} = \frac{df(y)}{dy} \cdot \frac{dy(x)}{dx}$$

The chain rule

Composition of two univariate functions

The derivative of a composite function $g(x) = f(y(x))$ for $x \in \mathbb{R}$

$$g'(x) = f'(y(x)) \cdot y'(x),$$

or in Leibniz's notation:

$$\frac{dg(x)}{dx} = \frac{df(y)}{dy} \cdot \frac{dy(x)}{dx}$$

Composition of a multivariate function with a univariate function

The derivative of a composite function $g(x) = f(\mathbf{y}(x))$ for $x \in \mathbb{R}$,
where $\mathbf{y}(x) = (y_1(x), \dots, y_n(x))$

$$\frac{dg(x)}{dx} = \sum_{i=1}^n \frac{\partial f(\mathbf{y})}{\partial y_i} \cdot \frac{dy_i(x)}{dx}$$

Chain rule – example

Consider a function $g(\alpha)$ for $\alpha \in \mathbb{R}$ of the form:

$$g(\alpha) = f(\mathbf{x} + \alpha \mathbf{v})$$

with two fixed vectors $\mathbf{x}, \mathbf{v} \in \mathbb{R}^n$

Chain rule – example

Consider a function $g(\alpha)$ for $\alpha \in \mathbb{R}$ of the form:

$$g(\alpha) = f(\mathbf{x} + \alpha \mathbf{v})$$

with two fixed vectors $\mathbf{x}, \mathbf{v} \in \mathbb{R}^n$

Define $\mathbf{y}(\alpha) = \mathbf{x} + \alpha \mathbf{v}$, therefore $g(\alpha) = f(\mathbf{y}(\alpha))$

Chain rule – example

Consider a function $g(\alpha)$ for $\alpha \in \mathbb{R}$ of the form:

$$g(\alpha) = f(\mathbf{x} + \alpha \mathbf{v})$$

with two fixed vectors $\mathbf{x}, \mathbf{v} \in \mathbb{R}^n$

Define $\mathbf{y}(\alpha) = \mathbf{x} + \alpha \mathbf{v}$, therefore $g(\alpha) = f(\mathbf{y}(\alpha))$

Using the chain rule:

$$\frac{dg(\alpha)}{d\alpha} = \sum_{i=1}^n \frac{\partial f(\mathbf{y})}{\partial y_i} \cdot \frac{dy_i(\alpha)}{d\alpha}$$

Chain rule – example

Consider a function $g(\alpha)$ for $\alpha \in \mathbb{R}$ of the form:

$$g(\alpha) = f(\mathbf{x} + \alpha \mathbf{v})$$

with two fixed vectors $\mathbf{x}, \mathbf{v} \in \mathbb{R}^n$

Define $\mathbf{y}(\alpha) = \mathbf{x} + \alpha \mathbf{v}$, therefore $g(\alpha) = f(\mathbf{y}(\alpha))$

Using the chain rule:

$$\frac{dg(\alpha)}{d\alpha} = \sum_{i=1}^n \frac{\partial f(\mathbf{y})}{\partial y_i} \cdot \underbrace{\frac{dy_i(\alpha)}{d\alpha}}_{x_i + \alpha v_i}$$

Chain rule – example

Consider a function $g(\alpha)$ for $\alpha \in \mathbb{R}$ of the form:

$$g(\alpha) = f(\mathbf{x} + \alpha \mathbf{v})$$

with two fixed vectors $\mathbf{x}, \mathbf{v} \in \mathbb{R}^n$

Define $\mathbf{y}(\alpha) = \mathbf{x} + \alpha \mathbf{v}$, therefore $g(\alpha) = f(\mathbf{y}(\alpha))$

Using the chain rule:

$$\begin{aligned}\frac{dg(\alpha)}{d\alpha} &= \sum_{i=1}^n \frac{\partial f(\mathbf{y})}{\partial y_i} \cdot \underbrace{\frac{dy_i(\alpha)}{d\alpha}}_{x_i + \alpha v_i} \\ &= \sum_{i=1}^n \frac{\partial f(\mathbf{y})}{\partial y_i} \cdot v_i = \nabla f(\mathbf{y})^\top \mathbf{v}\end{aligned}$$

Chain rule – example

Consider a function $g(\alpha)$ for $\alpha \in \mathbb{R}$ of the form:

$$g(\alpha) = f(\mathbf{x} + \alpha \mathbf{v})$$

with two fixed vectors $\mathbf{x}, \mathbf{v} \in \mathbb{R}^n$

Define $\mathbf{y}(\alpha) = \mathbf{x} + \alpha \mathbf{v}$, therefore $g(\alpha) = f(\mathbf{y}(\alpha))$

Using the chain rule:

$$\begin{aligned}\frac{dg(\alpha)}{d\alpha} &= \sum_{i=1}^n \frac{\partial f(\mathbf{y})}{\partial y_i} \cdot \underbrace{\frac{dy_i(\alpha)}{d\alpha}}_{x_i + \alpha v_i} \\ &= \sum_{i=1}^n \frac{\partial f(\mathbf{y})}{\partial y_i} \cdot v_i = \nabla f(\mathbf{y})^\top \mathbf{v}\end{aligned}$$

$$\boxed{\frac{df(\mathbf{x} + \alpha \mathbf{v})}{d\alpha} = \nabla f(\mathbf{x} + \alpha \mathbf{v})^\top \mathbf{v}}$$

The directional derivative

A **directional derivative** of function $f(x)$ along vector v at point x is the following limit:

$$\frac{\partial f(x)}{\partial v} = \lim_{\Delta \rightarrow 0} \frac{f(x + \Delta v) - f(x)}{\Delta}$$

Remark: note that $\frac{\partial f(x)}{\partial x_i}$ is a directional derivative along vector e_i

The directional derivative

A **directional derivative** of function $f(\mathbf{x})$ along vector \mathbf{v} at point \mathbf{x} is the following limit:

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{v}} = \lim_{\Delta \rightarrow 0} \frac{f(\mathbf{x} + \Delta \mathbf{v}) - f(\mathbf{x})}{\Delta}$$

Remark: note that $\frac{\partial f(\mathbf{x})}{\partial x_i}$ is a directional derivative along vector e_i

Defining a function $g(\Delta) = f(\mathbf{x} + \Delta \mathbf{v})$, we get:

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{v}} = \lim_{\Delta \rightarrow 0} \frac{g(\Delta) - g(0)}{\Delta} = g'(0)$$

The directional derivative

A **directional derivative** of function $f(\mathbf{x})$ along vector \mathbf{v} at point \mathbf{x} is the following limit:

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{v}} = \lim_{\Delta \rightarrow 0} \frac{f(\mathbf{x} + \Delta \mathbf{v}) - f(\mathbf{x})}{\Delta}$$

Remark: note that $\frac{\partial f(\mathbf{x})}{\partial x_i}$ is a directional derivative along vector e_i

Defining a function $g(\Delta) = f(\mathbf{x} + \Delta \mathbf{v})$, we get:

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{v}} = \lim_{\Delta \rightarrow 0} \frac{g(\Delta) - g(0)}{\Delta} = g'(0)$$

Using a rule from the previous slide (with Δ playing the role of α):

$$g'(0) = \left. \frac{df(\mathbf{x} + \Delta \mathbf{v})}{d\Delta} \right|_{\Delta=0} = \left. \nabla f(\mathbf{x} + \Delta \mathbf{v})^\top \mathbf{v} \right|_{\Delta=0} = \nabla f(\mathbf{x})^\top \mathbf{v}$$

The directional derivative

A **directional derivative** of function $f(\mathbf{x})$ along vector \mathbf{v} at point \mathbf{x} is the following limit:

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{v}} = \lim_{\Delta \rightarrow 0} \frac{f(\mathbf{x} + \Delta \mathbf{v}) - f(\mathbf{x})}{\Delta}$$

Remark: note that $\frac{\partial f(\mathbf{x})}{\partial x_i}$ is a directional derivative along vector e_i

Defining a function $g(\Delta) = f(\mathbf{x} + \Delta \mathbf{v})$, we get:

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{v}} = \lim_{\Delta \rightarrow 0} \frac{g(\Delta) - g(0)}{\Delta} = g'(0)$$

Using a rule from the previous slide (with Δ playing the role of α):

$$g'(0) = \left. \frac{df(\mathbf{x} + \Delta \mathbf{v})}{d\Delta} \right|_{\Delta=0} = \left. \nabla f(\mathbf{x} + \Delta \mathbf{v})^\top \mathbf{v} \right|_{\Delta=0} = \nabla f(\mathbf{x})^\top \mathbf{v}$$

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{v}} = \nabla f(\mathbf{x})^\top \mathbf{v}$$

Higher-order derivatives

$$\frac{\partial^2 f(\mathbf{x})}{\partial x_i^2} = \frac{\partial}{\partial x_i} \left(\frac{\partial f(\mathbf{x})}{\partial x_i} \right), \quad \underbrace{\frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j} = \frac{\partial}{\partial x_i} \left(\frac{\partial f(\mathbf{x})}{\partial x_j} \right)}_{\text{mixed derivatives}},$$

Higher-order derivatives

$$\frac{\partial^2 f(\mathbf{x})}{\partial x_i^2} = \frac{\partial}{\partial x_i} \left(\frac{\partial f(\mathbf{x})}{\partial x_i} \right), \quad \underbrace{\frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j} = \frac{\partial}{\partial x_i} \left(\frac{\partial f(\mathbf{x})}{\partial x_j} \right)}_{\text{mixed derivatives}},$$

Example: for $f(x_1, x_2) = x_1^2 e^{x_2} + x_2$

$$\frac{\partial f}{\partial x_1} = 2x_1 e^{x_2}, \quad \frac{\partial f}{\partial x_2} = x_1^2 e^{x_2} + 1$$

$$\frac{\partial^2 f}{\partial x_1^2} = 2e^{x_2}, \quad \frac{\partial^2 f}{\partial x_2 \partial x_1} = 2x_1 e^{x_2}, \quad \frac{\partial^2 f}{\partial x_2^2} = x_1^2 e^{x_2}, \quad \frac{\partial^2 f}{\partial x_1 \partial x_2} = 2x_1 e^{x_2}$$

Higher-order derivatives

$$\frac{\partial^2 f(\mathbf{x})}{\partial x_i^2} = \frac{\partial}{\partial x_i} \left(\frac{\partial f(\mathbf{x})}{\partial x_i} \right), \quad \underbrace{\frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j} = \frac{\partial}{\partial x_i} \left(\frac{\partial f(\mathbf{x})}{\partial x_j} \right)}_{\text{mixed derivatives}},$$

Example: for $f(x_1, x_2) = x_1^2 e^{x_2} + x_2$

$$\frac{\partial f}{\partial x_1} = 2x_1 e^{x_2}, \quad \frac{\partial f}{\partial x_2} = x_1^2 e^{x_2} + 1$$

$$\frac{\partial^2 f}{\partial x_1^2} = 2e^{x_2}, \quad \frac{\partial^2 f}{\partial x_2 \partial x_1} = 2x_1 e^{x_2}, \quad \frac{\partial^2 f}{\partial x_2^2} = x_1^2 e^{x_2}, \quad \frac{\partial^2 f}{\partial x_1 \partial x_2} = 2x_1 e^{x_2}$$

Fact: If f has all mixed second-order derivatives continuous then

$$\frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j} = \frac{\partial^2 f(\mathbf{x})}{\partial x_j \partial x_i}$$

Hessian

The **Hessian** of a function $f(\mathbf{x})$ at point \mathbf{x} is a matrix of second-order partial derivatives:

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

The Hessian is a **symmetric** matrix

Hessian

The **Hessian** of a function $f(\mathbf{x})$ at point \mathbf{x} is a matrix of second-order partial derivatives:

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

The Hessian is a **symmetric** matrix

Example: for $f(x_1, x_2) = x_1^2 e^{x_2} + x_2$

$$\frac{\partial^2 f}{\partial x_1^2} = 2e^{x_2}, \quad \frac{\partial^2 f}{\partial x_2 \partial x_1} = 2x_1 e^{x_2}, \quad \frac{\partial^2 f}{\partial x_2^2} = x_1^2 e^{x_2}, \quad \frac{\partial^2 f}{\partial x_1 \partial x_2} = 2x_1 e^{x_2}$$

$$\text{Therefore: } \nabla^2 f(\mathbf{x}) = \begin{bmatrix} 2e^{x_2} & 2x_1 e^{x_2} \\ 2x_1 e^{x_2} & x_1^2 e^{x_2} \end{bmatrix}$$

Hessian – example

Calculate the Hessian of a linear function $f(\boldsymbol{x}) = \boldsymbol{c} + \boldsymbol{b}^\top \boldsymbol{x}$

Hessian – example

Calculate the Hessian of a linear function $f(\mathbf{x}) = c + \mathbf{b}^\top \mathbf{x}$

Solution:

$$\frac{\partial f(\mathbf{x})}{\partial x_i} = b_i, \quad \frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j} = 0$$

hence $\nabla^2 f(\mathbf{x}) = \mathbf{0}$ is a **zero matrix**

Hessian – example

Calculate the Hessian of a quadratic form $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x}$ for symmetric matrix \mathbf{A}

Hessian – example

Calculate the Hessian of a quadratic form $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x}$ for symmetric matrix \mathbf{A}

Solution: we already calculated:

$$\frac{\partial f(\mathbf{x})}{\partial x_i} = 2 \sum_{j=1}^n A_{ij} x_j,$$

hence:

$$\frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j} = 2A_{ij}$$

Hessian – example

Calculate the Hessian of a quadratic form $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x}$ for symmetric matrix \mathbf{A}

Solution: we already calculated:

$$\frac{\partial f(\mathbf{x})}{\partial x_i} = 2 \sum_{j=1}^n A_{ij} x_j,$$

hence:

$$\frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j} = 2A_{ij}$$

$$\nabla^2 (\mathbf{x}^\top \mathbf{A} \mathbf{x}) = 2\mathbf{A}$$

Hessian – example

Calculate the Hessian of a quadratic form $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x}$ for symmetric matrix \mathbf{A}

Solution: we already calculated:

$$\frac{\partial f(\mathbf{x})}{\partial x_i} = 2 \sum_{j=1}^n A_{ij} x_j,$$

hence:

$$\frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j} = 2A_{ij}$$

$$\boxed{\nabla^2 (\mathbf{x}^\top \mathbf{A} \mathbf{x}) = 2\mathbf{A}}$$

Conclusion: The Hessian of a general quadratic function:

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x} + \mathbf{b}^\top \mathbf{x} + c, \quad \nabla^2 f(\mathbf{x}) = 2\mathbf{A}$$

Hessian – example

Calculate the second derivative of a composite function $g(\alpha) = f(x + \alpha v)$

Hessian – example

Calculate the second derivative of a composite function $g(\alpha) = f(\mathbf{x} + \alpha \mathbf{v})$

We already showed that $g'(\alpha) = \nabla f(\mathbf{x} + \alpha \mathbf{v})^\top \mathbf{v}$

Hessian – example

Calculate the second derivative of a composite function $g(\alpha) = f(\mathbf{x} + \alpha \mathbf{v})$

We already showed that $g'(\alpha) = \nabla f(\mathbf{x} + \alpha \mathbf{v})^\top \mathbf{v}$

We can treat $g'(\alpha)$ as a composite function $h(\mathbf{y}(\alpha))$, where

$$\mathbf{y}(\alpha) = \mathbf{x} + \alpha \mathbf{v}, \quad h(\mathbf{y}) = \nabla f(\mathbf{y})^\top \mathbf{v} = \sum_{i=1}^n \frac{\partial f(\mathbf{y})}{\partial y_i} \cdot v_i$$

Hessian – example

Calculate the second derivative of a composite function $g(\alpha) = f(\mathbf{x} + \alpha \mathbf{v})$

We already showed that $g'(\alpha) = \nabla f(\mathbf{x} + \alpha \mathbf{v})^\top \mathbf{v}$

We can treat $g'(\alpha)$ as a composite function $h(\mathbf{y}(\alpha))$, where

$$\mathbf{y}(\alpha) = \mathbf{x} + \alpha \mathbf{v}, \quad h(\mathbf{y}) = \nabla f(\mathbf{y})^\top \mathbf{v} = \sum_{i=1}^n \frac{\partial f(\mathbf{y})}{\partial y_i} \cdot v_i$$

From the chain rule:

$$g''(\alpha) = \frac{dg'(\alpha)}{d\alpha} = \sum_{i=1}^n \frac{\partial h(\mathbf{y})}{\partial y_i} \cdot \overbrace{\frac{d y_i}{d\alpha}}^{x_i + \alpha v_i}$$

Hessian – example

Calculate the second derivative of a composite function $g(\alpha) = f(\mathbf{x} + \alpha \mathbf{v})$

We already showed that $g'(\alpha) = \nabla f(\mathbf{x} + \alpha \mathbf{v})^\top \mathbf{v}$

We can treat $g'(\alpha)$ as a composite function $h(\mathbf{y}(\alpha))$, where

$$\mathbf{y}(\alpha) = \mathbf{x} + \alpha \mathbf{v}, \quad h(\mathbf{y}) = \nabla f(\mathbf{y})^\top \mathbf{v} = \sum_{i=1}^n \frac{\partial f(\mathbf{y})}{\partial y_i} \cdot v_i$$

From the chain rule:

$$g''(\alpha) = \frac{dg'(\alpha)}{d\alpha} = \sum_{i=1}^n \frac{\partial h(\mathbf{y})}{\partial y_i} \cdot \overbrace{\frac{d y_i}{d\alpha}}^{x_i + \alpha v_i} = \sum_{i=1}^n \frac{\partial}{\partial y_i} \left(\sum_{j=1}^n \frac{\partial f(\mathbf{y})}{\partial y_j} v_j \right) v_i$$

Hessian – example

Calculate the second derivative of a composite function $g(\alpha) = f(\mathbf{x} + \alpha \mathbf{v})$

We already showed that $g'(\alpha) = \nabla f(\mathbf{x} + \alpha \mathbf{v})^\top \mathbf{v}$

We can treat $g'(\alpha)$ as a composite function $h(\mathbf{y}(\alpha))$, where

$$\mathbf{y}(\alpha) = \mathbf{x} + \alpha \mathbf{v}, \quad h(\mathbf{y}) = \nabla f(\mathbf{y})^\top \mathbf{v} = \sum_{i=1}^n \frac{\partial f(\mathbf{y})}{\partial y_i} \cdot v_i$$

From the chain rule:

$$\begin{aligned} g''(\alpha) &= \frac{dg'(\alpha)}{d\alpha} = \sum_{i=1}^n \frac{\partial h(\mathbf{y})}{\partial y_i} \cdot \overbrace{\frac{d y_i}{d\alpha}}^{x_i + \alpha v_i} = \sum_{i=1}^n \frac{\partial}{\partial y_i} \left(\sum_{j=1}^n \frac{\partial f(\mathbf{y})}{\partial y_j} v_j \right) v_i \\ &= \sum_{i=1}^n \sum_{j=1}^n \frac{\partial^2 f(\mathbf{y})}{\partial y_i \partial y_j} v_i v_j \end{aligned}$$

Hessian – example

Calculate the second derivative of a composite function $g(\alpha) = f(\mathbf{x} + \alpha \mathbf{v})$

We already showed that $g'(\alpha) = \nabla f(\mathbf{x} + \alpha \mathbf{v})^\top \mathbf{v}$

We can treat $g'(\alpha)$ as a composite function $h(\mathbf{y}(\alpha))$, where

$$\mathbf{y}(\alpha) = \mathbf{x} + \alpha \mathbf{v}, \quad h(\mathbf{y}) = \nabla f(\mathbf{y})^\top \mathbf{v} = \sum_{i=1}^n \frac{\partial f(\mathbf{y})}{\partial y_i} \cdot v_i$$

From the chain rule:

$$\begin{aligned} g''(\alpha) &= \frac{dg'(\alpha)}{d\alpha} = \sum_{i=1}^n \frac{\partial h(\mathbf{y})}{\partial y_i} \cdot \underbrace{\frac{d y_i}{d\alpha}}_{x_i + \alpha v_i} = \sum_{i=1}^n \frac{\partial}{\partial y_i} \left(\sum_{j=1}^n \frac{\partial f(\mathbf{y})}{\partial y_j} v_j \right) v_i \\ &= \sum_{i=1}^n \sum_{j=1}^n \frac{\partial^2 f(\mathbf{y})}{\partial y_i \partial y_j} v_i v_j = \mathbf{v}^\top \nabla^2 f(\mathbf{y}) \mathbf{v} \end{aligned}$$

Hessian – example

Calculate the second derivative of a composite function $g(\alpha) = f(\mathbf{x} + \alpha \mathbf{v})$

We already showed that $g'(\alpha) = \nabla f(\mathbf{x} + \alpha \mathbf{v})^\top \mathbf{v}$

We can treat $g'(\alpha)$ as a composite function $h(\mathbf{y}(\alpha))$, where

$$\mathbf{y}(\alpha) = \mathbf{x} + \alpha \mathbf{v}, \quad h(\mathbf{y}) = \nabla f(\mathbf{y})^\top \mathbf{v} = \sum_{i=1}^n \frac{\partial f(\mathbf{y})}{\partial y_i} \cdot v_i$$

From the chain rule:

$$\begin{aligned} g''(\alpha) &= \frac{dg'(\alpha)}{d\alpha} = \sum_{i=1}^n \frac{\partial h(\mathbf{y})}{\partial y_i} \cdot \overbrace{\frac{d y_i}{d\alpha}}^{x_i + \alpha v_i} = \sum_{i=1}^n \frac{\partial}{\partial y_i} \left(\sum_{j=1}^n \frac{\partial f(\mathbf{y})}{\partial y_j} v_j \right) v_i \\ &= \sum_{i=1}^n \sum_{j=1}^n \frac{\partial^2 f(\mathbf{y})}{\partial y_i \partial y_j} v_i v_j = \mathbf{v}^\top \nabla^2 f(\mathbf{x} + \alpha \mathbf{v}) \mathbf{v} \end{aligned}$$

$$\frac{d^2 f(\mathbf{x} + \alpha \mathbf{v})}{d\alpha^2} = \mathbf{v}^\top \nabla^2 f(\mathbf{x} + \alpha \mathbf{v}) \mathbf{v}$$

Local minimum

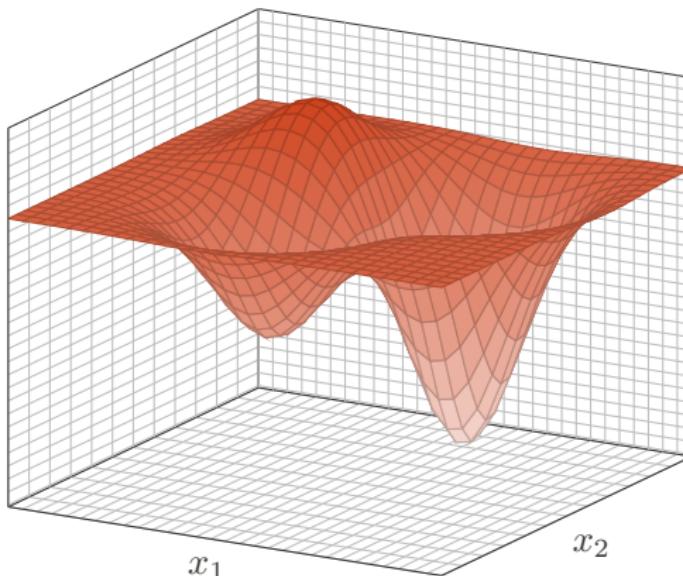
Local minimum: A function $f(x)$ has a local minimum at x_0 , if there exists $\epsilon > 0$, such that for all x satisfying $\|x - x_0\| \leq \epsilon$, it holds $f(x_0) \leq f(x)$.

A local maximum defined analogously.

Local minimum

Local minimum: A function $f(x)$ has a local minimum at x_0 , if there exists $\epsilon > 0$, such that for all x satisfying $\|x - x_0\| \leq \epsilon$, it holds $f(x_0) \leq f(x)$.

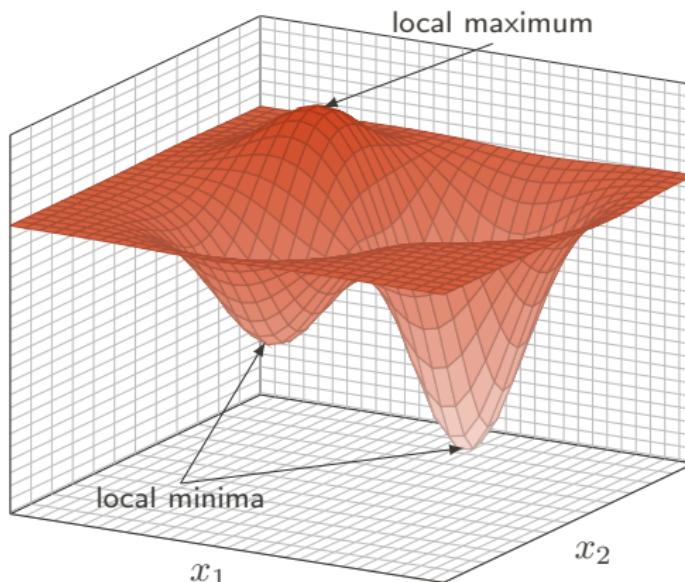
A local maximum defined analogously.



Local minimum

Local minimum: A function $f(x)$ has a local minimum at x_0 , if there exists $\epsilon > 0$, such that for all x satisfying $\|x - x_0\| \leq \epsilon$, it holds $f(x_0) \leq f(x)$.

A local maximum defined analogously.



Stationary points

Let $f(x)$ be differentiable over its entire domain. Points, for which $\nabla f(x) = \mathbf{0}$, are called **stationary points (critical points)**

Stationary points

Let $f(x)$ be differentiable over its entire domain. Points, for which $\nabla f(x) = \mathbf{0}$, are called **stationary points (critical points)**

The necessary condition for an extremum: If x_0 is an extremum (minimum or maximum) of a differentiable function $f(x)$, then x_0 is a stationary point of $f(x)$.

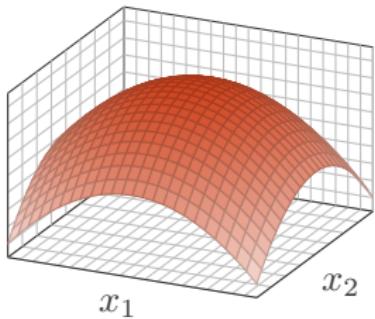
Stationary points

Let $f(x)$ be differentiable over its entire domain. Points, for which $\nabla f(x) = \mathbf{0}$, are called **stationary points (critical points)**

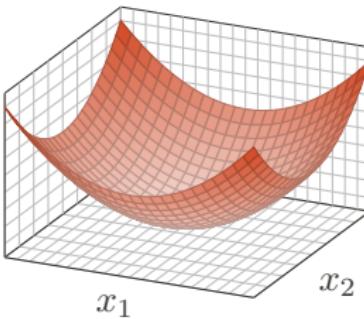
The necessary condition for an extremum: If x_0 is an extremum (minimum or maximum) of a differentiable function $f(x)$, then x_0 is a stationary point of $f(x)$.

Remark: Not all stationary points are extrema.

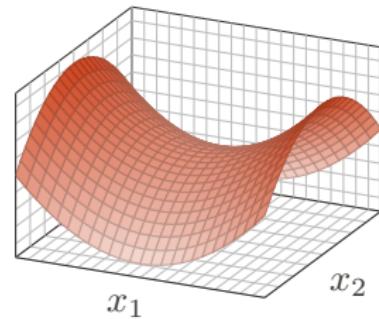
local maximum



local minimum



saddle point



The sufficient condition for a local minimum

If x_0 is a stationary point of twice differentiable function $f(x)$, i.e., $\nabla f(x_0) = \mathbf{0}$, and the Hessian $\nabla^2 f(x_0)$ is **positive-definite**, then $f(x)$ has a **local minimum at x_0**

The sufficient condition for a local minimum

If x_0 is a stationary point of twice differentiable function $f(x)$, i.e., $\nabla f(x_0) = \mathbf{0}$, and the Hessian $\nabla^2 f(x_0)$ is **positive-definite**, then $f(x)$ has a **local minimum at x_0**

Remainder: a symmetric matrix A is positive-definite if $v^\top A v > 0$ for any non-zero vector v

Equivalently, all eigenvalues of A are positive

The minimum of a quadratic function

Consider a quadratic function of a general form:

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A}\mathbf{x} + \mathbf{b}^\top \mathbf{x} + c$$

where \mathbf{A} is symmetric and invertible.

The minimum of a quadratic function

Consider a quadratic function of a general form:

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x} + \mathbf{b}^\top \mathbf{x} + c$$

where \mathbf{A} is symmetric and invertible.

Calculate the gradient and the Hessian:

$$\nabla f(\mathbf{x}) = 2\mathbf{A}\mathbf{x} + \mathbf{b}, \quad \nabla^2 f(\mathbf{x}) = 2\mathbf{A}$$

The minimum of a quadratic function

Consider a quadratic function of a general form:

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A}\mathbf{x} + \mathbf{b}^\top \mathbf{x} + c$$

where \mathbf{A} is symmetric and invertible.

Calculate the gradient and the Hessian:

$$\nabla f(\mathbf{x}) = 2\mathbf{A}\mathbf{x} + \mathbf{b}, \quad \nabla^2 f(\mathbf{x}) = 2\mathbf{A}$$

Find the stationary point:

$$\nabla f(\mathbf{x}) = \mathbf{0} \iff 2\mathbf{A}\mathbf{x} = -\mathbf{b} \iff \mathbf{x} = -\frac{1}{2}\mathbf{A}^{-1}\mathbf{b}$$

It is a **minimum**, if \mathbf{A} is **positive-definite**

The minimum of a quadratic function

Consider a quadratic function of a general form:

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x} + \mathbf{b}^\top \mathbf{x} + c$$

where \mathbf{A} is symmetric and invertible.

Calculate the gradient and the Hessian:

$$\nabla f(\mathbf{x}) = 2\mathbf{A}\mathbf{x} + \mathbf{b}, \quad \nabla^2 f(\mathbf{x}) = 2\mathbf{A}$$

Find the stationary point:

$$\nabla f(\mathbf{x}) = \mathbf{0} \iff 2\mathbf{A}\mathbf{x} = -\mathbf{b} \iff \mathbf{x} = -\frac{1}{2}\mathbf{A}^{-1}\mathbf{b}$$

It is a **minimum**, if \mathbf{A} is **positive-definite**

Conclusion: The minimum of a quadratic function:

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x} + \mathbf{b}^\top \mathbf{x} + c, \quad \mathbf{A} - \text{positive-definite},$$

is given by $\mathbf{x}^* = -\frac{1}{2}\mathbf{A}^{-1}\mathbf{b}$

Quadratic function – example

$$f(x_1, x_2) = 3x_1^2 + 3x_2^2 - 2x_1x_2 + x_1 - 2x_2 + 1$$

Quadratic function – example

$$f(x_1, x_2) = 3x_1^2 + 3x_2^2 - 2x_1x_2 + x_1 - 2x_2 + 1$$

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x} + \mathbf{b}^\top \mathbf{x} + c \quad \text{where } c = 1, \mathbf{b} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}, \mathbf{A} = \begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix}$$

Quadratic function – example

$$f(x_1, x_2) = 3x_1^2 + 3x_2^2 - 2x_1x_2 + x_1 - 2x_2 + 1$$

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x} + \mathbf{b}^\top \mathbf{x} + c \quad \text{where } c = 1, \mathbf{b} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}, \mathbf{A} = \begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix}$$

$$\mathbf{A}^{-1} = \frac{1}{8} \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}$$

Quadratic function – example

$$f(x_1, x_2) = 3x_1^2 + 3x_2^2 - 2x_1x_2 + x_1 - 2x_2 + 1$$

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x} + \mathbf{b}^\top \mathbf{x} + c \quad \text{where } c = 1, \mathbf{b} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}, \mathbf{A} = \begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix}$$

$$\mathbf{A}^{-1} = \frac{1}{8} \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix} \quad \Rightarrow \quad \mathbf{x}^* = -\mathbf{A}^{-1}\mathbf{b} = \begin{bmatrix} -\frac{1}{16} \\ \frac{5}{16} \end{bmatrix}$$

Point \mathbf{x}^* is a stationary point

Quadratic function – example

$$f(x_1, x_2) = 3x_1^2 + 3x_2^2 - 2x_1x_2 + x_1 - 2x_2 + 1$$

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A}\mathbf{x} + \mathbf{b}^\top \mathbf{x} + c \quad \text{where } c = 1, \mathbf{b} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}, \mathbf{A} = \begin{bmatrix} 3 & -1 \\ -1 & 3 \end{bmatrix}$$

$$\mathbf{A}^{-1} = \frac{1}{8} \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix} \quad \Rightarrow \quad \mathbf{x}^* = -\mathbf{A}^{-1}\mathbf{b} = \begin{bmatrix} -\frac{1}{16} \\ \frac{5}{16} \end{bmatrix}$$

Point \mathbf{x}^* is a stationary point

Since $\det \mathbf{A} = 8$ and $A_{11} = 3$, \mathbf{A} is **positive-definite**, therefore the function has a **minimum at \mathbf{x}^***

Taylor polynomials for a multivariate function

For any function $f(\boldsymbol{x})$ and any point \boldsymbol{x}_0 , consider a **univariate function**:

$$g(\alpha) = f(\boldsymbol{x}_0 + \alpha(\boldsymbol{x} - \boldsymbol{x}_0))$$

It holds $g(0) = f(\boldsymbol{x}_0)$ i $g(1) = f(\boldsymbol{x})$.

Taylor polynomials for a multivariate function

For any function $f(\mathbf{x})$ and any point \mathbf{x}_0 , consider a **univariate function**:

$$g(\alpha) = f(\mathbf{x}_0 + \alpha(\mathbf{x} - \mathbf{x}_0))$$

It holds $g(0) = f(\mathbf{x}_0)$ i $g(1) = f(\mathbf{x})$.

Constructing Taylor polynomials for a function $f(\mathbf{x})$ at \mathbf{x}_0 :

- We compute the Taylor polynomial $P_k(\alpha; 0)$ for $g(\alpha)$ at $\alpha = 0$
- Since $P_k(\alpha; 0)$ approximates $g(\alpha)$ around $\alpha = 0$, and $g(0) = f(\mathbf{x}_0)$ and $g(1) = f(\mathbf{x})$, then $P_k(1; 0)$ approximates $f(\mathbf{x})$ around $\mathbf{x} = \mathbf{x}_0$

Taylor polynomials for a multivariate function

For any function $f(\mathbf{x})$ and any point \mathbf{x}_0 , consider a **univariate function**:

$$g(\alpha) = f(\mathbf{x}_0 + \alpha(\mathbf{x} - \mathbf{x}_0))$$

It holds $g(0) = f(\mathbf{x}_0)$ i $g(1) = f(\mathbf{x})$.

Constructing Taylor polynomials for a function $f(\mathbf{x})$ at \mathbf{x}_0 :

- We compute the Taylor polynomial $P_k(\alpha; 0)$ for $g(\alpha)$ at $\alpha = 0$
- Since $P_k(\alpha; 0)$ approximates $g(\alpha)$ around $\alpha = 0$, and $g(0) = f(\mathbf{x}_0)$ and $g(1) = f(\mathbf{x})$, then $P_k(1; 0)$ approximates $f(\mathbf{x})$ around $\mathbf{x} = \mathbf{x}_0$

We earlier showed (for $\mathbf{v} = \mathbf{x} - \mathbf{x}_0$) that:

$$g'(\alpha) = \nabla f(\mathbf{x}_0 + \alpha(\mathbf{x} - \mathbf{x}_0))^{\top} (\mathbf{x} - \mathbf{x}_0)$$

$$g''(\alpha) = (\mathbf{x} - \mathbf{x}_0)^{\top} \nabla^2 f(\mathbf{x}_0 + \alpha(\mathbf{x} - \mathbf{x}_0)) (\mathbf{x} - \mathbf{x}_0)$$

Taylor polynomials for a multivariate function

For any function $f(\mathbf{x})$ and any point \mathbf{x}_0 , consider a **univariate function**:

$$g(\alpha) = f(\mathbf{x}_0 + \alpha(\mathbf{x} - \mathbf{x}_0))$$

It holds $g(0) = f(\mathbf{x}_0)$ i $g(1) = f(\mathbf{x})$.

Constructing Taylor polynomials for a function $f(\mathbf{x})$ at \mathbf{x}_0 :

- We compute the Taylor polynomial $P_k(\alpha; 0)$ for $g(\alpha)$ at $\alpha = 0$
- Since $P_k(\alpha; 0)$ approximates $g(\alpha)$ around $\alpha = 0$, and $g(0) = f(\mathbf{x}_0)$ and $g(1) = f(\mathbf{x})$, then $P_k(1; 0)$ approximates $f(\mathbf{x})$ around $\mathbf{x} = \mathbf{x}_0$

We earlier showed (for $\mathbf{v} = \mathbf{x} - \mathbf{x}_0$) that:

$$g'(0) = \nabla f(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0)$$

$$g''(0) = (\mathbf{x} - \mathbf{x}_0)^\top \nabla^2 f(\mathbf{x}_0) (\mathbf{x} - \mathbf{x}_0)$$

Taylor polynomials for a multivariate function

For any function $f(\mathbf{x})$ and any point \mathbf{x}_0 , consider a **univariate function**:

$$g(\alpha) = f(\mathbf{x}_0 + \alpha(\mathbf{x} - \mathbf{x}_0))$$

It holds $g(0) = f(\mathbf{x}_0)$ i $g(1) = f(\mathbf{x})$.

Constructing Taylor polynomials for a function $f(\mathbf{x})$ at \mathbf{x}_0 :

- We compute the Taylor polynomial $P_k(\alpha; 0)$ for $g(\alpha)$ at $\alpha = 0$
- Since $P_k(\alpha; 0)$ approximates $g(\alpha)$ around $\alpha = 0$, and $g(0) = f(\mathbf{x}_0)$ and $g(1) = f(\mathbf{x})$, then $P_k(1; 0)$ approximates $f(\mathbf{x})$ around $\mathbf{x} = \mathbf{x}_0$

We earlier showed (for $\mathbf{v} = \mathbf{x} - \mathbf{x}_0$) that:

$$g'(0) = \nabla f(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0)$$

$$g''(0) = (\mathbf{x} - \mathbf{x}_0)^\top \nabla^2 f(\mathbf{x}_0) (\mathbf{x} - \mathbf{x}_0)$$

Therefore:

$$P_0(\alpha; 0) = g(0)$$

$$P_1(\alpha; 0) = g(0) + g'(0)\alpha$$

$$P_2(\alpha; 0) = g(0) + g'(0)\alpha + \frac{g''(0)}{2}\alpha^2$$

Taylor polynomials for a multivariate function

For any function $f(\mathbf{x})$ and any point \mathbf{x}_0 , consider a **univariate function**:

$$g(\alpha) = f(\mathbf{x}_0 + \alpha(\mathbf{x} - \mathbf{x}_0))$$

It holds $g(0) = f(\mathbf{x}_0)$ i $g(1) = f(\mathbf{x})$.

Constructing Taylor polynomials for a function $f(\mathbf{x})$ at \mathbf{x}_0 :

- We compute the Taylor polynomial $P_k(\alpha; 0)$ for $g(\alpha)$ at $\alpha = 0$
- Since $P_k(\alpha; 0)$ approximates $g(\alpha)$ around $\alpha = 0$, and $g(0) = f(\mathbf{x}_0)$ and $g(1) = f(\mathbf{x})$, then $P_k(1; 0)$ approximates $f(\mathbf{x})$ around $\mathbf{x} = \mathbf{x}_0$

We earlier showed (for $\mathbf{v} = \mathbf{x} - \mathbf{x}_0$) that:

$$g'(0) = \nabla f(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0)$$

$$g''(0) = (\mathbf{x} - \mathbf{x}_0)^\top \nabla^2 f(\mathbf{x}_0) (\mathbf{x} - \mathbf{x}_0)$$

Therefore:

$$P_0(\alpha; 0) = f(\mathbf{x})$$

$$P_1(\alpha; 0) = f(\mathbf{x}_0) + \alpha \nabla f(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0)$$

$$P_2(\alpha; 0) = f(\mathbf{x}_0) + \alpha \nabla f(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0) + \frac{\alpha^2}{2} (\mathbf{x} - \mathbf{x}_0)^\top \nabla^2 f(\mathbf{x}_0) (\mathbf{x} - \mathbf{x}_0)$$

Taylor polynomials for a multivariate function

The Taylor polynomials for a function $f(\mathbf{x})$ at point \mathbf{x}_0 :

$$P_0(\mathbf{x}; \mathbf{x}_0) = f(\mathbf{x}_0)$$

$$P_1(\mathbf{x}; \mathbf{x}_0) = f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0)$$

$$P_2(\mathbf{x}; \mathbf{x}_0) = f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0) + \frac{1}{2} (\mathbf{x} - \mathbf{x}_0)^\top \nabla^2 f(\mathbf{x}_0) (\mathbf{x} - \mathbf{x}_0)$$

$$\dots = \dots$$

Taylor polynomials for a multivariate function

The Taylor polynomials for a function $f(\mathbf{x})$ at point \mathbf{x}_0 :

$$P_0(\mathbf{x}; \mathbf{x}_0) = f(\mathbf{x}_0)$$

$$P_1(\mathbf{x}; \mathbf{x}_0) = f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0)$$

$$P_2(\mathbf{x}; \mathbf{x}_0) = f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^\top \nabla^2 f(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0)$$

$$\dots = \dots$$

Taylor's theorem (only for the second order case):

$$f(\mathbf{x}) = f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^\top \nabla^2 f(\boldsymbol{\xi})(\mathbf{x} - \mathbf{x}_0),$$

where $\boldsymbol{\xi}$ is a point in between \mathbf{x}_0 and \mathbf{x}

Taylor polynomials for a quadratic function

Consider a quadratic function with a positive-definite matrix A

$$f(\mathbf{x}) = \mathbf{x}^\top A \mathbf{x} + \mathbf{b}^\top \mathbf{x} + c, \quad \nabla f(\mathbf{x}) = 2A\mathbf{x} + \mathbf{b}, \quad \nabla^2 f(\mathbf{x}) = 2A$$

Taylor polynomials for a quadratic function

Consider a quadratic function with a positive-definite matrix A

$$f(\mathbf{x}) = \mathbf{x}^\top A \mathbf{x} + \mathbf{b}^\top \mathbf{x} + c, \quad \nabla f(\mathbf{x}) = 2A\mathbf{x} + \mathbf{b}, \quad \nabla^2 f(\mathbf{x}) = 2A$$

Approximating the quadratic function with a second-order Taylor polynomial (also quadratic function!) is **exact**, and from the Taylor's theorem for $\mathbf{x}_0 = \mathbf{x}^*$:

Taylor polynomials for a quadratic function

Consider a quadratic function with a positive-definite matrix A

$$f(\mathbf{x}) = \mathbf{x}^\top A \mathbf{x} + \mathbf{b}^\top \mathbf{x} + c, \quad \nabla f(\mathbf{x}) = 2A\mathbf{x} + \mathbf{b}, \quad \nabla^2 f(\mathbf{x}) = 2A$$

Approximating the quadratic function with a second-order Taylor polynomial (also quadratic function!) is **exact**, and from the Taylor's theorem for $\mathbf{x}_0 = \mathbf{x}^*$:

$$f(\mathbf{x}) = f(\mathbf{x}^*) + \nabla f(\mathbf{x}^*)^\top (\mathbf{x} - \mathbf{x}^*) + \frac{1}{2}(\mathbf{x} - \mathbf{x}^*)^\top \nabla^2 f(\boldsymbol{\xi})(\mathbf{x} - \mathbf{x}^*)$$

Taylor polynomials for a quadratic function

Consider a quadratic function with a positive-definite matrix A

$$f(\mathbf{x}) = \mathbf{x}^\top A \mathbf{x} + \mathbf{b}^\top \mathbf{x} + c, \quad \nabla f(\mathbf{x}) = 2A\mathbf{x} + \mathbf{b}, \quad \nabla^2 f(\mathbf{x}) = 2A$$

Approximating the quadratic function with a second-order Taylor polynomial (also quadratic function!) is **exact**, and from the Taylor's theorem for $\mathbf{x}_0 = \mathbf{x}^*$:

$$f(\mathbf{x}) = f(\mathbf{x}^*) + \nabla f(\mathbf{x}^*)^\top (\mathbf{x} - \mathbf{x}^*) + \frac{1}{2}(\mathbf{x} - \mathbf{x}^*)^\top \nabla^2 f(\boldsymbol{\xi})(\mathbf{x} - \mathbf{x}^*)$$

Since $\nabla f(\mathbf{x}^*) = \mathbf{0}$ (because \mathbf{x}^* is the minimizer), and $\nabla^2 f(\boldsymbol{\xi}) = 2A$:

For a quadratic function $f(\mathbf{x}) = \mathbf{x}^\top A \mathbf{x} + \mathbf{b}^\top \mathbf{x} + c$ with a positive-definite matrix A :

$$f(\mathbf{x}) = f(\mathbf{x}^*) + (\mathbf{x} - \mathbf{x}^*)^\top A(\mathbf{x} - \mathbf{x}^*)$$

Optimization Methods for Data Analysis

3. Convex sets and functions

Wojciech Kotłowski

Institute of Computing Science, PUT
<http://www.cs.put.poznan.pl/wkotlowski/>

24.03.2022

Convex combination

A **convex combination** of vectors x_1, \dots, x_k is a vector:

$$x = \lambda_1 x_1 + \dots + \lambda_k x_k,$$

where the coefficients λ_i are **nonnegative** and such that $\lambda_1 + \dots + \lambda_k = 1$.

Convex combination

A **convex combination** of vectors x_1, \dots, x_k is a vector:

$$x = \lambda_1 x_1 + \dots + \lambda_k x_k,$$

where the coefficients λ_i are **nonnegative** and such that $\lambda_1 + \dots + \lambda_k = 1$.

Example: a set of all convex combinations of two points x_1 i x_2

$\{x : x = \lambda x_1 + (1 - \lambda) x_2, \lambda \in [0, 1]\}$ is an interval joining x_1 and x_2 :



Convex combination

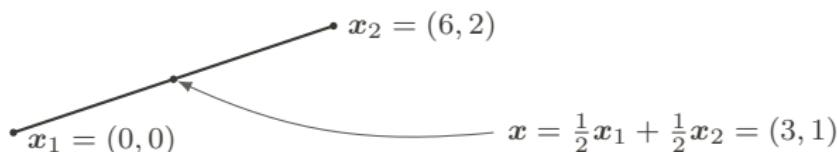
A **convex combination** of vectors x_1, \dots, x_k is a vector:

$$x = \lambda_1 x_1 + \dots + \lambda_k x_k,$$

where the coefficients λ_i are **nonnegative** and such that $\lambda_1 + \dots + \lambda_k = 1$.

Example: a set of all convex combinations of two points x_1 i x_2

$\{x : x = \lambda x_1 + (1 - \lambda) x_2, \lambda \in [0, 1]\}$ is an interval joining x_1 and x_2 :



Convex combination

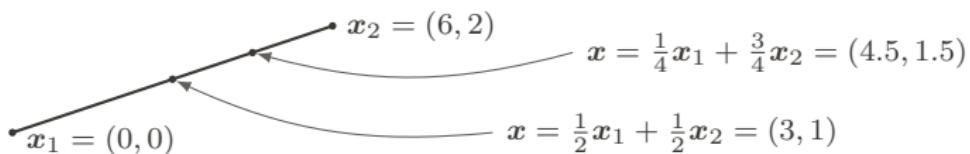
A **convex combination** of vectors x_1, \dots, x_k is a vector:

$$x = \lambda_1 x_1 + \dots + \lambda_k x_k,$$

where the coefficients λ_i are **nonnegative** and such that $\lambda_1 + \dots + \lambda_k = 1$.

Example: a set of all convex combinations of two points x_1 i x_2

$\{x : x = \lambda x_1 + (1 - \lambda) x_2, \lambda \in [0, 1]\}$ is an interval joining x_1 and x_2 :



Convex combination

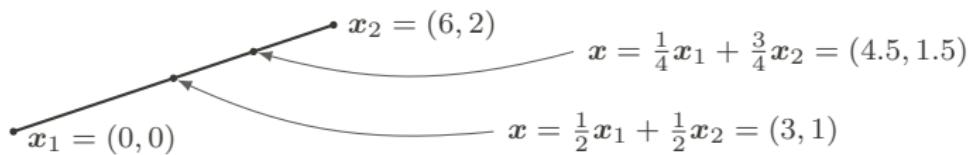
A **convex combination** of vectors x_1, \dots, x_k is a vector:

$$x = \lambda_1 x_1 + \dots + \lambda_k x_k,$$

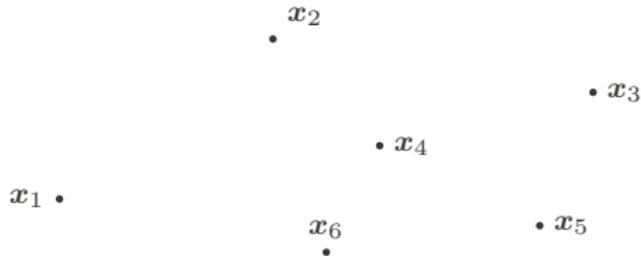
where the coefficients λ_i are **nonnegative** and such that $\lambda_1 + \dots + \lambda_k = 1$.

Example: a set of all convex combinations of two points x_1 i x_2

$\{x : x = \lambda x_1 + (1 - \lambda) x_2, \lambda \in [0, 1]\}$ is an interval joining x_1 and x_2 :



Example: a set of all convex combinations of k points?



Convex combination

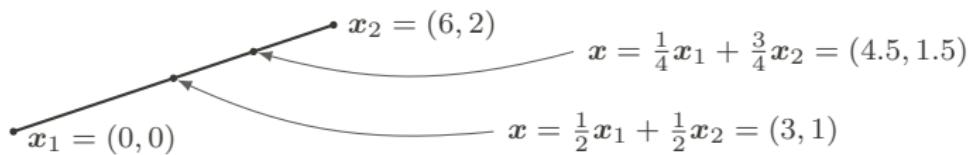
A **convex combination** of vectors x_1, \dots, x_k is a vector:

$$x = \lambda_1 x_1 + \dots + \lambda_k x_k,$$

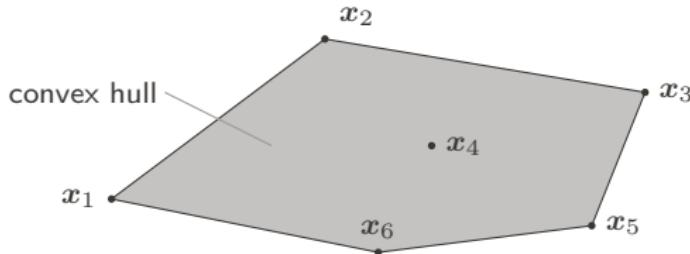
where the coefficients λ_i are **nonnegative** and such that $\lambda_1 + \dots + \lambda_k = 1$.

Example: a set of all convex combinations of two points x_1 i x_2

$\{x : x = \lambda x_1 + (1 - \lambda) x_2, \lambda \in [0, 1]\}$ is an interval joining x_1 and x_2 :



Example: a set of all convex combinations of k points?



Convex set

A **convex set** \mathcal{X} is a set such that for any points $x_1, \dots, x_k \in \mathcal{X}$, their every convex combination belongs to \mathcal{X} .

Convex set

A **convex set** \mathcal{X} is a set such that for any points $x_1, \dots, x_k \in \mathcal{X}$, their every convex combination belongs to \mathcal{X} .

Alternatively, a convex set \mathcal{X} is a set such that the interval joining any two points $x_1, x_2 \in \mathcal{X}$ entirely belongs to \mathcal{X} .

$$\forall x_1, x_2 \in \mathcal{X} \quad \{x : x = \lambda x_1 + (1 - \lambda)x_2, \lambda \in [0, 1]\} \subseteq \mathcal{X}$$

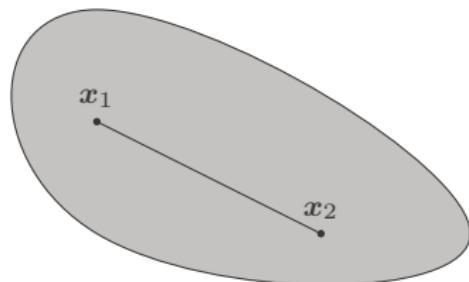
Convex set

A **convex set** \mathcal{X} is a set such that for any points $x_1, \dots, x_k \in \mathcal{X}$, their every convex combination belongs to \mathcal{X} .

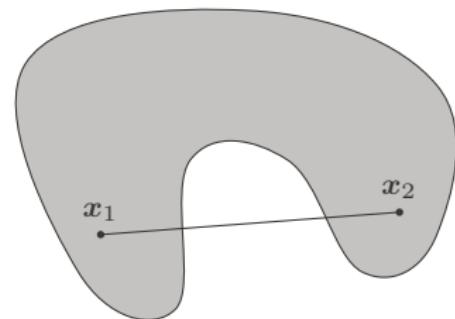
Alternatively, a convex set \mathcal{X} is a set such that the interval joining any two points $x_1, x_2 \in \mathcal{X}$ entirely belongs to \mathcal{X} .

$$\forall x_1, x_2 \in \mathcal{X} \quad \{x : x = \lambda x_1 + (1 - \lambda)x_2, \lambda \in [0, 1]\} \subseteq \mathcal{X}$$

convex

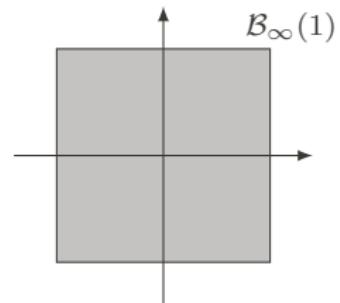
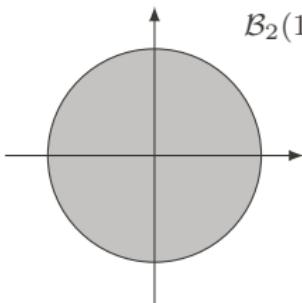
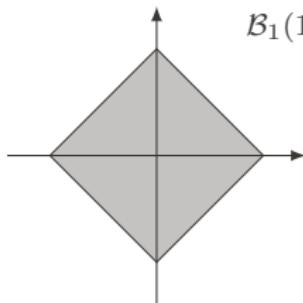


non-convex



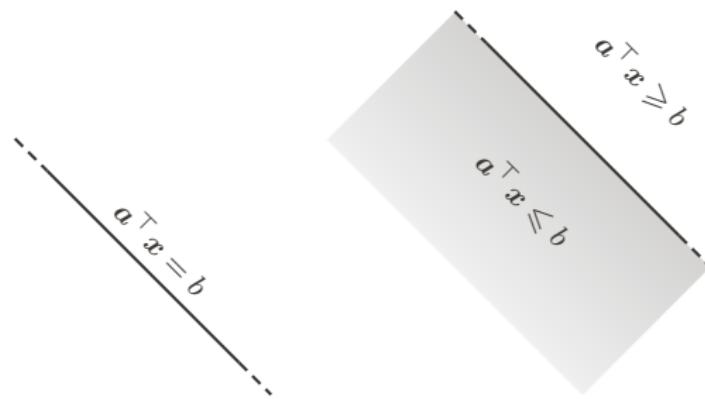
Examples of convex sets

- A p -norm ball $\mathcal{B}_p(r) = \{\mathbf{x}: |x_1|^p + \dots + |x_n|^p \leq r^p\}$



Examples of convex sets

- Hyperplane $\{x: \mathbf{a}^\top x = b\}$ or half-space $\{x: \mathbf{a}^\top x \leq b\}$



Intersection of convex sets

The intersection of convex sets is a convex set

Intersection of convex sets

The intersection of convex sets is a convex set

Proof: Let $\mathcal{C}_1, \dots, \mathcal{C}_m$ – convex sets and let $\mathcal{C} = \bigcap_{j=1}^m \mathcal{C}_j$

Intersection of convex sets

The intersection of convex sets is a convex set

Proof: Let $\mathcal{C}_1, \dots, \mathcal{C}_m$ – convex sets and let $\mathcal{C} = \bigcap_{j=1}^m \mathcal{C}_j$

- Take any two $x_1, x_2 \in \mathcal{C}$

From the definition of \mathcal{C} we have $x_1, x_2 \in \mathcal{C}_j$ for any $j = 1, \dots, m$

Intersection of convex sets

The intersection of convex sets is a convex set

Proof: Let $\mathcal{C}_1, \dots, \mathcal{C}_m$ – convex sets and let $\mathcal{C} = \bigcap_{j=1}^m \mathcal{C}_j$

- Take any two $x_1, x_2 \in \mathcal{C}$
From the definition of \mathcal{C} we have $x_1, x_2 \in \mathcal{C}_j$ for any $j = 1, \dots, m$
- From convexity of \mathcal{C}_j we have $x = \lambda x_1 + (1 - \lambda)x_2 \in \mathcal{C}_j$ for each j

Intersection of convex sets

The intersection of convex sets is a convex set

Proof: Let $\mathcal{C}_1, \dots, \mathcal{C}_m$ – convex sets and let $\mathcal{C} = \bigcap_{j=1}^m \mathcal{C}_j$

- Take any two $x_1, x_2 \in \mathcal{C}$
From the definition of \mathcal{C} we have $x_1, x_2 \in \mathcal{C}_j$ for any $j = 1, \dots, m$
- From convexity of \mathcal{C}_j we have $x = \lambda x_1 + (1 - \lambda)x_2 \in \mathcal{C}_j$ for each j
- Thus $x \in \mathcal{C}$, so \mathcal{C} is a convex set

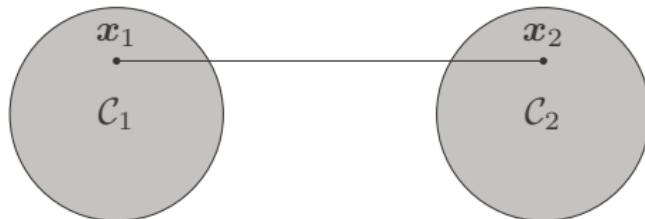
Intersection of convex sets

The intersection of convex sets is a convex set

Proof: Let $\mathcal{C}_1, \dots, \mathcal{C}_m$ – convex sets and let $\mathcal{C} = \bigcap_{j=1}^m \mathcal{C}_j$

- Take any two $x_1, x_2 \in \mathcal{C}$
From the definition of \mathcal{C} we have $x_1, x_2 \in \mathcal{C}_j$ for any $j = 1, \dots, m$
- From convexity of \mathcal{C}_j we have $x = \lambda x_1 + (1 - \lambda)x_2 \in \mathcal{C}_j$ for each j
- Thus $x \in \mathcal{C}$, so \mathcal{C} is a convex set

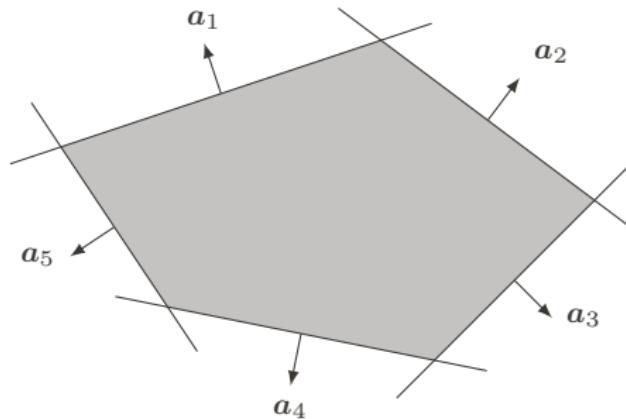
Remark: the sum of two convex sets is not necessarily convex (and usually is not)!



Examples of convex sets

- Polytope:

$$\{x: Ax \leq b\} = \bigcap_{j=1}^m \{x: a_j^\top x \leq b_j\}, \quad A = \begin{bmatrix} a_1^\top \\ a_2^\top \\ \vdots \\ a_m^\top \end{bmatrix}$$



Convex function

A function $f(x)$ is **convex**, if for any two points x, y in its domain and any $\lambda \in [0, 1]$,

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y)$$

In other words, the interval joint any two points on a function graph lies entirely above or at the graph.

If the inequality is strict („ $<$ ”), the function is called **strictly convex**

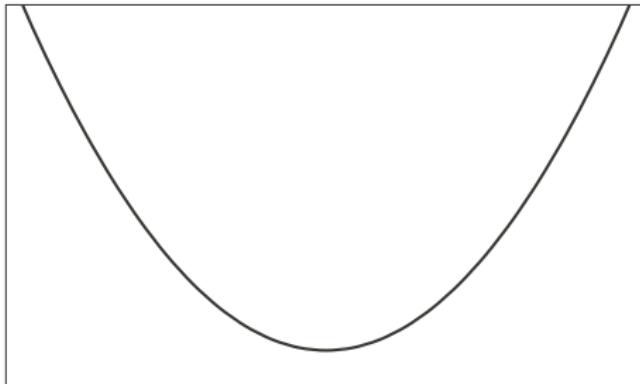
Convex function

A function $f(x)$ is **convex**, if for any two points x, y in its domain and any $\lambda \in [0, 1]$,

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y)$$

In other words, the interval joint any two points on a function graph lies entirely above or at the graph.

If the inequality is strict („ $<$ ”), the function is called **strictly convex**



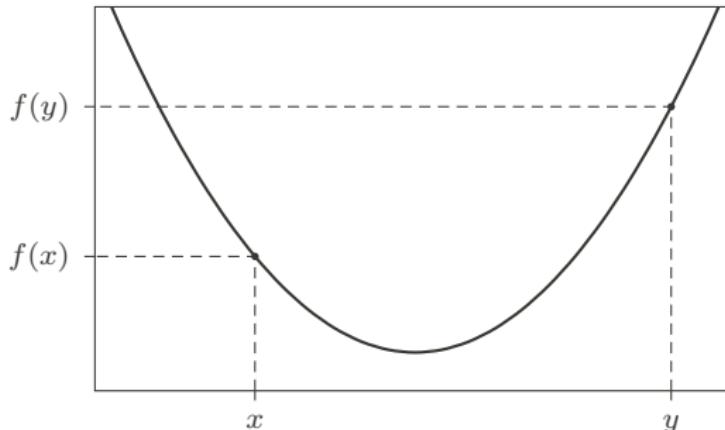
Convex function

A function $f(x)$ is **convex**, if for any two points x, y in its domain and any $\lambda \in [0, 1]$,

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y)$$

In other words, the interval joint any two points on a function graph lies entirely above or at the graph.

If the inequality is strict („ $<$ ”), the function is called **strictly convex**



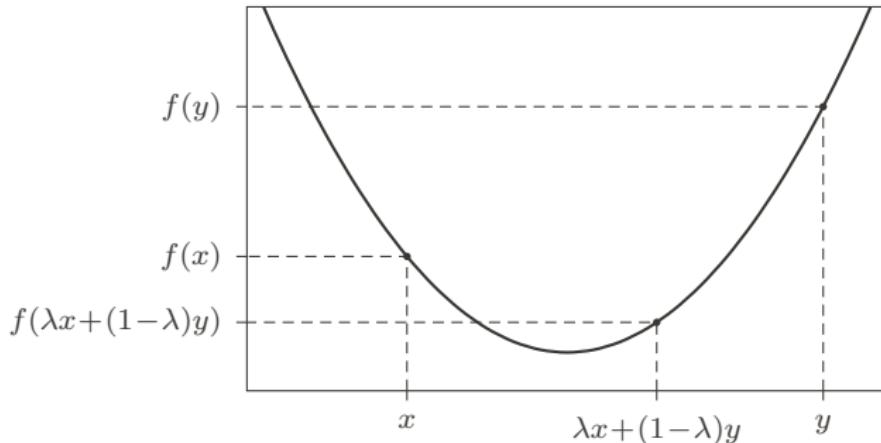
Convex function

A function $f(x)$ is **convex**, if for any two points x, y in its domain and any $\lambda \in [0, 1]$,

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y)$$

In other words, the interval joint any two points on a function graph lies entirely above or at the graph.

If the inequality is strict („ $<$ ”), the function is called **strictly convex**



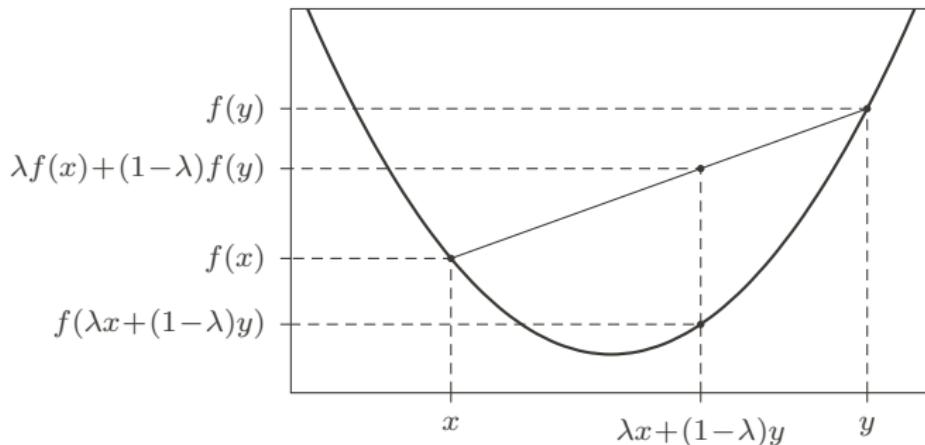
Convex function

A function $f(x)$ is **convex**, if for any two points x, y in its domain and any $\lambda \in [0, 1]$,

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y)$$

In other words, the interval joint any two points on a function graph lies entirely above or at the graph.

If the inequality is strict („ $<$ ”), the function is called **strictly convex**



Convex function

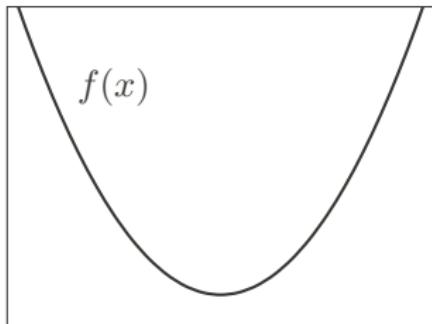
A function $f(x)$ is **convex**, if for any two points x, y in its domain and any $\lambda \in [0, 1]$,

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y)$$

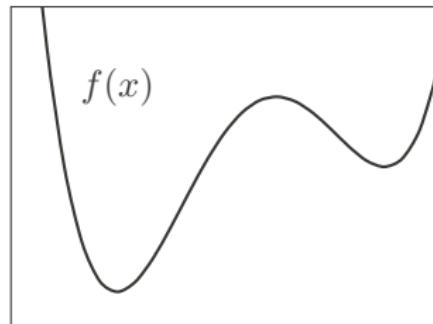
In other words, the interval joint any two points on a function graph lies entirely above or at the graph.

If the inequality is strict („ $<$ ”), the function is called **strictly convex**

convex function



non-convex function



Convex function

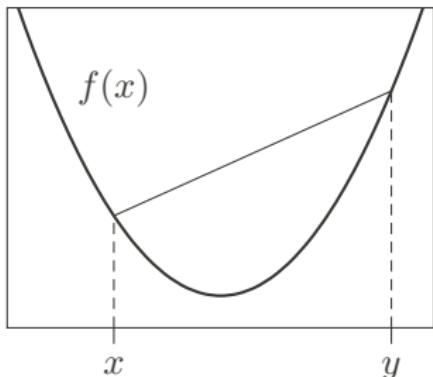
A function $f(x)$ is **convex**, if for any two points x, y in its domain and any $\lambda \in [0, 1]$,

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y)$$

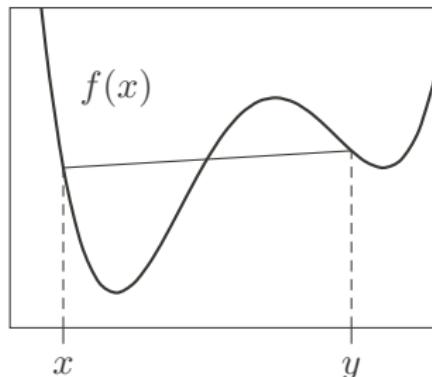
In other words, the interval joint any two points on a function graph lies entirely above or at the graph.

If the inequality is strict („ $<$ ”), the function is called **strictly convex**

convex function

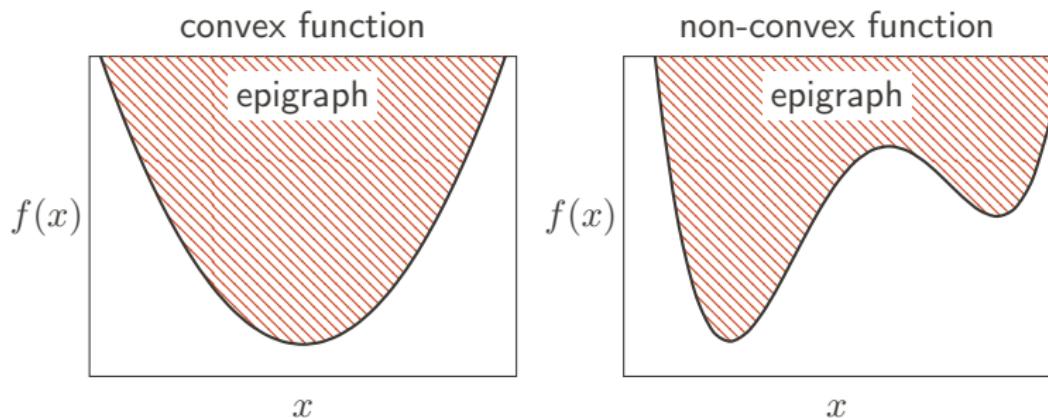


non-convex function



Convex function

A function is convex if and only if its **epigraph** (the set bounded from below by the function graph) is a **convex set**



Convex function

If a function $f(\mathbf{x})$ is **convex**, then for any $\mathbf{x}_1, \dots, \mathbf{x}_k$ from the domain and any $\lambda_1, \dots, \lambda_k$ such that $\lambda_1 + \dots + \lambda_k = 1$,

$$f(\lambda_1 \mathbf{x}_1 + \dots + \lambda_k \mathbf{x}_k) \leq \lambda_1 f(\mathbf{x}_1) + \dots + \lambda_k f(\mathbf{x}_k)$$

In words: the value of the function at the convex combination of points is not larger than the convex combination of function values at these points.

The proof by a simple induction on k (omitted)

Minimization of convex functions

Local minimum: A function $f(x)$ has a local minimum at x_0 , if there exists $\epsilon > 0$, such that for every x satisfying $\|x - x_0\| \leq \epsilon$, we have $f(x_0) \leq f(x)$

Minimization of convex functions

Local minimum: A function $f(x)$ has a local minimum at x_0 , if there exists $\epsilon > 0$, such that for every x satisfying $\|x - x_0\| \leq \epsilon$, we have $f(x_0) \leq f(x)$

Theorem: Consider the optimization problem:

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & x \in \mathcal{X}, \end{aligned}$$

in which both the function $f(x)$ and the set of feasible solutions \mathcal{X} are **convex**. Then every local minimum of $f(x)$ is also its global minimum

Proof

Proof

- Let $x_0 \in \mathcal{X}$ – a local minimum. Assume the contrary that x_0 is **not** a global minimum, i.e. there exists $x^* \in \mathcal{X}$ such that $f(x^*) < f(x_0)$

Proof

- Let $x_0 \in \mathcal{X}$ – a local minimum. Assume the contrary that x_0 is **not** a global minimum, i.e. there exists $x^* \in \mathcal{X}$ such that $f(x^*) < f(x_0)$
- Since x_0 is a local minimum, there exists $\epsilon > 0$ such that if $\|x - x_0\| \leq \epsilon$, then $f(x_0) \leq f(x)$

Proof

- Let $x_0 \in \mathcal{X}$ – a local minimum. Assume the contrary that x_0 is **not** a global minimum, i.e. there exists $x^* \in \mathcal{X}$ such that $f(x^*) < f(x_0)$
- Since x_0 is a local minimum, there exists $\epsilon > 0$ such that if $\|x - x_0\| \leq \epsilon$, then $f(x_0) \leq f(x)$
- Take a convex combination $y = \lambda x^* + (1 - \lambda)x_0$. Note that:

$$\|y - x_0\| = \|\lambda(x^* - x_0)\| = \lambda\|x^* - x_0\|$$

so for sufficiently small λ we will have $\|y - x_0\| \leq \epsilon$, and hence $f(x_0) \leq f(y)$

Proof

- Let $x_0 \in \mathcal{X}$ – a local minimum. Assume the contrary that x_0 is **not** a global minimum, i.e. there exists $x^* \in \mathcal{X}$ such that $f(x^*) < f(x_0)$
- Since x_0 is a local minimum, there exists $\epsilon > 0$ such that if $\|x - x_0\| \leq \epsilon$, then $f(x_0) \leq f(x)$
- Take a convex combination $y = \lambda x^* + (1 - \lambda)x_0$. Note that:

$$\|y - x_0\| = \|\lambda(x^* - x_0)\| = \lambda\|x^* - x_0\|$$

so for sufficiently small λ we will have $\|y - x_0\| \leq \epsilon$, and hence $f(x_0) \leq f(y)$

- On the other hand, from the convexity of $f(x)$:

$$\begin{aligned} f(y) &\leq \lambda f(x^*) + (1 - \lambda)f(x_0) \\ &< \lambda f(x_0) + (1 - \lambda)f(x_0) = f(x_0) \end{aligned}$$

Proof

- Let $x_0 \in \mathcal{X}$ – a local minimum. Assume the contrary that x_0 is **not** a global minimum, i.e. there exists $x^* \in \mathcal{X}$ such that $f(x^*) < f(x_0)$
- Since x_0 is a local minimum, there exists $\epsilon > 0$ such that if $\|x - x_0\| \leq \epsilon$, then $f(x_0) \leq f(x)$
- Take a convex combination $y = \lambda x^* + (1 - \lambda)x_0$. Note that:

$$\|y - x_0\| = \|\lambda(x^* - x_0)\| = \lambda\|x^* - x_0\|$$

so for sufficiently small λ we will have $\|y - x_0\| \leq \epsilon$, and hence $f(x_0) \leq f(y)$

- On the other hand, from the convexity of $f(x)$:

$$\begin{aligned} f(y) &\leq \lambda f(x^*) + (1 - \lambda)f(x_0) \\ &< \lambda f(x_0) + (1 - \lambda)f(x_0) = f(x_0) \end{aligned}$$

contradiction!

Digression: concave functions and maximization

A function $f(x)$ is **concave** if $-f(x)$ is convex

Theorem: Consider the optimization problem:

$$\begin{aligned} \max \quad & f(x) \\ \text{s.t.} \quad & x \in \mathcal{X}, \end{aligned}$$

in which the function $f(x)$ is **concave**, while the set of feasible solutions \mathcal{X} is **convex**. Then every local maximum of $f(x)$ is also its global maximum

Does there exist a function both convex and concave?

Does there exist a function both convex and concave?

For a **convex** function, for any $\mathbf{x}, \mathbf{y}, \lambda \in [0, 1]$:

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

For a **concave** function, for any $\mathbf{x}, \mathbf{y}, \lambda \in [0, 1]$:

$$f(\lambda\mathbf{x} + (1 - \lambda)\mathbf{y}) \geq \lambda f(\mathbf{x}) + (1 - \lambda)f(\mathbf{y})$$

Does there exist a function both convex and concave?

For a **convex** function, for any $\mathbf{x}, \mathbf{y}, \lambda \in [0, 1]$:

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

For a **concave** function, for any $\mathbf{x}, \mathbf{y}, \lambda \in [0, 1]$:

$$f(\lambda\mathbf{x} + (1 - \lambda)\mathbf{y}) \geq \lambda f(\mathbf{x}) + (1 - \lambda)f(\mathbf{y})$$

Conclusion: If $f(\mathbf{x})$ is both convex and concave, then for any $\mathbf{x}, \mathbf{y}, \lambda \in [0, 1]$ it holds:

$$f(\lambda\mathbf{x} + (1 - \lambda)\mathbf{y}) = \lambda f(\mathbf{x}) + (1 - \lambda)f(\mathbf{y}),$$

so $f(\mathbf{x})$ is a **linear function**:

$$f(\mathbf{x}) = \mathbf{b}^\top \mathbf{x} + c$$

Differentiable convex functions

If a function $f(\mathbf{x})$ is **differentiable**, then it is **convex** if and only if for any \mathbf{x}, \mathbf{x}_0 it holds:

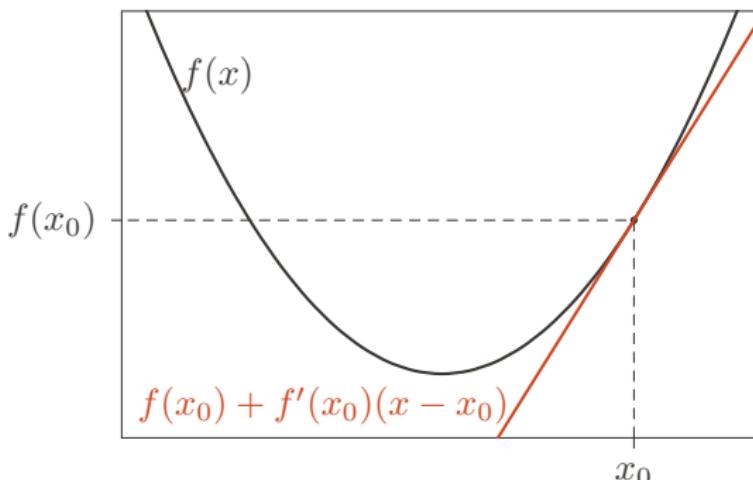
$$f(\mathbf{x}) \geq f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0)$$

Differentiable convex functions

If a function $f(x)$ is **differentiable**, then it is **convex** if and only if for any x, x_0 it holds:

$$f(x) \geq f(x_0) + \nabla f(x_0)^\top (x - x_0)$$

Interpretation: for a univariate function the condition above reduces to $f(x) \geq f(x_0) + f'(x_0)(x - x_0)$, i.e., the tangent to the graph is always below (or at) the graph.

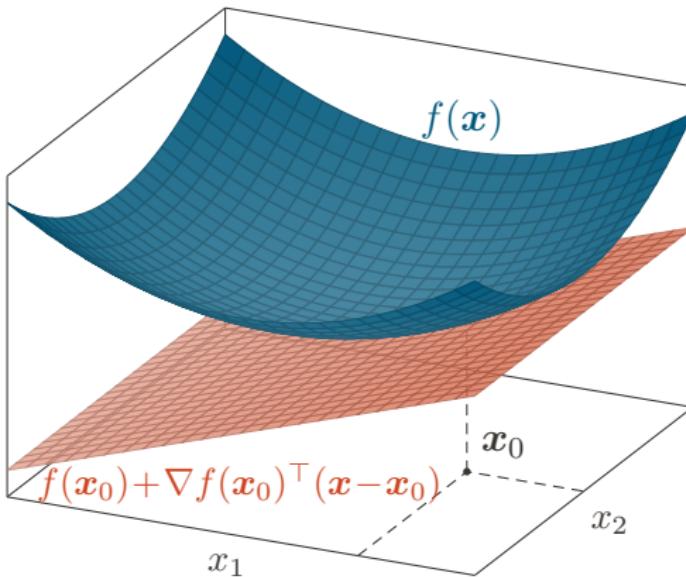


Differentiable convex functions

If a function $f(\mathbf{x})$ is **differentiable**, then it is **convex** if and only if for any \mathbf{x}, \mathbf{x}_0 it holds:

$$f(\mathbf{x}) \geq f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0)$$

Interpretation: in general, the tangent hyperplane is below (or at) the graph



Differentiable convex functions

If a function $f(x)$ is **differentiable**, then it is **convex** if and only if for any x, x_0 it holds:

$$f(x) \geq f(x_0) + \nabla f(x_0)^\top (x - x_0)$$

Proof: in both directions

Differentiable convex functions

If a function $f(\mathbf{x})$ is **differentiable**, then it is **convex** if and only if for any \mathbf{x}, \mathbf{x}_0 it holds:

$$f(\mathbf{x}) \geq f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0)$$

Proof: in both directions

\Rightarrow Assume that $f(\mathbf{x})$ is convex, i.e. for any $\lambda \in [0, 1]$:

$$\underbrace{f(\lambda \mathbf{x} + (1 - \lambda) \mathbf{x}_0)}_{f(\mathbf{x}_0 + \lambda(\mathbf{x} - \mathbf{x}_0))} \leq \underbrace{\lambda f(\mathbf{x}) + (1 - \lambda) f(\mathbf{x}_0)}_{f(\mathbf{x}_0) + \lambda(f(\mathbf{x}) - f(\mathbf{x}_0))}$$

Differentiable convex functions

If a function $f(\mathbf{x})$ is **differentiable**, then it is **convex** if and only if for any \mathbf{x}, \mathbf{x}_0 it holds:

$$f(\mathbf{x}) \geq f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0)$$

Proof: in both directions

⇒ Assume that $f(\mathbf{x})$ is convex, i.e. for any $\lambda \in [0, 1]$:

$$\underbrace{f(\lambda \mathbf{x} + (1 - \lambda) \mathbf{x}_0)}_{f(\mathbf{x}_0 + \lambda(\mathbf{x} - \mathbf{x}_0))} \leq \underbrace{\lambda f(\mathbf{x}) + (1 - \lambda) f(\mathbf{x}_0)}_{f(\mathbf{x}_0) + \lambda(f(\mathbf{x}) - f(\mathbf{x}_0))}$$

Therefore:

$$\frac{f(\mathbf{x}_0 + \lambda(\mathbf{x} - \mathbf{x}_0)) - f(\mathbf{x}_0)}{\lambda} \leq f(\mathbf{x}) - f(\mathbf{x}_0)$$

Differentiable convex functions

If a function $f(\mathbf{x})$ is **differentiable**, then it is **convex** if and only if for any \mathbf{x}, \mathbf{x}_0 it holds:

$$f(\mathbf{x}) \geq f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0)$$

Proof: in both directions

⇒ Assume that $f(\mathbf{x})$ is convex, i.e. for any $\lambda \in [0, 1]$:

$$\underbrace{f(\lambda\mathbf{x} + (1 - \lambda)\mathbf{x}_0)}_{f(\mathbf{x}_0 + \lambda(\mathbf{x} - \mathbf{x}_0))} \leq \underbrace{\lambda f(\mathbf{x}) + (1 - \lambda)f(\mathbf{x}_0)}_{f(\mathbf{x}_0) + \lambda(f(\mathbf{x}) - f(\mathbf{x}_0))}$$

Therefore:

$$\lim_{\lambda \rightarrow 0} \frac{f(\mathbf{x}_0 + \lambda(\mathbf{x} - \mathbf{x}_0)) - f(\mathbf{x}_0)}{\lambda} \leq f(\mathbf{x}) - f(\mathbf{x}_0)$$

Differentiable convex functions

If a function $f(\mathbf{x})$ is **differentiable**, then it is **convex** if and only if for any \mathbf{x}, \mathbf{x}_0 it holds:

$$f(\mathbf{x}) \geq f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0)$$

Proof: in both directions

⇒ Assume that $f(\mathbf{x})$ is convex, i.e. for any $\lambda \in [0, 1]$:

$$\underbrace{f(\lambda\mathbf{x} + (1 - \lambda)\mathbf{x}_0)}_{f(\mathbf{x}_0 + \lambda(\mathbf{x} - \mathbf{x}_0))} \leq \underbrace{\lambda f(\mathbf{x}) + (1 - \lambda)f(\mathbf{x}_0)}_{f(\mathbf{x}_0) + \lambda(f(\mathbf{x}) - f(\mathbf{x}_0))}$$

Therefore:

$$\lim_{\lambda \rightarrow 0} \underbrace{\frac{f(\mathbf{x}_0 + \lambda(\mathbf{x} - \mathbf{x}_0)) - f(\mathbf{x}_0)}{\lambda}}_{\text{directional derivative along vector } \mathbf{x} - \mathbf{x}_0} \leq f(\mathbf{x}) - f(\mathbf{x}_0)$$

Differentiable convex functions

If a function $f(\mathbf{x})$ is **differentiable**, then it is **convex** if and only if for any \mathbf{x}, \mathbf{x}_0 it holds:

$$f(\mathbf{x}) \geq f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0)$$

Proof: in both directions

\Rightarrow Assume that $f(\mathbf{x})$ is convex, i.e. for any $\lambda \in [0, 1]$:

$$\underbrace{f(\lambda \mathbf{x} + (1 - \lambda) \mathbf{x}_0)}_{f(\mathbf{x}_0 + \lambda(\mathbf{x} - \mathbf{x}_0))} \leq \underbrace{\lambda f(\mathbf{x}) + (1 - \lambda) f(\mathbf{x}_0)}_{f(\mathbf{x}_0) + \lambda(f(\mathbf{x}) - f(\mathbf{x}_0))}$$

Therefore:

$$\nabla f(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0) \leq f(\mathbf{x}) - f(\mathbf{x}_0)$$

which proves the inequality in the box

Differentiable convex functions

If a function $f(\mathbf{x})$ is **differentiable**, then it is **convex** if and only if for any \mathbf{x}, \mathbf{x}_0 it holds:

$$f(\mathbf{x}) \geq f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0)$$

Proof: in both directions

\Leftarrow Assume the inequality in the box holds and we prove that the function $f(\mathbf{x})$ is convex

Take any $\mathbf{x}_1, \mathbf{x}_2, \lambda \in [0, 1]$, and let $\mathbf{x}_0 = \lambda \mathbf{x}_1 + (1 - \lambda) \mathbf{x}_2$

We have: $\mathbf{x}_1 - \mathbf{x}_0 = (1 - \lambda)(\mathbf{x}_1 - \mathbf{x}_2)$, $\mathbf{x}_2 - \mathbf{x}_0 = \lambda(\mathbf{x}_2 - \mathbf{x}_1)$

Differentiable convex functions

If a function $f(\mathbf{x})$ is **differentiable**, then it is **convex** if and only if for any \mathbf{x}, \mathbf{x}_0 it holds:

$$f(\mathbf{x}) \geq f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0)$$

Proof: in both directions

\Leftarrow Assume the inequality in the box holds and we prove that the function $f(\mathbf{x})$ is convex

Take any $\mathbf{x}_1, \mathbf{x}_2, \lambda \in [0, 1]$, and let $\mathbf{x}_0 = \lambda \mathbf{x}_1 + (1 - \lambda) \mathbf{x}_2$

We have: $\mathbf{x}_1 - \mathbf{x}_0 = (1 - \lambda)(\mathbf{x}_1 - \mathbf{x}_2)$, $\mathbf{x}_2 - \mathbf{x}_0 = \lambda(\mathbf{x}_2 - \mathbf{x}_1)$

Using the inequality in the box **twice** for $\mathbf{x} = \mathbf{x}_1$ and $\mathbf{x} = \mathbf{x}_2$:

$$f(\mathbf{x}_1) \geq f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top (\mathbf{x}_1 - \mathbf{x}_0)$$

$$f(\mathbf{x}_2) \geq f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top (\mathbf{x}_2 - \mathbf{x}_0)$$

Differentiable convex functions

If a function $f(\mathbf{x})$ is **differentiable**, then it is **convex** if and only if for any \mathbf{x}, \mathbf{x}_0 it holds:

$$f(\mathbf{x}) \geq f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0)$$

Proof: in both directions

\Leftarrow Assume the inequality in the box holds and we prove that the function $f(\mathbf{x})$ is convex

Take any $\mathbf{x}_1, \mathbf{x}_2, \lambda \in [0, 1]$, and let $\mathbf{x}_0 = \lambda \mathbf{x}_1 + (1 - \lambda) \mathbf{x}_2$

We have: $\mathbf{x}_1 - \mathbf{x}_0 = (1 - \lambda)(\mathbf{x}_1 - \mathbf{x}_2)$, $\mathbf{x}_2 - \mathbf{x}_0 = \lambda(\mathbf{x}_2 - \mathbf{x}_1)$

Using the inequality in the box **twice** for $\mathbf{x} = \mathbf{x}_1$ and $\mathbf{x} = \mathbf{x}_2$:

$$f(\mathbf{x}_1) \geq f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top (1 - \lambda)(\mathbf{x}_1 - \mathbf{x}_2)$$

$$f(\mathbf{x}_2) \geq f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top \lambda(\mathbf{x}_2 - \mathbf{x}_1)$$

Differentiable convex functions

If a function $f(\mathbf{x})$ is **differentiable**, then it is **convex** if and only if for any \mathbf{x}, \mathbf{x}_0 it holds:

$$f(\mathbf{x}) \geq f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0)$$

Proof: in both directions

\Leftarrow Assume the inequality in the box holds and we prove that the function $f(\mathbf{x})$ is convex

Take any $\mathbf{x}_1, \mathbf{x}_2, \lambda \in [0, 1]$, and let $\mathbf{x}_0 = \lambda \mathbf{x}_1 + (1 - \lambda) \mathbf{x}_2$

We have: $\mathbf{x}_1 - \mathbf{x}_0 = (1 - \lambda)(\mathbf{x}_1 - \mathbf{x}_2)$, $\mathbf{x}_2 - \mathbf{x}_0 = \lambda(\mathbf{x}_2 - \mathbf{x}_1)$

Using the inequality in the box **twice** for $\mathbf{x} = \mathbf{x}_1$ and $\mathbf{x} = \mathbf{x}_2$:

$$\lambda f(\mathbf{x}_1) \geq f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top (1 - \lambda)(\mathbf{x}_1 - \mathbf{x}_2)$$

$$(1 - \lambda)f(\mathbf{x}_2) \geq f(\mathbf{x}_0) + (1 - \lambda)\nabla f(\mathbf{x}_0)^\top \lambda(\mathbf{x}_2 - \mathbf{x}_1)$$

Differentiable convex functions

If a function $f(\mathbf{x})$ is **differentiable**, then it is **convex** if and only if for any \mathbf{x}, \mathbf{x}_0 it holds:

$$f(\mathbf{x}) \geq f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0)$$

Proof: in both directions

\Leftarrow Assume the inequality in the box holds and we prove that the function $f(\mathbf{x})$ is convex

Take any $\mathbf{x}_1, \mathbf{x}_2, \lambda \in [0, 1]$, and let $\mathbf{x}_0 = \lambda \mathbf{x}_1 + (1 - \lambda) \mathbf{x}_2$

We have: $\mathbf{x}_1 - \mathbf{x}_0 = (1 - \lambda)(\mathbf{x}_1 - \mathbf{x}_2)$, $\mathbf{x}_2 - \mathbf{x}_0 = \lambda(\mathbf{x}_2 - \mathbf{x}_1)$

Using the inequality in the box **twice** for $\mathbf{x} = \mathbf{x}_1$ and $\mathbf{x} = \mathbf{x}_2$:

$$\lambda f(\mathbf{x}_1) \geq f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top (1 - \lambda)(\mathbf{x}_1 - \mathbf{x}_2)$$

$$(1 - \lambda)f(\mathbf{x}_2) \geq f(\mathbf{x}_0) + (1 - \lambda)\nabla f(\mathbf{x}_0)^\top \lambda(\mathbf{x}_2 - \mathbf{x}_1)$$

Summing both inequalities, the term with gradient vanishes, giving:

$$\lambda f(\mathbf{x}_1) + (1 - \lambda)f(\mathbf{x}_2) \geq f(\mathbf{x}_0),$$

so $f(\mathbf{x})$ is convex

Differentiable convex functions

Let $f(\mathbf{x})$ be twice differentiable with continuous second derivatives. Then $f(\mathbf{x})$ is convex if and only if its Hessian $\nabla^2 f(\mathbf{x})$ is **positive semi-definite** at any \mathbf{x}

- Positive semi-definiteness means that for any vector \mathbf{v} ,

$$\mathbf{v}^\top \nabla^2 f(\mathbf{x}) \mathbf{v} \geq 0$$

- For a univariate function this condition reduces to:

$$f''(x) \geq 0$$

Differentiable convex functions

Let $f(\mathbf{x})$ be twice differentiable with continuous second derivatives. Then $f(\mathbf{x})$ is convex if and only if its Hessian $\nabla^2 f(\mathbf{x})$ is **positive semi-definite** at any \mathbf{x}

Proof: in both directions

Differentiable convex functions

Let $f(\mathbf{x})$ be twice differentiable with continuous second derivatives. Then $f(\mathbf{x})$ is convex if and only if its Hessian $\nabla^2 f(\mathbf{x})$ is **positive semi-definite** at any \mathbf{x}

Proof: in both directions

\Leftarrow Assume the Hessian is positive semi-definite. From the Taylor's theorem:

$$f(\mathbf{x}) = f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0) + \frac{1}{2} \underbrace{(\mathbf{x} - \mathbf{x}_0)^\top \nabla^2 f(\boldsymbol{\xi})(\mathbf{x} - \mathbf{x}_0)}_{\geq 0 \text{ (positive semi-def.)}}$$

Differentiable convex functions

Let $f(\mathbf{x})$ be twice differentiable with continuous second derivatives. Then $f(\mathbf{x})$ is convex if and only if its Hessian $\nabla^2 f(\mathbf{x})$ is **positive semi-definite** at any \mathbf{x}

Proof: in both directions

\Leftarrow Assume the Hessian is positive semi-definite. From the Taylor's theorem:

$$\begin{aligned}f(\mathbf{x}) &= f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0) + \frac{1}{2} \underbrace{(\mathbf{x} - \mathbf{x}_0)^\top \nabla^2 f(\boldsymbol{\xi})(\mathbf{x} - \mathbf{x}_0)}_{\geq 0 \text{ (positive semi-def.)}} \\&\geq f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0),\end{aligned}$$

so the function is convex from the previous theorem

Differentiable convex functions

Let $f(\mathbf{x})$ be twice differentiable with continuous second derivatives. Then $f(\mathbf{x})$ is convex if and only if its Hessian $\nabla^2 f(\mathbf{x})$ is **positive semi-definite** at any \mathbf{x}

Proof: in both directions

⇒ Let $f(\mathbf{x})$ be convex. Assume the contrary, that there exists \mathbf{x}_0 and vector \mathbf{v} such that:

$$\mathbf{v}^\top \nabla^2 f(\mathbf{x}_0) \mathbf{v} < 0$$

Differentiable convex functions

Let $f(\mathbf{x})$ be twice differentiable with continuous second derivatives. Then $f(\mathbf{x})$ is convex if and only if its Hessian $\nabla^2 f(\mathbf{x})$ is **positive semi-definite** at any \mathbf{x}

Proof: in both directions

⇒ Let $f(\mathbf{x})$ be convex. Assume the contrary, that there exists \mathbf{x}_0 and vector \mathbf{v} such that:

$$\mathbf{v}^\top \nabla^2 f(\mathbf{x}_0) \mathbf{v} < 0$$

From the continuity of Hessian the above also holds within a small neighborhood \mathcal{S} of \mathbf{x}_0 . Take \mathbf{v} with the length sufficiently small to have $\mathbf{x}_0 + \mathbf{v} \in \mathcal{S}$

Differentiable convex functions

Let $f(\mathbf{x})$ be twice differentiable with continuous second derivatives. Then $f(\mathbf{x})$ is convex if and only if its Hessian $\nabla^2 f(\mathbf{x})$ is **positive semi-definite** at any \mathbf{x}

Proof: in both directions

⇒ Let $f(\mathbf{x})$ be convex. Assume the contrary, that there exists \mathbf{x}_0 and vector \mathbf{v} such that:

$$\mathbf{v}^\top \nabla^2 f(\mathbf{x}_0) \mathbf{v} < 0$$

From the continuity of Hessian the above also holds within a small neighborhood \mathcal{S} of \mathbf{x}_0 . Take \mathbf{v} with the length sufficiently small to have $\mathbf{x}_0 + \mathbf{v} \in \mathcal{S}$

From the Taylor's theorem for $\mathbf{x} = \mathbf{x}_0 + \mathbf{v}$:

$$\begin{aligned} f(\mathbf{x}) &= f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0) + \frac{1}{2} \underbrace{(\mathbf{x} - \mathbf{x}_0)^\top}_{\mathbf{v}} \underbrace{\nabla^2 f(\boldsymbol{\xi})}_{<0} \underbrace{(\mathbf{x} - \mathbf{x}_0)}_{\mathbf{v}} \\ &< f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0), \end{aligned}$$

because $\boldsymbol{\xi}$ lies in between \mathbf{x}_0 and $\mathbf{x}_0 + \mathbf{v}$, so $\boldsymbol{\xi} \in \mathcal{S}$.

Differentiable convex functions

Let $f(\mathbf{x})$ be twice differentiable with continuous second derivatives. Then $f(\mathbf{x})$ is convex if and only if its Hessian $\nabla^2 f(\mathbf{x})$ is **positive semi-definite** at any \mathbf{x}

Proof: in both directions

⇒ Let $f(\mathbf{x})$ be convex. Assume the contrary, that there exists \mathbf{x}_0 and vector \mathbf{v} such that:

$$\mathbf{v}^\top \nabla^2 f(\mathbf{x}_0) \mathbf{v} < 0$$

From the continuity of Hessian the above also holds within a small neighborhood \mathcal{S} of \mathbf{x}_0 . Take \mathbf{v} with the length sufficiently small to have $\mathbf{x}_0 + \mathbf{v} \in \mathcal{S}$

From the Taylor's theorem for $\mathbf{x} = \mathbf{x}_0 + \mathbf{v}$:

$$\begin{aligned} f(\mathbf{x}) &= f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0) + \frac{1}{2} \underbrace{(\mathbf{x} - \mathbf{x}_0)^\top}_{\mathbf{v}} \underbrace{\nabla^2 f(\boldsymbol{\xi})(\mathbf{x} - \mathbf{x}_0)}_{\mathbf{v}} < 0 \\ &< f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0), \end{aligned}$$

because $\boldsymbol{\xi}$ lies in between \mathbf{x}_0 and $\mathbf{x}_0 + \mathbf{v}$, so $\boldsymbol{\xi} \in \mathcal{S}$. Thus, the function does not satisfy the previous theorem, and so it is not convex

Differentiable convex functions

Let $f(\mathbf{x})$ be twice differentiable with continuous second derivatives. Then $f(\mathbf{x})$ is convex if and only if its Hessian $\nabla^2 f(\mathbf{x})$ is **positive semi-definite** at any \mathbf{x}

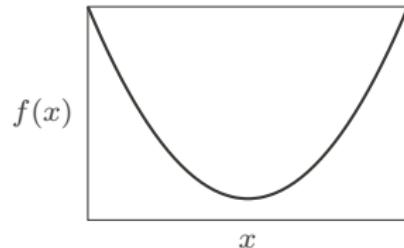
Let $f(\mathbf{x})$ be twice differentiable with continuous second derivatives. If its Hessian $\nabla^2 f(\mathbf{x})$ is **positive-definite** for any \mathbf{x} , the function is **strictly convex**. The converse statement does not hold in general.

Examples of convex function (univariate)

Quadratic function

$$f(x) = ax^2 + bx + c \text{ for } a \geq 0$$

$$f'(x) = 2ax + b, \quad f''(x) = 2a \geq 0$$

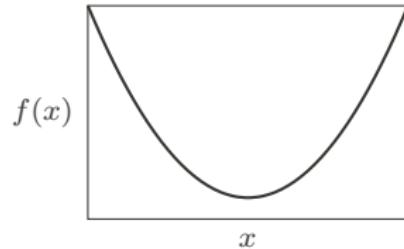


Examples of convex function (univariate)

Quadratic function

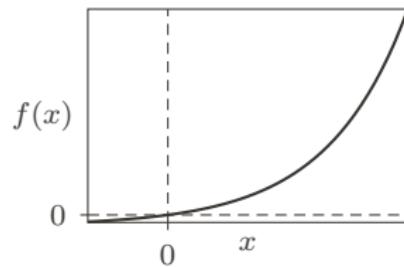
$$f(x) = ax^2 + bx + c \text{ for } a \geq 0$$

$$f'(x) = 2ax + b, \quad f''(x) = 2a \geq 0$$



Exponential function $f(x) = e^x$

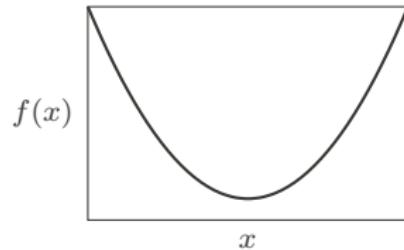
$$f'(x) = e^x, \quad f''(x) = e^x > 0$$



Examples of convex function (univariate)

Quadratic function

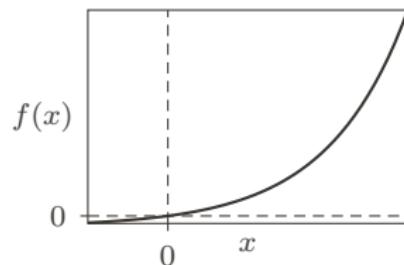
$$f(x) = ax^2 + bx + c \text{ for } a \geq 0$$



$$f'(x) = 2ax + b, \quad f''(x) = 2a \geq 0$$

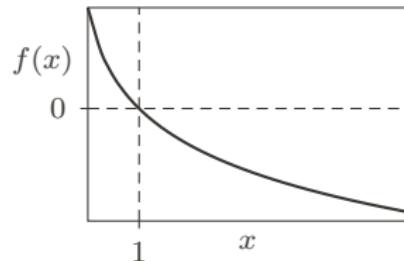
Exponential function $f(x) = e^x$

$$f'(x) = e^x, \quad f''(x) = e^x > 0$$



Negative logarithm $f(x) = -\ln(x)$
(for $x > 0$)

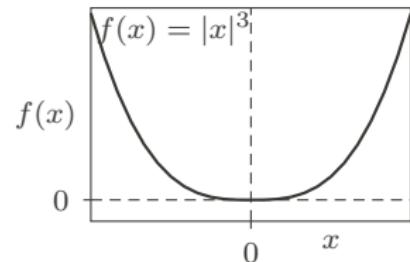
$$f'(x) = -\frac{1}{x}, \quad f''(x) = \frac{1}{x^2} > 0$$



Examples of convex functions (univariate)

Function $f(x) = |x|^\alpha$ for $\alpha \geq 1$
(in particular $f(x) = |x|$)

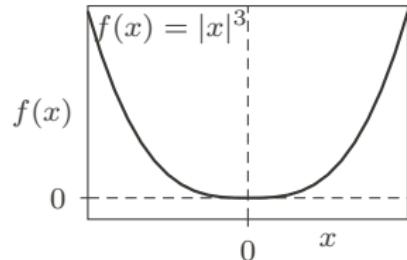
Non-differentiable



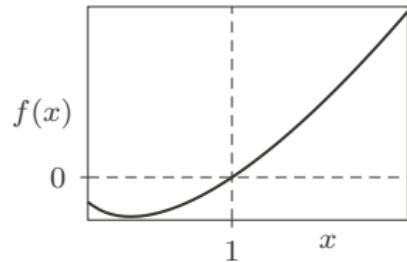
Examples of convex functions (univariate)

Function $f(x) = |x|^\alpha$ for $\alpha \geq 1$
(in particular $f(x) = |x|$)

Non-differentiable



Function $f(x) = x \ln x$ for $x > 0$

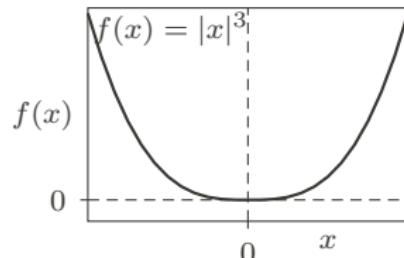


$$f'(x) = \ln x + 1, \quad f''(x) = \frac{1}{x} > 0$$

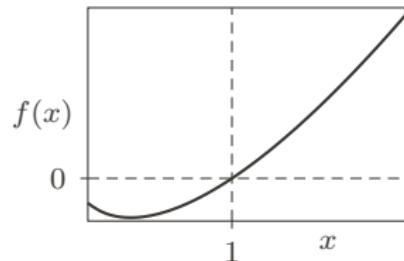
Examples of convex functions (univariate)

Function $f(x) = |x|^\alpha$ for $\alpha \geq 1$
(in particular $f(x) = |x|$)

Non-differentiable

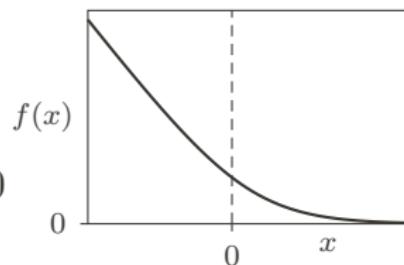


Function $f(x) = x \ln x$ for $x > 0$



Function $f(x) = \ln(1 + e^{-x})$

$f'(x) = -\frac{1}{1 + e^x}$, $f''(x) = \frac{e^x}{(1 + e^x)^2} > 0$



Geometric and arithmetic mean

From the convexity of negative logarithm, for any x_1, \dots, x_n i

$\lambda_1 = \dots = \lambda_n = \frac{1}{n}$:

$$-\ln\left(\frac{1}{n}x_1 + \dots + \frac{1}{n}x_n\right) \leq \frac{1}{n}(-\ln(x_1)) + \dots + \frac{1}{n}(-\ln(x_n))$$

Geometric and arithmetic mean

From the convexity of negative logarithm, for any x_1, \dots, x_n i

$\lambda_1 = \dots = \lambda_n = \frac{1}{n}$:

$$-\ln\left(\frac{1}{n}x_1 + \dots + \frac{1}{n}x_n\right) \leq \frac{1}{n}(-\ln(x_1)) + \dots + \frac{1}{n}(-\ln(x_n))$$

After merging all logarithms on the right-hand side:

$$-\ln\left(\frac{1}{n}\sum_{i=1}^n x_i\right) \leq -\ln\left(\left(\prod_{i=1}^n x_i\right)^{1/n}\right)$$

Geometric and arithmetic mean

From the convexity of negative logarithm, for any x_1, \dots, x_n i

$\lambda_1 = \dots = \lambda_n = \frac{1}{n}$:

$$-\ln\left(\frac{1}{n}x_1 + \dots + \frac{1}{n}x_n\right) \leq \frac{1}{n}(-\ln(x_1)) + \dots + \frac{1}{n}(-\ln(x_n))$$

After merging all logarithms on the right-hand side:

$$-\ln\left(\frac{1}{n}\sum_{i=1}^n x_i\right) \leq -\ln\left(\left(\prod_{i=1}^n x_i\right)^{1/n}\right)$$

After multiplying by -1 and dropping the logarithms:

$$\frac{1}{n}\sum_{i=1}^n x_i \geq \left(\prod_{i=1}^n x_i\right)^{1/n}$$

We obtained a known theorem: the **arithmetic mean** is **not smaller** than the **geometric mean**

Examples of convex functions (multivariate)

Any **norm** $f(x) = \|x\|$ (e.g., Euclidean) is a convex function

Examples of convex functions (multivariate)

Any **norm** $f(\mathbf{x}) = \|\mathbf{x}\|$ (e.g., Euclidean) is a convex function

Proof: From the triangle inequality $\|\mathbf{u} + \mathbf{v}\| \leq \|\mathbf{u}\| + \|\mathbf{v}\|$ and scalability $\|\alpha\mathbf{v}\| = \alpha\|\mathbf{v}\|$ (for $\alpha \geq 0$):

$$f(\lambda\mathbf{x} + (1 - \lambda)\mathbf{y}) = \|\lambda\mathbf{x} + (1 - \lambda)\mathbf{y}\|$$

Examples of convex functions (multivariate)

Any **norm** $f(\mathbf{x}) = \|\mathbf{x}\|$ (e.g., Euclidean) is a convex function

Proof: From the triangle inequality $\|\mathbf{u} + \mathbf{v}\| \leq \|\mathbf{u}\| + \|\mathbf{v}\|$ and scalability $\|\alpha\mathbf{v}\| = \alpha\|\mathbf{v}\|$ (for $\alpha \geq 0$):

$$\begin{aligned} f(\lambda\mathbf{x} + (1 - \lambda)\mathbf{y}) &= \|\lambda\mathbf{x} + (1 - \lambda)\mathbf{y}\| \\ &\leq \|\lambda\mathbf{x}\| + \|(1 - \lambda)\mathbf{y}\| \end{aligned}$$

Examples of convex functions (multivariate)

Any **norm** $f(\mathbf{x}) = \|\mathbf{x}\|$ (e.g., Euclidean) is a convex function

Proof: From the triangle inequality $\|\mathbf{u} + \mathbf{v}\| \leq \|\mathbf{u}\| + \|\mathbf{v}\|$ and scalability $\|\alpha\mathbf{v}\| = \alpha\|\mathbf{v}\|$ (for $\alpha \geq 0$):

$$\begin{aligned} f(\lambda\mathbf{x} + (1 - \lambda)\mathbf{y}) &= \|\lambda\mathbf{x} + (1 - \lambda)\mathbf{y}\| \\ &\leq \|\lambda\mathbf{x}\| + \|(1 - \lambda)\mathbf{y}\| \\ &= \lambda\|\mathbf{x}\| + (1 - \lambda)\|\mathbf{y}\| \end{aligned}$$

Examples of convex functions (multivariate)

Any **norm** $f(\mathbf{x}) = \|\mathbf{x}\|$ (e.g., Euclidean) is a convex function

Proof: From the triangle inequality $\|\mathbf{u} + \mathbf{v}\| \leq \|\mathbf{u}\| + \|\mathbf{v}\|$ and scalability $\|\alpha\mathbf{v}\| = \alpha\|\mathbf{v}\|$ (for $\alpha \geq 0$):

$$\begin{aligned} f(\lambda\mathbf{x} + (1 - \lambda)\mathbf{y}) &= \|\lambda\mathbf{x} + (1 - \lambda)\mathbf{y}\| \\ &\leq \|\lambda\mathbf{x}\| + \|(1 - \lambda)\mathbf{y}\| \\ &= \lambda\|\mathbf{x}\| + (1 - \lambda)\|\mathbf{y}\| \\ &= \lambda f(\mathbf{x}) + (1 - \lambda)f(\mathbf{y}) \end{aligned}$$

Examples of convex functions (multivariate)

Maximum $f(x) = \max_{i=1,\dots,n} \{x_i\}$ is a convex function

Examples of convex functions (multivariate)

Maximum $f(\mathbf{x}) = \max_{i=1,\dots,n} \{x_i\}$ is a convex function

Proof:

$$f(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y}) = \max_i \{\lambda x_i + (1 - \lambda) y_i\}$$

Examples of convex functions (multivariate)

Maximum $f(\mathbf{x}) = \max_{i=1,\dots,n} \{x_i\}$ is a convex function

Proof:

$$\begin{aligned} f(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y}) &= \max_i \{\lambda x_i + (1 - \lambda) y_i\} \\ &\leq \max_i \{\lambda x_i\} + \max_i \{(1 - \lambda) y_i\} \end{aligned}$$

Examples of convex functions (multivariate)

Maximum $f(\mathbf{x}) = \max_{i=1,\dots,n} \{x_i\}$ is a convex function

Proof:

$$\begin{aligned} f(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y}) &= \max_i \{\lambda x_i + (1 - \lambda) y_i\} \\ &\leq \max_i \{\lambda x_i\} + \max_i \{(1 - \lambda) y_i\} \\ &= \lambda \max_i \{x_i\} + (1 - \lambda) \max_i \{y_i\} \end{aligned}$$

Examples of convex functions (multivariate)

Maximum $f(\mathbf{x}) = \max_{i=1,\dots,n} \{x_i\}$ is a convex function

Proof:

$$\begin{aligned} f(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y}) &= \max_i \{\lambda x_i + (1 - \lambda) y_i\} \\ &\leq \max_i \{\lambda x_i\} + \max_i \{(1 - \lambda) y_i\} \\ &= \lambda \max_i \{x_i\} + (1 - \lambda) \max_i \{y_i\} \\ &= \lambda f(\mathbf{x}) + (1 - \lambda) f(\mathbf{y}) \end{aligned}$$

Examples of convex functions (multivariate)

A **quadratic function** $f(x) = x^\top Ax + b^\top x + c$ for any **positive semi-definite** matrix A is convex

Examples of convex functions (multivariate)

A **quadratic function** $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A}\mathbf{x} + \mathbf{b}^\top \mathbf{x} + c$ for any **positive semi-definite** matrix \mathbf{A} is convex

Proof: Follows from $\nabla^2 f(\mathbf{x}) = 2\mathbf{A}$ (we already calculated it!)

Examples of convex functions (multivariate)

A **quadratic function** $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A}\mathbf{x} + \mathbf{b}^\top \mathbf{x} + c$ for any **positive semi-definite** matrix \mathbf{A} is convex

Proof: Follows from $\nabla^2 f(\mathbf{x}) = 2\mathbf{A}$ (we already calculated it!)

Example 1: Linear function $f(\mathbf{x}) = \mathbf{b}^\top \mathbf{x} + c$ is convex

Examples of convex functions (multivariate)

A **quadratic function** $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A}\mathbf{x} + \mathbf{b}^\top \mathbf{x} + c$ for any **positive semi-definite** matrix \mathbf{A} is convex

Proof: Follows from $\nabla^2 f(\mathbf{x}) = 2\mathbf{A}$ (we already calculated it!)

Example 1: Linear function $f(\mathbf{x}) = \mathbf{b}^\top \mathbf{x} + c$ is convex

We have $\mathbf{A} = \mathbf{0}$, which is a positive semi-definite matrix, since for any vector \mathbf{v} :

$$\mathbf{v}^\top \mathbf{A}\mathbf{v} = \mathbf{0} \geqslant 0$$

Examples of convex functions (multivariate)

A **quadratic function** $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A}\mathbf{x} + \mathbf{b}^\top \mathbf{x} + c$ for any **positive semi-definite** matrix \mathbf{A} is convex

Proof: Follows from $\nabla^2 f(\mathbf{x}) = 2\mathbf{A}$ (we already calculated it!)

Example 2: Function $f(\mathbf{x}) = \|\mathbf{x}\|^2$ is convex

Examples of convex functions (multivariate)

A **quadratic function** $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A}\mathbf{x} + \mathbf{b}^\top \mathbf{x} + c$ for any **positive semi-definite** matrix \mathbf{A} is convex

Proof: Follows from $\nabla^2 f(\mathbf{x}) = 2\mathbf{A}$ (we already calculated it!)

Example 2: Function $f(\mathbf{x}) = \|\mathbf{x}\|^2$ is convex

Since $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{x} = \mathbf{x}^\top \mathbf{I}\mathbf{x}$, we have $\mathbf{A} = \mathbf{I}$

\mathbf{I} is positive-definite: for any non-zero vector \mathbf{v}

$$\mathbf{v}^\top \mathbf{I}\mathbf{v} = \|\mathbf{v}\|^2 > 0$$

Examples of convex functions (multivariate)

A **quadratic function** $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A}\mathbf{x} + \mathbf{b}^\top \mathbf{x} + c$ for any **positive semi-definite** matrix \mathbf{A} is convex

Proof: Follows from $\nabla^2 f(\mathbf{x}) = 2\mathbf{A}$ (we already calculated it!)

Example 3: Function $f(\mathbf{x}) = (y - \mathbf{a}^\top \mathbf{x})^2$, $y \in \mathbb{R}$, $\mathbf{a}, \mathbf{x} \in \mathbb{R}^n$ is convex

Examples of convex functions (multivariate)

A **quadratic function** $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A}\mathbf{x} + \mathbf{b}^\top \mathbf{x} + c$ for any **positive semi-definite** matrix \mathbf{A} is convex

Proof: Follows from $\nabla^2 f(\mathbf{x}) = 2\mathbf{A}$ (we already calculated it!)

Example 3: Function $f(\mathbf{x}) = (y - \mathbf{a}^\top \mathbf{x})^2$, $y \in \mathbb{R}$, $\mathbf{a}, \mathbf{x} \in \mathbb{R}^n$ is convex

$$f(\mathbf{x}) = y^2 - 2y\mathbf{a}^\top \mathbf{x} + (\mathbf{a}^\top \mathbf{x})^2 = \underbrace{y^2}_c + \underbrace{(-2y\mathbf{a})^\top}_{\mathbf{b}} \mathbf{x} + \mathbf{x}^\top \underbrace{\mathbf{a}\mathbf{a}^\top}_A \mathbf{x}$$

Examples of convex functions (multivariate)

A **quadratic function** $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A}\mathbf{x} + \mathbf{b}^\top \mathbf{x} + c$ for any **positive semi-definite** matrix \mathbf{A} is convex

Proof: Follows from $\nabla^2 f(\mathbf{x}) = 2\mathbf{A}$ (we already calculated it!)

Example 3: Function $f(\mathbf{x}) = (y - \mathbf{a}^\top \mathbf{x})^2$, $y \in \mathbb{R}$, $\mathbf{a}, \mathbf{x} \in \mathbb{R}^n$ is convex

$$f(\mathbf{x}) = y^2 - 2y\mathbf{a}^\top \mathbf{x} + (\mathbf{a}^\top \mathbf{x})^2 = \underbrace{y^2}_c + \underbrace{(-2y\mathbf{a})^\top}_{\mathbf{b}} \mathbf{x} + \mathbf{x}^\top \underbrace{\mathbf{a}\mathbf{a}^\top}_A \mathbf{x}$$

$\mathbf{A} = \mathbf{a}\mathbf{a}^\top \in \mathbb{R}^{n \times n}$ is positive semi-definite, since for any vector \mathbf{v} :

$$\mathbf{v}^\top \mathbf{A} \mathbf{v} = \mathbf{v}^\top \mathbf{a} \mathbf{a}^\top \mathbf{v} = (\mathbf{v}^\top \mathbf{a})^2 \geq 0$$

Operations preserving convexity

Maximum of **convex** functions is a convex function

$$f(\mathbf{x}) = \max_{j=1,\dots,m} \{f_j(\mathbf{x})\}, \quad (f_1, \dots, f_m \text{ convex})$$

Operations preserving convexity

Maximum of **convex** functions is a convex function

$$f(\mathbf{x}) = \max_{j=1,\dots,m} \{f_j(\mathbf{x})\}, \quad (f_1, \dots, f_m \text{ convex})$$

Proof:

$$f(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y}) = \max_j \{f_j(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y})\}$$

Operations preserving convexity

Maximum of convex functions is a convex function

$$f(\mathbf{x}) = \max_{j=1,\dots,m} \{f_j(\mathbf{x})\}, \quad (f_1, \dots, f_m \text{ convex})$$

Proof:

$$\begin{aligned} f(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y}) &= \max_j \{f_j(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y})\} \\ &\stackrel{(*)}{\leqslant} \max_j \{\lambda f_j(\mathbf{x}) + (1 - \lambda) f_j(\mathbf{y})\} \end{aligned}$$

Operations preserving convexity

Maximum of convex functions is a convex function

$$f(\mathbf{x}) = \max_{j=1,\dots,m} \{f_j(\mathbf{x})\}, \quad (f_1, \dots, f_m \text{ convex})$$

Proof:

$$\begin{aligned} f(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y}) &= \max_j \{f_j(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y})\} \\ &\stackrel{(*)}{\leqslant} \max_j \{\lambda f_j(\mathbf{x}) + (1 - \lambda) f_j(\mathbf{y})\} \end{aligned}$$

where in $(*)$ we used the convexity of f_1, \dots, f_m and the fact that if $a_j \leq b_j$ for $j = 1, \dots, m$ then $\max_{j=1,\dots,m} \{a_j\} \leq \max_{j=1,\dots,m} \{b_j\}$

Operations preserving convexity

Maximum of convex functions is a convex function

$$f(\mathbf{x}) = \max_{j=1,\dots,m} \{f_j(\mathbf{x})\}, \quad (f_1, \dots, f_m \text{ convex})$$

Proof:

$$\begin{aligned} f(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y}) &= \max_j \{f_j(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y})\} \\ &\stackrel{(*)}{\leqslant} \max_j \{\lambda f_j(\mathbf{x}) + (1 - \lambda) f_j(\mathbf{y})\} \\ &\leqslant \max_j \{\lambda f_j(\mathbf{x})\} + \max_j \{(1 - \lambda) f_j(\mathbf{y})\} \end{aligned}$$

where in $(*)$ we used the convexity of f_1, \dots, f_m and the fact that if $a_j \leq b_j$ for $j = 1, \dots, m$ then $\max_{j=1,\dots,m} \{a_j\} \leq \max_{j=1,\dots,m} \{b_j\}$

Operations preserving convexity

Maximum of convex functions is a convex function

$$f(\mathbf{x}) = \max_{j=1,\dots,m} \{f_j(\mathbf{x})\}, \quad (f_1, \dots, f_m \text{ convex})$$

Proof:

$$\begin{aligned} f(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y}) &= \max_j \{f_j(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y})\} \\ &\stackrel{(*)}{\leqslant} \max_j \{\lambda f_j(\mathbf{x}) + (1 - \lambda) f_j(\mathbf{y})\} \\ &\leqslant \max_j \{\lambda f_j(\mathbf{x})\} + \max_j \{(1 - \lambda) f_j(\mathbf{y})\} \\ &= \lambda \max_j \{f_j(\mathbf{x})\} + (1 - \lambda) \max_j \{f_j(\mathbf{y})\} \end{aligned}$$

where in $(*)$ we used the convexity of f_1, \dots, f_m and the fact that if $a_j \leq b_j$ for $j = 1, \dots, m$ then $\max_{j=1,\dots,m} \{a_j\} \leq \max_{j=1,\dots,m} \{b_j\}$

Operations preserving convexity

Maximum of convex functions is a convex function

$$f(\mathbf{x}) = \max_{j=1,\dots,m} \{f_j(\mathbf{x})\}, \quad (f_1, \dots, f_m \text{ convex})$$

Proof:

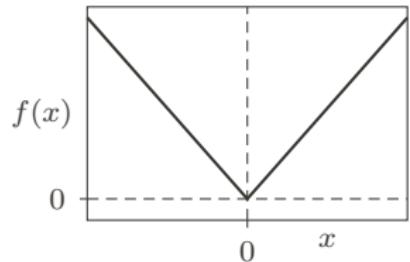
$$\begin{aligned} f(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y}) &= \max_j \{f_j(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y})\} \\ &\stackrel{(*)}{\leqslant} \max_j \{\lambda f_j(\mathbf{x}) + (1 - \lambda) f_j(\mathbf{y})\} \\ &\leqslant \max_j \{\lambda f_j(\mathbf{x})\} + \max_j \{(1 - \lambda) f_j(\mathbf{y})\} \\ &= \lambda \max_j \{f_j(\mathbf{x})\} + (1 - \lambda) \max_j \{f_j(\mathbf{y})\} \\ &= \lambda f(\mathbf{x}) + (1 - \lambda) f(\mathbf{y}), \end{aligned}$$

where in $(*)$ we used the convexity of f_1, \dots, f_m and the fact that if $a_j \leq b_j$ for $j = 1, \dots, m$ then $\max_{j=1,\dots,m} \{a_j\} \leq \max_{j=1,\dots,m} \{b_j\}$

Examples of convex functions

- Absolute value $f(x) = |x|$

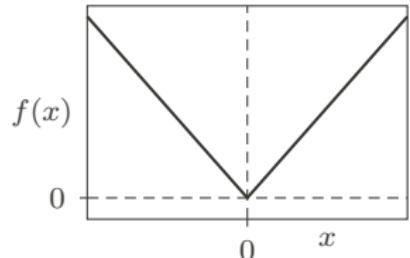
$$f(x) = |x| = \max\{x, -x\}$$



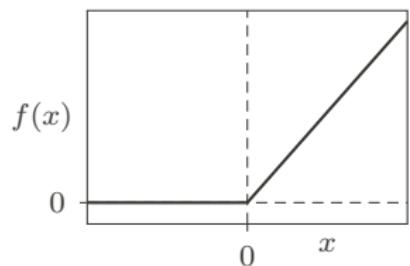
Examples of convex functions

- Absolute value $f(x) = |x|$

$$f(x) = |x| = \max\{x, -x\}$$



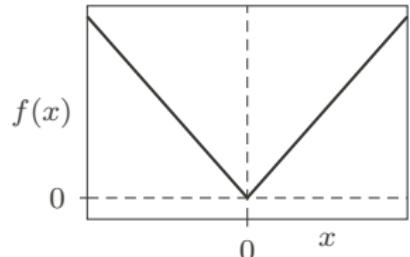
- Function $f(x) = \max\{0, x\}$



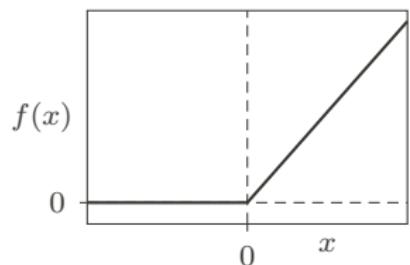
Examples of convex functions

- Absolute value $f(x) = |x|$

$$f(x) = |x| = \max\{x, -x\}$$



- Function $f(x) = \max\{0, x\}$



- Maximum of m linear functions:

$$f(\mathbf{x}) = \max \left\{ \mathbf{a}_1^\top \mathbf{x} + b_1, \mathbf{a}_2^\top \mathbf{x} + b_2, \dots, \mathbf{a}_m^\top \mathbf{x} + b_m \right\}$$

Operations preserving convexity

A **composition** of a **convex** function with a **linear** function is a convex function

$$f(\mathbf{x}) = g(\mathbf{A}\mathbf{x} + \mathbf{b}),$$

where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$, $g: \mathbb{R}^m \rightarrow \mathbb{R}$ is convex

Operations preserving convexity

A **composition** of a **convex** function with a **linear** function is a convex function

$$f(\mathbf{x}) = g(\mathbf{A}\mathbf{x} + \mathbf{b}),$$

where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$, $g: \mathbb{R}^m \rightarrow \mathbb{R}$ is convex

Proof:

$$f(\lambda\mathbf{x} + (1 - \lambda)\mathbf{y}) = g\left(\mathbf{A}(\lambda\mathbf{x} + (1 - \lambda)\mathbf{y}) + \mathbf{b}\right)$$

Operations preserving convexity

A **composition** of a **convex** function with a **linear** function is a convex function

$$f(\mathbf{x}) = g(\mathbf{A}\mathbf{x} + \mathbf{b}),$$

where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$, $g: \mathbb{R}^m \rightarrow \mathbb{R}$ is convex

Proof:

$$\begin{aligned} f(\lambda\mathbf{x} + (1 - \lambda)\mathbf{y}) &= g\left(\mathbf{A}(\lambda\mathbf{x} + (1 - \lambda)\mathbf{y}) + \mathbf{b}\right) \\ &= g\left(\lambda(\mathbf{A}\mathbf{x} + \mathbf{b}) + (1 - \lambda)(\mathbf{A}\mathbf{y} + \mathbf{b})\right) \end{aligned}$$

Operations preserving convexity

A **composition** of a **convex** function with a **linear** function is a convex function

$$f(\mathbf{x}) = g(\mathbf{A}\mathbf{x} + \mathbf{b}),$$

where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$, $g: \mathbb{R}^m \rightarrow \mathbb{R}$ is convex

Proof:

$$\begin{aligned} f(\lambda\mathbf{x} + (1 - \lambda)\mathbf{y}) &= g\left(\mathbf{A}(\lambda\mathbf{x} + (1 - \lambda)\mathbf{y}) + \mathbf{b}\right) \\ &= g\left(\lambda(\mathbf{A}\mathbf{x} + \mathbf{b}) + (1 - \lambda)(\mathbf{A}\mathbf{y} + \mathbf{b})\right) \\ (\text{convexity of } g) &\leq \lambda g(\mathbf{A}\mathbf{x} + \mathbf{b}) + (1 - \lambda)g(\mathbf{A}\mathbf{y} + \mathbf{b}) \end{aligned}$$

Operations preserving convexity

A **composition** of a **convex** function with a **linear** function is a convex function

$$f(\mathbf{x}) = g(\mathbf{A}\mathbf{x} + \mathbf{b}),$$

where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$, $g: \mathbb{R}^m \rightarrow \mathbb{R}$ is convex

Proof:

$$\begin{aligned} f(\lambda\mathbf{x} + (1 - \lambda)\mathbf{y}) &= g\left(\mathbf{A}(\lambda\mathbf{x} + (1 - \lambda)\mathbf{y}) + \mathbf{b}\right) \\ &= g\left(\lambda(\mathbf{A}\mathbf{x} + \mathbf{b}) + (1 - \lambda)(\mathbf{A}\mathbf{y} + \mathbf{b})\right) \\ (\text{convexity of } g) &\leq \lambda g(\mathbf{A}\mathbf{x} + \mathbf{b}) + (1 - \lambda)g(\mathbf{A}\mathbf{y} + \mathbf{b}) \\ &= \lambda f(\mathbf{x}) + (1 - \lambda)f(\mathbf{y}) \end{aligned}$$

Operations preserving convexity

A **composition** of a **convex** function with a **linear** function is a convex function

$$f(\mathbf{x}) = g(\mathbf{A}\mathbf{x} + \mathbf{b}),$$

where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$, $g: \mathbb{R}^m \rightarrow \mathbb{R}$ is convex

Proof:

$$\begin{aligned} f(\lambda\mathbf{x} + (1 - \lambda)\mathbf{y}) &= g\left(\mathbf{A}(\lambda\mathbf{x} + (1 - \lambda)\mathbf{y}) + \mathbf{b}\right) \\ &= g\left(\lambda(\mathbf{A}\mathbf{x} + \mathbf{b}) + (1 - \lambda)(\mathbf{A}\mathbf{y} + \mathbf{b})\right) \\ (\text{convexity of } g) &\leq \lambda g(\mathbf{A}\mathbf{x} + \mathbf{b}) + (1 - \lambda)g(\mathbf{A}\mathbf{y} + \mathbf{b}) \\ &= \lambda f(\mathbf{x}) + (1 - \lambda)f(\mathbf{y}) \end{aligned}$$

Remark: A special (and very useful) case with $m = 1$:

$$f(\mathbf{x}) = g(\mathbf{a}^\top \mathbf{x} + b)$$

Examples of convex function

The following functions are convex as a composition of a convex function g with a linear function:

- $f(\mathbf{x}) = (y - \mathbf{a}^\top \mathbf{x})^2$ (for $g(x) = x^2$)
- $f(\mathbf{x}) = |y - \mathbf{a}^\top \mathbf{x}|$ (for $g(x) = |x|$)
- $f(\mathbf{x}) = \max\{0, 1 - \mathbf{a}^\top \mathbf{x}\}$ (for $g(x) = \max\{0, x\}$)
- $f(\mathbf{x}) = \ln(1 + e^{-\mathbf{a}^\top \mathbf{x}})$ (for $g(x) = \ln(1 + e^{-x})$)
- $f(\mathbf{x}) = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2$ (for $g(\mathbf{y}) = \|\mathbf{y}\|^2$)

Operations preserving convexity

A **linear combination** of **convex** functions with **non-negative** linear coefficients is a convex function

$$f(\mathbf{x}) = \alpha_1 f_1(\mathbf{x}) + \dots + \alpha_m f_m(\mathbf{x}), \quad \alpha_1, \dots, \alpha_m \geq 0$$

Operations preserving convexity

A **linear combination** of **convex** functions with **non-negative** linear coefficients is a convex function

$$f(\mathbf{x}) = \alpha_1 f_1(\mathbf{x}) + \dots + \alpha_m f_m(\mathbf{x}), \quad \alpha_1, \dots, \alpha_m \geq 0$$

Proof:

$$f(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y}) = \sum_{j=1}^m \alpha_j f_j(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y})$$

Operations preserving convexity

A **linear combination** of **convex** functions with **non-negative** linear coefficients is a convex function

$$f(\mathbf{x}) = \alpha_1 f_1(\mathbf{x}) + \dots + \alpha_m f_m(\mathbf{x}), \quad \alpha_1, \dots, \alpha_m \geq 0$$

Proof:

$$\begin{aligned} f(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y}) &= \sum_{j=1}^m \alpha_j f_j(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y}) \\ (\text{convexity of } f_j) &\leq \sum_{j=1}^m \alpha_j (\lambda f_j(\mathbf{x}) + (1 - \lambda) f_j(\mathbf{y})) \end{aligned}$$

Operations preserving convexity

A **linear combination** of **convex** functions with **non-negative** linear coefficients is a convex function

$$f(\mathbf{x}) = \alpha_1 f_1(\mathbf{x}) + \dots + \alpha_m f_m(\mathbf{x}), \quad \alpha_1, \dots, \alpha_m \geq 0$$

Proof:

$$\begin{aligned} f(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y}) &= \sum_{j=1}^m \alpha_j f_j(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y}) \\ (\text{convexity of } f_j) &\leq \sum_{j=1}^m \alpha_j (\lambda f_j(\mathbf{x}) + (1 - \lambda) f_j(\mathbf{y})) \\ &= \lambda \left(\sum_{j=1}^m \alpha_j f_j(\mathbf{x}) \right) + (1 - \lambda) \left(\sum_{j=1}^m \alpha_j f_j(\mathbf{y}) \right) \end{aligned}$$

Operations preserving convexity

A **linear combination** of **convex** functions with **non-negative** linear coefficients is a convex function

$$f(\mathbf{x}) = \alpha_1 f_1(\mathbf{x}) + \dots + \alpha_m f_m(\mathbf{x}), \quad \alpha_1, \dots, \alpha_m \geq 0$$

Proof:

$$\begin{aligned} f(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y}) &= \sum_{j=1}^m \alpha_j f_j(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y}) \\ (\text{convexity of } f_j) &\leq \sum_{j=1}^m \alpha_j (\lambda f_j(\mathbf{x}) + (1 - \lambda) f_j(\mathbf{y})) \\ &= \lambda \left(\sum_{j=1}^m \alpha_j f_j(\mathbf{x}) \right) + (1 - \lambda) \left(\sum_{j=1}^m \alpha_j f_j(\mathbf{y}) \right) \\ &= \lambda f(\mathbf{x}) + (1 - \lambda) f(\mathbf{y}) \end{aligned}$$

Examples of convex functions

Conclusion: the following functions are convex (for $\mathbf{x} \in \mathbb{R}^n$)

$$f(\mathbf{x}) = e^{x_1} + \dots + e^{x_n}$$

$$f(\mathbf{x}) = x_1 \ln x_1 + \dots + x_n \ln x_n$$

$$f(\mathbf{x}) = (y_1 - \mathbf{a}_1^\top \mathbf{x}_1)^2 + \dots + (y_m - \mathbf{a}_m^\top \mathbf{x}_m)^2$$

$$f(\mathbf{x}) = \ln(1 + e^{-\mathbf{a}_1^\top \mathbf{x}}) + \dots + \ln(1 + e^{-\mathbf{a}_m^\top \mathbf{x}})$$

Operations preserving convexity

A function composition $f(x) = h(g(x))$, where $g: \mathbb{R}^n \rightarrow \mathbb{R}$ is **convex**, and $h: \mathbb{R} \rightarrow \mathbb{R}$ is **convex and non-decreasing**, is a convex function

Operations preserving convexity

A **function composition** $f(\mathbf{x}) = h(g(\mathbf{x}))$, where $g: \mathbb{R}^n \rightarrow \mathbb{R}$ is **convex**, and $h: \mathbb{R} \rightarrow \mathbb{R}$ is **convex and non-decreasing**, is a convex function

Proof:

$$f(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y}) = h\left(g(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y})\right)$$

Operations preserving convexity

A **function composition** $f(\mathbf{x}) = h(g(\mathbf{x}))$, where $g: \mathbb{R}^n \rightarrow \mathbb{R}$ is **convex**, and $h: \mathbb{R} \rightarrow \mathbb{R}$ is **convex and non-decreasing**, is a convex function

Proof:

$$\begin{aligned} f(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y}) &= h\left(g(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y})\right) \\ &\stackrel{(*)}{\leqslant} h\left(\lambda g(\mathbf{x}) + (1 - \lambda)g(\mathbf{y})\right) \end{aligned}$$

Operations preserving convexity

A **function composition** $f(\mathbf{x}) = h(g(\mathbf{x}))$, where $g: \mathbb{R}^n \rightarrow \mathbb{R}$ is **convex**, and $h: \mathbb{R} \rightarrow \mathbb{R}$ is **convex and non-decreasing**, is a convex function

Proof:

$$\begin{aligned} f(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y}) &= h\left(g(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y})\right) \\ &\stackrel{(*)}{\leqslant} h\left(\lambda g(\mathbf{x}) + (1 - \lambda)g(\mathbf{y})\right) \end{aligned}$$

where in $(*)$ we used the convexity of g and the monotonicity of h

Operations preserving convexity

A **function composition** $f(\mathbf{x}) = h(g(\mathbf{x}))$, where $g: \mathbb{R}^n \rightarrow \mathbb{R}$ is **convex**, and $h: \mathbb{R} \rightarrow \mathbb{R}$ is **convex and non-decreasing**, is a convex function

Proof:

$$\begin{aligned} f(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y}) &= h\left(g(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y})\right) \\ &\stackrel{(*)}{\leqslant} h\left(\lambda g(\mathbf{x}) + (1 - \lambda)g(\mathbf{y})\right) \\ (\text{wypukłość } h) &\leqslant \lambda h(g(\mathbf{x})) + (1 - \lambda)h(g(\mathbf{y})) \end{aligned}$$

where in $(*)$ we used the convexity of g and the monotonicity of h

Operations preserving convexity

A **function composition** $f(\mathbf{x}) = h(g(\mathbf{x}))$, where $g: \mathbb{R}^n \rightarrow \mathbb{R}$ is **convex**, and $h: \mathbb{R} \rightarrow \mathbb{R}$ is **convex and non-decreasing**, is a convex function

Proof:

$$\begin{aligned} f(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y}) &= h\left(g(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y})\right) \\ &\stackrel{(*)}{\leqslant} h\left(\lambda g(\mathbf{x}) + (1 - \lambda)g(\mathbf{y})\right) \\ (\text{wypukłość } h) &\leqslant \lambda h(g(\mathbf{x})) + (1 - \lambda)h(g(\mathbf{y})) \\ &= \lambda f(\mathbf{x}) + (1 - \lambda)f(\mathbf{y}), \end{aligned}$$

where in $(*)$ we used the convexity of g and the monotonicity of h

Operations preserving convexity

A **function composition** $f(\mathbf{x}) = h(g(\mathbf{x}))$, where $g: \mathbb{R}^n \rightarrow \mathbb{R}$ is **convex**, and $h: \mathbb{R} \rightarrow \mathbb{R}$ is **convex and non-decreasing**, is a convex function

Proof:

$$\begin{aligned} f(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y}) &= h\left(g(\lambda \mathbf{x} + (1 - \lambda) \mathbf{y})\right) \\ &\stackrel{(*)}{\leqslant} h\left(\lambda g(\mathbf{x}) + (1 - \lambda)g(\mathbf{y})\right) \\ (\text{wypukłość } h) &\leqslant \lambda h(g(\mathbf{x})) + (1 - \lambda)h(g(\mathbf{y})) \\ &= \lambda f(\mathbf{x}) + (1 - \lambda)f(\mathbf{y}), \end{aligned}$$

where in $(*)$ we used the convexity of g and the monotonicity of h

Example: function $f(\mathbf{x}) = e^{g(\mathbf{x})}$ is convex if $g(\mathbf{x})$ is convex

Optimization Methods for Data Analysis

4. Descent methods (gradient descent method)

Wojciech Kotłowski

Institute of Computing Science, PUT
<http://www.cs.put.poznan.pl/wkotlowski/>

31.03.2021

Outline

Unconstrained minimization of a **differentiable** function $f(x)$:

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

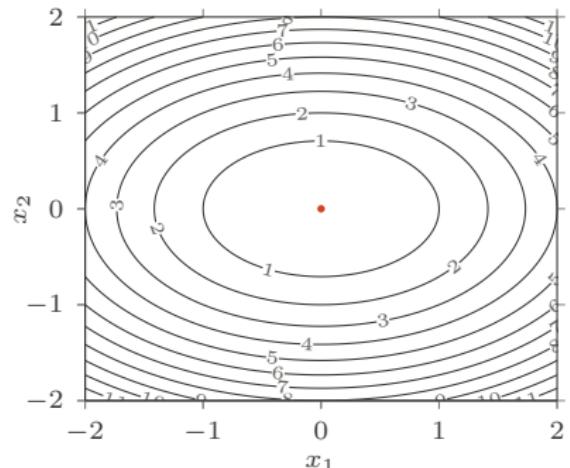
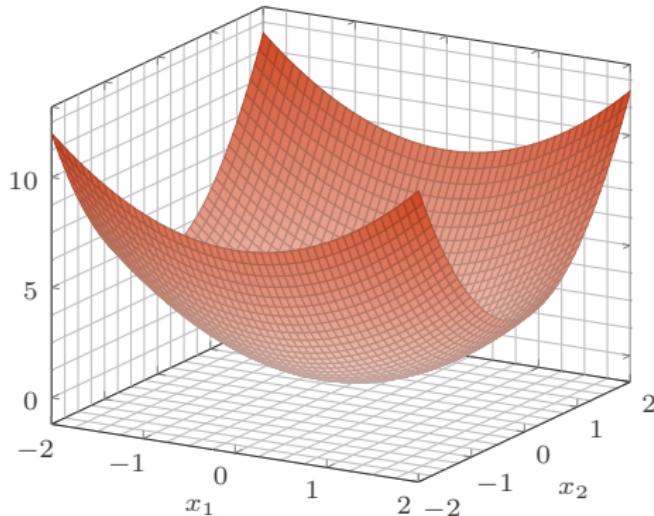
We will often assume the **convexity** of $f(x)$

1. Introduction
2. Gradient descent method
3. Newton-Raphson method (next lecture)
4. Modifications of Newton-Rapshon (next lecture)
5. Conjugate gradients method (next lecture)

Introduction

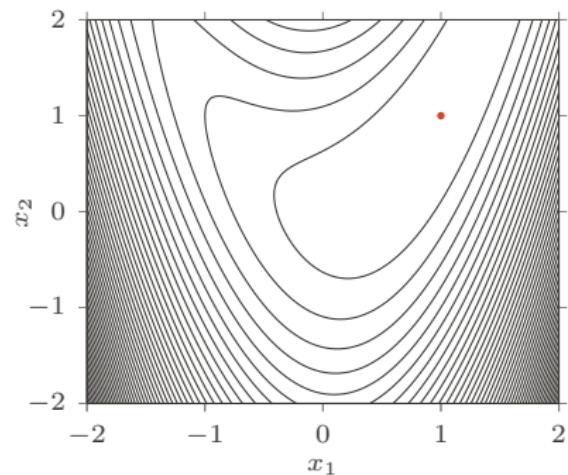
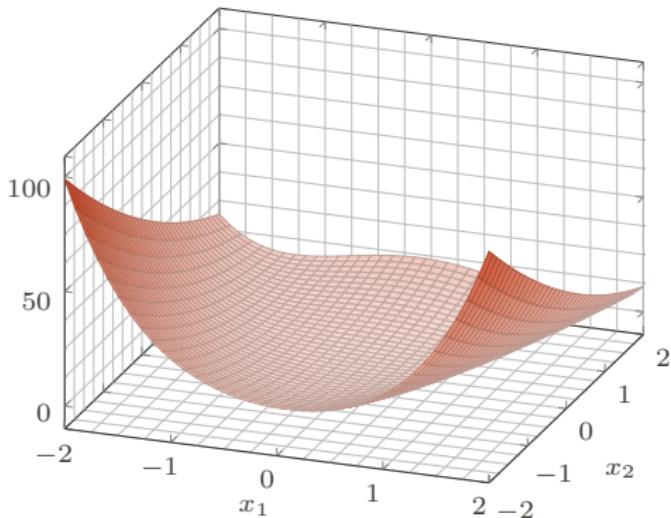
Surface and contour plots

$$f(x_1, x_2) = x_1^2 + 2x_2^2$$



Surface and contour plots

$$f(x_1, x_2) = 2.5(x_1^2 - x_2)^2 + (1 - x_1)^2$$



Contour plot of a quadratic function

Consider a quadratic function with positive-definite matrix A :

$$f(\mathbf{x}) = \mathbf{x}^\top A \mathbf{x} + \mathbf{b}^\top \mathbf{x} + c$$

Contour plot of a quadratic function

Consider a quadratic function with positive-definite matrix \mathbf{A} :

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x} + \mathbf{b}^\top \mathbf{x} + c$$

Approximating with a (exact!) Taylor polynomial of second order around the minimizer \mathbf{x}^* we can rewrite the function as:

$$f(\mathbf{x}) = f(\mathbf{x}^*) + (\mathbf{x} - \mathbf{x}^*)^\top \mathbf{A} (\mathbf{x} - \mathbf{x}^*)$$

Contour plot of a quadratic function

Consider a quadratic function with positive-definite matrix A :

$$f(\mathbf{x}) = \mathbf{x}^\top A \mathbf{x} + \mathbf{b}^\top \mathbf{x} + c$$

Approximating with a (exact!) Taylor polynomial of second order around the minimizer \mathbf{x}^* we can rewrite the function as:

$$f(\mathbf{x}) = f(\mathbf{x}^*) + (\mathbf{x} - \mathbf{x}^*)^\top A (\mathbf{x} - \mathbf{x}^*)$$

Example: $f(x) = x^2 + 2x + 2$

Contour plot of a quadratic function

Consider a quadratic function with positive-definite matrix \mathbf{A} :

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x} + \mathbf{b}^\top \mathbf{x} + c$$

Approximating with a (exact!) Taylor polynomial of second order around the minimizer \mathbf{x}^* we can rewrite the function as:

$$f(\mathbf{x}) = f(\mathbf{x}^*) + (\mathbf{x} - \mathbf{x}^*)^\top \mathbf{A} (\mathbf{x} - \mathbf{x}^*)$$

Example: $f(x) = x^2 + 2x + 2$

Minimum at $x^* = -\frac{b}{2a} = -1$, $f(x^*) = 1$, and therefore

$$f(x) = 1 + (x + 1)^2$$

Contour plot of a quadratic function

Consider a quadratic function with positive-definite matrix \mathbf{A} :

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x} + \mathbf{b}^\top \mathbf{x} + c$$

Approximating with a (exact!) Taylor polynomial of second order around the minimizer \mathbf{x}^* we can rewrite the function as:

$$f(\mathbf{x}) = f(\mathbf{x}^*) + (\mathbf{x} - \mathbf{x}^*)^\top \mathbf{A} (\mathbf{x} - \mathbf{x}^*)$$

Example: $f(x) = x^2 + 2x + 2$

Minimum at $x^* = -\frac{b}{2a} = -1$, $f(x^*) = 1$, and therefore
 $f(x) = 1 + (x + 1)^2$

Example: $f(\mathbf{x}) = x_1^2 + 2x_2^2 - 2x_1 + 4x_2 + 1$

Contour plot of a quadratic function

Consider a quadratic function with positive-definite matrix \mathbf{A} :

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x} + \mathbf{b}^\top \mathbf{x} + c$$

Approximating with a (exact!) Taylor polynomial of second order around the minimizer \mathbf{x}^* we can rewrite the function as:

$$f(\mathbf{x}) = f(\mathbf{x}^*) + (\mathbf{x} - \mathbf{x}^*)^\top \mathbf{A} (\mathbf{x} - \mathbf{x}^*)$$

Example: $f(x) = x^2 + 2x + 2$

Minimum at $x^* = -\frac{b}{2a} = -1$, $f(x^*) = 1$, and therefore
 $f(x) = 1 + (x + 1)^2$

Example: $f(\mathbf{x}) = x_1^2 + 2x_2^2 - 2x_1 + 4x_2 + 1$

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}, \quad \mathbf{b} = (-2, 4), \quad \mathbf{x}^* = -\frac{1}{2} \mathbf{A}^{-1} \mathbf{b} = -\frac{1}{2} \begin{bmatrix} 1 & 0 \\ 0 & 0.5 \end{bmatrix} \begin{bmatrix} -2 \\ 4 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Contour plot of a quadratic function

Consider a quadratic function with positive-definite matrix \mathbf{A} :

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x} + \mathbf{b}^\top \mathbf{x} + c$$

Approximating with a (exact!) Taylor polynomial of second order around the minimizer \mathbf{x}^* we can rewrite the function as:

$$f(\mathbf{x}) = f(\mathbf{x}^*) + (\mathbf{x} - \mathbf{x}^*)^\top \mathbf{A} (\mathbf{x} - \mathbf{x}^*)$$

Example: $f(x) = x^2 + 2x + 2$

Minimum at $x^* = -\frac{b}{2a} = -1$, $f(x^*) = 1$, and therefore
 $f(x) = 1 + (x + 1)^2$

Example: $f(\mathbf{x}) = x_1^2 + 2x_2^2 - 2x_1 + 4x_2 + 1$

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}, \quad \mathbf{b} = (-2, 4), \quad \mathbf{x}^* = -\frac{1}{2}\mathbf{A}^{-1}\mathbf{b} = -\frac{1}{2} \begin{bmatrix} 1 & 0 \\ 0 & 0.5 \end{bmatrix} \begin{bmatrix} -2 \\ 4 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

We have $f(\mathbf{x}^*) = -2$, so:

$$f(\mathbf{x}) = -2 + [x_1 - 1 \ x_2 + 1] \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} x_1 - 1 \\ x_2 + 1 \end{bmatrix} = -2 + (x_1 - 1)^2 + 2(x_2 + 1)^2$$

Contour plot of a quadratic function

$$f(\mathbf{x}) = f(\mathbf{x}^*) + (\mathbf{x} - \mathbf{x}^*)^\top \mathbf{A}(\mathbf{x} - \mathbf{x}^*)$$

Contour plot of a quadratic function

$$f(\mathbf{x}) = f(\mathbf{x}^*) + (\mathbf{x} - \mathbf{x}^*)^\top \mathbf{A}(\mathbf{x} - \mathbf{x}^*)$$

Equation:

$$(\mathbf{x} - \mathbf{x}^*)^\top \mathbf{A}(\mathbf{x} - \mathbf{x}^*) = c$$

is the equation of an **ellipsoid** with center \mathbf{x}^* , principal axes of which correspond to the **eigenvectors** of \mathbf{A} , and the lengths of these axes are inversely proportional to the square roots of the **eigenvalues**

In particular, for $n = 2$ this is an equation of **ellipse** on a plane

Contour plot of a quadratic function

$$f(\mathbf{x}) = f(\mathbf{x}^*) + (\mathbf{x} - \mathbf{x}^*)^\top \mathbf{A}(\mathbf{x} - \mathbf{x}^*)$$

Equation:

$$(\mathbf{x} - \mathbf{x}^*)^\top \mathbf{A}(\mathbf{x} - \mathbf{x}^*) = c$$

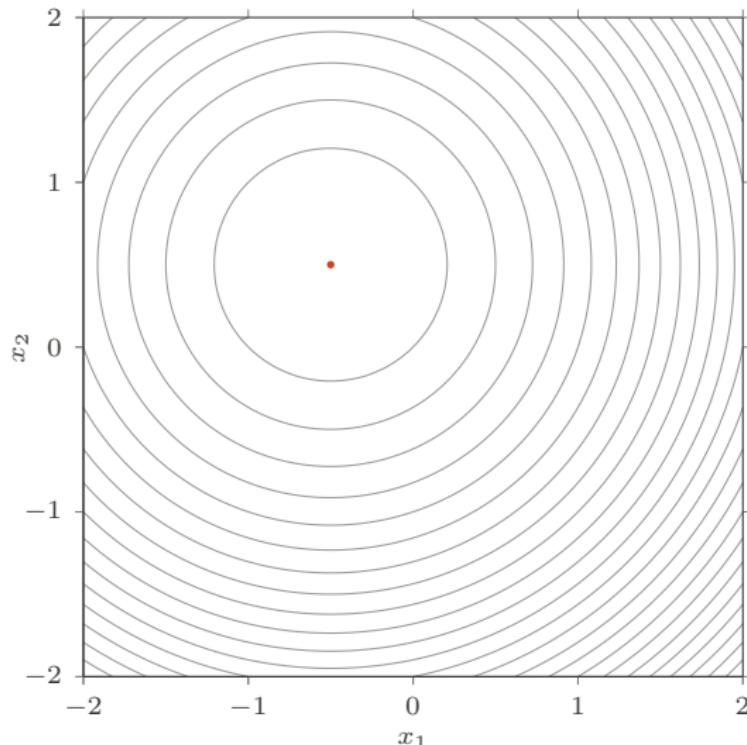
is the equation of an **ellipsoid** with center \mathbf{x}^* , principal axes of which correspond to the **eigenvectors** of \mathbf{A} , and the lengths of these axes are inversely proportional to the square roots of the **eigenvalues**

In particular, for $n = 2$ this is an equation of **ellipse** on a plane

Conclusion: the contour lines of a square functions are ellipses

Contour plot of a quadratic function

$$f(x_1, x_2) = x_1^2 + x_2^2 + x_1 - x_2$$



$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Eigenvectors:

$$\mathbf{v}_1 = (1, 0), \mathbf{v}_2 = (0, 1)$$

Eigenvalues:

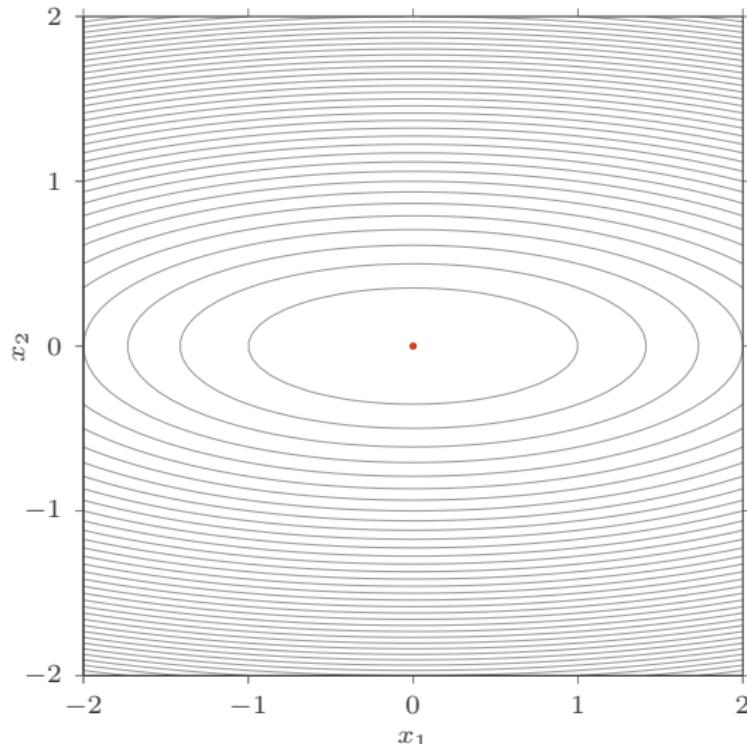
$$\lambda_1 = 1, \lambda_2 = 1$$

Minimum:

$$\mathbf{x}^* = (-0.5, 0.5)$$

Contour plot of a quadratic function

$$f(x_1, x_2) = 0.5x_1^2 + 2x_2^2$$



$$\mathbf{A} = \begin{bmatrix} 0.5 & 0 \\ 0 & 2 \end{bmatrix}$$

Eigenvectors:

$$\mathbf{v}_1 = (1, 0), \mathbf{v}_2 = (0, 1)$$

Eigenvalues:

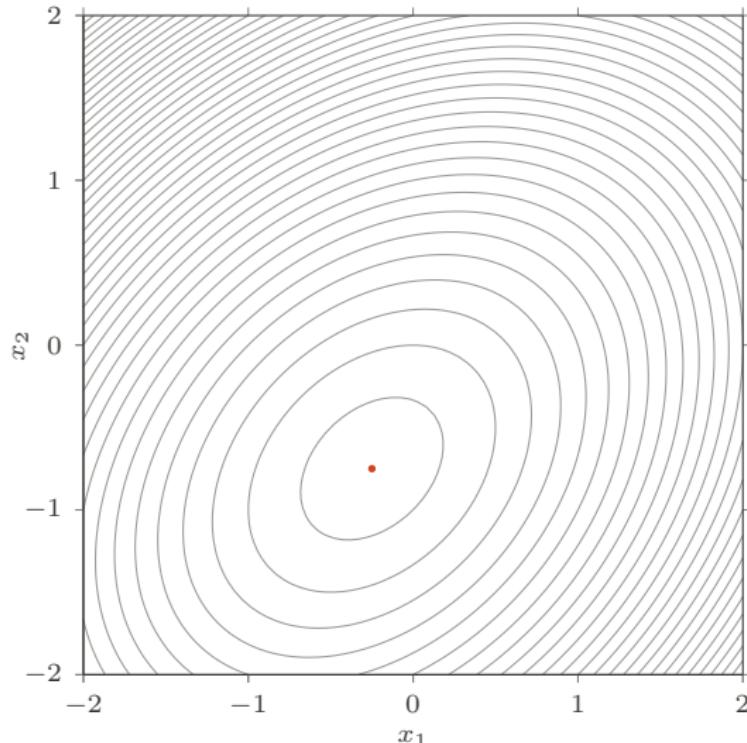
$$\lambda_1 = 0.5, \lambda_2 = 2$$

Minimum:

$$\mathbf{x}^* = (0, 0)$$

Contour plot of a quadratic function

$$f(x_1, x_2) = 1.5x_1^2 - x_1x_2 + 1.5x_2^2 + 2x_2$$



$$\mathbf{A} = \frac{1}{2} \begin{bmatrix} 3 & -1 \\ 1 & 3 \end{bmatrix}$$

Eigenvectors:

$$\mathbf{v}_1 = \frac{1}{\sqrt{2}}(1, 1), \mathbf{v}_2 = \frac{1}{\sqrt{2}}(1, -1)$$

Eigenvalues:

$$\lambda_1 = 1, \lambda_2 = 2$$

Minimum:

$$\mathbf{x}^* = (-0.25, -0.75)$$

Descent methods

- Iterative methods generating points $\mathbf{x}_1, \mathbf{x}_2, \dots$ in such a way that the objective function is decreased in each iteration: $f(\mathbf{x}_{k+1}) < f(\mathbf{x}_k)$
- In each iteration $k = 1, 2, \dots$ a **descent direction** \mathbf{v}_k , is found, i.e. a direction with the property that for sufficiently small $\alpha \in \mathbb{R}$:

$$f(\mathbf{x}_k + \alpha \mathbf{v}_k) < f(\mathbf{x}_k)$$

- The next point is calculated as follows:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{v}_k,$$

where the **step size** α_k is chosen to ensure a sufficiently large decrease of the objective function

Descent directions

Remainder: Taylor's theorem

$$f(\mathbf{x}) = f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0) + \frac{1}{2} (\mathbf{x} - \mathbf{x}_0)^\top \nabla^2 f(\boldsymbol{\xi}) (\mathbf{x} - \mathbf{x}_0),$$

where $\boldsymbol{\xi}$ is a point in between \mathbf{x}_0 and \mathbf{x}

Descent directions

Remainder: Taylor's theorem

$$f(\mathbf{x}) = f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0) + \frac{1}{2} (\mathbf{x} - \mathbf{x}_0)^\top \nabla^2 f(\boldsymbol{\xi}) (\mathbf{x} - \mathbf{x}_0),$$

where $\boldsymbol{\xi}$ is a point in between \mathbf{x}_0 and \mathbf{x}

Choosing $\mathbf{x}_0 = \mathbf{x}_k$ and $\mathbf{x} = \mathbf{x}_k + \alpha \mathbf{v}_k$ we get:

$$f(\mathbf{x}_k + \alpha \mathbf{v}_k) = f(\mathbf{x}_k) + \alpha \nabla f(\mathbf{x}_k)^\top \mathbf{v}_k + \frac{\alpha^2}{2} \mathbf{v}_k^\top \nabla^2 f(\boldsymbol{\xi}) \mathbf{v}_k$$

Descent directions

Remainder: Taylor's theorem

$$f(\mathbf{x}) = f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0) + \frac{1}{2} (\mathbf{x} - \mathbf{x}_0)^\top \nabla^2 f(\boldsymbol{\xi})(\mathbf{x} - \mathbf{x}_0),$$

where $\boldsymbol{\xi}$ is a point in between \mathbf{x}_0 and \mathbf{x}

Choosing $\mathbf{x}_0 = \mathbf{x}_k$ and $\mathbf{x} = \mathbf{x}_k + \alpha \mathbf{v}_k$ we get:

$$f(\mathbf{x}_k + \alpha \mathbf{v}_k) - f(\mathbf{x}_k) = \alpha \nabla f(\mathbf{x}_k)^\top \mathbf{v}_k + O(\alpha^2)$$

Descent directions

Remainder: Taylor's theorem

$$f(\mathbf{x}) = f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0) + \frac{1}{2} (\mathbf{x} - \mathbf{x}_0)^\top \nabla^2 f(\boldsymbol{\xi})(\mathbf{x} - \mathbf{x}_0),$$

where $\boldsymbol{\xi}$ is a point in between \mathbf{x}_0 and \mathbf{x}

Choosing $\mathbf{x}_0 = \mathbf{x}_k$ and $\mathbf{x} = \mathbf{x}_k + \alpha \mathbf{v}_k$ we get:

$$f(\mathbf{x}_k + \alpha \mathbf{v}_k) - f(\mathbf{x}_k) = \alpha \nabla f(\mathbf{x}_k)^\top \mathbf{v}_k + O(\alpha^2)$$

For \mathbf{v}_k to be a descent direction (i.e., to guarantee a drop of the objective function with sufficiently small step size α) the **directional derivative** along \mathbf{v}_k must be **negative**, i.e.:

$$\frac{\partial f(\mathbf{x}_k)}{\partial \mathbf{v}_k} = \nabla f(\mathbf{x}_k)^\top \mathbf{v}_k < 0$$

Descent directions

Remainder: Taylor's theorem

$$f(\mathbf{x}) = f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0) + \frac{1}{2} (\mathbf{x} - \mathbf{x}_0)^\top \nabla^2 f(\boldsymbol{\xi})(\mathbf{x} - \mathbf{x}_0),$$

where $\boldsymbol{\xi}$ is a point in between \mathbf{x}_0 and \mathbf{x}

Choosing $\mathbf{x}_0 = \mathbf{x}_k$ and $\mathbf{x} = \mathbf{x}_k + \alpha \mathbf{v}_k$ we get:

$$f(\mathbf{x}_k + \alpha \mathbf{v}_k) - f(\mathbf{x}_k) = \alpha \nabla f(\mathbf{x}_k)^\top \mathbf{v}_k + O(\alpha^2)$$

For \mathbf{v}_k to be a descent direction (i.e., to guarantee a drop of the objective function with sufficiently small step size α) the **directional derivative** along \mathbf{v}_k must be **negative**, i.e.:

$$\frac{\partial f(\mathbf{x}_k)}{\partial \mathbf{v}_k} = \nabla f(\mathbf{x}_k)^\top \mathbf{v}_k < 0$$

A descent direction must form an **obtuse angle** with the gradient ($\cos \theta < 0$)

One can always find a descent direction **as long as** $\nabla f(\mathbf{x}_k) \neq \mathbf{0}$

The convergence of descent methods

- We use the property that as long as $\nabla f(\mathbf{x}_k) \neq \mathbf{0}$, we can always find a descent direction \mathbf{v}_k and a step size α_k so that:

$$f(\mathbf{x}_{k+1}) < f(\mathbf{x}_k)$$

The convergence of descent methods

- We use the property that as long as $\nabla f(\mathbf{x}_k) \neq \mathbf{0}$, we can always find a descent direction \mathbf{v}_k and a step size α_k so that:

$$f(\mathbf{x}_{k+1}) < f(\mathbf{x}_k)$$

- Assuming the objective function $f(\mathbf{x})$ is bounded from below and the decrease of the function value in each iteration is **sufficiently large**, we conclude that the limit point of the algorithm \mathbf{x}_∞ must have the property $\nabla f(\mathbf{x}_\infty) = \mathbf{0}$

(rough justification: if $\nabla f(\mathbf{x}_\infty) \neq \mathbf{0}$, we could still improve the objective function, which would contradict the claim that \mathbf{x}_∞ is the limit point)

The convergence of descent methods

- We use the property that as long as $\nabla f(\mathbf{x}_k) \neq \mathbf{0}$, we can always find a descent direction \mathbf{v}_k and a step size α_k so that:

$$f(\mathbf{x}_{k+1}) < f(\mathbf{x}_k)$$

- Assuming the objective function $f(\mathbf{x})$ is bounded from below and the decrease of the function value in each iteration is **sufficiently large**, we conclude that the limit point of the algorithm \mathbf{x}_∞ must have the property $\nabla f(\mathbf{x}_\infty) = \mathbf{0}$

(rough justification: if $\nabla f(\mathbf{x}_\infty) \neq \mathbf{0}$, we could still improve the objective function, which would contradict the claim that \mathbf{x}_∞ is the limit point)

- In general, there are no guarantees that \mathbf{x}_∞ is even a **local minimum** (it could be a **saddle point**)

The convergence of descent methods

- We use the property that as long as $\nabla f(\mathbf{x}_k) \neq \mathbf{0}$, we can always find a descent direction \mathbf{v}_k and a step size α_k so that:

$$f(\mathbf{x}_{k+1}) < f(\mathbf{x}_k)$$

- Assuming the objective function $f(\mathbf{x})$ is bounded from below and the decrease of the function value in each iteration is **sufficiently large**, we conclude that the limit point of the algorithm \mathbf{x}_∞ must have the property $\nabla f(\mathbf{x}_\infty) = \mathbf{0}$

(rough justification: if $\nabla f(\mathbf{x}_\infty) \neq \mathbf{0}$, we could still improve the objective function, which would contradict the claim that \mathbf{x}_∞ is the limit point)

- In general, there are no guarantees that \mathbf{x}_∞ is even a **local minimum** (it could be a **saddle point**)
- For **convex** functions we can show that \mathbf{x}_∞ is a **global minimum** and compute the **order of convergence**

Gradient descent method

Gradient as a maximum ascent direction

We showed before that for a unit vector \mathbf{v}_k :

$$-\|\nabla f(\mathbf{x}_k)\| \leq \nabla f(\mathbf{x}_k)^\top \mathbf{v}_k \leq \|\nabla f(\mathbf{x}_k)\|,$$

and both inequalities are satisfied with equality for $\mathbf{v}_k = \pm \frac{\nabla f(\mathbf{x}_k)}{\|\nabla f(\mathbf{x}_k)\|}$

Gradient as a maximum ascent direction

We showed before that for a unit vector \mathbf{v}_k :

$$-\|\nabla f(\mathbf{x}_k)\| \leq \nabla f(\mathbf{x}_k)^\top \mathbf{v}_k \leq \|\nabla f(\mathbf{x}_k)\|,$$

and both inequalities are satisfied with equality for $\mathbf{v}_k = \pm \frac{\nabla f(\mathbf{x}_k)}{\|\nabla f(\mathbf{x}_k)\|}$

Conclusion: the gradient is a (local) direction of **maximum ascent** of the objective, while the negative gradient – of **maximum descent**

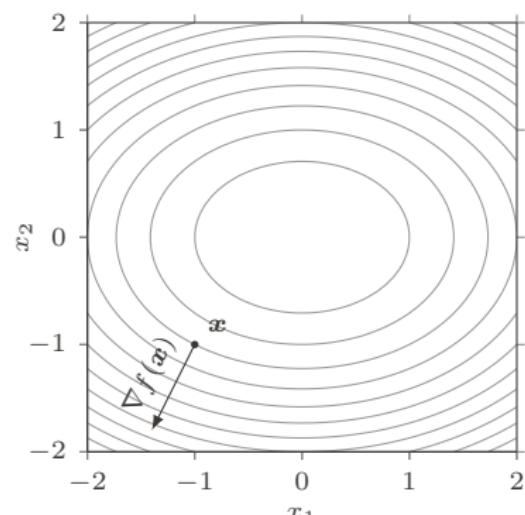
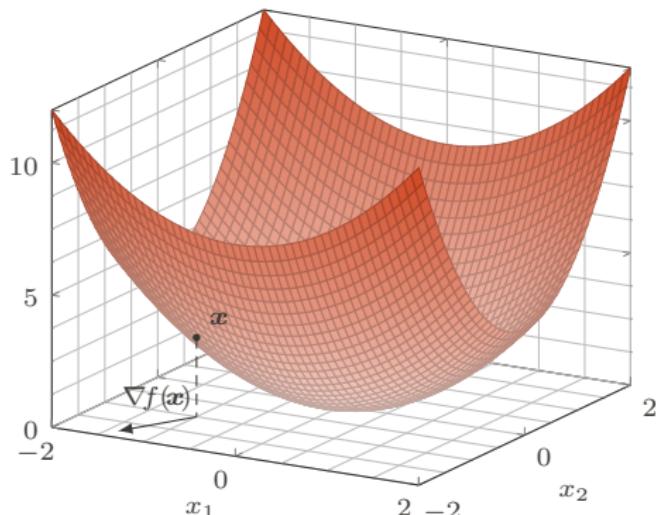
Gradient as a maximum ascent direction

We showed before that for a unit vector \mathbf{v}_k :

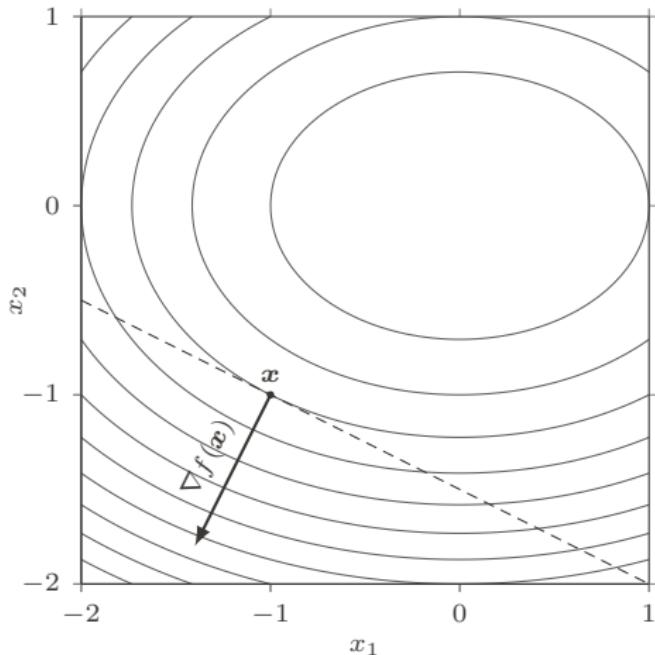
$$-\|\nabla f(\mathbf{x}_k)\| \leq \nabla f(\mathbf{x}_k)^\top \mathbf{v}_k \leq \|\nabla f(\mathbf{x}_k)\|,$$

and both inequalities are satisfied with equality for $\mathbf{v}_k = \pm \frac{\nabla f(\mathbf{x}_k)}{\|\nabla f(\mathbf{x}_k)\|}$

Conclusion: the gradient is a (local) direction of **maximum ascent** of the objective, while the negative gradient – of **maximum descent**



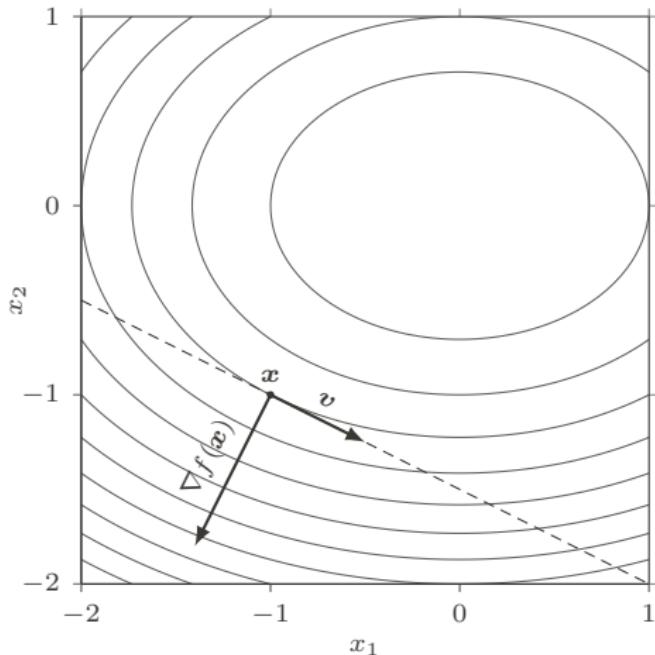
Gradient on a contour plot



The gradient vector is always **perpendicular** to the contours of the function

Why?

Gradient on a contour plot



The gradient vector is always **perpendicular** to the contours of the function

Why?

Because the directional derivative along v tangent to the contour must be zero:

$$\frac{\partial f(x + v)}{\partial v} = \nabla f(x)^\top v = 0$$

Therefore $\nabla f(x) \perp v$

Gradient descent method

Starting from x_1 , in iterations $k = 1, 2, \dots$:

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k)$$

Gradient descent method

Starting from x_1 , in iterations $k = 1, 2, \dots$:

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k)$$

Question: how to choose step sizes α_k ?

Gradient descent method

Starting from x_1 , in iterations $k = 1, 2, \dots$:

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k)$$

Question: how to choose step sizes α_k ?

- **Constant** step size $\alpha_k = \alpha$: simplest method, but ...
 - ▶ the value of α needs to be chosen by trial and error
 - ▶ does not guarantee that $f(x_{k+1}) < f(x_k)$

Gradient descent method

Starting from x_1 , in iterations $k = 1, 2, \dots$:

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k)$$

Question: how to choose step sizes α_k ?

- **Constant** step size $\alpha_k = \alpha$: simplest method, but ...
 - ▶ the value of α needs to be chosen by trial and error
 - ▶ does not guarantee that $f(x_{k+1}) < f(x_k)$
- **Optimal** step size: the **steepest descent method**

Gradient descent method

Starting from x_1 , in iterations $k = 1, 2, \dots$:

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k)$$

Question: how to choose step sizes α_k ?

- **Constant** step size $\alpha_k = \alpha$: simplest method, but ...
 - ▶ the value of α needs to be chosen by trial and error
 - ▶ does not guarantee that $f(x_{k+1}) < f(x_k)$
- **Optimal** step size: the **steepest descent method**
- **Sufficiently good** step size: not necessarily optimal, but guaranteeing $f(x_{k+1}) < f(x_k)$;
 - ▶ most often used in practice

Steepest descent (Cauchy) method

The algorithm chooses the step size α_k , which maximizes the improvement (decrease) in the objective function along the negative gradient:

$$\alpha_k = \underset{\alpha \geq 0}{\operatorname{argmin}} f(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k))$$

Steepest descent (Cauchy) method

The algorithm chooses the step size α_k , which maximizes the improvement (decrease) in the objective function along the negative gradient:

$$\alpha_k = \underset{\alpha \geq 0}{\operatorname{argmin}} f(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k))$$

We need to solve a **univariate minimization problem**

$$g(\alpha) = f(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k))$$

Steepest descent (Cauchy) method

The algorithm chooses the step size α_k , which maximizes the improvement (decrease) in the objective function along the negative gradient:

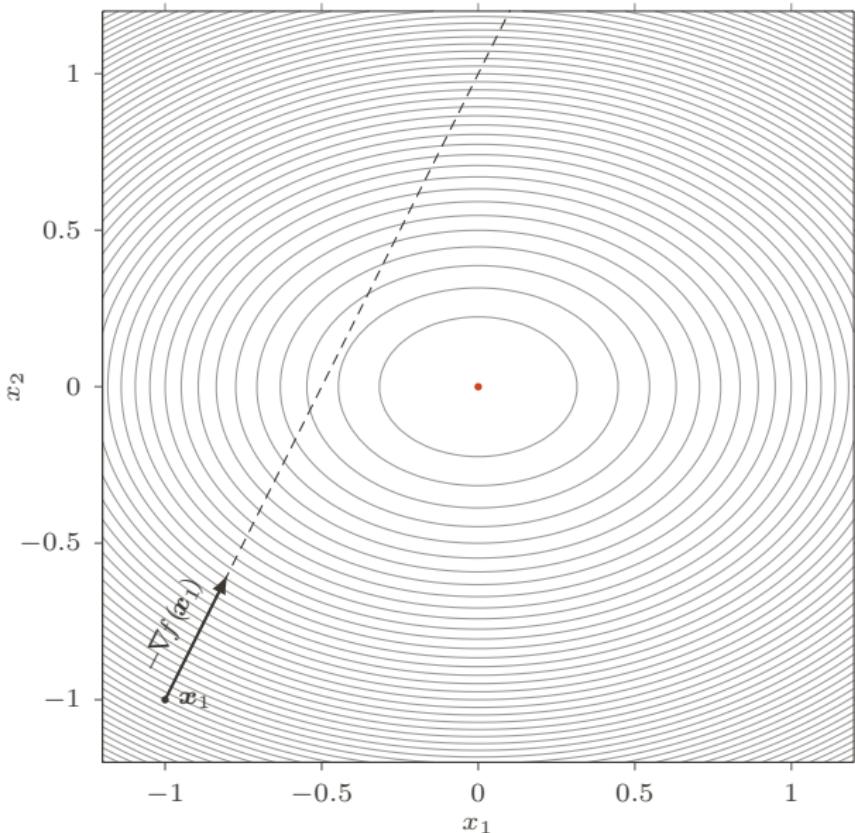
$$\alpha_k = \underset{\alpha \geq 0}{\operatorname{argmin}} f(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k))$$

We need to solve a **univariate minimization problem**

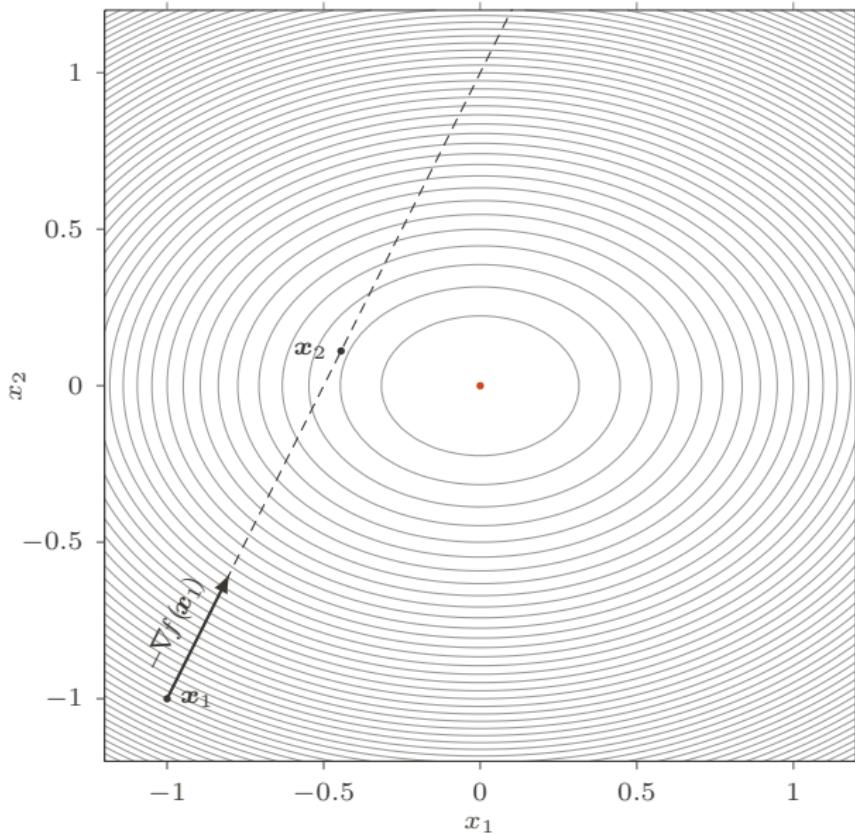
$$g(\alpha) = f(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k))$$

- Golden section method
- Bisection method
- Newton method
- ...

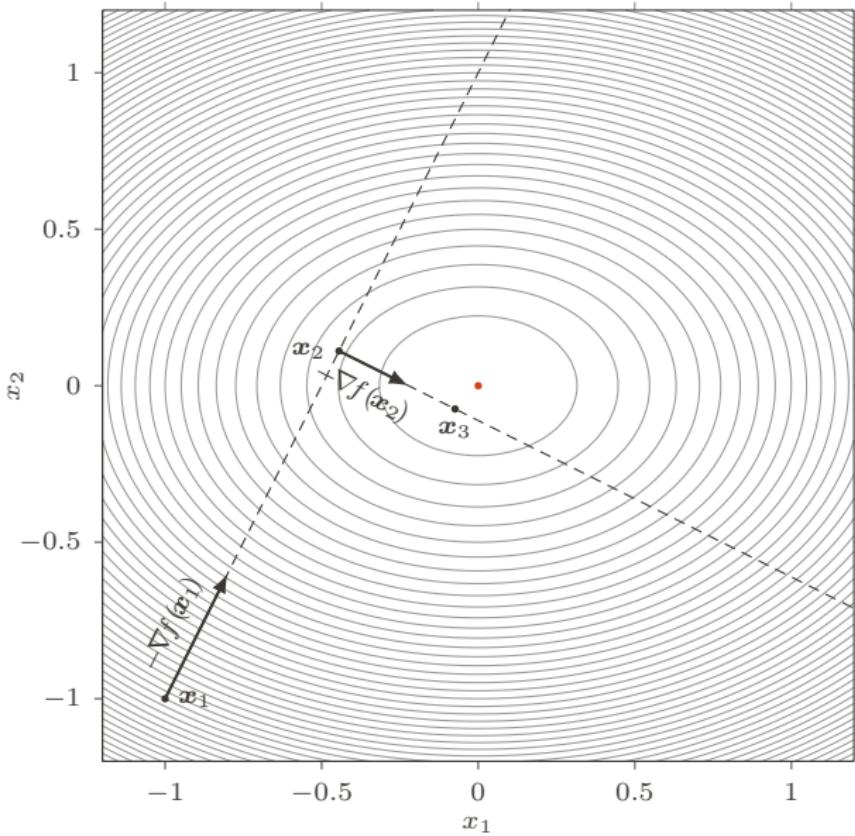
Steepest descent method



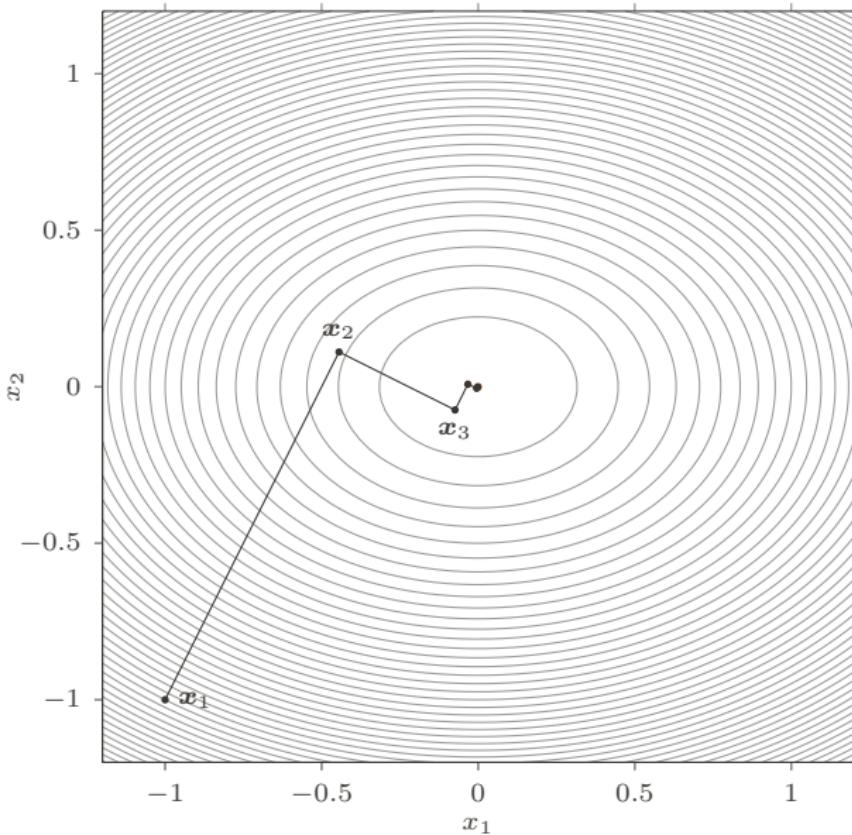
Steepest descent method



Steepest descent method

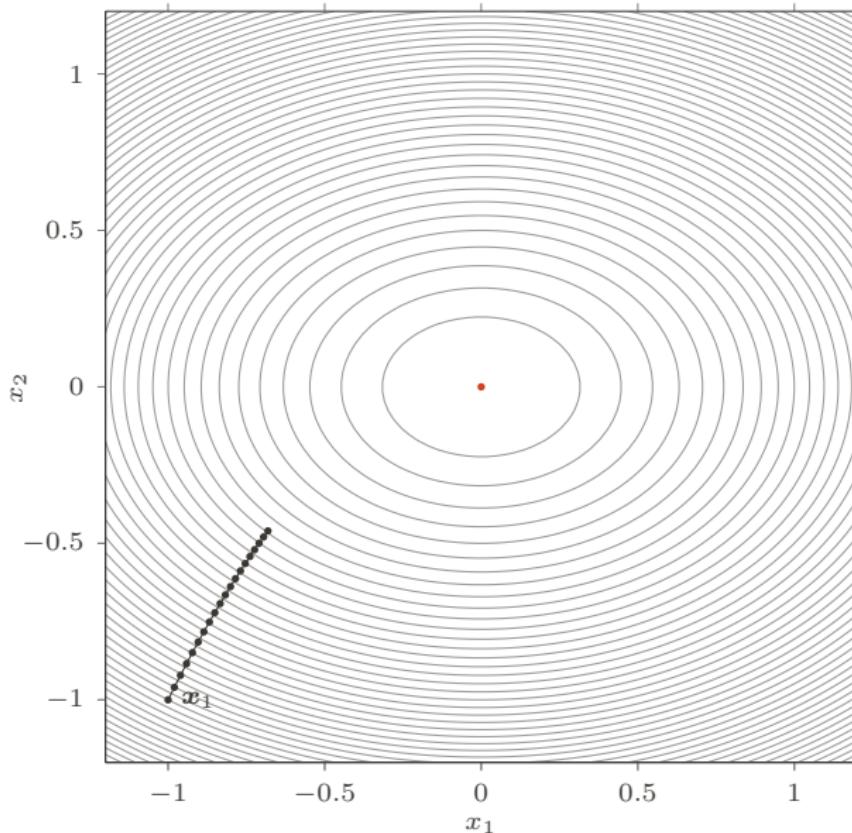


Steepest descent method



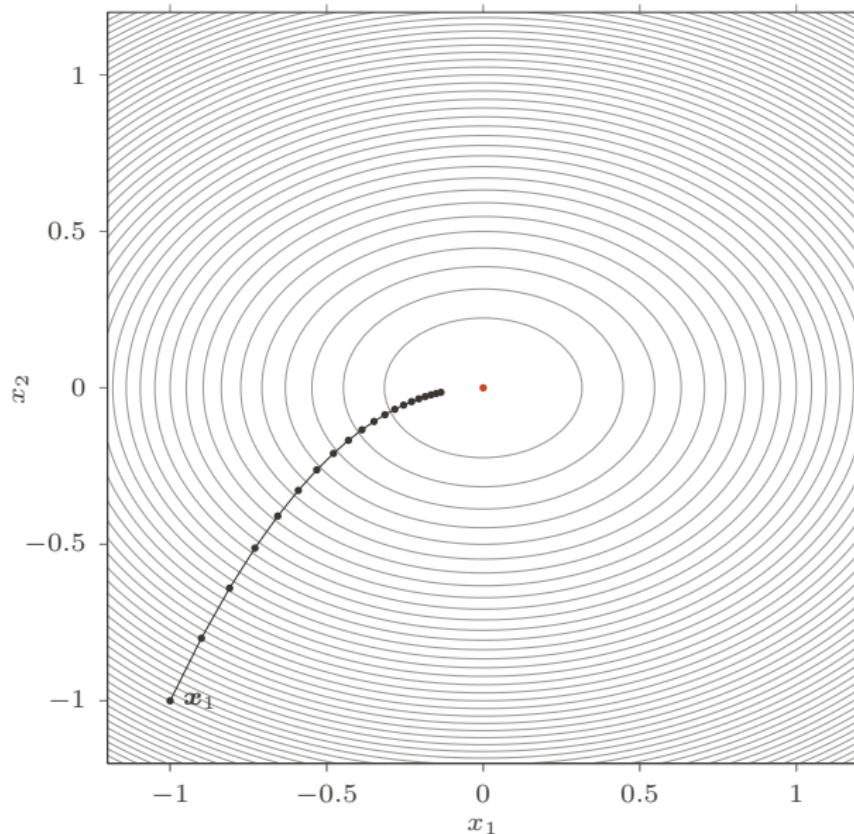
Compare with the constant step size method

$$\alpha = 0.01$$



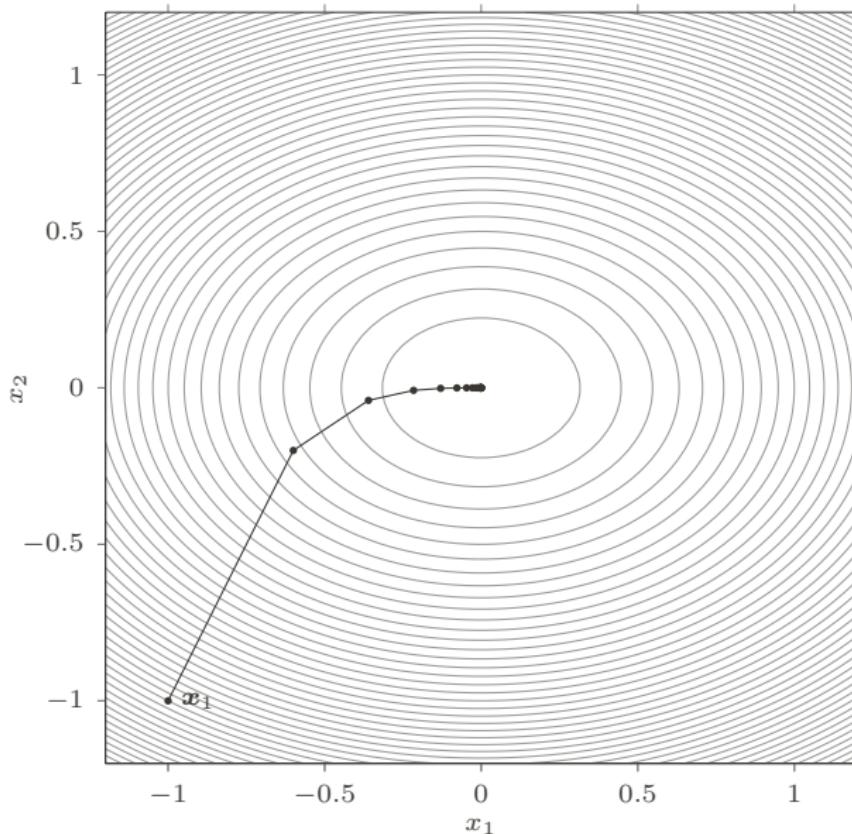
Compare with the constant step size method

$$\alpha = 0.05$$



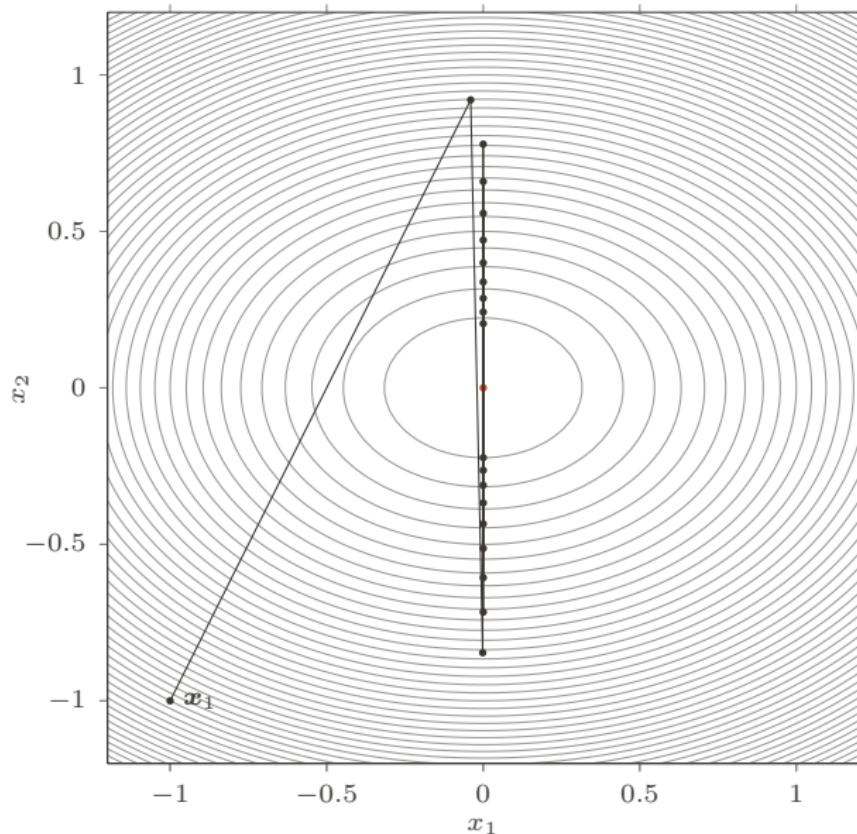
Compare with the constant step size method

$$\alpha = 0.2$$



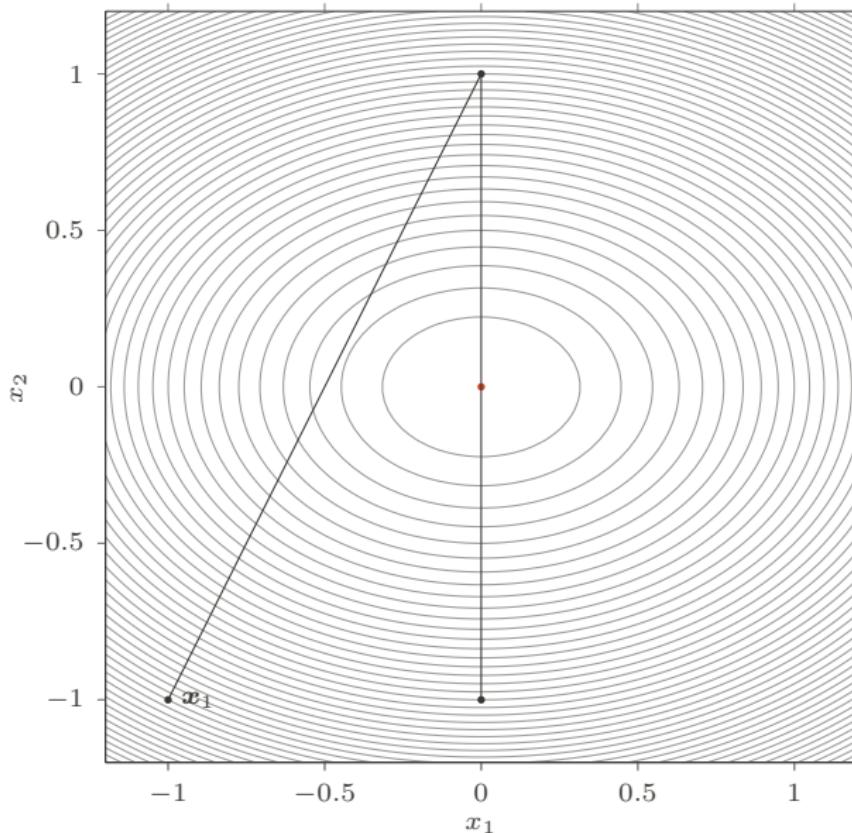
Compare with the constant step size method

$$\alpha = 0.48$$



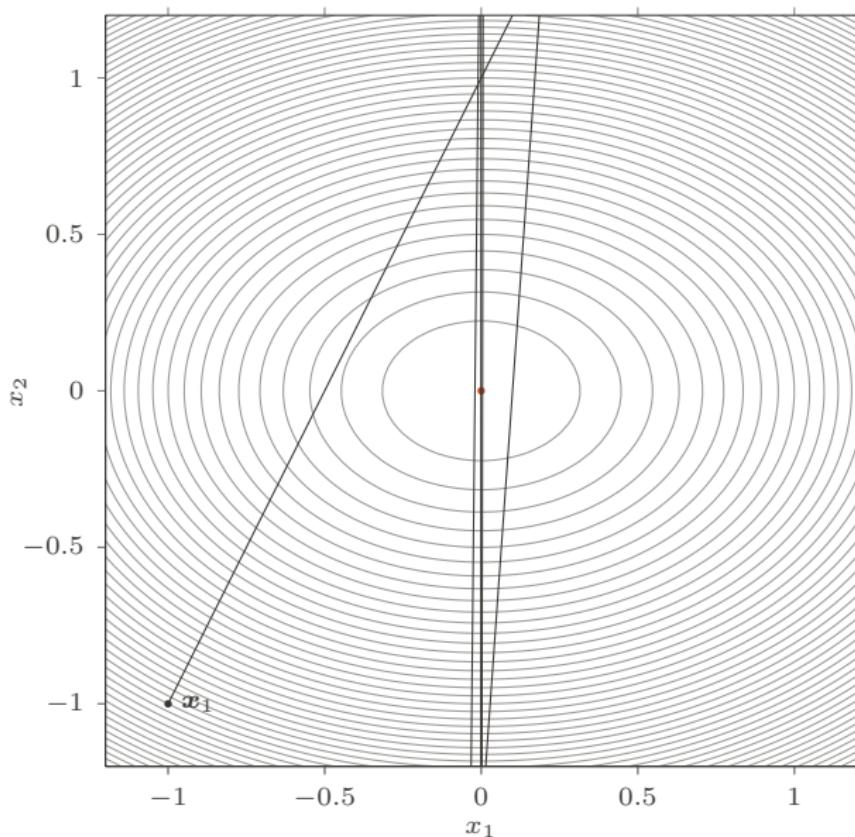
Compare with the constant step size method

$$\alpha = 0.5$$



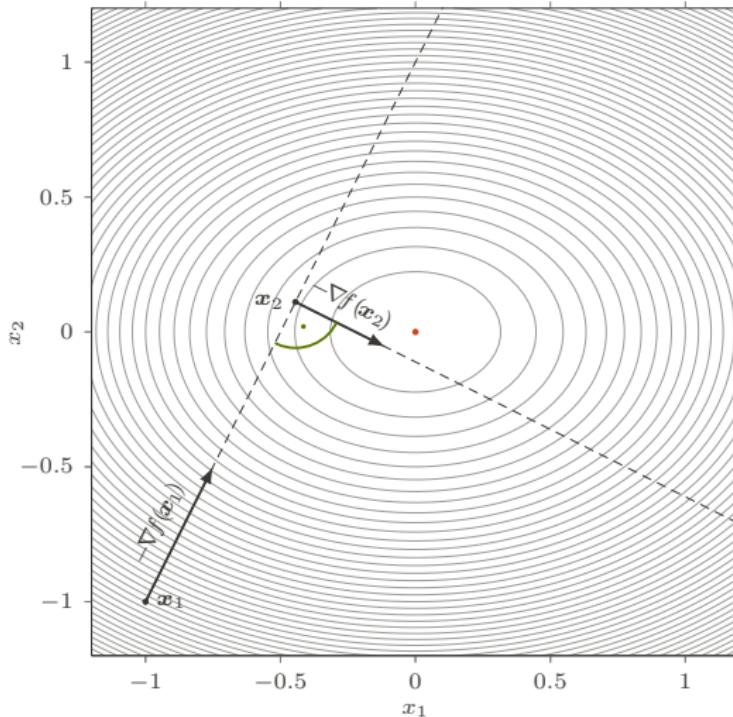
Compare with the constant step size method

$$\alpha = 0.6$$



Zig-zag theorem

In the steepest descent method, the successive descent directions (gradients) are **orthogonal**



Zig-zag theorem

In the steepest descent method, the successive descent directions (gradients) are **orthogonal**

Proof: The step size α_k in the update $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k)$ is obtained by minimizing a univariate function:

$$g(\alpha) = f(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k))$$

Zig-zag theorem

In the steepest descent method, the successive descent directions (gradients) are **orthogonal**

Proof: The step size α_k in the update $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k)$ is obtained by minimizing a univariate function:

$$g(\alpha) = f(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k))$$

We have already shown that:

$$\frac{df(\mathbf{x} + \alpha \mathbf{v})}{d\alpha} = \nabla f(\mathbf{x} + \alpha \mathbf{v})^\top \mathbf{v},$$

therefore:

$$\frac{dg(\alpha)}{d\alpha} = -\nabla f(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k))^\top \nabla f(\mathbf{x}_k)$$

Zig-zag theorem

In the steepest descent method, the successive descent directions (gradients) are **orthogonal**

Proof: The step size α_k in the update $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k)$ is obtained by minimizing a univariate function:

$$g(\alpha) = f(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k))$$

We have already shown that:

$$\frac{df(\mathbf{x} + \alpha \mathbf{v})}{d\alpha} = \nabla f(\mathbf{x} + \alpha \mathbf{v})^\top \mathbf{v},$$

therefore:

$$\frac{dg(\alpha)}{d\alpha} = -\nabla f(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k))^\top \nabla f(\mathbf{x}_k)$$

Since $g(\alpha)$ has a **minimum** at α_k , we must have $\frac{dg(\alpha)}{d\alpha} = 0$ at $\alpha = \alpha_k$.

Zig-zag theorem

In the steepest descent method, the successive descent directions (gradients) are **orthogonal**

Proof: The step size α_k in the update $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k)$ is obtained by minimizing a univariate function:

$$g(\alpha) = f(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k))$$

We have already shown that:

$$\frac{df(\mathbf{x} + \alpha \mathbf{v})}{d\alpha} = \nabla f(\mathbf{x} + \alpha \mathbf{v})^\top \mathbf{v},$$

therefore:

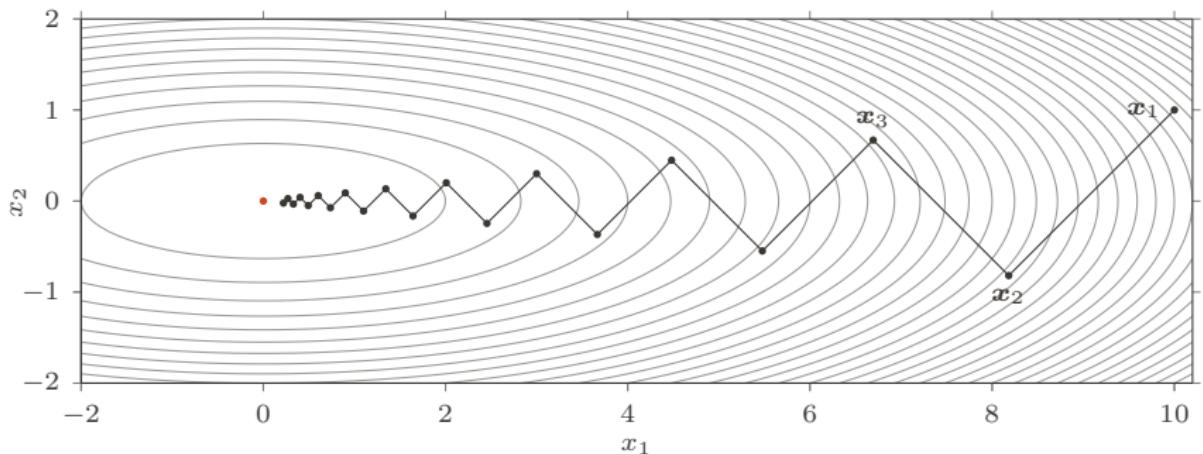
$$\frac{dg(\alpha)}{d\alpha} = -\nabla f(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k))^\top \nabla f(\mathbf{x}_k)$$

Since $g(\alpha)$ has a **minimum** at α_k , we must have $\frac{dg(\alpha)}{d\alpha} = 0$ at $\alpha = \alpha_k$.
But:

$$0 = \left. \frac{dg(\alpha)}{d\alpha} \right|_{\alpha=\alpha_k} = -\nabla f(\mathbf{x}_{k+1})^\top \nabla f(\mathbf{x}_k)$$

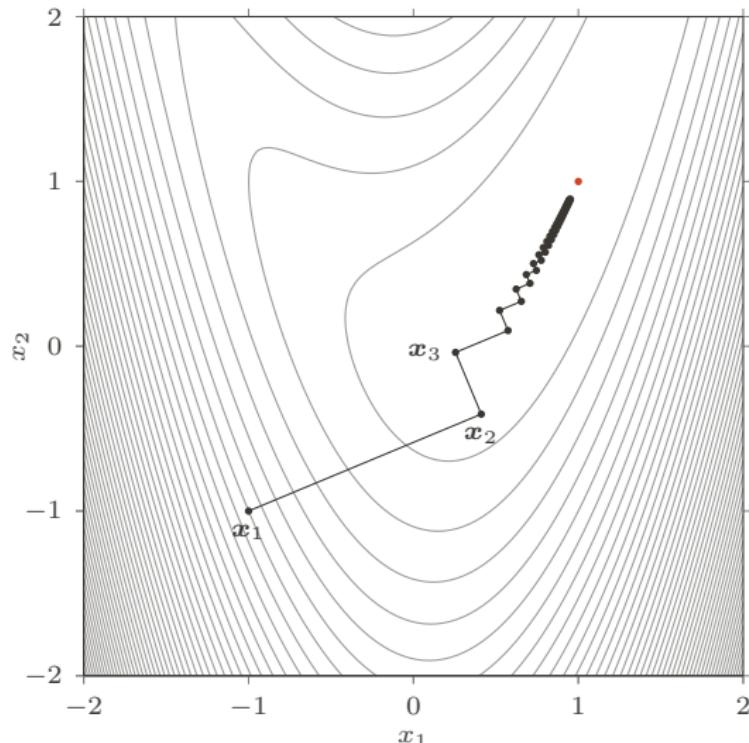
Zig-zagging – example

$$f(x_1, x_2) = x_1^2 + 10x_2^2$$



Zig-zagging – example

$$f(x_1, x_2) = 2.5(x_1^2 - x_2)^2 + (1 - x_1)^2$$



The steepest descent method for a quadratic function*

A convex quadratic function with a **positive-definite** matrix A :

$$f(x) = x^\top Ax + x^\top b + c,$$

The steepest descent method for a quadratic function*

A convex quadratic function with a **positive-definite** matrix A :

$$f(\mathbf{x}) = \mathbf{x}^\top A \mathbf{x} + \mathbf{x}^\top \mathbf{b} + c,$$

Denote the gradient $\nabla f(\mathbf{x}_k) = 2A\mathbf{x}_k + \mathbf{b}$ as \mathbf{g}_k , so

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k, \quad \text{where } \alpha_k = \underset{\alpha \geq 0}{\operatorname{argmin}} \underbrace{f(\mathbf{x}_k - \alpha \mathbf{g}_k)}_{g(\alpha)}$$

The steepest descent method for a quadratic function*

A convex quadratic function with a **positive-definite** matrix A :

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x} + \mathbf{x}^\top \mathbf{b} + c,$$

Denote the gradient $\nabla f(\mathbf{x}_k) = 2\mathbf{A}\mathbf{x}_k + \mathbf{b}$ as \mathbf{g}_k , so

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k, \quad \text{where } \alpha_k = \underset{\alpha \geq 0}{\operatorname{argmin}} \underbrace{f(\mathbf{x}_k - \alpha \mathbf{g}_k)}_{g(\alpha)}$$

$$g(\alpha) = (\mathbf{x}_k - \alpha \mathbf{g}_k)^\top \mathbf{A} (\mathbf{x}_k - \alpha \mathbf{g}_k) + (\mathbf{x}_k - \alpha \mathbf{g}_k)^\top \mathbf{b} + c$$

The steepest descent method for a quadratic function*

A convex quadratic function with a **positive-definite** matrix A :

$$f(\mathbf{x}) = \mathbf{x}^\top A \mathbf{x} + \mathbf{x}^\top \mathbf{b} + c,$$

Denote the gradient $\nabla f(\mathbf{x}_k) = 2A\mathbf{x}_k + \mathbf{b}$ as \mathbf{g}_k , so

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k, \quad \text{where } \alpha_k = \underset{\alpha \geq 0}{\operatorname{argmin}} \underbrace{f(\mathbf{x}_k - \alpha \mathbf{g}_k)}_{g(\alpha)}$$

$$g(\alpha) = \alpha^2 \mathbf{g}_k^\top A \mathbf{g}_k - \alpha \mathbf{g}_k^\top \underbrace{(2A\mathbf{x}_k + \mathbf{b})}_{\mathbf{g}_k} + \text{const}$$

The steepest descent method for a quadratic function*

A convex quadratic function with a **positive-definite** matrix A :

$$f(\mathbf{x}) = \mathbf{x}^\top A \mathbf{x} + \mathbf{x}^\top \mathbf{b} + c,$$

Denote the gradient $\nabla f(\mathbf{x}_k) = 2A\mathbf{x}_k + \mathbf{b}$ as \mathbf{g}_k , so

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k, \quad \text{where } \alpha_k = \underset{\alpha \geq 0}{\operatorname{argmin}} \underbrace{f(\mathbf{x}_k - \alpha \mathbf{g}_k)}_{g(\alpha)}$$

$$g(\alpha) = \alpha^2 \mathbf{g}_k^\top A \mathbf{g}_k - \alpha \|\mathbf{g}_k\|^2 + \text{const}$$

The steepest descent method for a quadratic function*

A convex quadratic function with a positive-definite matrix A :

$g(\alpha)$ is a **quadratic** function:

$$g(\alpha) = a\alpha^2 + b\alpha + c, \quad \text{for } b = -\|\mathbf{g}_k\|^2, \quad a = \mathbf{g}_k^\top A \mathbf{g}_k.$$

Denote the grad

$$\mathbf{x}_{k+1} =$$

$$\text{Minimum at } \alpha = -\frac{b}{2a} = \underbrace{\frac{\|\mathbf{g}_k\|^2}{2\mathbf{g}_k^\top A \mathbf{g}_k}}_{\alpha \geq 0} \quad g(\alpha)$$

$$g(\alpha) = \alpha^2 \mathbf{g}_k^\top A \mathbf{g}_k - \alpha \|\mathbf{g}_k\|^2 + \text{const}$$

The steepest descent method for a quadratic function*

A convex quadratic function with a **positive-definite** matrix A :

$$f(\mathbf{x}) = \mathbf{x}^\top A \mathbf{x} + \mathbf{x}^\top \mathbf{b} + c,$$

Denote the gradient $\nabla f(\mathbf{x}_k) = 2A\mathbf{x}_k + \mathbf{b}$ as \mathbf{g}_k , so

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k, \quad \text{where } \alpha_k = \underset{\alpha \geq 0}{\operatorname{argmin}} \underbrace{f(\mathbf{x}_k - \alpha \mathbf{g}_k)}_{g(\alpha)}$$

$$g(\alpha) = \alpha^2 \mathbf{g}_k^\top A \mathbf{g}_k - \alpha \|\mathbf{g}_k\|^2 + \text{const}$$

Optimal step size: $\alpha_k = \frac{\|\mathbf{g}_k\|^2}{2\mathbf{g}_k^\top A \mathbf{g}_k}$

The steepest descent method for a quadratic function*

A convex quadratic function with a **positive-definite** matrix A :

$$f(\mathbf{x}) = \mathbf{x}^\top A\mathbf{x} + \mathbf{x}^\top \mathbf{b} + c,$$

Denote the gradient $\nabla f(\mathbf{x}_k) = 2A\mathbf{x}_k + \mathbf{b}$ as \mathbf{g}_k , so

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k, \quad \text{where } \alpha_k = \underset{\alpha \geq 0}{\operatorname{argmin}} \underbrace{f(\mathbf{x}_k - \alpha \mathbf{g}_k)}_{g(\alpha)}$$

$$g(\alpha) = \alpha^2 \mathbf{g}_k^\top A \mathbf{g}_k - \alpha \|\mathbf{g}_k\|^2 + \text{const}$$

Optimal step size: $\alpha_k = \frac{\|\mathbf{g}_k\|^2}{2\mathbf{g}_k^\top A \mathbf{g}_k}$

We can also calculate the decrease in the objective function:

$$f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k) = g(\alpha_k) - g(0) = \alpha_k^2 \mathbf{g}_k^\top A \mathbf{g}_k - \alpha_k \|\mathbf{g}_k\|^2$$

The steepest descent method for a quadratic function*

A convex quadratic function with a **positive-definite** matrix A :

$$f(\mathbf{x}) = \mathbf{x}^\top A \mathbf{x} + \mathbf{x}^\top \mathbf{b} + c,$$

Denote the gradient $\nabla f(\mathbf{x}_k) = 2A\mathbf{x}_k + \mathbf{b}$ as \mathbf{g}_k , so

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k, \quad \text{where } \alpha_k = \underset{\alpha \geq 0}{\operatorname{argmin}} \underbrace{f(\mathbf{x}_k - \alpha \mathbf{g}_k)}_{g(\alpha)}$$

$$g(\alpha) = \alpha^2 \mathbf{g}_k^\top A \mathbf{g}_k - \alpha \|\mathbf{g}_k\|^2 + \text{const}$$

Optimal step size: $\alpha_k = \frac{\|\mathbf{g}_k\|^2}{2\mathbf{g}_k^\top A \mathbf{g}_k}$

We can also calculate the decrease in the objective function:

$$f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k) = \frac{1}{4} \frac{\|\mathbf{g}_k\|^4}{\mathbf{g}_k^\top A \mathbf{g}_k} - \frac{1}{2} \frac{\|\mathbf{g}_k\|^4}{\mathbf{g}_k^\top A \mathbf{g}_k}$$

The steepest descent method for a quadratic function*

A convex quadratic function with a **positive-definite** matrix A :

$$f(\mathbf{x}) = \mathbf{x}^\top A\mathbf{x} + \mathbf{x}^\top \mathbf{b} + c,$$

Denote the gradient $\nabla f(\mathbf{x}_k) = 2A\mathbf{x}_k + \mathbf{b}$ as \mathbf{g}_k , so

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k, \quad \text{where } \alpha_k = \underset{\alpha \geq 0}{\operatorname{argmin}} \underbrace{f(\mathbf{x}_k - \alpha \mathbf{g}_k)}_{g(\alpha)}$$

$$g(\alpha) = \alpha^2 \mathbf{g}_k^\top A \mathbf{g}_k - \alpha \|\mathbf{g}_k\|^2 + \text{const}$$

Optimal step size: $\alpha_k = \frac{\|\mathbf{g}_k\|^2}{2\mathbf{g}_k^\top A \mathbf{g}_k}$

We can also calculate the decrease in the objective function:

$$f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k) = -\frac{1}{4} \frac{\|\mathbf{g}_k\|^4}{\mathbf{g}_k^\top A \mathbf{g}_k}$$

The steepest descent method for a quadratic function*

A convex quadratic function with a **positive-definite** matrix A :

$$f(x) = x^\top Ax + x^\top b + c,$$

The objective has a **minimum** at $x^* = -\frac{1}{2}A^{-1}b$

The steepest descent method for a quadratic function*

A convex quadratic function with a **positive-definite** matrix A :

$$f(x) = x^\top Ax + x^\top b + c,$$

The objective has a **minimum** at $x^* = -\frac{1}{2}A^{-1}b$

Representing the objective function by its second-order Taylor polynomial at point x^* :

The steepest descent method for a quadratic function*

A convex quadratic function with a **positive-definite** matrix A :

$$f(x) = \mathbf{x}^\top \mathbf{A} \mathbf{x} + \mathbf{x}^\top \mathbf{b} + c,$$

The objective has a **minimum** at $\mathbf{x}^* = -\frac{1}{2}\mathbf{A}^{-1}\mathbf{b}$

Representing the objective function by its second-order Taylor polynomial at point \mathbf{x}^* :

$$f(\mathbf{x}) - f(\mathbf{x}^*) = (\mathbf{x} - \mathbf{x}^*)^\top \mathbf{A} (\mathbf{x} - \mathbf{x}^*)$$

The steepest descent method for a quadratic function*

A convex quadratic function with a **positive-definite** matrix A :

$$f(x) = x^\top Ax + x^\top b + c,$$

The objective has a **minimum** at $x^* = -\frac{1}{2}A^{-1}b$

Representing the objective function by its second-order Taylor polynomial at point x^* :

$$f(x_k) - f(x^*) = (x_k - x^*)^\top A(x_k - x^*)$$

The steepest descent method for a quadratic function*

A condition $g_k = 2Ax_k + b$, so with a **positive-definite** matrix A :
$$x_k = \frac{1}{2}A^{-1}g_k - \frac{1}{2}A^{-1}b$$
$$x^\top Ax + x^\top b + c,$$

The objective has a **minimum** at $x^* = -\frac{1}{2}A^{-1}b$

Representing the objective function by its second derivative polynomial at point x^* :

$$f(x_k) - f(x^*) = \underbrace{(x_k - x^*)^\top}_{\frac{1}{2}A^{-1}g_k} A \underbrace{(x_k - x^*)}_{\frac{1}{2}A^{-1}g_k}$$

The steepest descent method for a quadratic function*

A convex quadratic function with a **positive-definite** matrix A :

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x} + \mathbf{x}^\top \mathbf{b} + c,$$

The objective has a **minimum** at $\mathbf{x}^* = -\frac{1}{2}\mathbf{A}^{-1}\mathbf{b}$

Representing the objective function by its second-order Taylor polynomial at point \mathbf{x}^* :

$$f(\mathbf{x}_k) - f(\mathbf{x}^*) = \frac{1}{4}(\mathbf{A}^{-1}\mathbf{g}_k)^\top \mathbf{A} \mathbf{A}^{-1}\mathbf{g}_k$$

The steepest descent method for a quadratic function*

A convex quadratic function with a **positive-definite** matrix A :

$$f(x) = x^\top Ax + x^\top b + c,$$

The objective has a **minimum** at $x^* = -\frac{1}{2}A^{-1}b$

Representing the objective function by its second-order Taylor polynomial at point x^* :

$$f(x_k) - f(x^*) = \frac{1}{4}g_k^\top A^{-1}AA^{-1}g_k$$

The steepest descent method for a quadratic function*

A convex quadratic function with a **positive-definite** matrix A :

$$f(x) = x^\top Ax + x^\top b + c,$$

The objective has a **minimum** at $x^* = -\frac{1}{2}A^{-1}b$

Representing the objective function by its second-order Taylor polynomial at point x^* :

$$f(x_k) - f(x^*) = \frac{1}{4}g_k^\top A^{-1}g_k$$

The steepest descent method for a quadratic function*

A convex quadratic function with a **positive-definite** matrix A :

Optimization error: difference between the value
of the objective at \mathbf{x}_k and the optimal value

The objective has

Representing the objective function by its second-order Taylor polynomial
at point \mathbf{x}^* :

$$E(\mathbf{x}_k) = f(\mathbf{x}_k) - f(\mathbf{x}^*) = \frac{1}{4} \mathbf{g}_k^\top \mathbf{A}^{-1} \mathbf{g}_k$$

The steepest descent method for a quadratic function*

A convex quadratic function with a **positive-definite** matrix A :

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x} + \mathbf{x}^\top \mathbf{b} + c,$$

The objective has a **minimum** at $\mathbf{x}^* = -\frac{1}{2}\mathbf{A}^{-1}\mathbf{b}$

Representing the objective function by its second-order Taylor polynomial at point \mathbf{x}^* :

$$E(\mathbf{x}_k) = f(\mathbf{x}_k) - f(\mathbf{x}^*) = \frac{1}{4}\mathbf{g}_k^\top \mathbf{A}^{-1} \mathbf{g}_k$$

Taking equation from the previous slide:

$$f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k) = -\frac{1}{4} \frac{\|\mathbf{g}_k\|^4}{\mathbf{g}_k^\top \mathbf{A} \mathbf{g}_k}$$

The steepest descent method for a quadratic function*

A convex quadratic function with a **positive-definite** matrix A :

$$f(\mathbf{x}) = \mathbf{x}^\top A \mathbf{x} + \mathbf{x}^\top \mathbf{b} + c,$$

The objective has a **minimum** at $\mathbf{x}^* = -\frac{1}{2}A^{-1}\mathbf{b}$

Representing the objective function by its second-order Taylor polynomial at point \mathbf{x}^* :

$$E(\mathbf{x}_k) = f(\mathbf{x}_k) - f(\mathbf{x}^*) = \frac{1}{4} \mathbf{g}_k^\top A^{-1} \mathbf{g}_k$$

Taking equation from the previous slide:

$$\left(f(\mathbf{x}_{k+1}) - f(\mathbf{x}^*) \right) - \left(f(\mathbf{x}_k) - f(\mathbf{x}^*) \right) = -\frac{1}{4} \frac{\|\mathbf{g}_k\|^4}{\mathbf{g}_k^\top A \mathbf{g}_k}$$

The steepest descent method for a quadratic function*

A convex quadratic function with a **positive-definite** matrix A :

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x} + \mathbf{x}^\top \mathbf{b} + c,$$

The objective has a **minimum** at $\mathbf{x}^* = -\frac{1}{2}\mathbf{A}^{-1}\mathbf{b}$

Representing the objective function by its second-order Taylor polynomial at point \mathbf{x}^* :

$$E(\mathbf{x}_k) = f(\mathbf{x}_k) - f(\mathbf{x}^*) = \frac{1}{4} \mathbf{g}_k^\top \mathbf{A}^{-1} \mathbf{g}_k$$

Taking equation from the previous slide:

$$E(\mathbf{x}_{k+1}) - E(\mathbf{x}_k) = -\frac{1}{4} \frac{\|\mathbf{g}_k\|^4}{\mathbf{g}_k^\top \mathbf{A} \mathbf{g}_k}$$

The steepest descent method for a quadratic function*

A convex quadratic function with a **positive-definite** matrix A :

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x} + \mathbf{x}^\top \mathbf{b} + c,$$

The objective has a **minimum** at $\mathbf{x}^* = -\frac{1}{2}\mathbf{A}^{-1}\mathbf{b}$

Representing the objective function by its second-order Taylor polynomial at point \mathbf{x}^* :

$$E(\mathbf{x}_k) = f(\mathbf{x}_k) - f(\mathbf{x}^*) = \frac{1}{4}\mathbf{g}_k^\top \mathbf{A}^{-1} \mathbf{g}_k$$

Taking equation from the previous slide:

$$\frac{E(\mathbf{x}_{k+1}) - E(\mathbf{x}_k)}{E(\mathbf{x}_k)} = -\frac{\|\mathbf{g}_k\|^4}{(\mathbf{g}_k^\top \mathbf{A} \mathbf{g}_k)(\mathbf{g}_k^\top \mathbf{A}^{-1} \mathbf{g}_k)}$$

The steepest descent method for a quadratic function*

A convex quadratic function with a **positive-definite** matrix A :

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x} + \mathbf{x}^\top \mathbf{b} + c,$$

The objective has a **minimum** at $\mathbf{x}^* = -\frac{1}{2}\mathbf{A}^{-1}\mathbf{b}$

Representing the objective function by its second-order Taylor polynomial at point \mathbf{x}^* :

$$E(\mathbf{x}_k) = f(\mathbf{x}_k) - f(\mathbf{x}^*) = \frac{1}{4}\mathbf{g}_k^\top \mathbf{A}^{-1} \mathbf{g}_k$$

Taking equation from the previous slide:

$$\frac{E(\mathbf{x}_{k+1})}{E(\mathbf{x}_k)} = 1 - \frac{\|\mathbf{g}_k\|^4}{(\mathbf{g}_k^\top \mathbf{A} \mathbf{g}_k)(\mathbf{g}_k^\top \mathbf{A}^{-1} \mathbf{g}_k)}$$

The steepest descent method for a quadratic function*

A convex quadratic function with a **positive-definite** matrix A :

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x} + \mathbf{x}^\top \mathbf{b} + c,$$

The objective has a **minimum** at $\mathbf{x}^* = -\frac{1}{2}\mathbf{A}^{-1}\mathbf{b}$

Representing the objective function by its second-order Taylor polynomial at point \mathbf{x}^* :

$$E(\mathbf{x}_k) = f(\mathbf{x}_k) - f(\mathbf{x}^*) = \frac{1}{4}\mathbf{g}_k^\top \mathbf{A}^{-1} \mathbf{g}_k$$

Taking equation from the previous slide:

$$E(\mathbf{x}_{k+1}) = \left(1 - \frac{\|\mathbf{g}_k\|^4}{(\mathbf{g}_k^\top \mathbf{A} \mathbf{g}_k)(\mathbf{g}_k^\top \mathbf{A}^{-1} \mathbf{g}_k)}\right) E(\mathbf{x}_k)$$

The steepest descent method for a quadratic function*

A convex quadratic function with a **positive-definite** matrix A :

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x} + \mathbf{x}^\top \mathbf{b} + c,$$

The objective has a **minimum** at $\mathbf{x}^* = -\frac{1}{2}\mathbf{A}^{-1}\mathbf{b}$

Representing the objective function by its second-order Taylor polynomial at point \mathbf{x}^* :

$$E(\mathbf{x}_k) = f(\mathbf{x}_k) - f(\mathbf{x}^*) = \frac{1}{4}\mathbf{g}_k^\top \mathbf{A}^{-1} \mathbf{g}_k$$

Taking equation from the previous slide:

$$E(\mathbf{x}_{k+1}) = \left(1 - \frac{\|\mathbf{g}_k\|^4}{(\mathbf{g}_k^\top \mathbf{A} \mathbf{g}_k)(\mathbf{g}_k^\top \mathbf{A}^{-1} \mathbf{g}_k)}\right) E(\mathbf{x}_k)$$

Linear convergence of the optimization error!

The steepest descent method for a quadratic function*

Kantorovich inequality:

Let A be a positive-definite matrix with the smallest and the largest eigenvalues denoted by, respectively, m and M . Then, for any non-zero vector x :

$$\frac{\|x\|^4}{(x^\top Ax)(x^\top A^{-1}x)} \geq \frac{4Mm}{(M+m)^2}$$

The steepest descent method for a quadratic function*

Kantorovich inequality:

Let \mathbf{A} be a positive-definite matrix with the smallest and the largest eigenvalues denoted by, respectively, m and M . Then, for any non-zero vector \mathbf{x} :

$$\frac{\|\mathbf{x}\|^4}{(\mathbf{x}^\top \mathbf{A} \mathbf{x})(\mathbf{x}^\top \mathbf{A}^{-1} \mathbf{x})} \geq \frac{4Mm}{(M+m)^2}$$

$$E(\mathbf{x}_{k+1}) = \left(1 - \frac{\|\mathbf{g}_k\|^4}{(\mathbf{g}_k^\top \mathbf{A} \mathbf{g}_k)(\mathbf{g}_k^\top \mathbf{A}^{-1} \mathbf{g}_k)}\right) E(\mathbf{x}_k)$$

The steepest descent method for a quadratic function*

Kantorovich inequality:

Let \mathbf{A} be a positive-definite matrix with the smallest and the largest eigenvalues denoted by, respectively, m and M . Then, for any non-zero vector \mathbf{x} :

$$\frac{\|\mathbf{x}\|^4}{(\mathbf{x}^\top \mathbf{A} \mathbf{x})(\mathbf{x}^\top \mathbf{A}^{-1} \mathbf{x})} \geq \frac{4Mm}{(M+m)^2}$$

$$E(\mathbf{x}_{k+1}) \leq \left(1 - \frac{4Mm}{(M+m)^2}\right) E(\mathbf{x}_k)$$

The steepest descent method for a quadratic function*

Kantorovich inequality:

Let \mathbf{A} be a positive-definite matrix with the smallest and the largest eigenvalues denoted by, respectively, m and M . Then, for any non-zero vector \mathbf{x} :

$$\frac{\|\mathbf{x}\|^4}{(\mathbf{x}^\top \mathbf{A} \mathbf{x})(\mathbf{x}^\top \mathbf{A}^{-1} \mathbf{x})} \geq \frac{4Mm}{(M+m)^2}$$

$$E(\mathbf{x}_{k+1}) \leq \left(\frac{(M+m)^2 - 4Mm}{(M+m)^2} \right) E(\mathbf{x}_k)$$

The steepest descent method for a quadratic function*

Kantorovich inequality:

Let \mathbf{A} be a positive-definite matrix with the smallest and the largest eigenvalues denoted by, respectively, m and M . Then, for any non-zero vector \mathbf{x} :

$$\frac{\|\mathbf{x}\|^4}{(\mathbf{x}^\top \mathbf{A} \mathbf{x})(\mathbf{x}^\top \mathbf{A}^{-1} \mathbf{x})} \geq \frac{4Mm}{(M+m)^2}$$

$$E(\mathbf{x}_{k+1}) \leq \left(\frac{M^2 + 2Mm + m^2 - 4Mm}{(M+m)^2} \right) E(\mathbf{x}_k)$$

The steepest descent method for a quadratic function*

Kantorovich inequality:

Let \mathbf{A} be a positive-definite matrix with the smallest and the largest eigenvalues denoted by, respectively, m and M . Then, for any non-zero vector \mathbf{x} :

$$\frac{\|\mathbf{x}\|^4}{(\mathbf{x}^\top \mathbf{A} \mathbf{x})(\mathbf{x}^\top \mathbf{A}^{-1} \mathbf{x})} \geq \frac{4Mm}{(M+m)^2}$$

$$E(\mathbf{x}_{k+1}) \leq \left(\frac{M-m}{M+m}\right)^2 E(\mathbf{x}_k)$$

The steepest descent method for a quadratic function*

Kantorovich inequality:

Let \mathbf{A} be a positive-definite matrix with the smallest and the largest eigenvalues denoted by, respectively, m and M . Then, for any non-zero vector \mathbf{x} :

$$\frac{\|\mathbf{x}\|^4}{(\mathbf{x}^\top \mathbf{A} \mathbf{x})(\mathbf{x}^\top \mathbf{A}^{-1} \mathbf{x})} \geq \frac{4Mm}{(M+m)^2}$$

$$E(\mathbf{x}_{k+1}) \leq \left(\frac{M-m}{M+m}\right)^{2k} E(\mathbf{x}_1)$$

$$f(\mathbf{x}_{k+1}) - f(\mathbf{x}^*) \leq \beta^k (f(\mathbf{x}_1) - f(\mathbf{x}^*)), \quad \text{where } \beta = \left(\frac{M-m}{M+m}\right)^2$$

The coefficient of linear convergence β depends on the largest and smallest eigenvalues of \mathbf{A}

For $M \gg m$, β is close to one (convergence very slow!)

Condition number

A **condition number** of a positive-definite matrix A is the ratio of its largest and smallest eigenvalues:

$$\kappa = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}$$

Condition number

A **condition number** of a positive-definite matrix A is the ratio of its largest and smallest eigenvalues:

$$\kappa = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}$$

$$\beta = \left(\frac{M - m}{M + m} \right)^2$$

Condition number

A **condition number** of a positive-definite matrix A is the ratio of its largest and smallest eigenvalues:

$$\kappa = \frac{M}{m}$$

$$\kappa = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}$$

$$\beta = \left(\frac{\frac{M}{m} - 1}{\frac{M}{m} + 1} \right)^2$$

Condition number

A **condition number** of a positive-definite matrix A is the ratio of its largest and smallest eigenvalues:

$$\kappa = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}$$

$$\beta = \left(\frac{\kappa - 1}{\kappa + 1} \right)^2$$

Condition number

A **condition number** of a positive-definite matrix \mathbf{A} is the ratio of its largest and smallest eigenvalues:

$$\kappa = \frac{\lambda_{\max}(\mathbf{A})}{\lambda_{\min}(\mathbf{A})}$$

Theorem: For a convex quadratic function with positive-definite matrix \mathbf{A} :

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x} + \mathbf{x}^\top \mathbf{b} + c,$$

the steepest descent method converges **linearly** with the convergence coefficient β depending on the condition number κ of matrix \mathbf{A} :

$$f(\mathbf{x}_{k+1}) - f(\mathbf{x}^*) \leq \beta^k (f(\mathbf{x}_1) - f(\mathbf{x}^*)), \quad \text{where } \beta = \left(\frac{\kappa - 1}{\kappa + 1} \right)^2$$

The larger the condition number κ , the slower the convergence

Convergence on the steepest descent – example

For $\gamma > 1$ consider a function:

$$f(\mathbf{x}) = x_1^2 + \gamma x_2^2, \quad \mathbf{x} = (x_1, x_2),$$

with a minimum at $\mathbf{x}^* = (0, 0)$

Convergence on the steepest descent – example

For $\gamma > 1$ consider a function:

$$f(\mathbf{x}) = x_1^2 + \gamma x_2^2, \quad \mathbf{x} = (x_1, x_2),$$

with a minimum at $\mathbf{x}^* = (0, 0)$

The function can be rewritten as:

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x}, \quad \text{where } \mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & \gamma \end{bmatrix}$$

Convergence on the steepest descent – example

For $\gamma > 1$ consider a function:

$$f(\mathbf{x}) = x_1^2 + \gamma x_2^2, \quad \mathbf{x} = (x_1, x_2),$$

with a minimum at $\mathbf{x}^* = (0, 0)$

The function can be rewritten as:

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x}, \quad \text{where } \mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & \gamma \end{bmatrix}$$

The eigenvalues are $(1, \gamma)$, and therefore:

$$\kappa = \frac{\gamma}{1} = \gamma, \quad \beta = \left(\frac{\gamma - 1}{\gamma + 1} \right)^2$$

Convergence on the steepest descent – example

For $\gamma > 1$ consider a function:

$$f(\mathbf{x}) = x_1^2 + \gamma x_2^2, \quad \mathbf{x} = (x_1, x_2),$$

with a minimum at $\mathbf{x}^* = (0, 0)$

The function can be rewritten as:

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x}, \quad \text{where } \mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & \gamma \end{bmatrix}$$

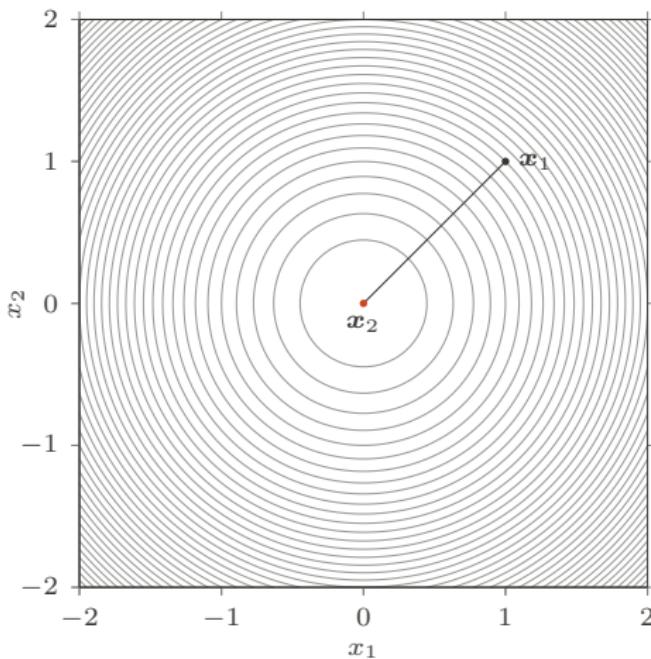
The eigenvalues are $(1, \gamma)$, and therefore:

$$\kappa = \frac{\gamma}{1} = \gamma, \quad \beta = \left(\frac{\gamma - 1}{\gamma + 1} \right)^2$$

Let us look at the convergence of the steepest descent method if we start from point $\mathbf{x}_1 = (\gamma, 1)$.

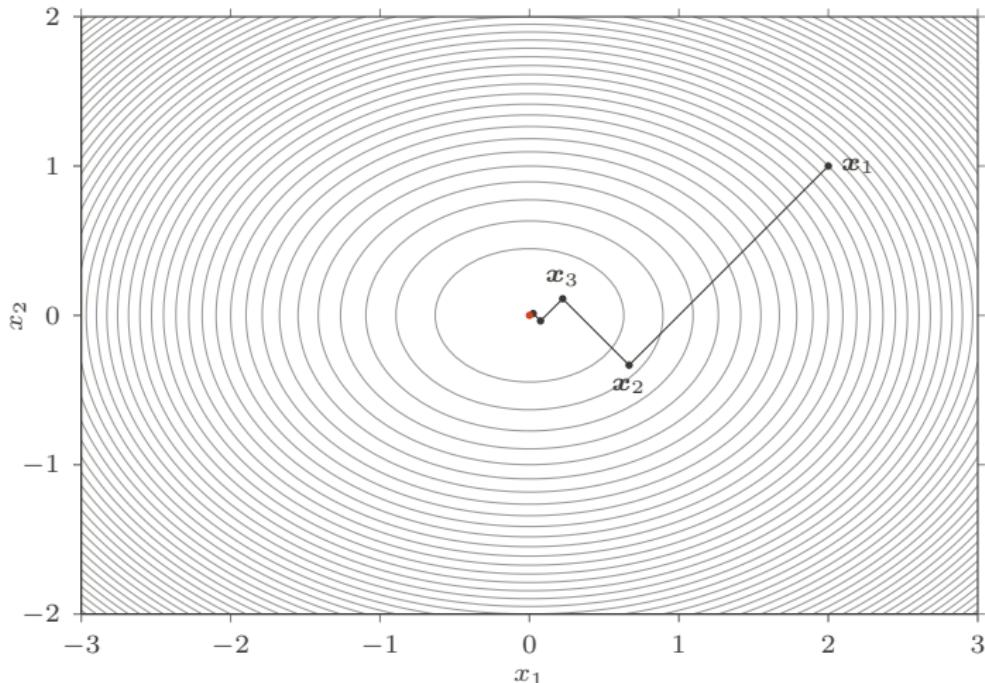
Example

$$f(x_1, x_2) = x_1^2 + x_2^2, \quad (\gamma = 1)$$



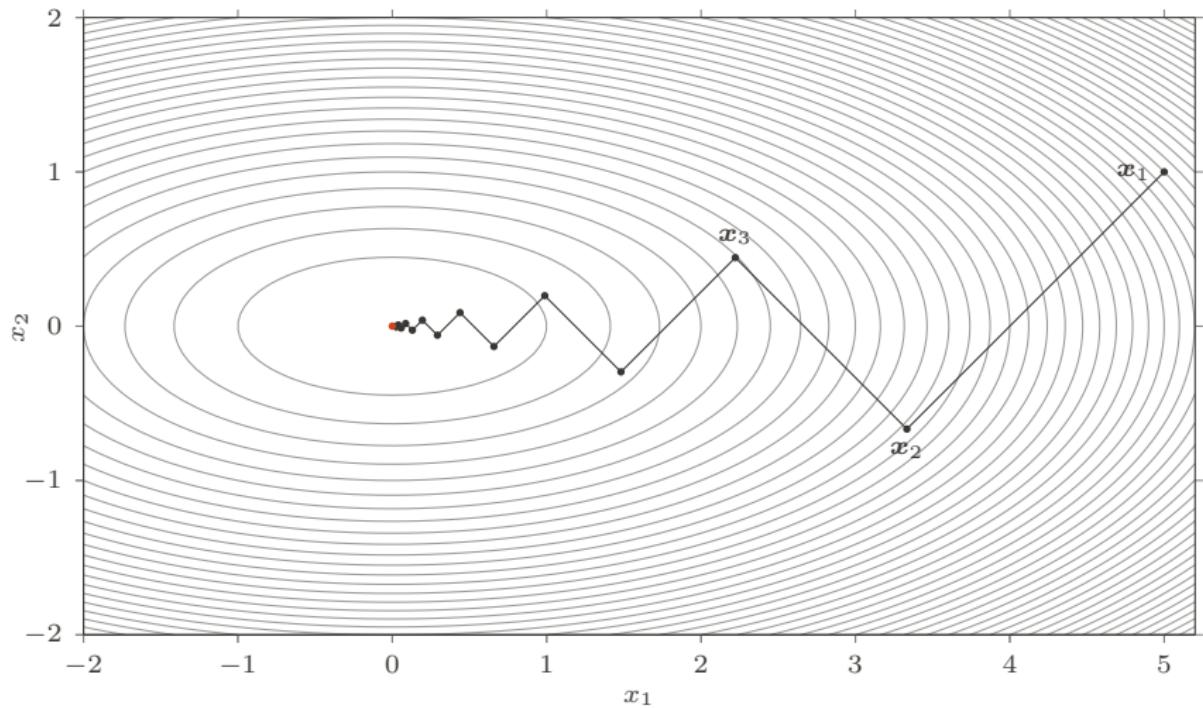
Example

$$f(x_1, x_2) = x_1^2 + 2x_2^2, \quad (\gamma = 2)$$



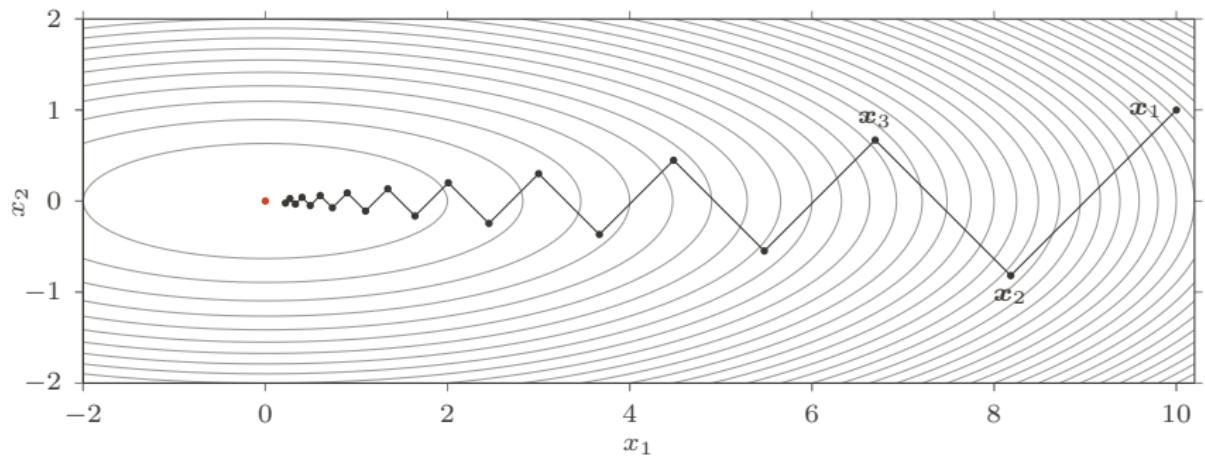
Example

$$f(x_1, x_2) = x_1^2 + 5x_2^2, \quad (\gamma = 5)$$



Example

$$f(x_1, x_2) = x_1^2 + 10x_2^2, \quad (\gamma = 10)$$



Convergence of the steepest descent method for convex functions

Let $f(x)$ be **convex** and **continuously twice differentiable**. Assume the Hessian $\nabla^2 f(x^*)$ at minimum x^* is **positive-definite**. Then, the steepest descent method converges linearly (in the optimization error sense) with the sequence of coefficients:

$$\beta_k \xrightarrow{k \rightarrow \infty} \beta = \left(\frac{\kappa - 1}{\kappa + 1} \right)^2$$

where κ is the condition number of matrix $\nabla^2 f(x^*)$

Step size choice: Armijo's rule (backtracking)

Motivation:

- An optimal choice of the step size requires solving a univariate optimization problem in each iteration, which can be computationally expensive
- On the other hand, a constant step size does not even guarantee the decrease in the objective, and must be tuned quite accurately to even work at all

We want to have a fast method of choosing the step sizes, which at the same time guarantees the decrease in the objective

Approximate minimization

The goal is to approximately minimize a univariate function

$$\min_{\alpha \geq 0} g(\alpha), \quad \text{where } g(\alpha) = f(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k))$$

Approximate minimization

The goal is to approximately minimize a univariate function

$$\min_{\alpha \geq 0} g(\alpha), \quad \text{where } g(\alpha) = f(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k))$$

We showed before (“zig-zag theorem”) that

$$g'(\alpha) = -\nabla f(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k))^{\top} \nabla f(\mathbf{x}_k),$$

which means that $g'(0) = -\|\nabla f(\mathbf{x}_k)\|^2 < 0$

Approximate minimization

The goal is to approximately minimize a univariate function

$$\min_{\alpha \geq 0} g(\alpha), \quad \text{where } g(\alpha) = f(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k))$$

We showed before (“zig-zag theorem”) that

$$g'(\alpha) = -\nabla f(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k))^{\top} \nabla f(\mathbf{x}_k),$$

which means that $g'(0) = -\|\nabla f(\mathbf{x}_k)\|^2 < 0$

Consequently, $g(\alpha)$ is **decreasing** around $\alpha = 0$, so taking a **very small** α guarantees the decrease, $g(\alpha) < g(0)$

Approximate minimization

The goal is to approximately minimize a univariate function

$$\min_{\alpha \geq 0} g(\alpha), \quad \text{where } g(\alpha) = f(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k))$$

We showed before (“zig-zag theorem”) that

$$g'(\alpha) = -\nabla f(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k))^{\top} \nabla f(\mathbf{x}_k),$$

which means that $g'(0) = -\|\nabla f(\mathbf{x}_k)\|^2 < 0$

Consequently, $g(\alpha)$ is **decreasing** around $\alpha = 0$, so taking a **very small** α guarantees the decrease, $g(\alpha) < g(0)$

Unfortunately, the decrease will be very small, and the algorithm will converge very slowly

Armijo's rule (backtracking)

- Consider a function $g(\alpha)$ with $g'(0) < 0$. We find $\alpha > 0$, for which the decrease in the function value $g(0) - g(\alpha)$ is **significant**

Armijo's rule (backtracking)

- Consider a function $g(\alpha)$ with $g'(0) < 0$. We find $\alpha > 0$, for which the decrease in the function value $g(0) - g(\alpha)$ is **significant**
- From Taylor's theorem:

$$g(\alpha) = g(0) + \alpha g'(0) + O(\alpha^2)$$

Armijo's rule (backtracking)

- Consider a function $g(\alpha)$ with $g'(0) < 0$. We find $\alpha > 0$, for which the decrease in the function value $g(0) - g(\alpha)$ is **significant**
- From Taylor's theorem:

$$g(\alpha) = g(0) + \alpha g'(0) + O(\alpha^2) \leq g(0) + \gamma \alpha g'(0),$$

for some $\gamma \in (0, 1)$ and sufficiently small α

Armijo's rule (backtracking)

- Consider a function $g(\alpha)$ with $g'(0) < 0$. We find $\alpha > 0$, for which the decrease in the function value $g(0) - g(\alpha)$ is **significant**
- From Taylor's theorem:

$$g(\alpha) = g(0) + \alpha g'(0) + O(\alpha^2) \leq g(0) + \gamma \alpha g'(0),$$

for some $\gamma \in (0, 1)$ and sufficiently small α

- We search for α **as large as possible**, still satisfying **Armijo's rule**:

$$g(\alpha) \leq g(0) + \gamma \alpha g'(0), \quad \text{for fixed } \gamma \in (0, 1)$$

Armijo's rule (backtracking)

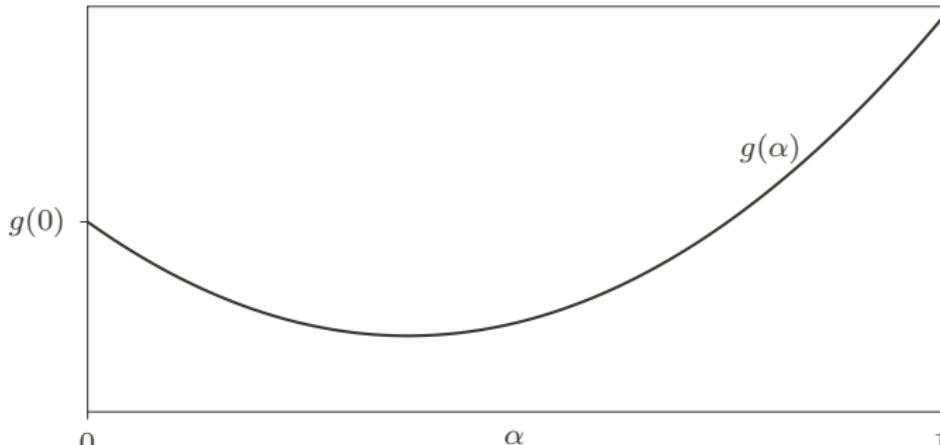
- Consider a function $g(\alpha)$ with $g'(0) < 0$. We find $\alpha > 0$, for which the decrease in the function value $g(0) - g(\alpha)$ is **significant**
- From Taylor's theorem:

$$g(\alpha) = g(0) + \alpha g'(0) + O(\alpha^2) \leq g(0) + \gamma \alpha g'(0),$$

for some $\gamma \in (0, 1)$ and sufficiently small α

- We search for α **as large as possible**, still satisfying **Armijo's rule**:

$$g(\alpha) \leq g(0) + \gamma \alpha g'(0), \quad \text{for fixed } \gamma \in (0, 1)$$



Armijo's rule (backtracking)

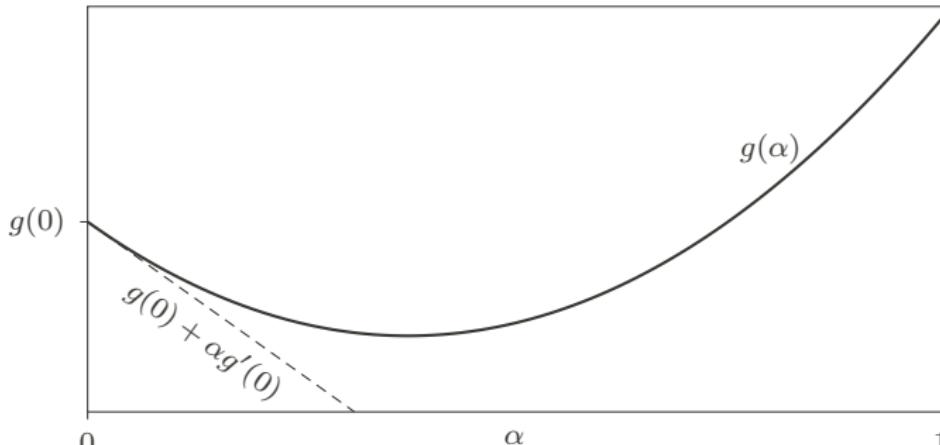
- Consider a function $g(\alpha)$ with $g'(0) < 0$. We find $\alpha > 0$, for which the decrease in the function value $g(0) - g(\alpha)$ is **significant**
- From Taylor's theorem:

$$g(\alpha) = g(0) + \alpha g'(0) + O(\alpha^2) \leq g(0) + \gamma \alpha g'(0),$$

for some $\gamma \in (0, 1)$ and sufficiently small α

- We search for α **as large as possible**, still satisfying **Armijo's rule**:

$$g(\alpha) \leq g(0) + \gamma \alpha g'(0), \quad \text{for fixed } \gamma \in (0, 1)$$



Armijo's rule (backtracking)

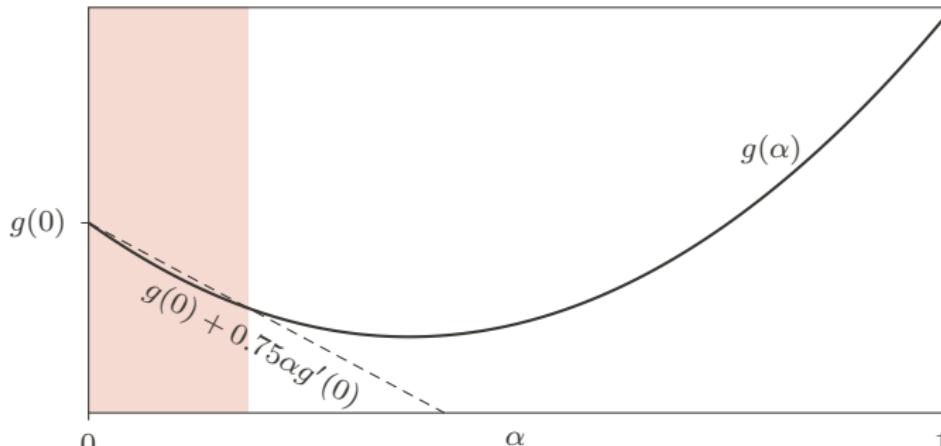
- Consider a function $g(\alpha)$ with $g'(0) < 0$. We find $\alpha > 0$, for which the decrease in the function value $g(0) - g(\alpha)$ is **significant**
- From Taylor's theorem:

$$g(\alpha) = g(0) + \alpha g'(0) + O(\alpha^2) \leq g(0) + \gamma \alpha g'(0),$$

for some $\gamma \in (0, 1)$ and sufficiently small α

- We search for α **as large as possible**, still satisfying **Armijo's rule**:

$$g(\alpha) \leq g(0) + \gamma \alpha g'(0), \quad \text{for fixed } \gamma \in (0, 1)$$



Armijo's rule (backtracking)

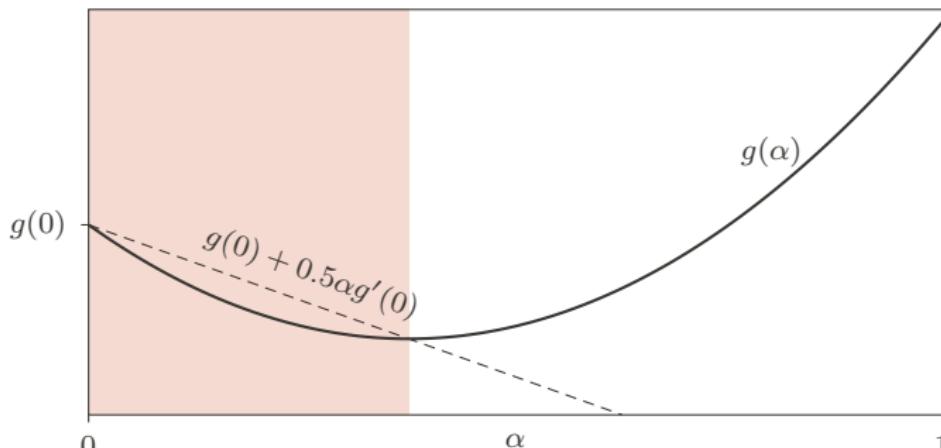
- Consider a function $g(\alpha)$ with $g'(0) < 0$. We find $\alpha > 0$, for which the decrease in the function value $g(0) - g(\alpha)$ is **significant**
- From Taylor's theorem:

$$g(\alpha) = g(0) + \alpha g'(0) + O(\alpha^2) \leq g(0) + \gamma \alpha g'(0),$$

for some $\gamma \in (0, 1)$ and sufficiently small α

- We search for α **as large as possible**, still satisfying **Armijo's rule**:

$$g(\alpha) \leq g(0) + \gamma \alpha g'(0), \quad \text{for fixed } \gamma \in (0, 1)$$



Armijo's rule (backtracking)

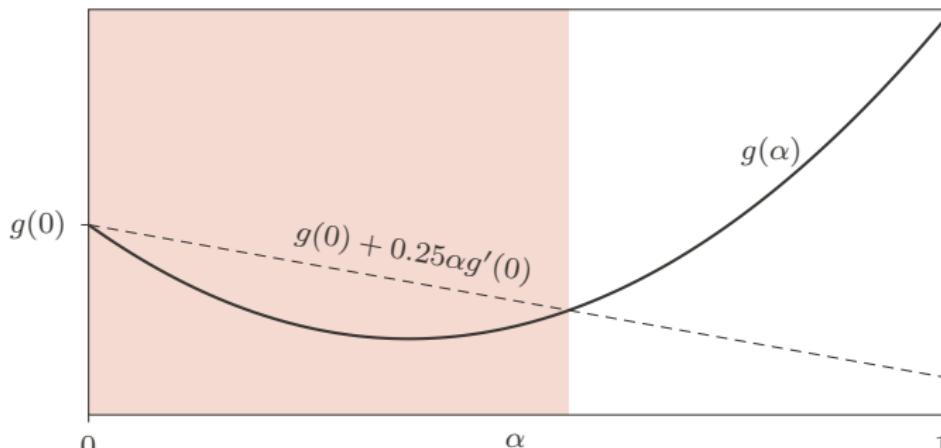
- Consider a function $g(\alpha)$ with $g'(0) < 0$. We find $\alpha > 0$, for which the decrease in the function value $g(0) - g(\alpha)$ is **significant**
- From Taylor's theorem:

$$g(\alpha) = g(0) + \alpha g'(0) + O(\alpha^2) \leq g(0) + \gamma \alpha g'(0),$$

for some $\gamma \in (0, 1)$ and sufficiently small α

- We search for α **as large as possible**, still satisfying **Armijo's rule**:

$$g(\alpha) \leq g(0) + \gamma \alpha g'(0), \quad \text{for fixed } \gamma \in (0, 1)$$



Armijo's rule (backtracking)

Start with a large value $\alpha = \alpha_{\max}$ and reduce α by multiplying by a constant $\eta \in (0, 1)$ as long as the Armijo's rule is violated

Input: Parameters: $\alpha_{\max} > 0$, $\gamma \in (0, 1)$ and $\eta \in (0, 1)$

Initialize: $\alpha = \alpha_{\max}$

As long as $g(\alpha) > g(0) + \gamma \alpha g'(0)$, **repeat**

$\alpha \leftarrow \eta \alpha$

Return step size $\alpha_k = \alpha$

Armijo's rule (backtracking)

Start with a large value $\alpha = \alpha_{\max}$ and reduce α by multiplying by a constant $\eta \in (0, 1)$ as long as the Armijo's rule is violated

Input: Parameters: $\alpha_{\max} > 0$, $\gamma \in (0, 1)$ and $\eta \in (0, 1)$

Initialize: $\alpha = \alpha_{\max}$

As long as $f(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k)) > f(\mathbf{x}_k) - \alpha \gamma \|\nabla f(\mathbf{x}_k)\|^2$, **repeat**
 $\alpha \leftarrow \eta \alpha$

Return step size $\alpha_k = \alpha$

Armijo's rule (backtracking)

Start with a large value $\alpha = \alpha_{\max}$ and reduce α by multiplying by a constant $\eta \in (0, 1)$ as long as the Armijo's rule is violated

Input: Parameters: $\alpha_{\max} > 0$, $\gamma \in (0, 1)$ and $\eta \in (0, 1)$

Initialize: $\alpha = \alpha_{\max}$

As long as $f(\mathbf{x}_k - \alpha \nabla f(\mathbf{x}_k)) > f(\mathbf{x}_k) - \alpha \gamma \|\nabla f(\mathbf{x}_k)\|^2$, **repeat**
 $\alpha \leftarrow \eta \alpha$

Return step size $\alpha_k = \alpha$

The resulting step size α_k guarantees:

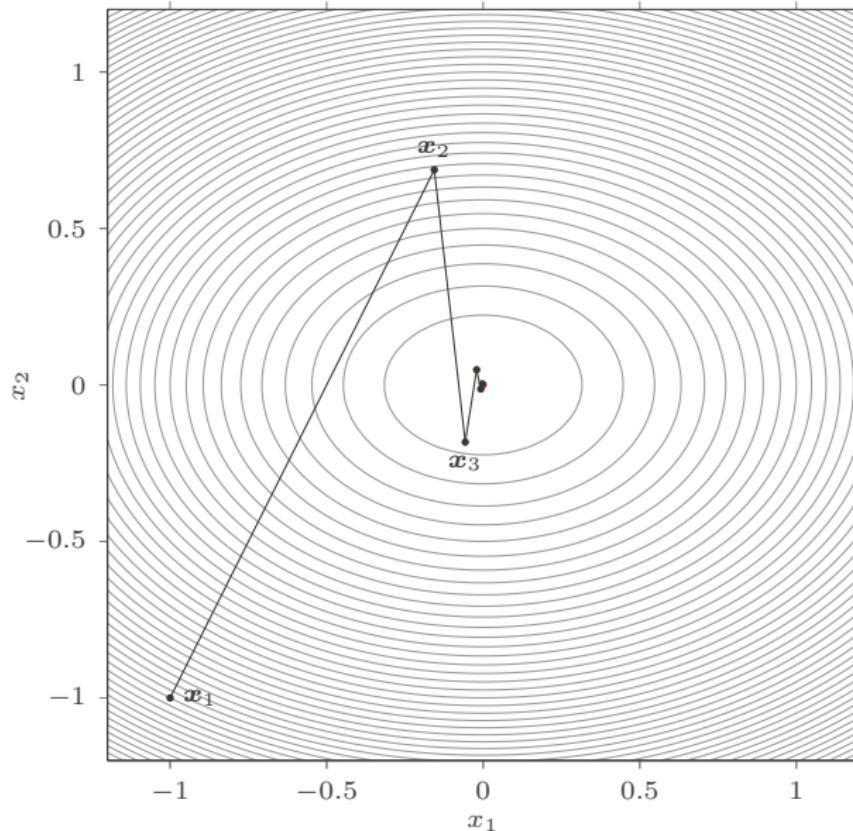
$$f(\mathbf{x}_k) - f(\mathbf{x}_{k+1}) \geq \alpha_k \gamma \|\nabla f(\mathbf{x}_k)\|^2$$

Typical parameter values:

- $\alpha_{\max} \simeq 1$
- $\gamma \in [0.01, 0.3]$
- $\eta \in [0.1, 0.8]$ (determines the accuracy of the optimization, a trade-off between the quality of the solution and computational complexity)

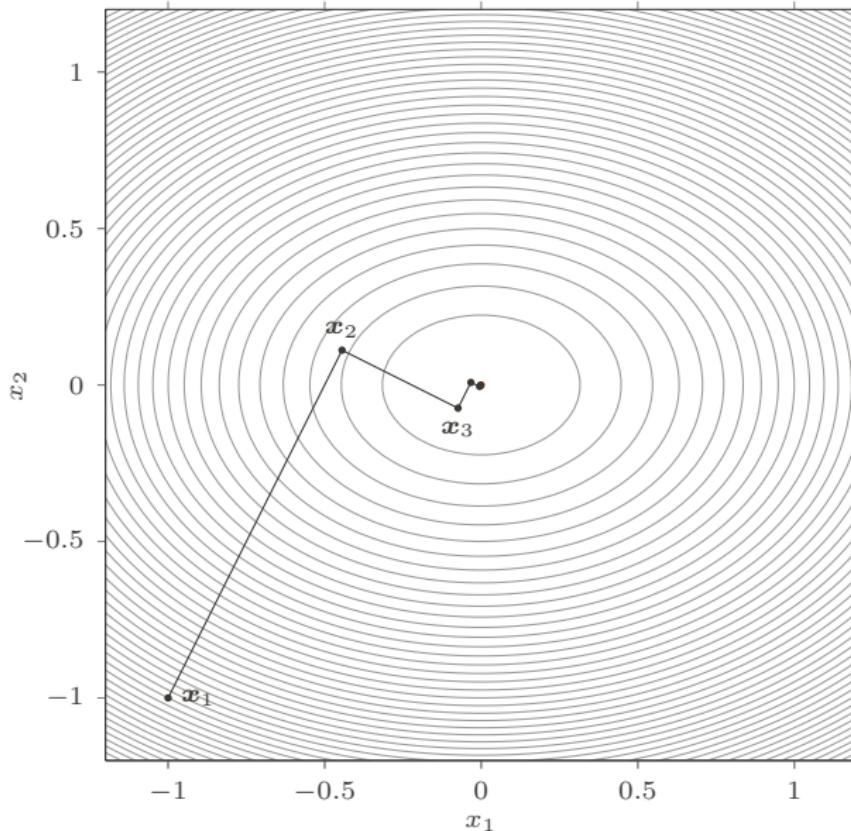
Backtracking for $\alpha_{\max} = 1$, $\gamma = 0.2$, $\eta = 0.75$

$$f(x_1, x_2) = x_1^2 + 2x_2^2$$



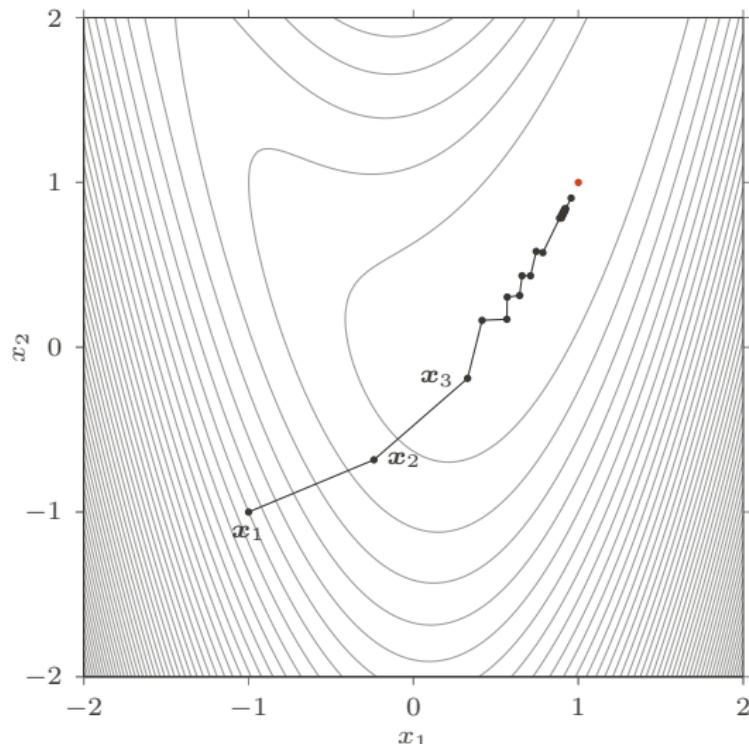
Compare with the steepest descent method

$$f(x_1, x_2) = x_1^2 + 2x_2^2$$



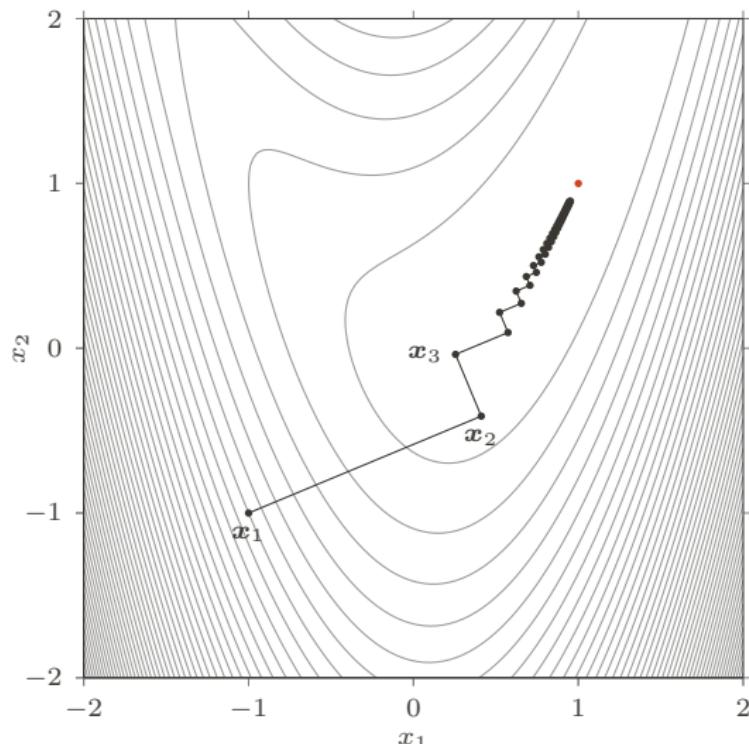
Backtracking for $\alpha_{\max} = 1$, $\gamma = 0.5$, $\eta = 0.75$

$$f(x_1, x_2) = 2.5(x_1^2 - x_2)^2 + (1 - x_1)^2$$



Compare with the steepest descent method

$$f(x_1, x_2) = 2.5(x_1^2 - x_2)^2 + (1 - x_1)^2$$



Optimization Methods for Data Analysis

5. Descent methods (Newton-Raphson method)

Wojciech Kotłowski

Institute of Computing Science, PUT
<http://www.cs.put.poznan.pl/wkotlowski/>

12.04.2022

Outline

Unconstrained minimization of a **differentiable** function $f(x)$:

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

We will often assume the **convexity** of $f(x)$

1. Introduction (previous lecture)
2. Gradient descent method (previous lecture)
3. Newton-Raphson method
4. Modifications of Newton-Rapshon
5. Conjugate gradients method (next lecture)

Reminder: descent methods

- Iterative methods generating a sequence of solutions x_1, x_2, \dots according to:

$$x_{k+1} = x_k + \alpha_k v_k,$$

- Direction v_k is a **descent direction**: guarantees that with sufficient small $\alpha \in \mathbb{R}$:

$$f(x_k + \alpha v_k) < f(x_k)$$

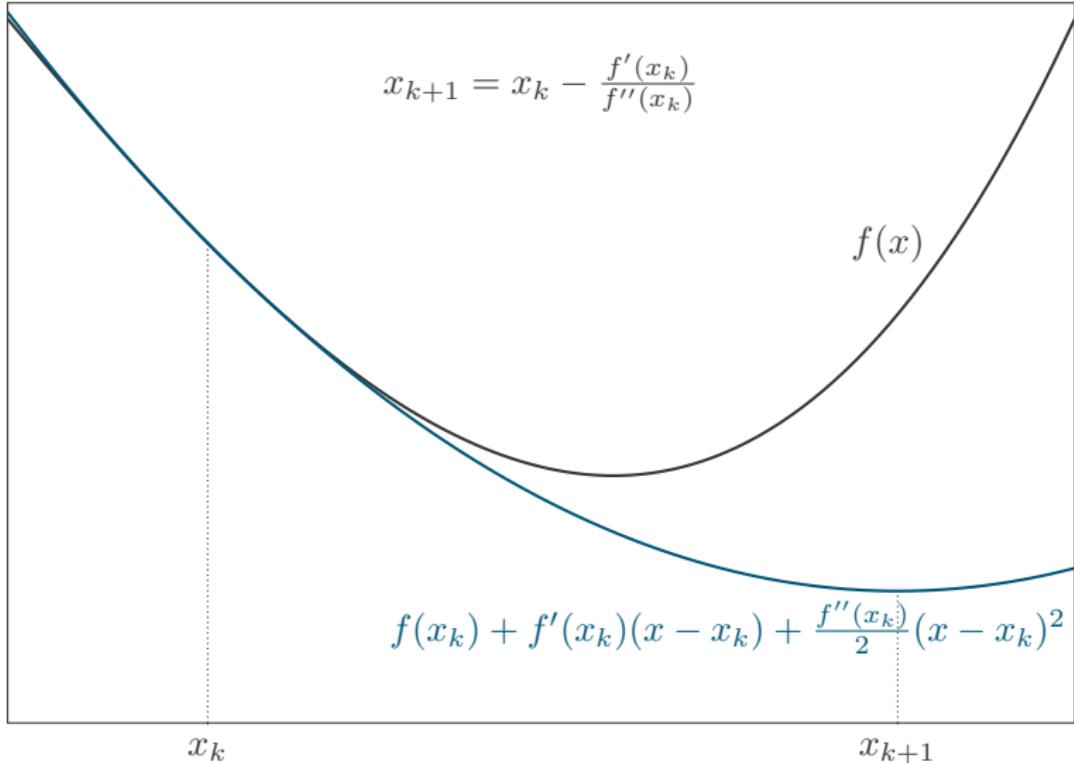
Equivalently, the **directional derivative** along v_k must be **negative**:

$$\frac{\partial f(x_k)}{\partial v_k} = \nabla f(x_k)^\top v_k < 0$$

- **Step size** α_k chosen to obtain a sufficient improvement (decrease) of the objective function.

Newton-Raphson method

Remainder: Newton method



Newton-Raphson method

Multivariate version of the Newton method

Iteratively generates a sequence of solutions x_1, x_2, \dots

In each iteration $k = 1, 2, \dots$, approximate $f(x)$ with its **second-order Taylor polynomial** at current solution x_k :

$$P_2(x; x_k) = f(x_k) + \nabla f(x_k)^\top (x - x_k) + \frac{1}{2}(x - x_k)^\top \nabla^2 f(x_k)(x - x_k)$$

Assumption: the Hessian $\nabla^2 f(x_k)$ is **positive definite**

As the next point x_{k+1} we take the minimizer of $P_2(x; x_k)$

Newton-Raphson method

Remainder: the minimizer of a quadratic function $x^\top Ax + b^\top x + c$ with positive-definite A is given by: $x^* = -\frac{1}{2}A^{-1}b$

Newton-Raphson method

Remainder: the minimizer of a quadratic function $x^\top Ax + b^\top x + c$ with positive-definite A is given by: $x^* = -\frac{1}{2}A^{-1}b$

Conclusion: the minimizer of a quadratic function

$$F(v) = \frac{1}{2}v^\top \nabla^2 f(x_k)v + \nabla f(x_k)^\top v + f(x_k)$$

is $v = -(\nabla^2 f(x_k))^{-1} \nabla f(x_k)$

Newton-Raphson method

Remainder: the minimizer of a quadratic function $\mathbf{x}^\top \mathbf{A} \mathbf{x} + \mathbf{b}^\top \mathbf{x} + c$ with positive-definite \mathbf{A} is given by: $\mathbf{x}^* = -\frac{1}{2}\mathbf{A}^{-1}\mathbf{b}$

Conclusion: the minimizer of a quadratic function

$$F(\mathbf{v}) = \frac{1}{2}\mathbf{v}^\top \nabla^2 f(\mathbf{x}_k) \mathbf{v} + \nabla f(\mathbf{x}_k)^\top \mathbf{v} + f(\mathbf{x}_k)$$

is $\mathbf{v} = -(\nabla^2 f(\mathbf{x}_k))^{-1} \nabla f(\mathbf{x}_k)$

Conclusion: Taking $\mathbf{v} = \mathbf{x} - \mathbf{x}_k$, the minimizer of

$$P_2(\mathbf{x}; \mathbf{x}_k) = f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^\top (\mathbf{x} - \mathbf{x}_k) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_k)^\top \nabla^2 f(\mathbf{x}_k)(\mathbf{x} - \mathbf{x}_k)$$

satisfies:

$$\mathbf{x} - \mathbf{x}_k = -(\nabla^2 f(\mathbf{x}_k))^{-1} \nabla f(\mathbf{x}_k)$$

Newton-Raphson method

Remainder: the minimizer of a quadratic function $\mathbf{x}^\top \mathbf{A} \mathbf{x} + \mathbf{b}^\top \mathbf{x} + c$ with positive-definite \mathbf{A} is given by: $\mathbf{x}^* = -\frac{1}{2}\mathbf{A}^{-1}\mathbf{b}$

Conclusion: the minimizer of a quadratic function

$$F(\mathbf{v}) = \frac{1}{2}\mathbf{v}^\top \nabla^2 f(\mathbf{x}_k) \mathbf{v} + \nabla f(\mathbf{x}_k)^\top \mathbf{v} + f(\mathbf{x}_k)$$

is $\mathbf{v} = -(\nabla^2 f(\mathbf{x}_k))^{-1} \nabla f(\mathbf{x}_k)$

Conclusion: Taking $\mathbf{v} = \mathbf{x} - \mathbf{x}_k$, the minimizer of

$$P_2(\mathbf{x}; \mathbf{x}_k) = f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^\top (\mathbf{x} - \mathbf{x}_k) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_k)^\top \nabla^2 f(\mathbf{x}_k)(\mathbf{x} - \mathbf{x}_k)$$

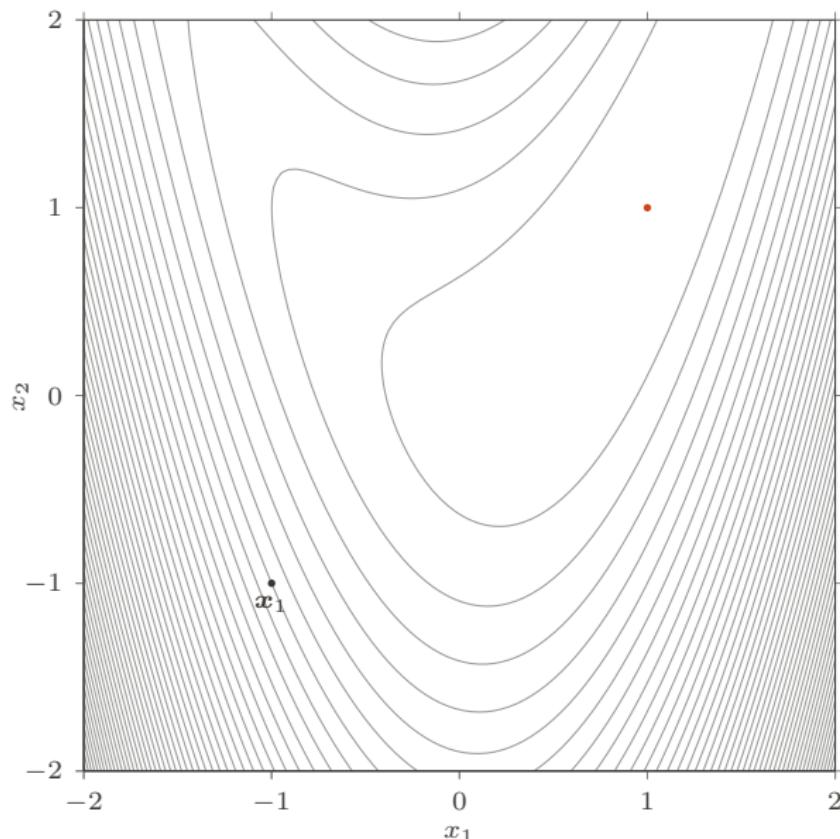
satisfies:

$$\mathbf{x} - \mathbf{x}_k = -(\nabla^2 f(\mathbf{x}_k))^{-1} \nabla f(\mathbf{x}_k)$$

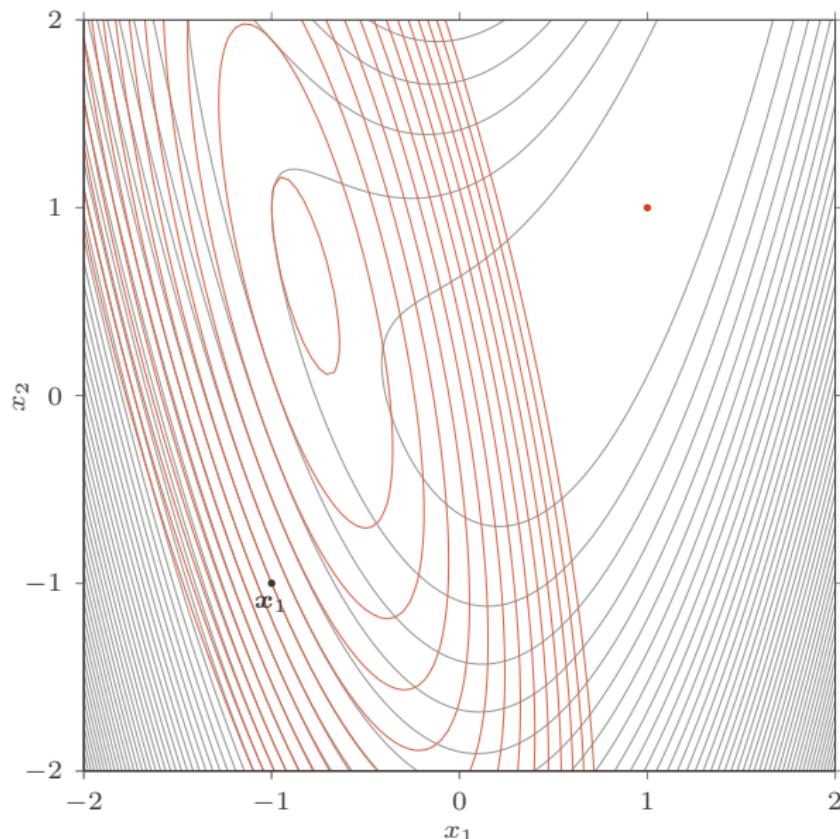
Newton-Raphson method:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - (\nabla^2 f(\mathbf{x}_k))^{-1} \nabla f(\mathbf{x}_k)$$

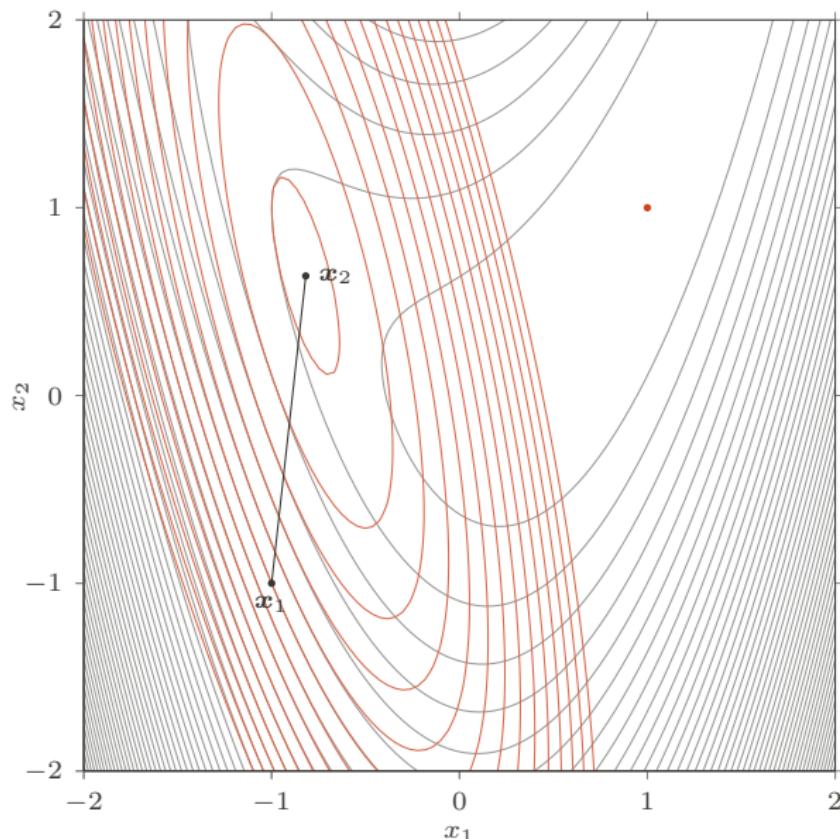
Newton-Raphson method



Newton-Raphson method



Newton-Raphson method



Newton-Raphson method

Start from point x_1 and repeat for $k = 1, 2, \dots$:

$$x_{k+1} = x_k - (\nabla^2 f(x_k))^{-1} \nabla f(x_k)$$

Newton-Raphson method

Start from point x_1 and repeat for $k = 1, 2, \dots$:

$$x_{k+1} = x_k - (\nabla^2 f(x_k))^{-1} \nabla f(x_k)$$

- Compare with univariate Newton method: $x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$

Newton-Raphson method

Start from point x_1 and repeat for $k = 1, 2, \dots$:

$$x_{k+1} = x_k - (\nabla^2 f(x_k))^{-1} \nabla f(x_k)$$

- Compare with univariate Newton method: $x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$
- **Computationally more costly** than gradient descent: requires inverting a matrix in each iteration
(in practice one solves a linear system of equations instead $\mathbf{A}x = \mathbf{b}$ with $\mathbf{A} = \nabla^2 f(x_k)$ and $\mathbf{b} = \nabla f(x_k)$)

Newton-Raphson method

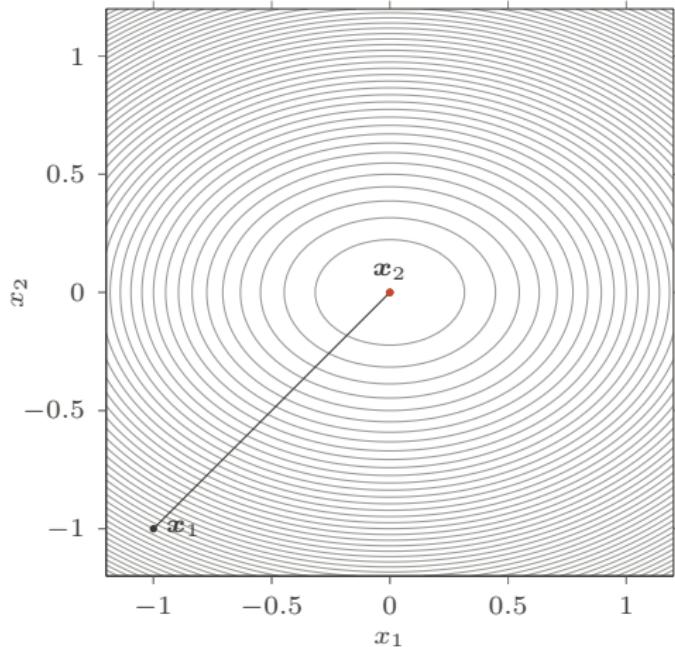
Start from point x_1 and repeat for $k = 1, 2, \dots$:

$$x_{k+1} = x_k - (\nabla^2 f(x_k))^{-1} \nabla f(x_k)$$

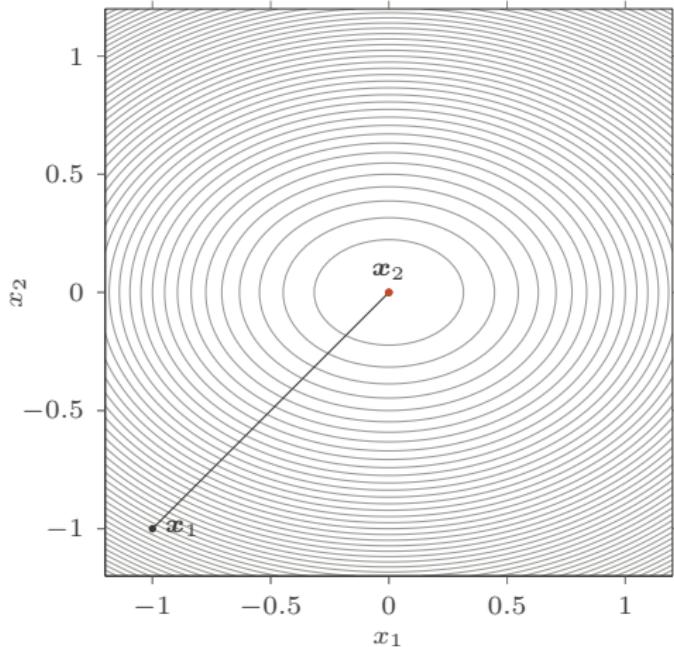
- Compare with univariate Newton method: $x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$
- **Computationally more costly** than gradient descent: requires inverting a matrix in each iteration
(in practice one solves a linear system of equations instead $\mathbf{A}x = \mathbf{b}$ with $\mathbf{A} = \nabla^2 f(x_k)$ and $\mathbf{b} = \nabla f(x_k)$)
- Converges fast when initialized sufficiently close to the minimum; when initialized wrongly, it can diverge!

Newton-Raphson method: example

$$f(x_1, x_2) = x_1^2 + 2x_2^2$$



Newton-Raphson method: example



$$f(x_1, x_2) = x_1^2 + 2x_2^2$$

Newton-Rapshon method finds the minimum of a quadratic function:

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x} + \mathbf{b}^\top \mathbf{x} + c$$

in a **single step**, independent of the initialization point!

Newton-Raphson method for a quadratic function

Consider minimization of a quadratic function with positive-definite matrix \mathbf{A} :

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x} + \mathbf{x}^\top \mathbf{b} + c,$$

Newton-Raphson method for a quadratic function

Consider minimization of a quadratic function with positive-definite matrix \mathbf{A} :

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x} + \mathbf{x}^\top \mathbf{b} + c,$$

$$\nabla f(\mathbf{x}) = 2\mathbf{A}\mathbf{x} + \mathbf{b}, \quad \nabla^2 f(\mathbf{x}) = 2\mathbf{A}, \quad (\nabla^2 f(\mathbf{x}))^{-1} = \frac{1}{2}\mathbf{A}^{-1}$$

Newton-Raphson method for a quadratic function

Consider minimization of a quadratic function with positive-definite matrix \mathbf{A} :

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x} + \mathbf{x}^\top \mathbf{b} + c,$$

$$\nabla f(\mathbf{x}) = 2\mathbf{A}\mathbf{x} + \mathbf{b}, \quad \nabla^2 f(\mathbf{x}) = 2\mathbf{A}, \quad (\nabla^2 f(\mathbf{x}))^{-1} = \frac{1}{2}\mathbf{A}^{-1}$$

Starting from any point \mathbf{x}_1 , we choose point \mathbf{x}_2 as:

$$\mathbf{x}_2 = \mathbf{x}_1 - (\nabla^2 f(\mathbf{x}_1))^{-1} \nabla f(\mathbf{x}_1)$$

Newton-Raphson method for a quadratic function

Consider minimization of a quadratic function with positive-definite matrix \mathbf{A} :

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x} + \mathbf{x}^\top \mathbf{b} + c,$$

$$\nabla f(\mathbf{x}) = 2\mathbf{A}\mathbf{x} + \mathbf{b}, \quad \nabla^2 f(\mathbf{x}) = 2\mathbf{A}, \quad (\nabla^2 f(\mathbf{x}))^{-1} = \frac{1}{2}\mathbf{A}^{-1}$$

Starting from any point \mathbf{x}_1 , we choose point \mathbf{x}_2 as:

$$\mathbf{x}_2 = \mathbf{x}_1 - \frac{1}{2}\mathbf{A}^{-1}(2\mathbf{A}\mathbf{x}_1 + \mathbf{b})$$

Newton-Raphson method for a quadratic function

Consider minimization of a quadratic function with positive-definite matrix \mathbf{A} :

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x} + \mathbf{x}^\top \mathbf{b} + c,$$

$$\nabla f(\mathbf{x}) = 2\mathbf{A}\mathbf{x} + \mathbf{b}, \quad \nabla^2 f(\mathbf{x}) = 2\mathbf{A}, \quad (\nabla^2 f(\mathbf{x}))^{-1} = \frac{1}{2}\mathbf{A}^{-1}$$

Starting from any point \mathbf{x}_1 , we choose point \mathbf{x}_2 as:

$$\mathbf{x}_2 = \mathbf{x}_1 - \underbrace{\mathbf{A}^{-1}}_I \mathbf{A} \mathbf{x}_1 - \frac{1}{2} \mathbf{A}^{-1} \mathbf{b}$$

Newton-Raphson method for a quadratic function

Consider minimization of a quadratic function with positive-definite matrix \mathbf{A} :

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x} + \mathbf{x}^\top \mathbf{b} + c,$$

$$\nabla f(\mathbf{x}) = 2\mathbf{A}\mathbf{x} + \mathbf{b}, \quad \nabla^2 f(\mathbf{x}) = 2\mathbf{A}, \quad (\nabla^2 f(\mathbf{x}))^{-1} = \frac{1}{2}\mathbf{A}^{-1}$$

Starting from any point \mathbf{x}_1 , we choose point \mathbf{x}_2 as:

$$\mathbf{x}_2 = \mathbf{x}_1 - \mathbf{x}_1 - \frac{1}{2}\mathbf{A}^{-1}\mathbf{b}$$

Newton-Raphson method for a quadratic function

Consider minimization of a quadratic function with positive-definite matrix \mathbf{A} :

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x} + \mathbf{x}^\top \mathbf{b} + c,$$

$$\nabla f(\mathbf{x}) = 2\mathbf{A}\mathbf{x} + \mathbf{b}, \quad \nabla^2 f(\mathbf{x}) = 2\mathbf{A}, \quad (\nabla^2 f(\mathbf{x}))^{-1} = \frac{1}{2}\mathbf{A}^{-1}$$

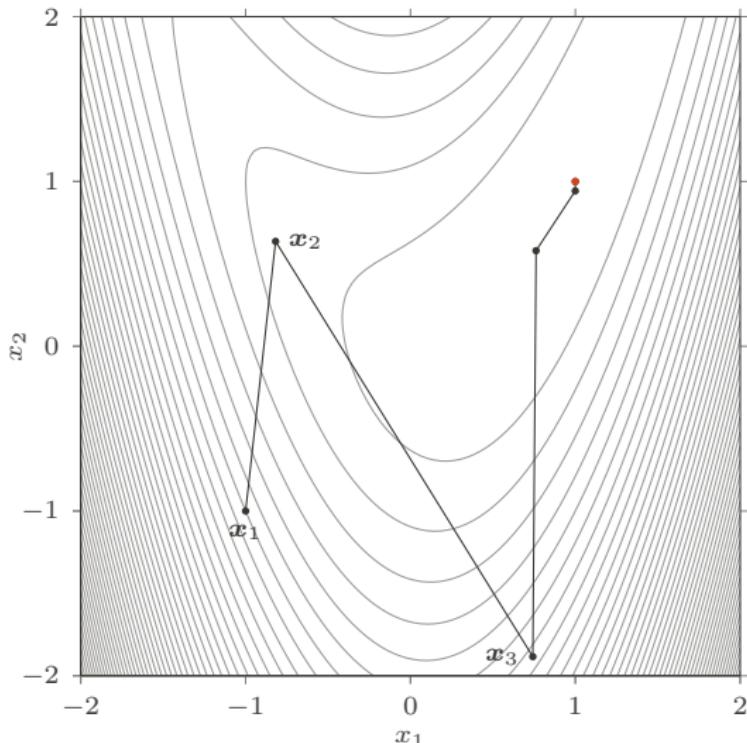
Starting from any point \mathbf{x}_1 , we choose point \mathbf{x}_2 as:

$$\mathbf{x}_2 = -\frac{1}{2}\mathbf{A}^{-1}\mathbf{b}$$

But this is the minimizer \mathbf{x}^* of a quadratic function!

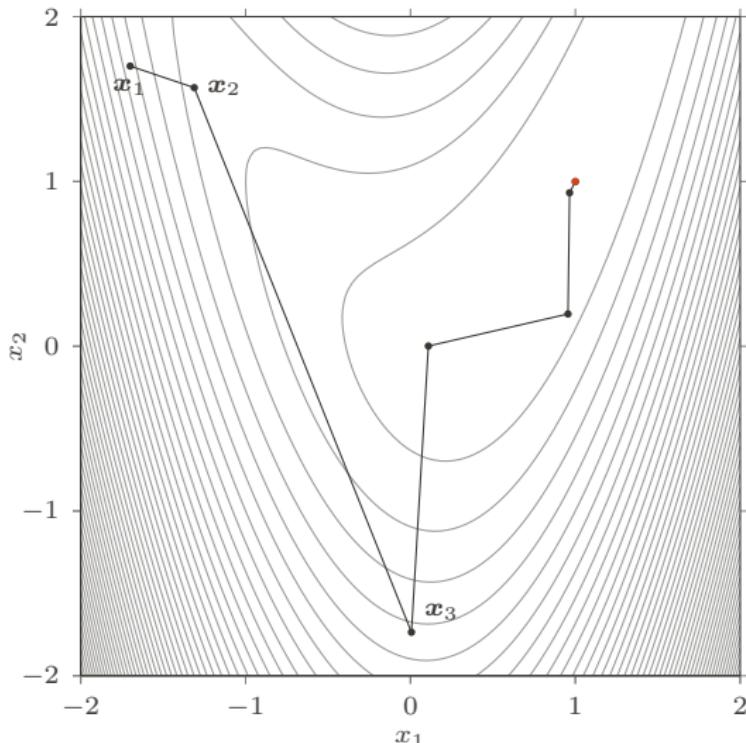
Newton-Raphson method: example

$$f(x_1, x_2) = 2.5(x_1^2 - x_2)^2 + (1 - x_1)^2$$



Newton-Raphson method: example

$$f(x_1, x_2) = 2.5(x_1^2 - x_2)^2 + (1 - x_1)^2$$



Newton-Raphson method as a descent method

Start with x_1 and repeat for $k = 1, 2, \dots$:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{v}_k, \quad \mathbf{v}_k = -(\nabla^2 f(\mathbf{x}_k))^{-1} \nabla f(\mathbf{x}_k)$$

From our assumption, $\nabla^2 f(\mathbf{x}_k)$ is positive-definite, so has all eigenvalues positive

Newton-Raphson method as a descent method

Start with x_1 and repeat for $k = 1, 2, \dots$:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{v}_k, \quad \mathbf{v}_k = -(\nabla^2 f(\mathbf{x}_k))^{-1} \nabla f(\mathbf{x}_k)$$

From our assumption, $\nabla^2 f(\mathbf{x}_k)$ is positive-definite, so has all eigenvalues positive

Then $(\nabla^2 f(\mathbf{x}_k))^{-1}$ is also positive-definite, since the eigenvalues of \mathbf{A}^{-1} are the reciprocals of the eigenvalues of \mathbf{A}

Newton-Raphson method as a descent method

Start with x_1 and repeat for $k = 1, 2, \dots$:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{v}_k, \quad \mathbf{v}_k = -(\nabla^2 f(\mathbf{x}_k))^{-1} \nabla f(\mathbf{x}_k)$$

From our assumption, $\nabla^2 f(\mathbf{x}_k)$ is positive-definite, so has all eigenvalues positive

Then $(\nabla^2 f(\mathbf{x}_k))^{-1}$ is also positive-definite, since the eigenvalues of \mathbf{A}^{-1} are the reciprocals of the eigenvalues of \mathbf{A}

So $\mathbf{x}^\top (\nabla^2 f(\mathbf{x}_k))^{-1} \mathbf{x} > 0$ for any non-zero vector \mathbf{x}

Newton-Raphson method as a descent method

Start with \mathbf{x}_1 and repeat for $k = 1, 2, \dots$:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{v}_k, \quad \mathbf{v}_k = -(\nabla^2 f(\mathbf{x}_k))^{-1} \nabla f(\mathbf{x}_k)$$

From our assumption, $\nabla^2 f(\mathbf{x}_k)$ is positive-definite, so has all eigenvalues positive

Then $(\nabla^2 f(\mathbf{x}_k))^{-1}$ is also positive-definite, since the eigenvalues of \mathbf{A}^{-1} are the reciprocals of the eigenvalues of \mathbf{A}

So $\mathbf{x}^\top (\nabla^2 f(\mathbf{x}_k))^{-1} \mathbf{x} > 0$ for any non-zero vector \mathbf{x}

Conclusion: $\mathbf{v}_k = -(\nabla^2 f(\mathbf{x}_k))^{-1} \nabla f(\mathbf{x}_k)$ is a **descent direction**

$$\nabla f(\mathbf{x}_k)^\top \mathbf{v}_k = -\nabla f(\mathbf{x}_k) (\nabla^2 f(\mathbf{x}_k))^{-1} \nabla f(\mathbf{x}_k) < 0$$

Newton-Raphson method as a descent method

Start with x_1 and repeat for $k = 1, 2, \dots$:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{v}_k, \quad \mathbf{v}_k = -(\nabla^2 f(\mathbf{x}_k))^{-1} \nabla f(\mathbf{x}_k)$$

From our assumption, $\nabla^2 f(\mathbf{x}_k)$ is positive-definite, so has all eigenvalues positive

Then $(\nabla^2 f(\mathbf{x}_k))^{-1}$ is also positive-definite, since the eigenvalues of \mathbf{A}^{-1} are the reciprocals of the eigenvalues of \mathbf{A}

So $\mathbf{x}^\top (\nabla^2 f(\mathbf{x}_k))^{-1} \mathbf{x} > 0$ for any non-zero vector \mathbf{x}

Conclusion: $\mathbf{v}_k = -(\nabla^2 f(\mathbf{x}_k))^{-1} \nabla f(\mathbf{x}_k)$ is a **descent direction**

$$\nabla f(\mathbf{x}_k)^\top \mathbf{v}_k = -\nabla f(\mathbf{x}_k) (\nabla^2 f(\mathbf{x}_k))^{-1} \nabla f(\mathbf{x}_k) < 0$$

Note: step size α_k is **constant** and equal to 1

Convergence of Newton-Raphson method

If the objective $f(\mathbf{x})$ is three times continuously differentiable, the Newton-Raphson method initialized **sufficiently close** to a local minimum \mathbf{x}^* with positive-definite Hessian $\nabla^2 f(\mathbf{x}^*)$ converges with a **quadratic order of convergence**

The proof conceptually very similar to the univariate case, but more complicated due to matrix-vector Taylor approximation.

Convergence of Newton-Raphson method

If the objective $f(\mathbf{x})$ is three times continuously differentiable, the Newton-Raphson method initialized **sufficiently close** to a local minimum \mathbf{x}^* with positive-definite Hessian $\nabla^2 f(\mathbf{x}^*)$ converges with a **quadratic order of convergence**

The proof conceptually very similar to the univariate case, but more complicated due to matrix-vector Taylor approximation.

Initialized wrongly, the Newton-Raphson method can even diverge!

There are **modifications** of the method, which prevent divergence issues.

Modifications of the Newton-Raphson method

Variable step size (damped Newton-Raphson)

Newton-Raphson method (original):

Start from x_1 and repeat for $k = 1, 2, \dots$:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{v}_k, \quad \mathbf{v}_k = -(\nabla^2 f(\mathbf{x}_k))^{-1} \nabla f(\mathbf{x}_k), \quad \alpha_k = 1$$

Does not guarantee the decrease of the objective $f(\mathbf{x}_{k+1}) < f(\mathbf{x}_k)$!

Variable step size (damped Newton-Raphson)

Newton-Raphson method (original):

Start from x_1 and repeat for $k = 1, 2, \dots$:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{v}_k, \quad \mathbf{v}_k = -(\nabla^2 f(\mathbf{x}_k))^{-1} \nabla f(\mathbf{x}_k), \quad \alpha_k = 1$$

Does not guarantee the decrease of the objective $f(\mathbf{x}_{k+1}) < f(\mathbf{x}_k)!$

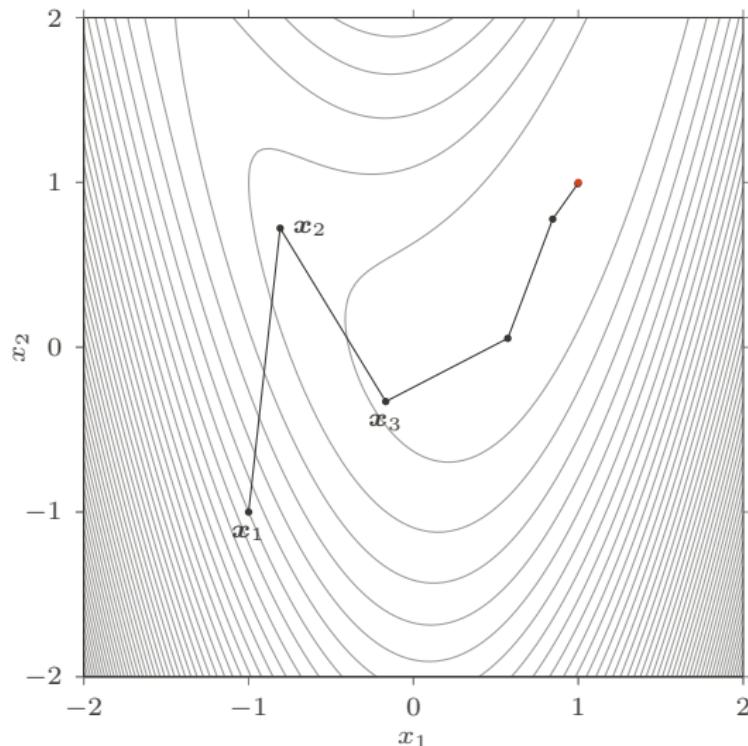
If the Hessian $\nabla^2 f(\mathbf{x}_k)$ is **positive-definite**, \mathbf{v}_k is a **descent direction** and we can modify the method by choosing α_k :

- optimally: $\alpha_k = \operatorname{argmin}_{\alpha \geq 0} f(\mathbf{x}_k + \alpha_k \mathbf{v}_k),$
- using the backtracking method,

which will guarantee $f(\mathbf{x}_{k+1}) < f(\mathbf{x}_k)$

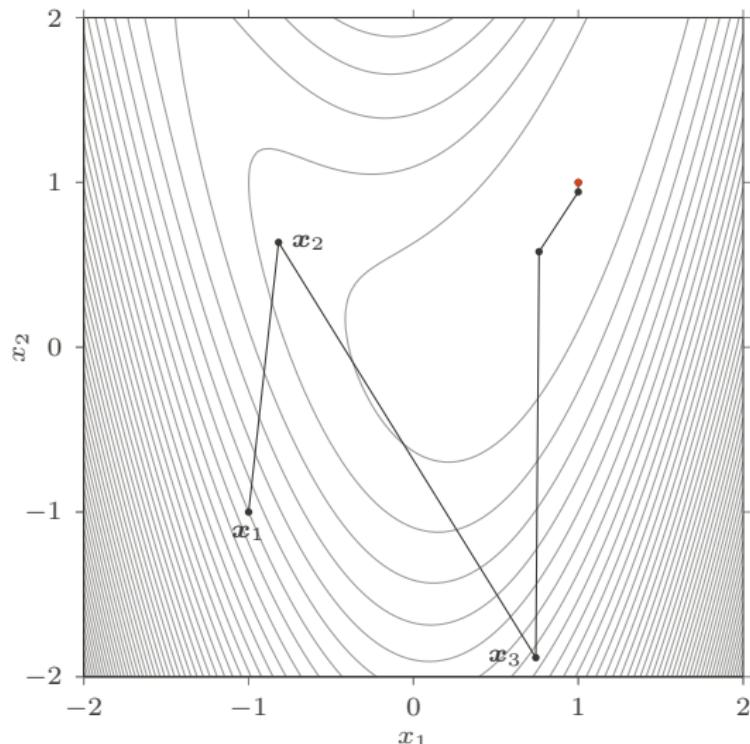
Newton-Raphson method with the optimal step sizes

$$f(x_1, x_2) = 2.5(x_1^2 - x_2)^2 + (1 - x_1)^2$$



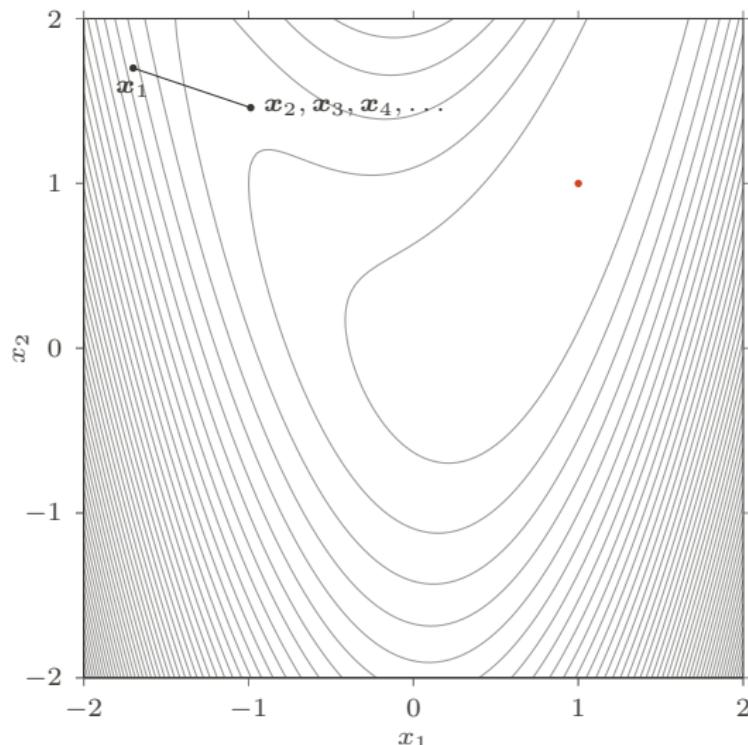
Compare with the original Netwon-Raphson

$$f(x_1, x_2) = 2.5(x_1^2 - x_2)^2 + (1 - x_1)^2$$



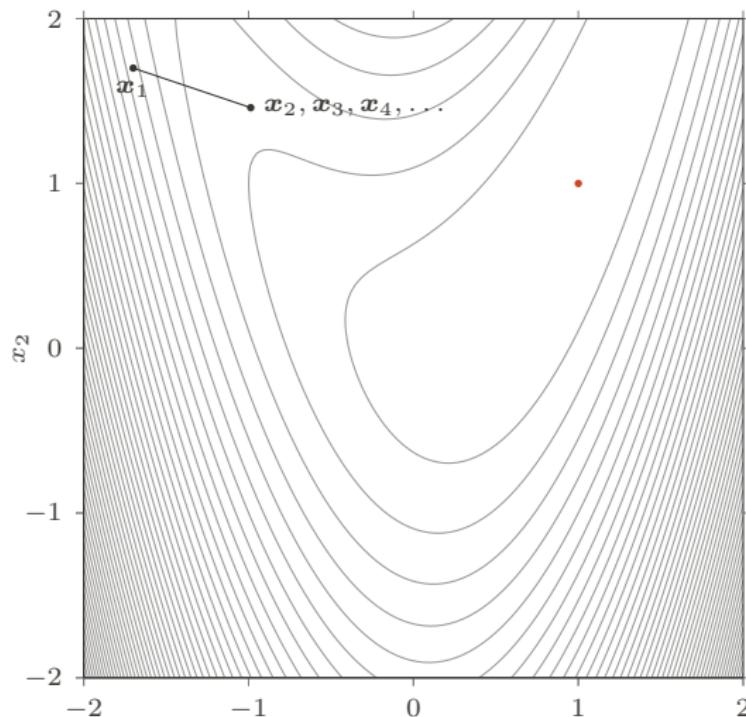
Newton-Raphson method with the optimal step sizes

$$f(x_1, x_2) = 2.5(x_1^2 - x_2)^2 + (1 - x_1)^2$$



Newton-Raphson method with the optimal step sizes

$$f(x_1, x_2) = 2.5(x_1^2 - x_2)^2 + (1 - x_1)^2$$



Why did the algorithm “get stuck” in \mathbf{x}_2 ?

Analysis

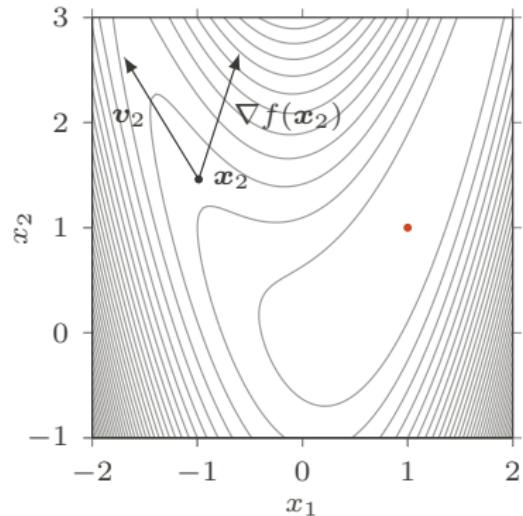
$$\mathbf{x}_2 = (-0.99, 1.46)$$

$$\nabla f(\mathbf{x}_2) = (0.77, 2.4)$$

$$\nabla^2 f(\mathbf{x}_2) = \begin{bmatrix} 16.8 & 9.9 \\ 9.9 & 5.0 \end{bmatrix}$$

$$\begin{aligned}\mathbf{v}_2 &= -(\nabla^2(\mathbf{x}_2))^{-1}\nabla(\mathbf{x}_2) \\ &= (-1.42, 2.33)\end{aligned}$$

$$\nabla f(\mathbf{x}_2)^\top \mathbf{v}_2 = 4.5 > 0$$



Analysis

$$\mathbf{x}_2 = (-0.99, 1.46)$$

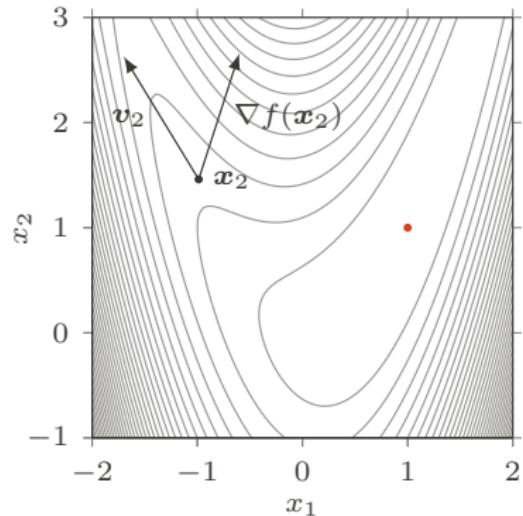
$$\nabla f(\mathbf{x}_2) = (0.77, 2.4)$$

$$\nabla^2 f(\mathbf{x}_2) = \begin{bmatrix} 16.8 & 9.9 \\ 9.9 & 5.0 \end{bmatrix}$$

$$\begin{aligned}\mathbf{v}_2 &= -(\nabla^2(\mathbf{x}_2))^{-1}\nabla(\mathbf{x}_2) \\ &= (-1.42, 2.33)\end{aligned}$$

$$\nabla f(\mathbf{x}_2)^\top \mathbf{v}_2 = 4.5 > 0$$

Direction \mathbf{v}_2 is not a descent direction



Analysis

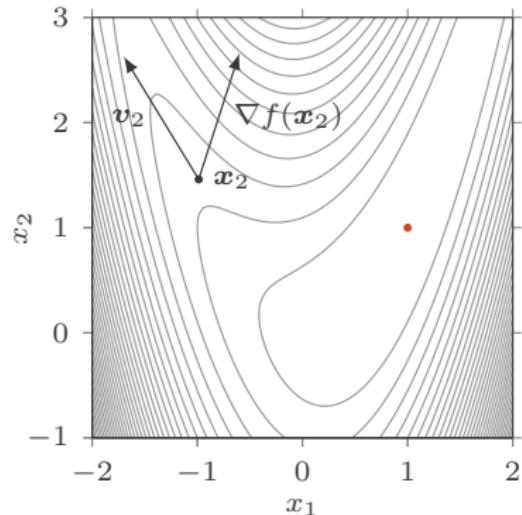
$$\mathbf{x}_2 = (-0.99, 1.46)$$

$$\nabla f(\mathbf{x}_2) = (0.77, 2.4)$$

$$\nabla^2 f(\mathbf{x}_2) = \begin{bmatrix} 16.8 & 9.9 \\ 9.9 & 5.0 \end{bmatrix}$$

$$\begin{aligned}\mathbf{v}_2 &= -(\nabla^2(\mathbf{x}_2))^{-1}\nabla(\mathbf{x}_2) \\ &= (-1.42, 2.33)\end{aligned}$$

$$\nabla f(\mathbf{x}_2)^\top \mathbf{v}_2 = 4.5 > 0$$



Direction \mathbf{v}_2 is not a descent direction

Eigenvalues of the Hessian: $(22.4, -0.62) \implies$ not positive-definite

Optimal step size is $\alpha_2 = 0$, since we cannot improve the objective function along \mathbf{v}_2

Analysis

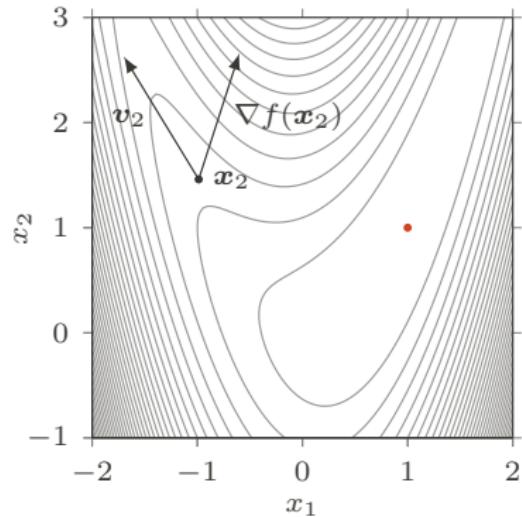
$$\mathbf{x}_2 = (-0.99, 1.46)$$

$$\nabla f(\mathbf{x}_2) = (0.77, 2.4)$$

$$\nabla^2 f(\mathbf{x}_2) = \begin{bmatrix} 16.8 & 9.9 \\ 9.9 & 5.0 \end{bmatrix}$$

$$\begin{aligned}\mathbf{v}_2 &= -(\nabla^2(\mathbf{x}_2))^{-1}\nabla(\mathbf{x}_2) \\ &= (-1.42, 2.33)\end{aligned}$$

$$\nabla f(\mathbf{x}_2)^\top \mathbf{v}_2 = 4.5 > 0$$



Direction \mathbf{v}_2 is not a descent direction

Eigenvalues of the Hessian: $(22.4, -0.62) \implies$ not positive-definite

Optimal step size is $\alpha_2 = 0$, since we cannot improve the objective function along \mathbf{v}_2

Conclusion: Optimization of step size can fail, if the Hessian is not positive-definite!

Eigenvalues of the Hessian

So far we assumed that the Hessian $\nabla^2 f(\mathbf{x}_k)$ is **positive-definite**, so that all its eigenvalues are **positive**, and thus is **invertible**.

In practice:

- Even for convex functions, the Hessian may only be positive semi-definite, and one or more of the eigenvalues can be zero; then the inverse $(\nabla^2 f(\mathbf{x}_k))^{-1}$ does not exist
- For **non-convex** functions, the eigenvalues can be negative, and then direction $\mathbf{v}_k = -(\nabla^2 f(\mathbf{x}_k))^{-1} \nabla f(\mathbf{x}_k)$ may not be a descent direction!

Example

Consider minimizing a function:

$$f(\mathbf{x}) = x_1^4 + x_2^2 - x_1 - x_2$$

Example

Consider minimizing a function:

$$f(\mathbf{x}) = x_1^4 + x_2^2 - x_1 - x_2$$

$$\frac{\partial f}{\partial x_1} = 4x_1^3 - 1, \quad \frac{\partial f}{\partial x_2} = 2x_2 - 1$$

Example

Consider minimizing a function:

$$f(\mathbf{x}) = x_1^4 + x_2^2 - x_1 - x_2$$

$$\frac{\partial f}{\partial x_1} = 4x_1^3 - 1, \quad \frac{\partial f}{\partial x_2} = 2x_2 - 1$$

$$\frac{\partial^2 f}{\partial x_1^2} = 12x_1^2, \quad \frac{\partial^2 f}{\partial x_2^2} = 2, \quad \frac{\partial^2 f}{\partial x_1 \partial x_2} = 0, \quad \nabla^2 f(\mathbf{x}) = \begin{bmatrix} 12x_1^2 & 0 \\ 0 & 2 \end{bmatrix}$$

Example

Consider minimizing a function:

$$f(\mathbf{x}) = x_1^4 + x_2^2 - x_1 - x_2$$

$$\frac{\partial f}{\partial x_1} = 4x_1^3 - 1, \quad \frac{\partial f}{\partial x_2} = 2x_2 - 1$$

$$\frac{\partial^2 f}{\partial x_1^2} = 12x_1^2, \quad \frac{\partial^2 f}{\partial x_2^2} = 2, \quad \frac{\partial^2 f}{\partial x_1 \partial x_2} = 0, \quad \nabla^2 f(\mathbf{x}) = \begin{bmatrix} 12x_1^2 & 0 \\ 0 & 2 \end{bmatrix}$$

Since the Hessian at \mathbf{x} has eigenvalues $(12x_1^2, 2)$, **nonnegative** for any x_1 , it is a **positive semi-definite matrix for every \mathbf{x}** , so $f(\mathbf{x})$ is a **convex** function

Example

Consider minimizing a function:

$$f(\mathbf{x}) = x_1^4 + x_2^2 - x_1 - x_2$$

$$\frac{\partial f}{\partial x_1} = 4x_1^3 - 1, \quad \frac{\partial f}{\partial x_2} = 2x_2 - 1$$

$$\frac{\partial^2 f}{\partial x_1^2} = 12x_1^2, \quad \frac{\partial^2 f}{\partial x_2^2} = 2, \quad \frac{\partial^2 f}{\partial x_1 \partial x_2} = 0, \quad \nabla^2 f(\mathbf{x}) = \begin{bmatrix} 12x_1^2 & 0 \\ 0 & 2 \end{bmatrix}$$

Since the Hessian at \mathbf{x} has eigenvalues $(12x_1^2, 2)$, **nonnegative** for any x_1 , it is a **positive semi-definite matrix for every x** , so $f(\mathbf{x})$ is a **convex** function

Starting the Newton-Raphson method from $\mathbf{x}_1 = \mathbf{0}$ we have:

$$\nabla^2 f(\mathbf{x}_1) = \begin{bmatrix} 0 & 0 \\ 0 & 2 \end{bmatrix},$$

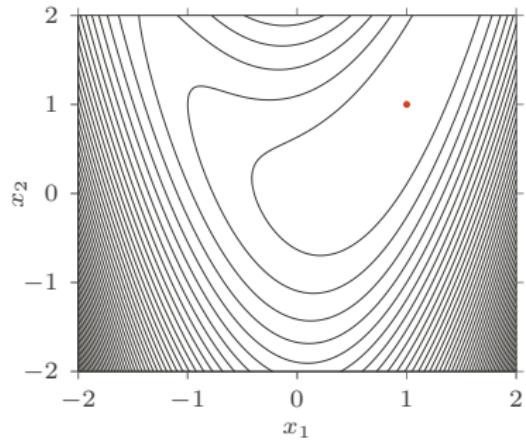
with eigenvalues $(0, 2)$, which is non-invertible!

Example

Consider minimizing a function:

$$f(\mathbf{x}) = 2.5(x_1^2 - x_2)^2 + (1 - x_1)^2$$

$$\nabla^2 f(\mathbf{x}) = \begin{bmatrix} 30x_1^2 - 10x_2 + 2 & -10x_1 \\ -10x_1 & 5 \end{bmatrix}$$

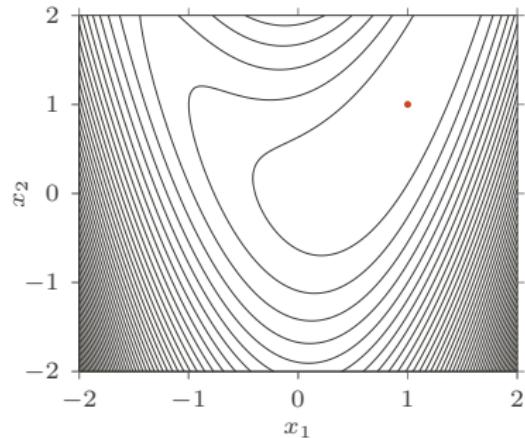


Example

Consider minimizing a function:

$$f(\mathbf{x}) = 2.5(x_1^2 - x_2)^2 + (1 - x_1)^2$$

$$\nabla^2 f(\mathbf{x}) = \begin{bmatrix} 30x_1^2 - 10x_2 + 2 & -10x_1 \\ -10x_1 & 5 \end{bmatrix}$$



Starting the Newton-Raphson method from $\mathbf{x}_1 = (0, 0.3)$ we have:

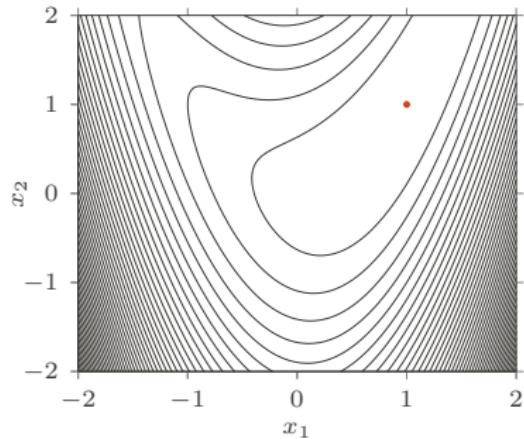
$$\nabla^2 f(\mathbf{x}_1) = \begin{bmatrix} -1 & 0 \\ 0 & 5 \end{bmatrix} \quad \text{with eigenvalues } (-1, 5)$$

Example

Consider minimizing a function:

$$f(\mathbf{x}) = 2.5(x_1^2 - x_2)^2 + (1 - x_1)^2$$

$$\nabla^2 f(\mathbf{x}) = \begin{bmatrix} 30x_1^2 - 10x_2 + 2 & -10x_1 \\ -10x_1 & 5 \end{bmatrix}$$



Starting the Newton-Raphson method from $\mathbf{x}_1 = (0, 0.3)$ we have:

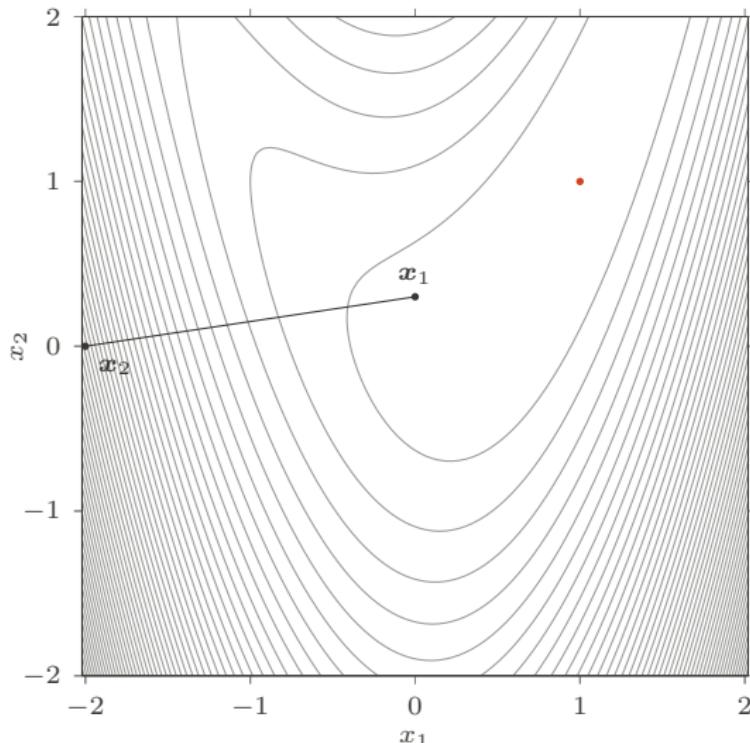
$$\nabla^2 f(\mathbf{x}_1) = \begin{bmatrix} -1 & 0 \\ 0 & 5 \end{bmatrix} \quad \text{with eigenvalues } (-1, 5)$$

The Taylor polynomial at \mathbf{x}_1 is non-convex, and the next point \mathbf{x}_2 is its **saddle point**, not its minimum!

Direction $-(\nabla^2 f(\mathbf{x}_1))^{-1} \nabla f(\mathbf{x}_1)$ is not a descent direction!

Example

$$f(\boldsymbol{x}) = 2.5(x_1^2 - x_2)^2 + (1 - x_1)^2, \quad \boldsymbol{x}_1 = (0, 0.3)$$



Eigenvalues shift

Fact: if a symmetric matrix \mathbf{A} has eigenvalues $(\lambda_1, \dots, \lambda_n)$ then matrix $\mathbf{A} + \epsilon \mathbf{I}$ has eigenvalues $(\lambda_1 + \epsilon, \dots, \lambda_n + \epsilon)$

Eigenvalues shift

Fact: if a symmetric matrix \mathbf{A} has eigenvalues $(\lambda_1, \dots, \lambda_n)$ then matrix $\mathbf{A} + \epsilon \mathbf{I}$ has eigenvalues $(\lambda_1 + \epsilon, \dots, \lambda_n + \epsilon)$

Proof: If \mathbf{v}_k is an eigenvector of \mathbf{A} with associated eigenvalue λ_k :

$$\mathbf{A}\mathbf{v}_k = \lambda_k \mathbf{v}_k,$$

then \mathbf{v}_k is also the eigenvector of $\mathbf{A} + \epsilon \mathbf{I}$ with eigenvalue $\lambda_k + \epsilon$:

$$(\mathbf{A} + \epsilon \mathbf{I})\mathbf{v}_k = \mathbf{A}\mathbf{v}_k + \epsilon \mathbf{I}\mathbf{v}_k = \lambda_k \mathbf{v}_k + \epsilon \mathbf{v}_k = (\lambda_k + \epsilon) \mathbf{v}_k$$

Eigenvalues shift

Fact: if a symmetric matrix \mathbf{A} has eigenvalues $(\lambda_1, \dots, \lambda_n)$ then matrix $\mathbf{A} + \epsilon \mathbf{I}$ has eigenvalues $(\lambda_1 + \epsilon, \dots, \lambda_n + \epsilon)$

Proof: If \mathbf{v}_k is an eigenvector of \mathbf{A} with associated eigenvalue λ_k :

$$\mathbf{A}\mathbf{v}_k = \lambda_k \mathbf{v}_k,$$

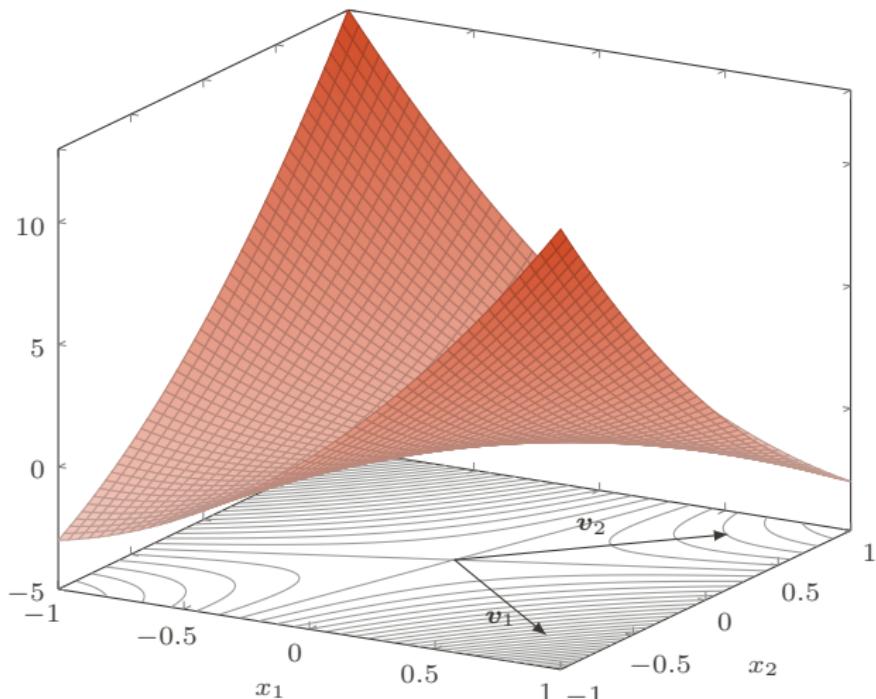
then \mathbf{v}_k is also the eigenvector of $\mathbf{A} + \epsilon \mathbf{I}$ with eigenvalue $\lambda_k + \epsilon$:

$$(\mathbf{A} + \epsilon \mathbf{I})\mathbf{v}_k = \mathbf{A}\mathbf{v}_k + \epsilon \mathbf{I}\mathbf{v}_k = \lambda_k \mathbf{v}_k + \epsilon \mathbf{v}_k = (\lambda_k + \epsilon) \mathbf{v}_k$$

Conclusion: With the appropriate choice of ϵ , we can guarantee that all eigenvalues of $\mathbf{A} + \epsilon \mathbf{I}$ will be **positive** (i.e., this matrix will be **positive-definite**)

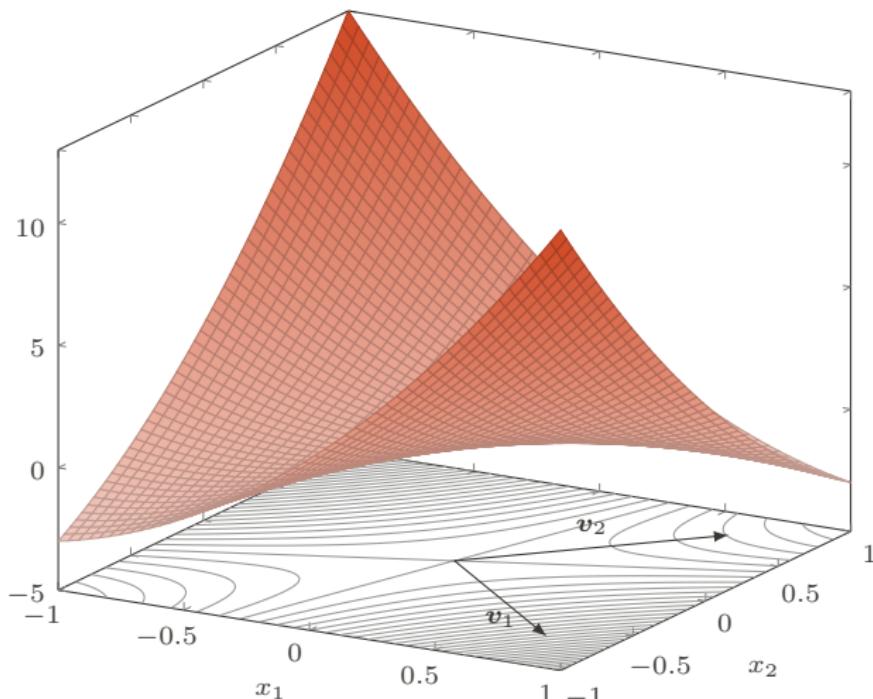
Example

$$f(\mathbf{x}) = 3x_1^2 - 8x_1x_2 + 2x_2^2, \quad \mathbf{x} = (x_1, x_2)$$



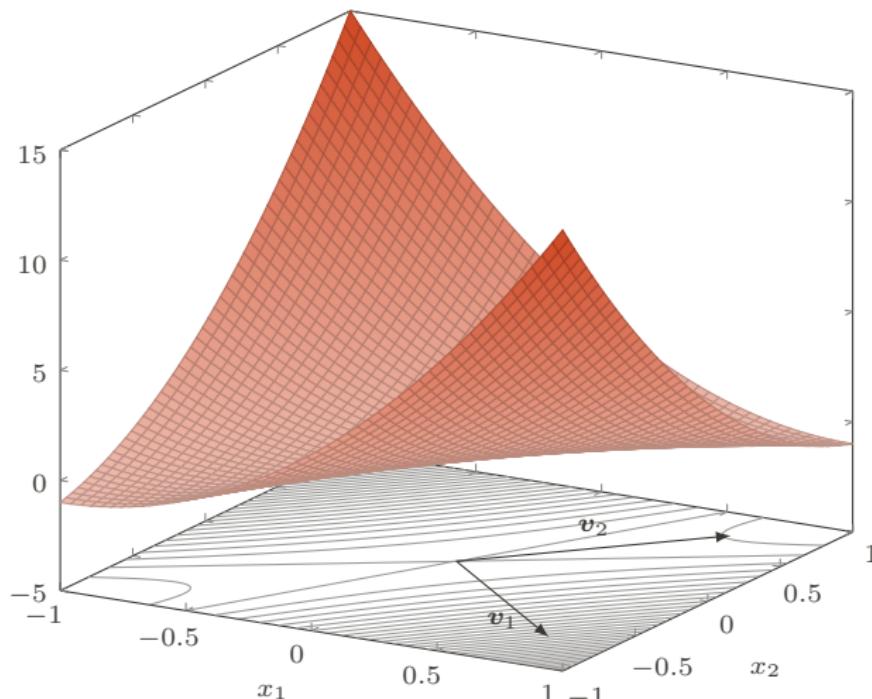
Example

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x}, \quad \mathbf{A} = \begin{bmatrix} 3 & -4 \\ -4 & 2 \end{bmatrix}, \quad \lambda_1 = 6.53, \lambda_2 = -1.53$$



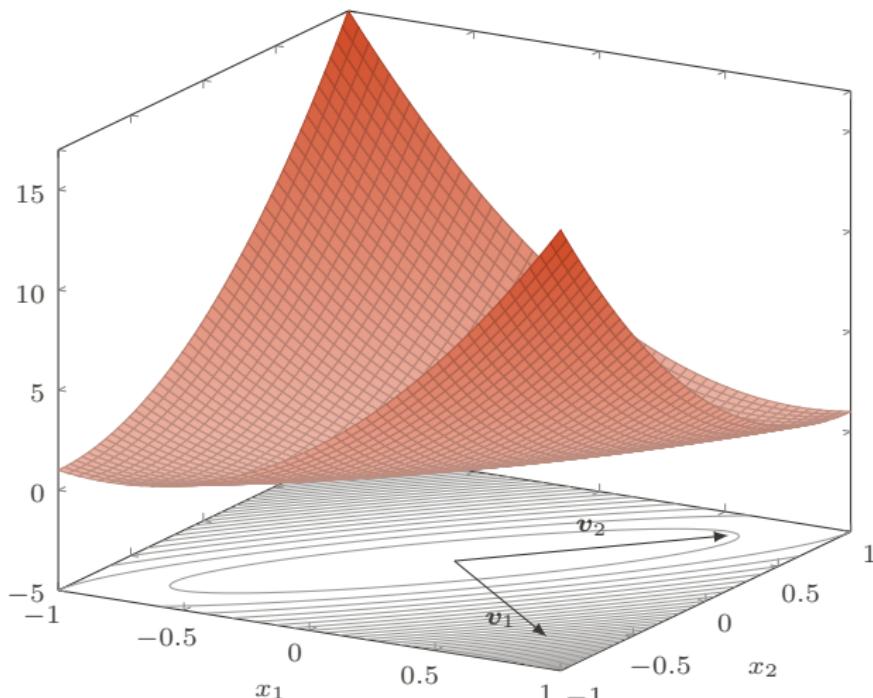
Example

$$f(x) = x^\top (A + I)x, \quad A + I = \begin{bmatrix} 4 & -4 \\ -4 & 3 \end{bmatrix}, \quad \lambda_1 = 7.53, \lambda_2 = -0.53$$



Example

$$f(\mathbf{x}) = \mathbf{x}^\top (\mathbf{A} + 2\mathbf{I})\mathbf{x}, \quad \mathbf{A} + 2\mathbf{I} = \begin{bmatrix} 5 & -4 \\ -4 & 4 \end{bmatrix}, \quad \lambda_1 = 8.53, \lambda_2 = 0.47$$



Levenberg-Marquardt method

In each iteration $k = 1, 2, \dots$ we add $\epsilon_k \mathbf{I}$ to the Hessian to ensure its positive-definiteness:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{v}_k, \quad \mathbf{v}_k = -(\nabla^2 f(\mathbf{x}_k) + \epsilon_k \mathbf{I})^{-1} \nabla f(\mathbf{x}_k)$$

Levenberg-Marquardt method

In each iteration $k = 1, 2, \dots$ we add $\epsilon_k \mathbf{I}$ to the Hessian to ensure its positive-definiteness:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{v}_k, \quad \mathbf{v}_k = -(\nabla^2 f(\mathbf{x}_k) + \epsilon_k \mathbf{I})^{-1} \nabla f(\mathbf{x}_k)$$

The choice of ϵ_k :

- For **convex** functions it suffices to add a very small, fixed value ϵ , e.g., $\epsilon = 10^{-5}$. The step size can be set to $\alpha_k = 1$ (as in the original Newton-Raphson method) or optimized
- For **non-convex** functions, in each iteration one needs to compute the eigenvalues of the Hessian and pick ϵ_k so that no eigenvalues end up being positive

In this case it is always recommended to use a variable (optimized) step size α_k !

Levenberg-Marquardt method: interpretation

Consider a general descent method $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{v}_k$

Levenberg-Marquardt method: interpretation

Consider a general descent method $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{v}_k$

In the Newton-Rapshon method $\mathbf{v}_k = -(\nabla^2 f(\mathbf{x}_k))^{-1} \nabla f(\mathbf{x}_k)$ can be obtained as a solution of the minimization problem:

$$g(\mathbf{v}) = \nabla f(\mathbf{x}_k)^\top \mathbf{v} + \frac{1}{2} \mathbf{v}^\top \nabla^2 f(\mathbf{x}_k) \mathbf{v}$$

Lagrange Multipliers method: interpretation

Minimizer of quadratic function $\mathbf{x}^\top \mathbf{b} + \frac{1}{2} \mathbf{x}^\top \mathbf{A} \mathbf{x}$
is $\mathbf{x}^* = -\mathbf{A}^{-1} \mathbf{b}$

In the Newton-Rapshon method $\mathbf{v}_k = -(\nabla^2 f(\mathbf{x}_k))^{-1} \nabla f(\mathbf{x}_k)$ can be obtained as a solution of the minimization problem:

$$g(\mathbf{v}) = \nabla f(\mathbf{x}_k)^\top \mathbf{v} + \frac{1}{2} \mathbf{v}^\top \nabla^2 f(\mathbf{x}_k) \mathbf{v}$$

Levenberg-Marquardt method: interpretation

Consider a general descent method $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{v}_k$

In the Newton-Rapshon method $\mathbf{v}_k = -(\nabla^2 f(\mathbf{x}_k))^{-1} \nabla f(\mathbf{x}_k)$ can be obtained as a solution of the minimization problem:

$$g(\mathbf{v}) = \nabla f(\mathbf{x}_k)^\top \mathbf{v} + \frac{1}{2} \mathbf{v}^\top \nabla^2 f(\mathbf{x}_k) \mathbf{v}$$

In the gradient descent method $\mathbf{v}_k = -\nabla f(\mathbf{x}_k)$ can be obtained as a solution of the minimization problem:

$$g(\mathbf{v}) = \nabla f(\mathbf{x}_k)^\top \mathbf{v} + \frac{1}{2} \mathbf{v}^\top \mathbf{I} \mathbf{v}$$

Levenberg-Marquardt method: interpretation

Consider a general descent method $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{v}_k$

In the Newton-Rapshon method $\mathbf{v}_k = -(\nabla^2 f(\mathbf{x}_k))^{-1} \nabla f(\mathbf{x}_k)$ can be obtained as a solution of the minimization problem:

$$g(\mathbf{v}) = \nabla f(\mathbf{x}_k)^\top \mathbf{v} + \frac{1}{2} \mathbf{v}^\top \nabla^2 f(\mathbf{x}_k) \mathbf{v}$$

In the gradient descent method $\mathbf{v}_k = -\nabla f(\mathbf{x}_k)$ can be obtained as a solution of the minimization problem:

$$g(\mathbf{v}) = \nabla f(\mathbf{x}_k)^\top \mathbf{v} + \frac{1}{2} \mathbf{v}^\top \mathbf{I} \mathbf{v}$$

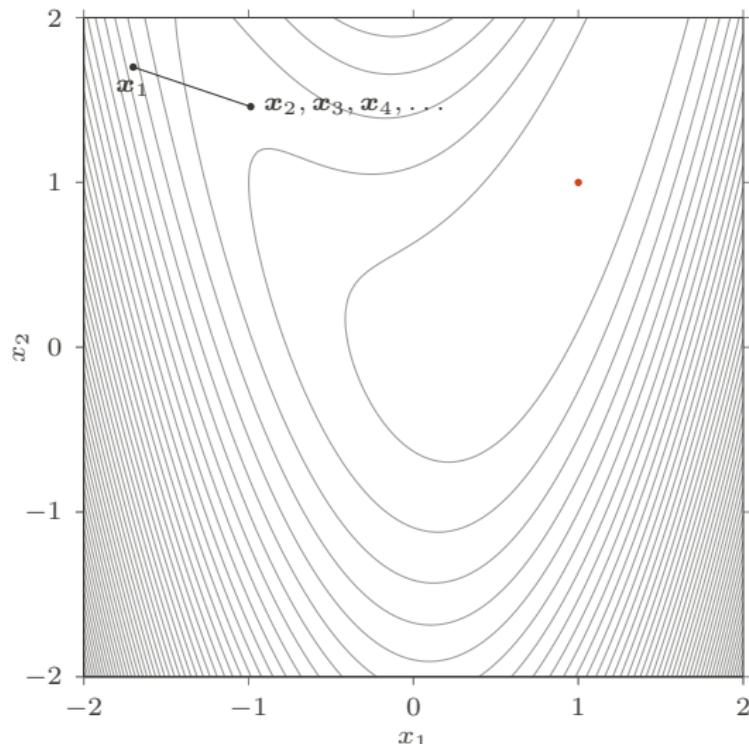
The Levenberg-Marquardt method is an **interpolation** of the two methods above, choosing its descent direction as a solution of the minimization problem:

$$g(\mathbf{v}) = \nabla f(\mathbf{x}_k)^\top \mathbf{v} + \frac{1}{2} \mathbf{v}^\top (\epsilon_k \mathbf{I} + \nabla^2 f(\mathbf{x}_k)) \mathbf{v}$$

It is a special case of **quasi-Newton method**, where in the quadratic function one chooses arbitrary positive-definite matrix A_k

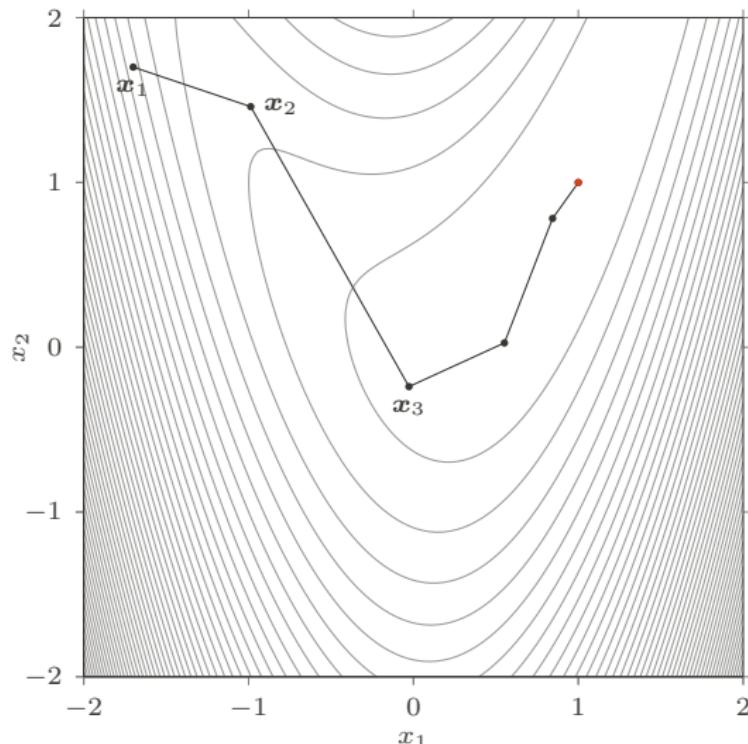
Newton-Raphson method with the optimal step size

$$f(x_1, x_2) = 2.5(x_1^2 - x_2)^2 + (1 - x_1)^2$$



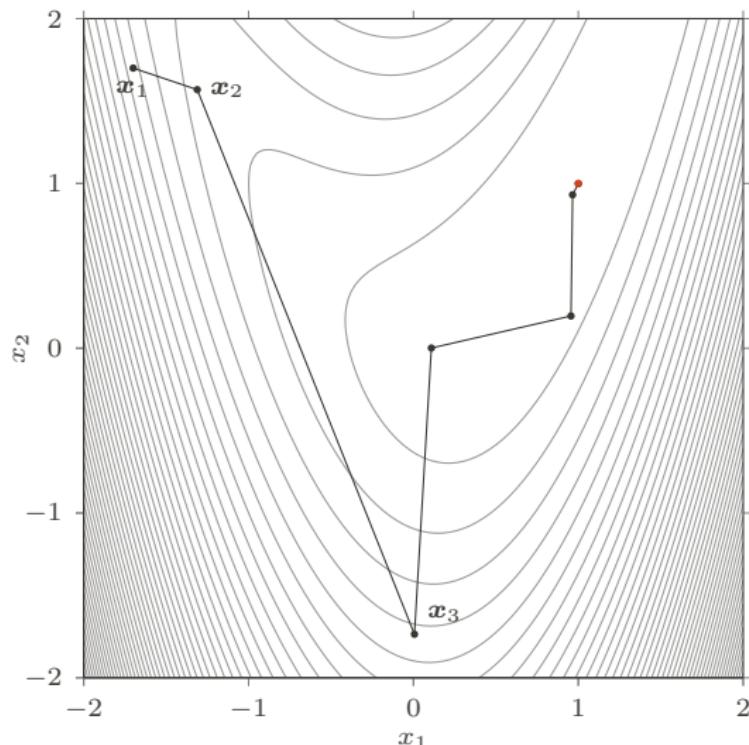
Levenberg-Marquardt method with the optimal step size

$$f(x_1, x_2) = 2.5(x_1^2 - x_2)^2 + (1 - x_1)^2$$



Compare with the original Newton-Raphson method

$$f(x_1, x_2) = 2.5(x_1^2 - x_2)^2 + (1 - x_1)^2$$



Summary

Two modifications:

1. Optimal step size α_k
2. Ensuring positive-definiteness of the Hessian (Levenberg-Marquardt)

Summary

Two modifications:

1. Optimal step size α_k
2. Ensuring positive-definiteness of the Hessian (Levenberg-Marquardt)

If we use (1), we should also use (2), otherwise the algorithm can get stuck at a suboptimal solution!

Summary

Two modifications:

1. Optimal step size α_k
2. Ensuring positive-definiteness of the Hessian (Levenberg-Marquardt)

If we use (1), we should also use (2), otherwise the algorithm can get stuck at a suboptimal solution!

Step size optimization ensures the convergence of the algorithm, but not necessarily improves its convergence speed, while increasing its computational complexity

Summary

Two modifications:

1. Optimal step size α_k
2. Ensuring positive-definiteness of the Hessian (Levenberg-Marquardt)

If we use (1), we should also use (2), otherwise the algorithm can get stuck at a suboptimal solution!

Step size optimization ensures the convergence of the algorithm, but not necessarily improves its convergence speed, while increasing its computational complexity

Good compromise: use the original Newton-Raphson method (with $\alpha_k = 1$), but if the objective does not decrease in a given iteration (or the Hessian is non-invertible), apply the Levenberg-Marquardt step with optimized α_k .

Conjugate gradient method

Reminder from linear algebra: vector basis

A set of vectors $\{v_1, \dots, v_m\}$ is called **linearly independent**, if none of the vectors from this set can be written as a linear combination of the others

In \mathbb{R}^n space, a linearly independent set of vectors can consist of at most n vectors, and such a set is called a **basis of a vector space**

Conclusion: If $\{v_1, \dots, v_n\}$ is a basis then **any vector** $x \in \mathbb{R}^n$ must be a linear combination of the basis vectors, i.e., for some coefficients

$$c_1, \dots, c_n \in \mathbb{R}$$

$$x = \sum_{k=1}^n c_k v_k$$

Conjugate vectors

Let A be a **positive-definite matrix**

A set of **non-zero** vectors $\{v_1, \dots, v_n\}$ is called **conjugate** (with respect to the matrix A), if

$$v_i^\top A v_j = 0, \quad \text{for any } i \neq j$$

Conjugate vectors

Let A be a **positive-definite matrix**

A set of **non-zero** vectors $\{v_1, \dots, v_n\}$ is called **conjugate** (with respect to the matrix A), if

$$v_i^\top A v_j = 0, \quad \text{for any } i \neq j$$

Fact: A set of conjugate vectors is a **basis** of the vector space

Conjugate vectors

Let A be a **positive-definite matrix**

A set of **non-zero** vectors $\{v_1, \dots, v_n\}$ is called **conjugate** (with respect to the matrix A), if

$$v_i^\top A v_j = 0, \quad \text{for any } i \neq j$$

Fact: A set of conjugate vectors is a **basis** of the vector space

Proof: It suffices to show that the vectors are linearly independent.

Assume the contrary, that there exists vector v_k which is linearly dependent of the others, i.e.:

$$v_k = \sum_{i \neq k} \alpha_i v_i$$

Conjugate vectors

Let A be a **positive-definite matrix**

A set of **non-zero** vectors $\{v_1, \dots, v_n\}$ is called **conjugate** (with respect to the matrix A), if

$$v_i^\top A v_j = 0, \quad \text{for any } i \neq j$$

Fact: A set of conjugate vectors is a **basis** of the vector space

Proof: It suffices to show that the vectors are linearly independent.

Assume the contrary, that there exists vector v_k which is linearly dependent of the others, i.e.:

$$v_k = \sum_{i \neq k} \alpha_i v_i$$

But this means that:

$$A v_k = \sum_{i \neq k} \alpha_i A v_i$$

Conjugate vectors

Let A be a **positive-definite matrix**

A set of **non-zero** vectors $\{v_1, \dots, v_n\}$ is called **conjugate** (with respect to the matrix A), if

$$v_i^\top A v_j = 0, \quad \text{for any } i \neq j$$

Fact: A set of conjugate vectors is a **basis** of the vector space

Proof: It suffices to show that the vectors are linearly independent.

Assume the contrary, that there exists vector v_k which is linearly dependent of the others, i.e.:

$$v_k = \sum_{i \neq k} \alpha_i v_i$$

But this means that:

$$v_k^\top A v_k = \sum_{i \neq k} \alpha_i v_k^\top A v_i$$

Conjugate vectors

Let A be a **positive-definite matrix**

A set of **non-zero** vectors $\{v_1, \dots, v_n\}$ is called **conjugate** (with respect to the matrix A), if

$$v_i^\top A v_j = 0, \quad \text{for any } i \neq j$$

Fact: A set of conjugate vectors is a **basis** of the vector space

Proof: It suffices to show that the vectors are linearly independent.

Assume the contrary, that there exists vector v_k which is linearly dependent of the others, i.e.:

$$v_k = \sum_{i \neq k} \alpha_i v_i$$

But this means that:

$$v_k^\top A v_k = \sum_{i \neq k} \alpha_i \underbrace{v_k^\top A v_i}_{=0}$$

Conjugate vectors

Let A be a **positive-definite matrix**

A set of **non-zero** vectors $\{v_1, \dots, v_n\}$ is called **conjugate** (with respect to the matrix A), if

$$v_i^\top A v_j = 0, \quad \text{for any } i \neq j$$

Fact: A set of conjugate vectors is a **basis** of the vector space

Proof: It suffices to show that the vectors are linearly independent.

Assume the contrary, that there exists vector v_k which is linearly dependent of the others, i.e.:

$$v_k = \sum_{i \neq k} \alpha_i v_i$$

But this means that:

$$v_k^\top A v_k = 0$$

which gives a contradiction, because A is positive-definite, and v_k is non-zero, so $v_k^\top A v_k > 0$

Quadratic function

We will consider minimizing a quadratic function:

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \mathbf{A} \mathbf{x} - \mathbf{x}^\top \mathbf{b}, \quad \mathbf{A} \text{ -- positive-def.}$$

We drop coefficient c (does not influence the optimization)

The other coefficients are chosen so that:

$$\nabla f(\mathbf{x}) = \mathbf{0} \iff \mathbf{A} \mathbf{x} = \mathbf{b},$$

i.e. the minimizer $\mathbf{x}^* = \mathbf{A}^{-1} \mathbf{b}$ is the solution to a system of equations
 $\mathbf{A} \mathbf{x}^* = \mathbf{b}$

Conjugate gradient method: introduction

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \mathbf{A} \mathbf{x} - \mathbf{x}^\top \mathbf{b}, \quad \mathbf{A} \text{ -- positive-def.}$$

Let $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ be a conjugate set with respect to matrix \mathbf{A}

We will consider a descent method for minimizing $f(\mathbf{x})$, in which the consecutive descent directions are **consecutive conjugate vectors**, i.e.:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{v}_k, \quad \text{i.e., } \mathbf{x}_{k+1} = \mathbf{x}_1 + \sum_{i=1}^k \alpha_i \mathbf{v}_i,$$

while coefficients α_k are obtained by solving:

$$\alpha_k = \underset{\alpha}{\operatorname{argmin}} \underbrace{f(\mathbf{x}_k + \alpha_k \mathbf{v}_k)}_{g(\alpha)}$$

Conjugate gradient method: introduction

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \mathbf{A} \mathbf{x} - \mathbf{x}^\top \mathbf{b}, \quad \mathbf{A} \text{ -- positive-def.}$$

Let $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ be a conjugate set with respect to matrix \mathbf{A}

We will consider a descent method for minimizing $f(\mathbf{x})$, in which the consecutive descent directions are **consecutive conjugate vectors**, i.e.:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{v}_k, \quad \text{i.e., } \mathbf{x}_{k+1} = \mathbf{x}_1 + \sum_{i=1}^k \alpha_i \mathbf{v}_i,$$

while coefficients α_k are obtained by solving:

$$\alpha_k = \underset{\alpha}{\operatorname{argmin}} \underbrace{f(\mathbf{x}_k + \alpha \mathbf{v}_k)}_{g(\alpha)}$$

$$g(\alpha) = \frac{1}{2} (\mathbf{x}_k + \alpha \mathbf{v}_k)^\top \mathbf{A} (\mathbf{x}_k + \alpha \mathbf{v}_k) - (\mathbf{x}_k + \alpha \mathbf{v}_k)^\top \mathbf{b}$$

Conjugate gradient method: introduction

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \mathbf{A} \mathbf{x} - \mathbf{x}^\top \mathbf{b}, \quad \mathbf{A} \text{ -- positive-def.}$$

Let $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ be a conjugate set with respect to matrix \mathbf{A}

We will consider a descent method for minimizing $f(\mathbf{x})$, in which the consecutive descent directions are **consecutive conjugate vectors**, i.e.:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{v}_k, \quad \text{i.e., } \mathbf{x}_{k+1} = \mathbf{x}_1 + \sum_{i=1}^k \alpha_i \mathbf{v}_i,$$

while coefficients α_k are obtained by solving:

$$\alpha_k = \underset{\alpha}{\operatorname{argmin}} \underbrace{f(\mathbf{x}_k + \alpha_k \mathbf{v}_k)}_{g(\alpha)}$$

$$g(\alpha) = \frac{\alpha^2}{2} \mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k + \alpha (\mathbf{v}_k^\top \mathbf{A} \mathbf{x}_k - \mathbf{v}_k^\top \mathbf{b}) + \text{const}$$

Conjugate gradient method: introduction

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \mathbf{A} \mathbf{x} - \mathbf{x}^\top \mathbf{b}, \quad \mathbf{A} \text{ -- positive-def.}$$

Let $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ be a conjugate set with respect to matrix \mathbf{A}

We will consider a descent method for minimizing $f(\mathbf{x})$, in which the consecutive descent directions are **consecutive conjugate vectors**, i.e.:

$$\mathbf{r}_{k+1} = \mathbf{r}_k + \alpha_k \mathbf{v}_k \quad \text{i.e.} \quad \mathbf{x}_{k+1} = \mathbf{x}_k + \sum_{i=1}^k \alpha_i \mathbf{v}_i,$$

gradient: $\mathbf{A}\mathbf{x}_k - \mathbf{b} = \nabla f(\mathbf{x}_k)$

while coefficients denoted by \mathbf{g}_k are being:

$$\alpha_k = \arg \min_{\alpha} \underbrace{f(\mathbf{x}_k + \alpha_k \mathbf{v}_k)}_{g(\alpha)}$$

$$g(\alpha) = \frac{\alpha^2}{2} \mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k + \alpha \mathbf{v}_k^\top (\mathbf{A}\mathbf{x}_k - \mathbf{b}) + \text{const}$$

Conjugate gradient method: introduction

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \mathbf{A} \mathbf{x} - \mathbf{x}^\top \mathbf{b}, \quad \mathbf{A} \text{ -- positive-def.}$$

Let $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ be a conjugate set with respect to matrix \mathbf{A}

We will consider a descent method for minimizing $f(\mathbf{x})$, in which the consecutive descent directions are **consecutive conjugate vectors**, i.e.:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{v}_k, \quad \text{i.e., } \mathbf{x}_{k+1} = \mathbf{x}_1 + \sum_{i=1}^k \alpha_i \mathbf{v}_i,$$

while coefficients α_k are obtained by solving:

$$\alpha_k = \underset{\alpha}{\operatorname{argmin}} \underbrace{f(\mathbf{x}_k + \alpha_k \mathbf{v}_k)}_{g(\alpha)}$$

$$g(\alpha) = \frac{\alpha^2}{2} \mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k + \alpha \mathbf{v}_k^\top \mathbf{g}_k + \text{const}$$

Conjugate gradient method: introduction

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \mathbf{A} \mathbf{x} - \mathbf{x}^\top \mathbf{b}, \quad \mathbf{A} \text{ -- positive-def.}$$

Let $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ be a conjugate set with respect to matrix \mathbf{A}

We will consider a descent method for minimizing $f(\mathbf{x})$, in which the consecutive descent directions are **consecutive conjugate vectors**, i.e.:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{v}_k, \quad \text{i.e., } \mathbf{x}_{k+1} = \mathbf{x}_1 + \sum_{i=1}^k \alpha_i \mathbf{v}_i,$$

while coefficients α_k are obtained by solving:

$$\alpha_k = \underset{\alpha}{\operatorname{argmin}} \underbrace{f(\mathbf{x}_k + \alpha \mathbf{v}_k)}_{g(\alpha)}$$

$$g'(\alpha) = \alpha \mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k + \mathbf{v}_k^\top \mathbf{g}_k$$

Conjugate gradient method: introduction

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \mathbf{A} \mathbf{x} - \mathbf{x}^\top \mathbf{b}, \quad \mathbf{A} \text{ -- positive-def.}$$

Let $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ be a conjugate set with respect to matrix \mathbf{A}

We will consider a descent method for minimizing $f(\mathbf{x})$, in which the consecutive descent directions are **consecutive conjugate vectors**, i.e.:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{v}_k, \quad \text{i.e., } \mathbf{x}_{k+1} = \mathbf{x}_1 + \sum_{i=1}^k \alpha_i \mathbf{v}_i,$$

while coefficients α_k are obtained by solving:

$$\alpha_k = \underset{\alpha}{\operatorname{argmin}} \underbrace{f(\mathbf{x}_k + \alpha_k \mathbf{v}_k)}_{g(\alpha)}$$

$$g'(\alpha) = 0 \iff \alpha \mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k = -\mathbf{v}_k^\top \mathbf{g}_k$$

Conjugate gradient method: introduction

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \mathbf{A} \mathbf{x} - \mathbf{x}^\top \mathbf{b}, \quad \mathbf{A} \text{ -- positive-def.}$$

Let $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ be a conjugate set with respect to matrix \mathbf{A}

We will consider a descent method for minimizing $f(\mathbf{x})$, in which the consecutive descent directions are **consecutive conjugate vectors**, i.e.:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{v}_k, \quad \text{i.e., } \mathbf{x}_{k+1} = \mathbf{x}_1 + \sum_{i=1}^k \alpha_i \mathbf{v}_i,$$

while coefficients α_k are obtained by solving:

$$\alpha_k = \underset{\alpha}{\operatorname{argmin}} \underbrace{f(\mathbf{x}_k + \alpha_k \mathbf{v}_k)}_{g(\alpha)}$$

$$\alpha_k = - \frac{\mathbf{v}_k^\top \mathbf{g}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k}$$

Conjugate gradient method: introduction

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \mathbf{A} \mathbf{x} - \mathbf{x}^\top \mathbf{b}, \quad \mathbf{A} \text{ -- positive-def.}$$

Let $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ be a conjugate set with respect to matrix \mathbf{A}

We will consider a descent method for minimizing $f(\mathbf{x})$, in which the consecutive descent directions are **consecutive conjugate vectors**, i.e.:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{v}_k, \quad \text{i.e., } \mathbf{x}_{k+1} = \mathbf{x}_1 + \sum_{i=1}^k \alpha_i \mathbf{v}_i,$$

while coefficients α_k are obtained by solving:

$$\begin{aligned}\alpha_k &= \underset{\alpha}{\operatorname{argmin}} \underbrace{f(\mathbf{x}_k + \alpha \mathbf{v}_k)}_{g(\alpha)} \\ \alpha_k &= - \frac{\mathbf{v}_k^\top \mathbf{g}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k} = - \frac{\mathbf{v}_k^\top (\mathbf{A} \mathbf{x}_k - \mathbf{b})}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k}\end{aligned}$$

Conjugate gradient method: introduction

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \mathbf{A} \mathbf{x} - \mathbf{x}^\top \mathbf{b}, \quad \mathbf{A} \text{ -- positive-def.}$$

Let $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ be a conjugate set with respect to matrix \mathbf{A}

We will consider a descent method for minimizing $f(\mathbf{x})$, in which the consecutive descent directions are **consecutive conjugate vectors**, i.e.:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{v}_k, \quad \text{i.e., } \mathbf{x}_{k+1} = \mathbf{x}_1 + \sum_{i=1}^k \alpha_i \mathbf{v}_i,$$

while coefficients α_k are obtained by solving:

$$\begin{aligned} \alpha_k &= \underset{\alpha}{\operatorname{argmin}} \underbrace{f(\mathbf{x}_k + \alpha_k \mathbf{v}_k)}_{g(\alpha)} \\ \alpha_k &= - \frac{\mathbf{v}_k^\top \mathbf{g}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k} = - \frac{\mathbf{v}_k^\top (\mathbf{A} \mathbf{x}_k - \mathbf{A} \mathbf{x}_1 + \mathbf{A} \mathbf{x}_1 - \mathbf{b})}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k} \end{aligned}$$

Conjugate gradient method: introduction

Since $\mathbf{x}_k - \mathbf{x}_1 = \sum_{i=1}^{k-1} \alpha_i \mathbf{v}_i$, we get

Let
We
con

$$\mathbf{v}_k^\top \mathbf{A}(\mathbf{x}_k - \mathbf{x}_1) = \sum_{i=1}^{k-1} \alpha_i \underbrace{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_i}_{=0} = 0$$

\mathbf{A} – positive-def.

respect to matrix \mathbf{A}
imizing $f(\mathbf{x})$, in which the
consecutive conjugate vectors, i.e.:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{v}_k, \quad \text{i.e., } \mathbf{x}_{k+1} = \mathbf{x}_1 + \sum_{i=1}^k \alpha_i \mathbf{v}_i,$$

while coefficients α_k are obtained by solving:

$$\begin{aligned}\alpha_k &= \underset{\alpha}{\operatorname{argmin}} \underbrace{f(\mathbf{x}_k + \alpha_k \mathbf{v}_k)}_{g(\alpha)} \\ \alpha_k &= -\frac{\mathbf{v}_k^\top \mathbf{g}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k} = -\frac{\mathbf{v}_k^\top (\mathbf{A}\mathbf{x}_1 - \mathbf{b}) + \mathbf{v}_k^\top \mathbf{A}(\mathbf{x}_k - \mathbf{x}_1)}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k}\end{aligned}$$

Conjugate gradient method: introduction

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \mathbf{A} \mathbf{x} - \mathbf{x}^\top \mathbf{b}, \quad \mathbf{A} \text{ -- positive-def.}$$

Let $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ be a conjugate set with respect to matrix \mathbf{A}

We will consider a descent method for minimizing $f(\mathbf{x})$, in which the consecutive descent directions are **consecutive conjugate vectors**, i.e.:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{v}_k, \quad \text{i.e., } \mathbf{x}_{k+1} = \mathbf{x}_1 + \sum_{i=1}^k \alpha_i \mathbf{v}_i,$$

while coefficients α_k are obtained by solving:

$$\begin{aligned}\alpha_k &= \underset{\alpha}{\operatorname{argmin}} \underbrace{f(\mathbf{x}_k + \alpha \mathbf{v}_k)}_{g(\alpha)} \\ \alpha_k &= - \frac{\mathbf{v}_k^\top \mathbf{g}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k} = \frac{\mathbf{v}_k^\top (\mathbf{b} - \mathbf{A} \mathbf{x}_1)}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k}\end{aligned}$$

The optimal α_k only depends on the starting point \mathbf{x}_1 !

We can select coefficient α_k independently of the other coefficients and previous steps of the algorithm

The quadratic function and the conjugate vectors

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \mathbf{A} \mathbf{x} - \mathbf{x}^\top \mathbf{b}, \quad \mathbf{A} \text{ -- positive-def.}$$

Let $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ – conjugate set. Since it is also a basis,

$$\mathbf{x}^* - \mathbf{x}_1 = \sum_{i=1}^n \alpha_i \mathbf{v}_i$$

Compute the optimal coefficients α_k

The quadratic function and the conjugate vectors

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \mathbf{A} \mathbf{x} - \mathbf{x}^\top \mathbf{b}, \quad \mathbf{A} \text{ -- positive-def.}$$

Let $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ – conjugate set. Since it is also a basis,

$$\mathbf{x}^* - \mathbf{x}_1 = \sum_{i=1}^n \alpha_i \mathbf{v}_i$$

Compute the optimal coefficients α_k

$$\mathbf{A}\mathbf{x}^* - \mathbf{A}\mathbf{x}_1 = \sum_{i=1}^n \alpha_i \mathbf{A}\mathbf{v}_i$$

The quadratic function and the conjugate vectors

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \mathbf{A} \mathbf{x} - \mathbf{x}^\top \mathbf{b}, \quad \mathbf{A} \text{ -- positive-def.}$$

Let $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ – conjugate set. Since it is also a basis,

$$\mathbf{x}^* - \mathbf{x}_1 = \sum_{i=1}^n \alpha_i \mathbf{v}_i$$

Compute the optimal coefficients α_k

$$\mathbf{b} - \mathbf{A} \mathbf{x}_1 = \sum_{i=1}^n \alpha_i \mathbf{A} \mathbf{v}_i$$

The quadratic function and the conjugate vectors

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \mathbf{A} \mathbf{x} - \mathbf{x}^\top \mathbf{b}, \quad \mathbf{A} \text{ -- positive-def.}$$

Let $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ – conjugate set. Since it is also a basis,

$$\mathbf{x}^* - \mathbf{x}_1 = \sum_{i=1}^n \alpha_i \mathbf{v}_i$$

Compute the optimal coefficients α_k

$$\mathbf{v}_k^\top (\mathbf{b} - \mathbf{A} \mathbf{x}_1) = \sum_{i=1}^n \alpha_k \underbrace{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_i}_{=0 \text{ dla } k \neq i}$$

The quadratic function and the conjugate vectors

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \mathbf{A} \mathbf{x} - \mathbf{x}^\top \mathbf{b}, \quad \mathbf{A} \text{ -- positive-def.}$$

Let $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ – conjugate set. Since it is also a basis,

$$\mathbf{x}^* - \mathbf{x}_1 = \sum_{i=1}^n \alpha_i \mathbf{v}_i$$

Compute the optimal coefficients α_k

$$\mathbf{v}_k^\top (\mathbf{b} - \mathbf{A} \mathbf{x}_1) = \alpha_k \mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k$$

The quadratic function and the conjugate vectors

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \mathbf{A} \mathbf{x} - \mathbf{x}^\top \mathbf{b}, \quad \mathbf{A} \text{ -- positive-def.}$$

Let $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ – conjugate set. Since it is also a basis,

$$\mathbf{x}^* - \mathbf{x}_1 = \sum_{i=1}^n \alpha_i \mathbf{v}_i$$

Compute the optimal coefficients α_k

$$\alpha_k = \frac{\mathbf{v}_k^\top (\mathbf{b} - \mathbf{A} \mathbf{x}_1)}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k},$$

The quadratic function and the conjugate vectors

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \mathbf{A} \mathbf{x} - \mathbf{x}^\top \mathbf{b}, \quad \mathbf{A} \text{ -- positive-def.}$$

Let $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ – conjugate set. Since it is also a basis,

$$\mathbf{x}^* - \mathbf{x}_1 = \sum_{i=1}^n \alpha_i \mathbf{v}_i$$

Compute the optimal coefficients α_k

$$\alpha_k = \frac{\mathbf{v}_k^\top (\mathbf{b} - \mathbf{A} \mathbf{x}_1)}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k},$$

But these are **the same** coefficients as those selected in consecutive steps by the conjugate gradient method, i.e.

$$\mathbf{x}_{n+1} = \mathbf{x}_1 + \sum_{k=1}^n \alpha_k \mathbf{v}_k = \mathbf{x}^*$$

The quadratic function and the conjugate vectors

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^\top \mathbf{A} \mathbf{x} - \mathbf{x}^\top \mathbf{b}, \quad \mathbf{A} \text{ -- positive-def.}$$

Let $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ – conjugate set. Since it is also a basis,

$$\mathbf{x}^* - \mathbf{x}_1 = \sum_{i=1}^n \alpha_i \mathbf{v}_i$$

Compute the optimal coefficients α_k

$$\alpha_k = \frac{\mathbf{v}_k^\top (\mathbf{b} - \mathbf{A}\mathbf{x}_1)}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k},$$

But these are **the same** coefficients as those selected in consecutive steps by the conjugate gradient method, i.e.

$$\mathbf{x}_{n+1} = \mathbf{x}_1 + \sum_{k=1}^n \alpha_k \mathbf{v}_k = \mathbf{x}^*$$

Conclusion: the conjugate gradient method chooses coefficients α_k in each step optimally, so the optimal solution \mathbf{x}^* is obtained after n steps!

Orthogonality of the gradients

The gradient in each iteration $\mathbf{g}_k = \nabla f(\mathbf{x}_k)$ is **orthogonal** to all previous descent directions $\mathbf{v}_1, \dots, \mathbf{v}_{k-1}$, i.e.:

$$\mathbf{v}_i^\top \mathbf{g}_k = 0, \quad \text{for } i = 1, \dots, k-1$$

Orthogonality of the gradients

The gradient in each iteration $\mathbf{g}_k = \nabla f(\mathbf{x}_k)$ is **orthogonal** to all previous descent directions $\mathbf{v}_1, \dots, \mathbf{v}_{k-1}$, i.e.:

$$\mathbf{v}_i^\top \mathbf{g}_k = 0, \quad \text{for } i = 1, \dots, k-1$$

Proof:

$$\mathbf{v}_i^\top \mathbf{g}_k = \mathbf{v}_i^\top (\mathbf{A}\mathbf{x}_k - \mathbf{b})$$

Orthogonality of the gradients

The gradient in each iteration $\mathbf{g}_k = \nabla f(\mathbf{x}_k)$ is **orthogonal** to all previous descent directions $\mathbf{v}_1, \dots, \mathbf{v}_{k-1}$, i.e.:

$$\mathbf{v}_i^\top \mathbf{g}_k = 0, \quad \text{for } i = 1, \dots, k-1$$

Proof:

$$\begin{aligned}\mathbf{v}_i^\top \mathbf{g}_k &= \mathbf{v}_i^\top (\mathbf{A}\mathbf{x}_k - \mathbf{b}) \\ &= \mathbf{v}_i^\top \left(\mathbf{A}\mathbf{x}_1 + \mathbf{A} \left(\sum_{j=1}^{k-1} \alpha_j \mathbf{v}_j \right) - \mathbf{b} \right)\end{aligned}$$

Orthogonality of the gradients

The gradient in each iteration $\mathbf{g}_k = \nabla f(\mathbf{x}_k)$ is **orthogonal** to all previous descent directions $\mathbf{v}_1, \dots, \mathbf{v}_{k-1}$, i.e.:

$$\mathbf{v}_i^\top \mathbf{g}_k = 0, \quad \text{for } i = 1, \dots, k-1$$

Proof:

$$\begin{aligned}\mathbf{v}_i^\top \mathbf{g}_k &= \mathbf{v}_i^\top (\mathbf{A}\mathbf{x}_k - \mathbf{b}) \\ &= \mathbf{v}_i^\top \left(\mathbf{A}\mathbf{x}_1 + \mathbf{A} \left(\sum_{j=1}^{k-1} \alpha_j \mathbf{v}_j \right) - \mathbf{b} \right) \\ &= \mathbf{v}_i^\top (\mathbf{A}\mathbf{x}_1 - \mathbf{b}) + \sum_{j=1}^{k-1} \alpha_j \underbrace{\mathbf{v}_i^\top \mathbf{A} \mathbf{v}_j}_{=0 \text{ dla } i \neq j}\end{aligned}$$

Orthogonality of the gradients

The gradient in each iteration $\mathbf{g}_k = \nabla f(\mathbf{x}_k)$ is **orthogonal** to all previous descent directions $\mathbf{v}_1, \dots, \mathbf{v}_{k-1}$, i.e.:

$$\mathbf{v}_i^\top \mathbf{g}_k = 0, \quad \text{for } i = 1, \dots, k-1$$

Proof:

$$\begin{aligned}\mathbf{v}_i^\top \mathbf{g}_k &= \mathbf{v}_i^\top (\mathbf{A}\mathbf{x}_k - \mathbf{b}) \\ &= \mathbf{v}_i^\top \left(\mathbf{A}\mathbf{x}_1 + \mathbf{A} \left(\sum_{j=1}^{k-1} \alpha_j \mathbf{v}_j \right) - \mathbf{b} \right) \\ &= \mathbf{v}_i^\top (\mathbf{A}\mathbf{x}_1 - \mathbf{b}) + \sum_{j=1}^{k-1} \alpha_j \underbrace{\mathbf{v}_i^\top \mathbf{A} \mathbf{v}_j}_{=0 \text{ dla } i \neq j} \\ &= \mathbf{v}_i^\top (\mathbf{A}\mathbf{x}_1 - \mathbf{b}) + \alpha_i \mathbf{v}_i^\top \mathbf{A} \mathbf{v}_i\end{aligned}$$

Orthogonality of the gradients

The gradient in each iteration $\mathbf{g}_k = \nabla f(\mathbf{x}_k)$ is **orthogonal** to all previous descent directions $\mathbf{v}_1, \dots, \mathbf{v}_{k-1}$, i.e.:

$$\mathbf{v}_i^\top \mathbf{g}_k = 0, \quad \text{for } i = 1, \dots, k-1$$

Proof:

$$\begin{aligned}\mathbf{v}_i^\top \mathbf{g}_k &= \mathbf{v}_i^\top (\mathbf{A}\mathbf{x}_k - \mathbf{b}) \\ &= \mathbf{v}_i^\top \left(\mathbf{A}\mathbf{x}_1 + \mathbf{A} \left(\sum_{j=1}^{k-1} \alpha_j \mathbf{v}_j \right) - \mathbf{b} \right) \\ &= \mathbf{v}_i^\top (\mathbf{A}\mathbf{x}_1 - \mathbf{b}) + \sum_{j=1}^{k-1} \alpha_j \underbrace{\mathbf{v}_i^\top \mathbf{A} \mathbf{v}_j}_{=0 \text{ dla } i \neq j} \\ &= \mathbf{v}_i^\top (\mathbf{A}\mathbf{x}_1 - \mathbf{b}) + \alpha_i \mathbf{v}_i^\top \mathbf{A} \mathbf{v}_i \\ &= \mathbf{v}_i^\top (\mathbf{A}\mathbf{x}_1 - \mathbf{b}) + \frac{\mathbf{v}_i^\top (\mathbf{b} - \mathbf{A}\mathbf{x}_1)}{\mathbf{v}_i^\top \mathbf{A} \mathbf{v}_i} \mathbf{v}_i^\top \mathbf{A} \mathbf{v}_i = 0\end{aligned}$$

How to find conjugate directions?

How to find conjugate directions?

- As the first direction, choose the negative gradient:

$$\mathbf{v}_1 = -\nabla f(\mathbf{x}_1) = -\mathbf{g}_1$$

How to find conjugate directions?

- As the first direction, choose the negative gradient:

$$\mathbf{v}_1 = -\nabla f(\mathbf{x}_1) = -\mathbf{g}_1$$

- Assume we already chose conjugate directions $\mathbf{v}_1, \dots, \mathbf{v}_k$ and want to choose \mathbf{v}_{k+1} . Again, a natural candidate is $-\mathbf{g}_{k+1}$, but it is not necessarily conjugate to $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$

How to find conjugate directions?

- As the first direction, choose the negative gradient:

$$\mathbf{v}_1 = -\nabla f(\mathbf{x}_1) = -\mathbf{g}_1$$

- Assume we already chose conjugate directions $\mathbf{v}_1, \dots, \mathbf{v}_k$ and want to choose \mathbf{v}_{k+1} . Again, a natural candidate is $-\mathbf{g}_{k+1}$, but it is not necessarily conjugate to $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$
- Idea:** add a linear combination of previous conjugate directions to the gradient to make it conjugate as well, i.e. $\mathbf{v}_j^\top \mathbf{A} \mathbf{v}_{k+1} = 0$ for $j \leq k$

$$\mathbf{v}_{k+1} = -\mathbf{g}_{k+1} + \sum_{i=1}^k \beta_{k,i} \mathbf{v}_i$$

How to find conjugate directions?

- As the first direction, choose the negative gradient:

$$\mathbf{v}_1 = -\nabla f(\mathbf{x}_1) = -\mathbf{g}_1$$

- Assume we already chose conjugate directions $\mathbf{v}_1, \dots, \mathbf{v}_k$ and want to choose \mathbf{v}_{k+1} . Again, a natural candidate is $-\mathbf{g}_{k+1}$, but it is not necessarily conjugate to $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$
- Idea:** add a linear combination of previous conjugate directions to the gradient to make it conjugate as well, i.e. $\mathbf{v}_j^\top \mathbf{A} \mathbf{v}_{k+1} = 0$ for $j \leq k$

$$\mathbf{v}_{k+1} = -\mathbf{g}_{k+1} + \sum_{i=1}^k \beta_{k,i} \mathbf{v}_i$$

- Compute these coefficients:

$$\mathbf{v}_{k+1} = -\mathbf{g}_{k+1} + \sum_{i=1}^k \beta_{k,i} \mathbf{v}_i$$

How to find conjugate directions?

- As the first direction, choose the negative gradient:

$$\mathbf{v}_1 = -\nabla f(\mathbf{x}_1) = -\mathbf{g}_1$$

- Assume we already chose conjugate directions $\mathbf{v}_1, \dots, \mathbf{v}_k$ and want to choose \mathbf{v}_{k+1} . Again, a natural candidate is $-\mathbf{g}_{k+1}$, but it is not necessarily conjugate to $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$
- Idea:** add a linear combination of previous conjugate directions to the gradient to make it conjugate as well, i.e. $\mathbf{v}_j^\top \mathbf{A} \mathbf{v}_{k+1} = 0$ for $j \leq k$

$$\mathbf{v}_{k+1} = -\mathbf{g}_{k+1} + \sum_{i=1}^k \beta_{k,i} \mathbf{v}_i$$

- Compute these coefficients:

$$\mathbf{A} \mathbf{v}_{k+1} = -\mathbf{A} \mathbf{g}_{k+1} + \sum_{i=1}^k \beta_{k,i} \mathbf{A} \mathbf{v}_i$$

How to find conjugate directions?

- As the first direction, choose the negative gradient:

$$\mathbf{v}_1 = -\nabla f(\mathbf{x}_1) = -\mathbf{g}_1$$

- Assume we already chose conjugate directions $\mathbf{v}_1, \dots, \mathbf{v}_k$ and want to choose \mathbf{v}_{k+1} . Again, a natural candidate is $-\mathbf{g}_{k+1}$, but it is not necessarily conjugate to $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$
- Idea:** add a linear combination of previous conjugate directions to the gradient to make it conjugate as well, i.e. $\mathbf{v}_j^\top A \mathbf{v}_{k+1} = 0$ for $j \leq k$

$$\mathbf{v}_{k+1} = -\mathbf{g}_{k+1} + \sum_{i=1}^k \beta_{k,i} \mathbf{v}_i$$

- Compute these coefficients:

$$\mathbf{v}_j^\top A \mathbf{v}_{k+1} = -\mathbf{v}_j^\top A \mathbf{g}_{k+1} + \sum_{i=1}^k \beta_{k,i} \underbrace{\mathbf{v}_j^\top A \mathbf{v}_i}_{=0 \text{ dla } j \neq i}$$

How to find conjugate directions?

- As the first direction, choose the negative gradient:

$$\mathbf{v}_1 = -\nabla f(\mathbf{x}_1) = -\mathbf{g}_1$$

- Assume we already chose conjugate directions $\mathbf{v}_1, \dots, \mathbf{v}_k$ and want to choose \mathbf{v}_{k+1} . Again, a natural candidate is $-\mathbf{g}_{k+1}$, but it is not necessarily conjugate to $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$
- Idea:** add a linear combination of previous conjugate directions to the gradient to make it conjugate as well, i.e. $\mathbf{v}_j^\top \mathbf{A} \mathbf{v}_{k+1} = 0$ for $j \leq k$

$$\mathbf{v}_{k+1} = -\mathbf{g}_{k+1} + \sum_{i=1}^k \beta_{k,i} \mathbf{v}_i$$

- Compute these coefficients:

$$\underbrace{\mathbf{v}_j^\top \mathbf{A} \mathbf{v}_{k+1}}_{=0} = -\mathbf{v}_j^\top \mathbf{A} \mathbf{g}_{k+1} + \beta_{k,j} \mathbf{v}_j^\top \mathbf{A} \mathbf{v}_j$$

How to find conjugate directions?

- As the first direction, choose the negative gradient:

$$\mathbf{v}_1 = -\nabla f(\mathbf{x}_1) = -\mathbf{g}_1$$

- Assume we already chose conjugate directions $\mathbf{v}_1, \dots, \mathbf{v}_k$ and want to choose \mathbf{v}_{k+1} . Again, a natural candidate is $-\mathbf{g}_{k+1}$, but it is not necessarily conjugate to $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$
- Idea:** add a linear combination of previous conjugate directions to the gradient to make it conjugate as well, i.e. $\mathbf{v}_j^\top \mathbf{A} \mathbf{v}_{k+1} = 0$ for $j \leq k$

$$\mathbf{v}_{k+1} = -\mathbf{g}_{k+1} + \sum_{i=1}^k \beta_{k,i} \mathbf{v}_i$$

- Compute these coefficients:

$$\mathbf{v}_j^\top \mathbf{A} \mathbf{g}_{k+1} = \beta_{k,j} \mathbf{v}_j^\top \mathbf{A} \mathbf{v}_j$$

How to find conjugate directions?

- As the first direction, choose the negative gradient:

$$\mathbf{v}_1 = -\nabla f(\mathbf{x}_1) = -\mathbf{g}_1$$

- Assume we already chose conjugate directions $\mathbf{v}_1, \dots, \mathbf{v}_k$ and want to choose \mathbf{v}_{k+1} . Again, a natural candidate is $-\mathbf{g}_{k+1}$, but it is not necessarily conjugate to $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$
- Idea:** add a linear combination of previous conjugate directions to the gradient to make it conjugate as well, i.e. $\mathbf{v}_j^\top \mathbf{A} \mathbf{v}_{k+1} = 0$ for $j \leq k$

$$\mathbf{v}_{k+1} = -\mathbf{g}_{k+1} + \sum_{i=1}^k \beta_{k,i} \mathbf{v}_i$$

- Compute these coefficients:

$$\beta_{k,j} = \frac{\mathbf{v}_j^\top \mathbf{A} \mathbf{g}_{k+1}}{\mathbf{v}_j^\top \mathbf{A} \mathbf{v}_j}$$

How to find conjugate directions?

- As the first direction, choose the negative gradient:

$$\mathbf{v}_1 = -\nabla f(\mathbf{x}_1) = -\mathbf{g}_1$$

- Assume we already chose conjugate directions $\mathbf{v}_1, \dots, \mathbf{v}_k$ and want to choose \mathbf{v}_{k+1} . Again, a natural candidate is $-\mathbf{g}_{k+1}$, but it is not necessarily conjugate to $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$
- Idea:** add a linear combination of previous conjugate directions to the gradient to make it conjugate as well, i.e. $\mathbf{v}_j^\top \mathbf{A} \mathbf{v}_{k+1} = 0$ for $j \leq k$

$$\mathbf{v}_{k+1} = -\mathbf{g}_{k+1} + \sum_{i=1}^k \beta_{k,i} \mathbf{v}_i$$

- Compute these coefficients:

$$\beta_{k,j} = \frac{\mathbf{g}_{k+1}^\top \mathbf{A} \mathbf{v}_j}{\mathbf{v}_j^\top \mathbf{A} \mathbf{v}_j}$$

Problem: need to find k coefficients $\beta_{k,1}, \dots, \beta_{k,k}$ in each step, and store all previous conjugate directions $\mathbf{v}_1, \dots, \mathbf{v}_k$

How to find conjugate directions?

- As the first direction, choose the negative gradient:

$$\mathbf{v}_1 = -\nabla f(\mathbf{x}_1) = -\mathbf{g}_1$$

- Assume we already chose conjugate directions $\mathbf{v}_1, \dots, \mathbf{v}_k$ and want to choose \mathbf{v}_{k+1} . Again, a natural candidate is $-\mathbf{g}_{k+1}$, but it is not necessarily conjugate to $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$
- Idea:** add a linear combination of previous conjugate directions to the gradient to make it conjugate as well, i.e. $\mathbf{v}_j^\top \mathbf{A} \mathbf{v}_{k+1} = 0$ for $j \leq k$

$$\mathbf{v}_{k+1} = -\mathbf{g}_{k+1} + \sum_{i=1}^k \beta_{k,i} \mathbf{v}_i$$

- Compute these coefficients:

$$\beta_{k,j} = \frac{\mathbf{g}_{k+1}^\top \mathbf{A} \mathbf{v}_j}{\mathbf{v}_j^\top \mathbf{A} \mathbf{v}_j}$$

Problem **The magic of conjugate gradients:** It holds $\beta_{k,j} = 0$ for $j = 1, \dots, k-1$, so there is only one non-zero coefficient $\beta_{k,k}$, and store all

The key point in the construction of the algorithm

It holds $\mathbf{g}_{k+1}^\top \mathbf{A} \mathbf{v}_j = 0$ for $j = 1, \dots, k - 1$

The key point in the construction of the algorithm

It holds $\mathbf{g}_{k+1}^\top \mathbf{A} \mathbf{v}_j = 0$ for $j = 1, \dots, k - 1$

Proof: First we show that the gradient \mathbf{g}_{k+1} is **orthogonal** to all past gradients, i.e.

$$\mathbf{g}_{k+1}^\top \mathbf{g}_j = 0, \quad \text{for } j = 1, \dots, k$$

The key point in the construction of the algorithm

It holds $\mathbf{g}_{k+1}^\top \mathbf{A} \mathbf{v}_j = 0$ for $j = 1, \dots, k - 1$

Proof: First we show that the gradient \mathbf{g}_{k+1} is **orthogonal** to all past gradients, i.e.

$$\mathbf{g}_{k+1}^\top \mathbf{g}_j = 0, \quad \text{for } j = 1, \dots, k$$

- We have already shown that the gradient is orthogonal to all past descent directions, i.e. for $j \leq k$ we have $\mathbf{g}_{k+1}^\top \mathbf{v}_j = 0$

The key point in the construction of the algorithm

It holds $\mathbf{g}_{k+1}^\top \mathbf{A} \mathbf{v}_j = 0$ for $j = 1, \dots, k - 1$

Proof: First we show that the gradient \mathbf{g}_{k+1} is **orthogonal** to all past gradients, i.e.

$$\mathbf{g}_{k+1}^\top \mathbf{g}_j = 0, \quad \text{for } j = 1, \dots, k$$

- We have already shown that the gradient is orthogonal to all past descent directions, i.e. for $j \leq k$ we have $\mathbf{g}_{k+1}^\top \mathbf{v}_j = 0$
- From the rule for choosing a descent direction:

$$\mathbf{v}_j = -\mathbf{g}_j + \sum_{i=1}^{j-1} \beta_{j-1,i} \mathbf{v}_i \implies \mathbf{g}_j = -\mathbf{v}_j + \sum_{i=1}^{j-1} \beta_{j-1,i} \mathbf{v}_i$$

The key point in the construction of the algorithm

It holds $\mathbf{g}_{k+1}^\top \mathbf{A} \mathbf{v}_j = 0$ for $j = 1, \dots, k - 1$

Proof: First we show that the gradient \mathbf{g}_{k+1} is **orthogonal** to all past gradients, i.e.

$$\mathbf{g}_{k+1}^\top \mathbf{g}_j = 0, \quad \text{for } j = 1, \dots, k$$

- We have already shown that the gradient is orthogonal to all past descent directions, i.e. for $j \leq k$ we have $\mathbf{g}_{k+1}^\top \mathbf{v}_j = 0$
- From the rule for choosing a descent direction:

$$\mathbf{v}_j = -\mathbf{g}_j + \sum_{i=1}^{j-1} \beta_{j-1,i} \mathbf{v}_i \implies \mathbf{g}_j = -\mathbf{v}_j + \sum_{i=1}^{j-1} \beta_{j-1,i} \mathbf{v}_i$$

- So for $j \leq k$:

$$\mathbf{g}_{k+1}^\top \mathbf{g}_j = -\underbrace{\mathbf{g}_{k+1}^\top \mathbf{v}_j}_{=0} + \sum_{i=1}^{j-1} \beta_{j-1,i} \underbrace{\mathbf{g}_{k+1}^\top \mathbf{v}_i}_{=0} = 0$$

The key point in the construction of the algorithm

It holds $\mathbf{g}_{k+1}^\top \mathbf{A} \mathbf{v}_j = 0$ for $j = 1, \dots, k - 1$

Proof: We showed that gradient \mathbf{g}_{k+1} is **orthogonal** to all previous gradients, i.e. $\mathbf{g}_{k+1}^\top \mathbf{g}_j = 0$ for $j = 1, \dots, k$

The key point in the construction of the algorithm

It holds $\mathbf{g}_{k+1}^\top \mathbf{A} \mathbf{v}_j = 0$ for $j = 1, \dots, k - 1$

Proof: We showed that gradient \mathbf{g}_{k+1} is **orthogonal** to all previous gradients, i.e. $\mathbf{g}_{k+1}^\top \mathbf{g}_j = 0$ for $j = 1, \dots, k$

Now we use the fact that:

$$\mathbf{g}_{k+1} = \mathbf{A}\mathbf{x}_{k+1} - \mathbf{b} = \mathbf{A}(\mathbf{x}_k + \alpha_k \mathbf{v}_k) - \mathbf{b} = \underbrace{\mathbf{A}\mathbf{x}_k - \mathbf{b}}_{= \mathbf{g}_k} + \mathbf{A}\alpha_k \mathbf{v}_k,$$

The key point in the construction of the algorithm

It holds $\mathbf{g}_{k+1}^\top \mathbf{A} \mathbf{v}_j = 0$ for $j = 1, \dots, k - 1$

Proof: We showed that gradient \mathbf{g}_{k+1} is **orthogonal** to all previous gradients, i.e. $\mathbf{g}_{k+1}^\top \mathbf{g}_j = 0$ for $j = 1, \dots, k$

Now we use the fact that:

$$\mathbf{g}_{k+1} = \mathbf{A}\mathbf{x}_{k+1} - \mathbf{b} = \mathbf{A}(\mathbf{x}_k + \alpha_k \mathbf{v}_k) - \mathbf{b} = \underbrace{\mathbf{A}\mathbf{x}_k - \mathbf{b}}_{= \mathbf{g}_k} + \mathbf{A}\alpha_k \mathbf{v}_k,$$

If $\alpha_k = 0$, then $\mathbf{g}_{k+1} = \mathbf{g}_k$, and since $\mathbf{g}_{k+1}^\top \mathbf{g}_k = 0$, we conclude that $\mathbf{g}_{k+1} = \nabla f(\mathbf{x}_{k+1}) = \mathbf{0}$, so **we are at the minimum.**

The key point in the construction of the algorithm

It holds $\mathbf{g}_{k+1}^\top \mathbf{A} \mathbf{v}_j = 0$ for $j = 1, \dots, k - 1$

Proof: We showed that gradient \mathbf{g}_{k+1} is **orthogonal** to all previous gradients, i.e. $\mathbf{g}_{k+1}^\top \mathbf{g}_j = 0$ for $j = 1, \dots, k$

Now we use the fact that:

$$\mathbf{g}_{k+1} = \mathbf{A}\mathbf{x}_{k+1} - \mathbf{b} = \mathbf{A}(\mathbf{x}_k + \alpha_k \mathbf{v}_k) - \mathbf{b} = \underbrace{\mathbf{A}\mathbf{x}_k - \mathbf{b}}_{= \mathbf{g}_k} + \mathbf{A}\alpha_k \mathbf{v}_k,$$

If $\alpha_k = 0$, then $\mathbf{g}_{k+1} = \mathbf{g}_k$, and since $\mathbf{g}_{k+1}^\top \mathbf{g}_k = 0$, we conclude that $\mathbf{g}_{k+1} = \nabla f(\mathbf{x}_{k+1}) = \mathbf{0}$, so **we are at the minimum.**

If $\alpha_k \neq 0$, we have:

$$\mathbf{A}\mathbf{v}_k = \frac{1}{\alpha_k} (\mathbf{g}_{k+1} - \mathbf{g}_k)$$

The key point in the construction of the algorithm

It holds $\mathbf{g}_{k+1}^\top \mathbf{A} \mathbf{v}_j = 0$ for $j = 1, \dots, k - 1$

Proof: We showed that gradient \mathbf{g}_{k+1} is **orthogonal** to all previous gradients, i.e. $\mathbf{g}_{k+1}^\top \mathbf{g}_j = 0$ for $j = 1, \dots, k$

Now we use the fact that:

$$\mathbf{g}_{k+1} = \mathbf{A} \mathbf{x}_{k+1} - \mathbf{b} = \mathbf{A}(\mathbf{x}_k + \alpha_k \mathbf{v}_k) - \mathbf{b} = \underbrace{\mathbf{A} \mathbf{x}_k - \mathbf{b}}_{= \mathbf{g}_k} + \mathbf{A} \alpha_k \mathbf{v}_k,$$

If $\alpha_k = 0$, then $\mathbf{g}_{k+1} = \mathbf{g}_k$, and since $\mathbf{g}_{k+1}^\top \mathbf{g}_k = 0$, we conclude that $\mathbf{g}_{k+1} = \nabla f(\mathbf{x}_{k+1}) = \mathbf{0}$, so **we are at the minimum**.

If $\alpha_k \neq 0$, we have:

$$\mathbf{A} \mathbf{v}_k = \frac{1}{\alpha_k} (\mathbf{g}_{k+1} - \mathbf{g}_k)$$

So we get for $j \leq k - 1$:

$$\mathbf{g}_{k+1}^\top \mathbf{A} \mathbf{v}_j = \frac{1}{\alpha_k} \mathbf{g}_{k+1}^\top (\mathbf{g}_{j+1} - \mathbf{g}_j) = \frac{1}{\alpha_k} \left(\underbrace{\mathbf{g}_{k+1}^\top \mathbf{g}_{j+1}}_{=0} - \underbrace{\mathbf{g}_{k+1}^\top \mathbf{g}_j}_{=0} \right) = 0$$

Choosing descent directions

Conclusion: In the conjugate gradient method:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{v}_k, \quad \alpha_k = -\frac{\mathbf{v}_k^\top \mathbf{g}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k}$$

the descent directions $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ are obtained as:

$$\mathbf{v}_1 = -\mathbf{g}_1$$

$$\mathbf{v}_{k+1} = -\mathbf{g}_{k+1} + \beta_k \mathbf{v}_k, \quad \beta_k = \frac{\mathbf{g}_{k+1}^\top \mathbf{A} \mathbf{v}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k},$$

Choosing descent directions

Conclusion: In the conjugate gradient method:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{v}_k, \quad \alpha_k = -\frac{\mathbf{v}_k^\top \mathbf{g}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k}$$

the descent directions $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ are obtained as:

$$\begin{aligned}\mathbf{v}_1 &= -\mathbf{g}_1 \\ \mathbf{v}_{k+1} &= -\mathbf{g}_{k+1} + \beta_k \mathbf{v}_k, \quad \beta_k = \frac{\mathbf{g}_{k+1}^\top \mathbf{A} \mathbf{v}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k},\end{aligned}$$

Remark: A different expression for the coefficients α_k is often used:

$$\alpha_k = \frac{-\mathbf{v}_k^\top \mathbf{g}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k}$$

Choosing descent directions

Conclusion: In the conjugate gradient method:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{v}_k, \quad \alpha_k = -\frac{\mathbf{v}_k^\top \mathbf{g}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k}$$

the descent directions $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ According to $\mathbf{v}_k = -\mathbf{g}_k + \beta_{k-1} \mathbf{v}_{k-1}$

$$\mathbf{v}_1 = -\mathbf{g}_1$$

$$\mathbf{v}_{k+1} = -\mathbf{g}_{k+1} + \beta_k \mathbf{v}_k, \quad \beta_k = \frac{\mathbf{g}_{k+1}^\top \mathbf{A} \mathbf{v}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k},$$

Remark: A different expression for the coefficients α_k is often used:

$$\alpha_k = \frac{(\mathbf{g}_k - \beta_{k-1} \mathbf{v}_{k-1})^\top \mathbf{g}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k}$$

Choosing descent directions

Conclusion: In the conjugate gradient method:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{v}_k, \quad \alpha_k = -\frac{\mathbf{v}_k^\top \mathbf{g}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k}$$

the descent direction is gradient orthogonal to past directions:

$$\mathbf{v}_j^\top \mathbf{g}_k = 0 \text{ dla } j < k$$

$$\mathbf{v}_{k+1} = -\mathbf{g}_{k+1} + \beta_k \mathbf{v}_k, \quad \beta_k = \frac{\mathbf{g}_{k+1}^\top \mathbf{A} \mathbf{v}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k},$$

Remark: A different expression for the coefficients α_k is often used:

$$\alpha_k = \frac{\mathbf{g}_k^\top \mathbf{g}_k - \beta_{k-1} \mathbf{v}_{k-1}^\top \mathbf{g}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k}$$

Choosing descent directions

Conclusion: In the conjugate gradient method:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{v}_k, \quad \alpha_k = -\frac{\mathbf{v}_k^\top \mathbf{g}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k}$$

the descent directions $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ are obtained as:

$$\begin{aligned}\mathbf{v}_1 &= -\mathbf{g}_1 \\ \mathbf{v}_{k+1} &= -\mathbf{g}_{k+1} + \beta_k \mathbf{v}_k, \quad \beta_k = \frac{\mathbf{g}_{k+1}^\top \mathbf{A} \mathbf{v}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k},\end{aligned}$$

Remark: A different expression for the coefficients α_k is often used:

$$\alpha_k = \frac{\mathbf{g}_k^\top \mathbf{g}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k}$$

Choosing descent directions

Conclusion: In the conjugate gradient method:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{v}_k, \quad \alpha_k = -\frac{\mathbf{v}_k^\top \mathbf{g}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k}$$

the descent directions $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ are obtained as:

$$\begin{aligned}\mathbf{v}_1 &= -\mathbf{g}_1 \\ \mathbf{v}_{k+1} &= -\mathbf{g}_{k+1} + \beta_k \mathbf{v}_k, \quad \beta_k = \frac{\mathbf{g}_{k+1}^\top \mathbf{A} \mathbf{v}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k},\end{aligned}$$

Remark: A different expression for the coefficients α_k is often used:

$$\alpha_k = \frac{\mathbf{g}_k^\top \mathbf{g}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k}$$

Likewise, for coefficient β_k :

$$\beta_k = \frac{\mathbf{g}_{k+1}^\top \mathbf{A} \mathbf{v}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k}$$

Choosing descent directions

Conclusion: In the conjugate gradient method:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{v}_k, \quad \alpha_k = -\frac{\mathbf{v}_k^\top \mathbf{g}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k}$$

the descent directions $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ are obtained as:

$$\begin{aligned}\mathbf{v}_1 &= -\mathbf{g}_1 \\ \mathbf{v}_{k+1} &= -\mathbf{g}_{k+1} + \beta_k \mathbf{v}_k, \quad \beta_k = \frac{\mathbf{g}_{k+1}^\top \mathbf{A} \mathbf{v}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k},\end{aligned}$$

Remark: A different expression for the coefficients α_k is often used:

$$\alpha_k = \frac{\mathbf{g}_k^\top \mathbf{g}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k}$$

Likewise, for coefficient β_k :

$$\beta_k = \frac{\mathbf{g}_{k+1}^\top (\alpha_k \mathbf{A} \mathbf{v}_k)}{\mathbf{v}_k^\top (\alpha_k \mathbf{A} \mathbf{v}_k)}$$

Choosing descent directions

Conclusion: In the conjugate gradient method:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{v}_k, \quad \alpha_k = -\frac{\mathbf{v}_k^\top \mathbf{g}_k}{\mathbf{v}_k^\top A \mathbf{v}_k}$$

the descent directions $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ are obtained as:

In the proof on the previous slide we showed that:

$$\alpha_k A \mathbf{v}_k = \mathbf{g}_{k+1} - \mathbf{g}_k$$

$$\mathbf{v}_k^\top A \mathbf{v}_k$$

Remark: A different expression for the coefficients α_k is often used:

$$\alpha_k = \frac{\mathbf{g}_k^\top \mathbf{g}_k}{\mathbf{v}_k^\top A \mathbf{v}_k}$$

Likewise, for coefficient β_k :

$$\beta_k = \frac{\mathbf{g}_{k+1}^\top (\alpha_k A \mathbf{v}_k)}{\mathbf{v}_k^\top (\alpha_k A \mathbf{v}_k)}$$

Choosing descent directions

Conclusion: In the conjugate gradient method:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{v}_k, \quad \alpha_k = -\frac{\mathbf{v}_k^\top \mathbf{g}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k}$$

the descent directions $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ are obtained as:

$$\begin{aligned}\mathbf{v}_1 &= -\mathbf{g}_1 \\ \mathbf{v}_{k+1} &= -\mathbf{g}_{k+1} + \beta_k \mathbf{v}_k, \quad \beta_k = \frac{\mathbf{g}_{k+1}^\top \mathbf{A} \mathbf{v}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k},\end{aligned}$$

Remark: A different expression for the coefficients α_k is often used:

$$\alpha_k = \frac{\mathbf{g}_k^\top \mathbf{g}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k}$$

Likewise, for coefficient β_k :

$$\beta_k = \frac{\mathbf{g}_{k+1}^\top (\mathbf{g}_{k+1} - \mathbf{g}_k)}{\mathbf{v}_k^\top (\mathbf{g}_{k+1} - \mathbf{g}_k)}$$

Choosing descent directions

Conclusion: In the conjugate gradient method:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{v}_k, \quad \alpha_k = -\frac{\mathbf{v}_k^\top \mathbf{g}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k}$$

the descent directions $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ are obtained as:

gradient orthogonal to previous gradients:

$$\mathbf{g}_{k+1}^\top \mathbf{g}_j = 0 \text{ for } j \leq k, \quad \mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k$$

Remark: A different expression for the coefficients α_k is often used:

$$\alpha_k = \frac{\mathbf{g}_k^\top \mathbf{g}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k}$$

Likewise, for coefficient β_k :

$$\beta_k = \frac{\mathbf{g}_{k+1}^\top \mathbf{g}_{k+1} - \mathbf{g}_{k+1}^\top \mathbf{g}_k}{\mathbf{v}_k^\top \mathbf{g}_{k+1} - \mathbf{v}_k^\top \mathbf{g}_k}$$

Choosing descent directions

Conclusion: In the conjugate gradient method:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{v}_k, \quad \alpha_k = -\frac{\mathbf{v}_k^\top \mathbf{g}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k}$$

the descent directions $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ are obtained as:

$$\mathbf{v}_1 = -\mathbf{g}_1$$

\mathbf{v}_j gradient orthogonal to previous directions: $\frac{\mathbf{g}_j^\top \mathbf{A} \mathbf{v}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k}$,

$$\mathbf{v}_j^\top \mathbf{g}_{k+1} = 0 \text{ for } j \leq k$$

Remark: A different choice of \mathbf{v}_k is often used:

$$\alpha_k = \frac{\mathbf{g}_k^\top \mathbf{g}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k}$$

Likewise, for coefficient β_k :

$$\beta_k = \frac{\mathbf{g}_{k+1}^\top \mathbf{g}_{k+1}}{\mathbf{v}_k^\top \mathbf{g}_{k+1} - \mathbf{v}_k^\top \mathbf{g}_k}$$

Choosing descent directions

Conclusion: In the conjugate gradient method:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{v}_k, \quad \alpha_k = -\frac{\mathbf{v}_k^\top \mathbf{g}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k}$$

the descent directions $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ are obtained as:

$$\mathbf{v}_1 = -\mathbf{g}_1$$

We have shown when re-expressing α_k

$$\mathbf{v}_{k+1} = -\mathbf{g}_{k+1} + \beta$$

that $-\mathbf{v}_k^\top \mathbf{g}_k = \mathbf{g}_k^\top \mathbf{g}_k$

$$\gamma_k \mathbf{A} \mathbf{v}_k$$

Remark: A different expression for the coefficients α_k is often used:

$$\alpha_k = \frac{\mathbf{g}_k^\top \mathbf{g}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k}$$

Likewise, for coefficient β_k :

$$\beta_k = \frac{\mathbf{g}_{k+1}^\top \mathbf{g}_{k+1}}{-\mathbf{v}_k^\top \mathbf{g}_k}$$

Choosing descent directions

Conclusion: In the conjugate gradient method:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{v}_k, \quad \alpha_k = -\frac{\mathbf{v}_k^\top \mathbf{g}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k}$$

the descent directions $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ are obtained as:

$$\begin{aligned}\mathbf{v}_1 &= -\mathbf{g}_1 \\ \mathbf{v}_{k+1} &= -\mathbf{g}_{k+1} + \beta_k \mathbf{v}_k, \quad \beta_k = \frac{\mathbf{g}_{k+1}^\top \mathbf{A} \mathbf{v}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k},\end{aligned}$$

Remark: A different expression for the coefficients α_k is often used:

$$\alpha_k = \frac{\mathbf{g}_k^\top \mathbf{g}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k}$$

Likewise, for coefficient β_k :

$$\beta_k = \frac{\mathbf{g}_{k+1}^\top \mathbf{g}_{k+1}}{\mathbf{g}_k^\top \mathbf{g}_k}$$

Choosing descent directions

Conclusion: In the conjugate gradient method:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{v}_k, \quad \alpha_k = -\frac{\mathbf{v}_k^\top \mathbf{g}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k}$$

the descent directions $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$ are obtained as:

$$\begin{aligned}\mathbf{v}_1 &= -\mathbf{g}_1 \\ \mathbf{v}_{k+1} &= -\mathbf{g}_{k+1} + \beta_k \mathbf{v}_k, \quad \beta_k = \frac{\mathbf{g}_{k+1}^\top \mathbf{A} \mathbf{v}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k},\end{aligned}$$

Remark: A different expression for the coefficients α_k is often used:

$$\alpha_k = \frac{\mathbf{g}_k^\top \mathbf{g}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k}$$

Likewise, for coefficient β_k :

$$\beta_k = \frac{\|\mathbf{g}_{k+1}\|^2}{\|\mathbf{g}_k\|^2}$$

Conjugate directions are descent directions

Using the definition of the conjugate direction:

$$\mathbf{v}_{k+1} = -\mathbf{g}_{k+1} + \beta_k \mathbf{v}_k$$

we have:

$$\mathbf{g}_{k+1}^\top \mathbf{v}_{k+1} = -\mathbf{g}_{k+1}^\top \mathbf{g}_{k+1} + \beta_k \mathbf{g}_{k+1}^\top \mathbf{v}_k$$

Conjugate directions are descent directions

Using the definition of the conjugate direction:

$$\mathbf{v}_{k+1} = -\mathbf{g}_{k+1} + \beta_k \mathbf{v}_k$$

we have:

$$\mathbf{g}_{k+1}^\top \mathbf{v}_{k+1} = -\mathbf{g}_{k+1}^\top \mathbf{g}_{k+1} + \beta_k \mathbf{g}_{k+1}^\top \mathbf{v}_k$$

gradient orthogonal to past directions:

$$\mathbf{g}_{k+1}^\top \mathbf{v}_j = 0 \text{ for } j \leq k$$

Conjugate directions are descent directions

Using the definition of the conjugate direction:

$$\mathbf{v}_{k+1} = -\mathbf{g}_{k+1} + \beta_k \mathbf{v}_k$$

we have:

$$\mathbf{g}_{k+1}^\top \mathbf{v}_{k+1} = -\mathbf{g}_{k+1}^\top \mathbf{g}_{k+1}$$

Conjugate directions are descent directions

Using the definition of the conjugate direction:

$$\mathbf{v}_{k+1} = -\mathbf{g}_{k+1} + \beta_k \mathbf{v}_k$$

we have:

$$\mathbf{g}_{k+1}^\top \mathbf{v}_{k+1} = -\|\mathbf{g}_{k+1}\|^2 < 0,$$

so \mathbf{v}_{k+1} is a **descent direction**

Conjugate gradient method for a quadratic function

Choose a starting point \mathbf{x}_1

Compute gradient $\mathbf{g}_1 = \nabla f(\mathbf{x}_1)$ and a starting point $\mathbf{v}_1 = -\mathbf{g}_1$

For $k = 1, 2, \dots$:

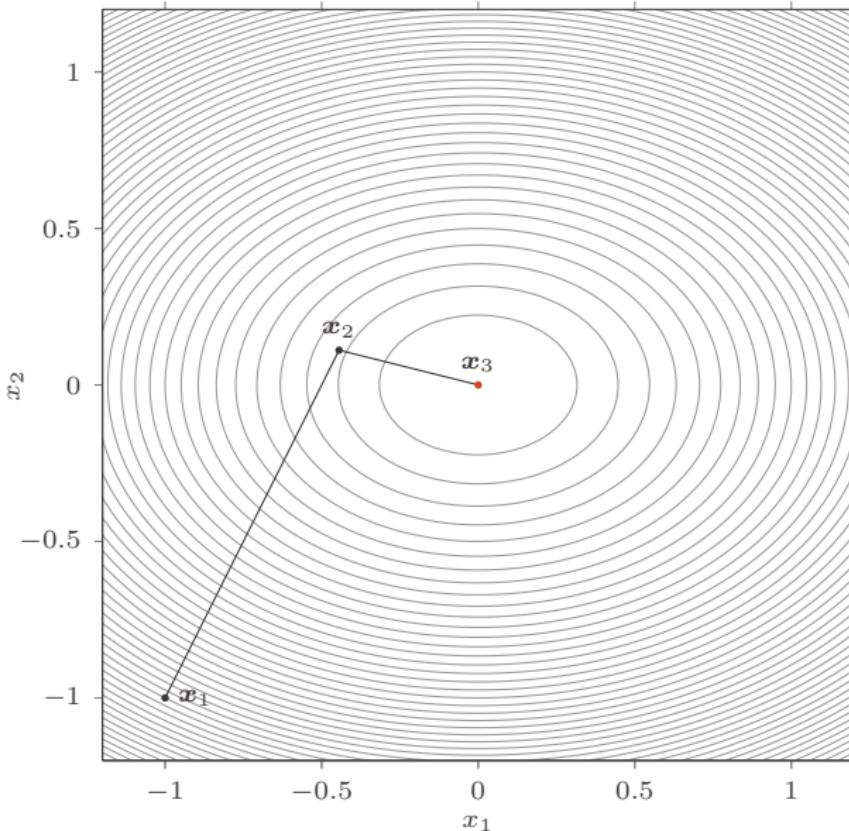
1. Compute the optimal step size $\alpha_k = \frac{\mathbf{g}_k^\top \mathbf{g}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k}$
2. Update the solution: $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{v}_k$
3. Compute the gradient: $\mathbf{g}_{k+1} = \nabla f(\mathbf{x}_{k+1})$
4. If $\mathbf{g}_{k+1} = \mathbf{0}$, stop the algorithm, otherwise choose a new descent direction:

$$\mathbf{v}_{k+1} = -\mathbf{g}_{k+1} + \beta_k \mathbf{v}_k, \quad \beta_k = \frac{\|\mathbf{g}_{k+1}\|^2}{\|\mathbf{g}_k\|^2},$$

The algorithm will stop after at most n steps, returning the optimal solution (minimizer) \mathbf{x}^*

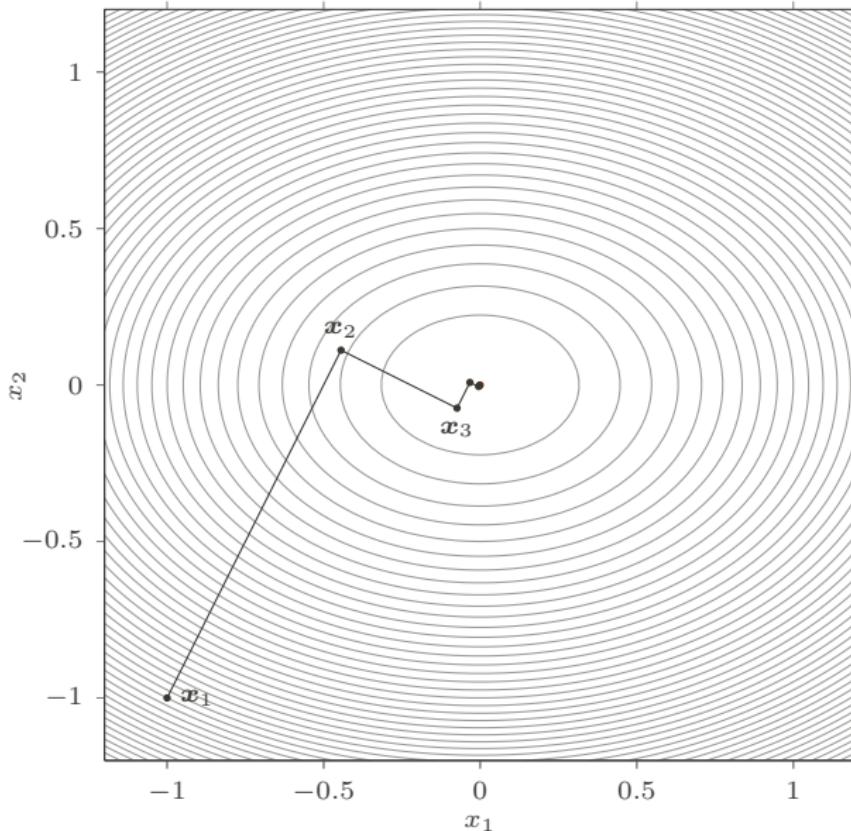
Conjugate gradient method – example

$$f(x_1, x_2) = x_1^2 + 2x_2^2$$



Compare with the steepest descent method

$$f(x_1, x_2) = x_1^2 + 2x_2^2$$



Conjugate gradient method for general function

The method can be easily extended to minimize arbitrary functions, not necessarily quadratic; several changes need to be introduced:

Conjugate gradient method for general function

The method can be easily extended to minimize arbitrary functions, not necessarily quadratic; several changes need to be introduced:

- The optimal step size for a quadratic function $\alpha_k = \frac{\mathbf{g}_k^\top \mathbf{g}_k}{\mathbf{v}_k^\top \mathbf{A} \mathbf{v}_k}$ was obtained from:

$$\alpha_k = \operatorname{argmin}_{\alpha} f(\mathbf{x}_k + \alpha \mathbf{v}_k)$$

For non-quadratic function one needs to numerically solve this univariate optimization problem

Conjugate gradient method for general function

The method can be easily extended to minimize arbitrary functions, not necessarily quadratic; several changes need to be introduced:

- The optimal step size for a quadratic function $\alpha_k = \frac{\mathbf{g}_k^\top \mathbf{g}_k}{\mathbf{v}_k^\top A \mathbf{v}_k}$ was obtained from:

$$\alpha_k = \operatorname{argmin}_{\alpha} f(\mathbf{x}_k + \alpha \mathbf{v}_k)$$

For non-quadratic function one needs to numerically solve this univariate optimization problem

- Since we do not have guarantees that subsequent gradients \mathbf{g}_k are orthogonal, one can replace:

$$\beta_k = \frac{\|\mathbf{g}_{k+1}\|^2}{\|\mathbf{g}_k\|^2} = \frac{\mathbf{g}_{k+1}^\top \mathbf{g}_{k+1}}{\|\mathbf{g}_k\|^2} \quad \Rightarrow \quad \beta_k = \frac{\mathbf{g}_{k+1}^\top (\mathbf{g}_{k+1} - \mathbf{g}_k)}{\|\mathbf{g}_k\|^2}$$

Depending on which expression is used, there are two versions of the algorithm: **Fletcher-Reeves** and **Polak-Ribière**.

Conjugate gradient method for general function

The method can be easily extended to minimize arbitrary functions, not necessarily quadratic; several changes need to be introduced:

- The optimal step size for a quadratic function $\alpha_k = \frac{\mathbf{g}_k^\top \mathbf{g}_k}{\mathbf{v}_k^\top A \mathbf{v}_k}$ was obtained from:

$$\alpha_k = \operatorname{argmin}_{\alpha} f(\mathbf{x}_k + \alpha \mathbf{v}_k)$$

For non-quadratic function one needs to numerically solve this univariate optimization problem

- Since we do not have guarantees that subsequent gradients \mathbf{g}_k are orthogonal, one can replace:

$$\beta_k = \frac{\|\mathbf{g}_{k+1}\|^2}{\|\mathbf{g}_k\|^2} = \frac{\mathbf{g}_{k+1}^\top \mathbf{g}_{k+1}}{\|\mathbf{g}_k\|^2} \quad \Rightarrow \quad \beta_k = \frac{\mathbf{g}_{k+1}^\top (\mathbf{g}_{k+1} - \mathbf{g}_k)}{\|\mathbf{g}_k\|^2}$$

Depending on which expression is used, there are two versions of the algorithm: **Fletcher-Reeves** and **Polak-Ribière**.

- No guarantee that the algorithm stops within n iterations, so we choose condition $\|\nabla f(\mathbf{x}_k)\| < \epsilon$ as a stopping rule (with small ϵ).

Conjugate gradient method for general function (Fletcher-Reeves and Polak-Ribi  re)

Choose a starting point \mathbf{x}_1

Compute gradient $\mathbf{g}_1 = \nabla f(\mathbf{x}_1)$ and initial direction $\mathbf{v}_1 = -\mathbf{g}_1$

For $k = 1, 2, \dots$:

1. Compute the optimal step size $\alpha_k = \operatorname{argmin}_{\alpha} f(\mathbf{x}_k + \alpha \mathbf{v}_k)$
2. Update the solution $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{v}_k$
3. Compute the gradient $\mathbf{g}_{k+1} = \nabla f(\mathbf{x}_{k+1})$
4. If $\|\mathbf{g}_{k+1}\| < \epsilon$, stop the algorithm, otherwise choose a new descent direction:

$$\mathbf{v}_{k+1} = -\mathbf{g}_{k+1} + \beta_k \mathbf{v}_k, \quad \text{where:}$$

$$(\text{F-R}) \quad \beta_k = \frac{\|\mathbf{g}_{k+1}\|^2}{\|\mathbf{g}_k\|^2} \quad \text{or} \quad (\text{P-R}) \quad \beta_k = \frac{\mathbf{g}_{k+1}^\top (\mathbf{g}_{k+1} - \mathbf{g}_k)}{\|\mathbf{g}_k\|^2}$$

The Polak-Ribi  re method preferred in practice

The conjugate directions are descent directions

Since:

$$\alpha_k = \operatorname{argmin}_{\alpha} f(\mathbf{x}_k + \alpha \mathbf{v}_k),$$

we have:

$$\frac{df(\mathbf{x}_k + \alpha \mathbf{v}_k)}{d\alpha} \Big|_{\alpha=\alpha_k} = 0$$

The conjugate directions are descent directions

Since:

$$\alpha_k = \operatorname{argmin}_{\alpha} f(\mathbf{x}_k + \alpha \mathbf{v}_k),$$

we have:

$$\frac{df(\mathbf{x}_k + \alpha \mathbf{v}_k)}{d\alpha} \Big|_{\alpha=\alpha_k} = 0$$

From the chain rule:

$$\frac{df(\mathbf{x}_k + \alpha \mathbf{v}_k)}{d\alpha} = \nabla f(\mathbf{x}_k + \alpha \mathbf{v}_k)^\top \mathbf{v}_k,$$

so:

$$\nabla f(\mathbf{x}_k + \alpha_k \mathbf{v}_k)^\top \mathbf{v}_k = 0$$

The conjugate directions are descent directions

Since:

$$\alpha_k = \underset{\alpha}{\operatorname{argmin}} f(\mathbf{x}_k + \alpha \mathbf{v}_k),$$

we have:

$$\frac{df(\mathbf{x}_k + \alpha \mathbf{v}_k)}{d\alpha} \Big|_{\alpha=\alpha_k} = 0$$

From the chain rule:

$$\frac{df(\mathbf{x}_k + \alpha \mathbf{v}_k)}{d\alpha} = \nabla f(\mathbf{x}_k + \alpha \mathbf{v}_k)^\top \mathbf{v}_k,$$

so:

$$\nabla f(\mathbf{x}_{k+1})^\top \mathbf{v}_k = 0$$

The conjugate directions are descent directions

Since:

$$\alpha_k = \underset{\alpha}{\operatorname{argmin}} f(\mathbf{x}_k + \alpha \mathbf{v}_k),$$

we have:

$$\frac{df(\mathbf{x}_k + \alpha \mathbf{v}_k)}{d\alpha} \Big|_{\alpha=\alpha_k} = 0$$

From the chain rule:

$$\frac{df(\mathbf{x}_k + \alpha \mathbf{v}_k)}{d\alpha} = \nabla f(\mathbf{x}_k + \alpha \mathbf{v}_k)^\top \mathbf{v}_k,$$

so:

$$\mathbf{g}_{k+1}^\top \mathbf{v}_k = 0$$

The conjugate directions are descent directions

Since:

$$\alpha_k = \underset{\alpha}{\operatorname{argmin}} f(\mathbf{x}_k + \alpha \mathbf{v}_k),$$

we have:

$$\frac{df(\mathbf{x}_k + \alpha \mathbf{v}_k)}{d\alpha} \Big|_{\alpha=\alpha_k} = 0$$

From the chain rule:

$$\frac{df(\mathbf{x}_k + \alpha \mathbf{v}_k)}{d\alpha} = \nabla f(\mathbf{x}_k + \alpha \mathbf{v}_k)^\top \mathbf{v}_k,$$

so:

$$\mathbf{g}_{k+1}^\top \mathbf{v}_k = 0$$

$$\mathbf{v}_{k+1} = -\mathbf{g}_{k+1} + \beta_k \mathbf{v}_k$$

The conjugate directions are descent directions

Since:

$$\alpha_k = \underset{\alpha}{\operatorname{argmin}} f(\mathbf{x}_k + \alpha \mathbf{v}_k),$$

we have:

$$\frac{df(\mathbf{x}_k + \alpha \mathbf{v}_k)}{d\alpha} \Big|_{\alpha=\alpha_k} = 0$$

From the chain rule:

$$\frac{df(\mathbf{x}_k + \alpha \mathbf{v}_k)}{d\alpha} = \nabla f(\mathbf{x}_k + \alpha \mathbf{v}_k)^\top \mathbf{v}_k,$$

so:

$$\mathbf{g}_{k+1}^\top \mathbf{v}_k = 0$$

$$\mathbf{g}_{k+1}^\top \mathbf{v}_{k+1} = -\mathbf{g}_{k+1}^\top \mathbf{g}_{k+1} + \beta_k \underbrace{\mathbf{g}_{k+1}^\top \mathbf{v}_k}_{=0}$$

The conjugate directions are descent directions

Since:

$$\alpha_k = \operatorname{argmin}_{\alpha} f(\mathbf{x}_k + \alpha \mathbf{v}_k),$$

we have:

$$\frac{df(\mathbf{x}_k + \alpha \mathbf{v}_k)}{d\alpha} \Big|_{\alpha=\alpha_k} = 0$$

From the chain rule:

$$\frac{df(\mathbf{x}_k + \alpha \mathbf{v}_k)}{d\alpha} = \nabla f(\mathbf{x}_k + \alpha \mathbf{v}_k)^\top \mathbf{v}_k,$$

so:

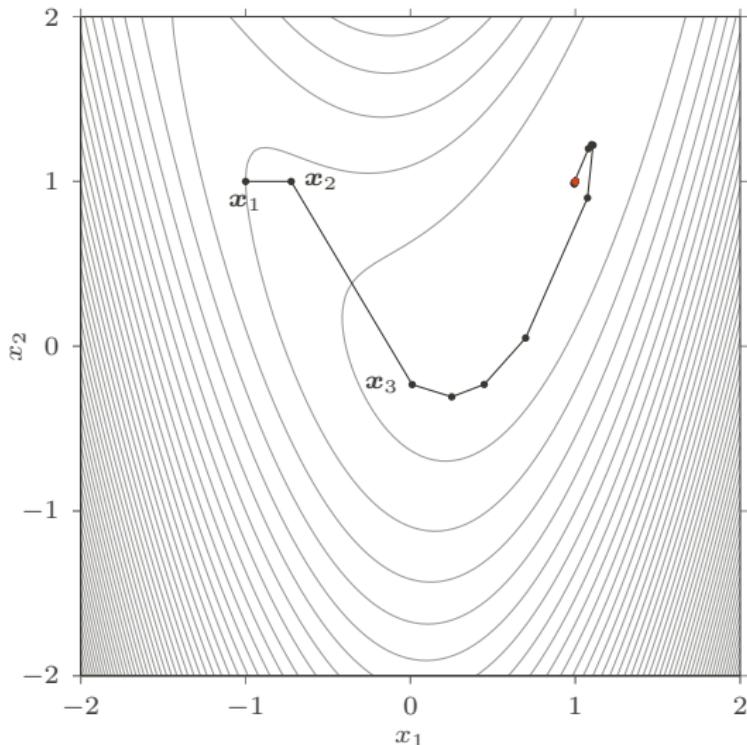
$$\mathbf{g}_{k+1}^\top \mathbf{v}_k = 0$$

$$\mathbf{g}_{k+1}^\top \mathbf{v}_{k+1} = -\|\mathbf{g}_{k+1}\|^2 < 0,$$

thus \mathbf{v}_{k+1} is a **descent direction**

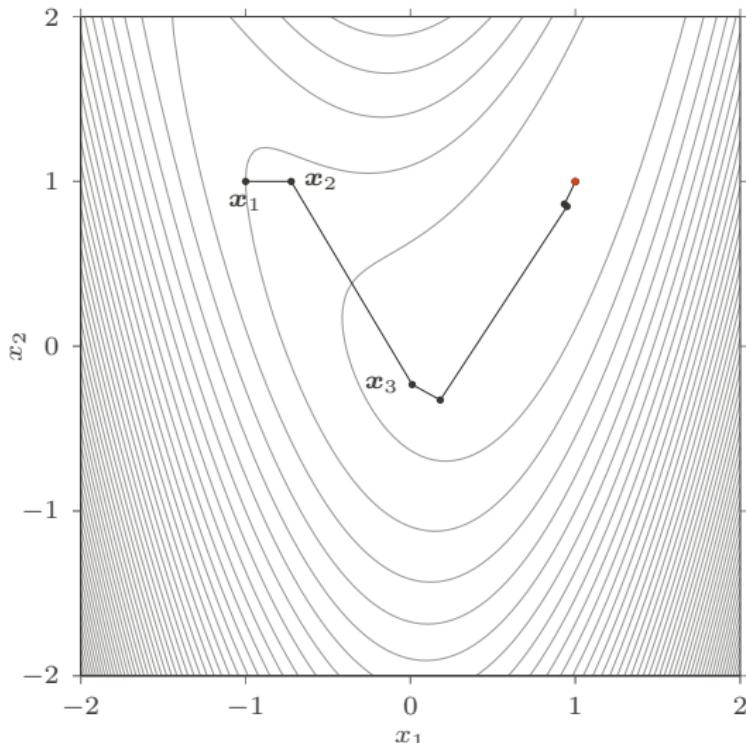
Fletcher-Reeves method – example

$$f(x_1, x_2) = 2.5(x_1^2 - x_2)^2 + (1 - x_1)^2$$



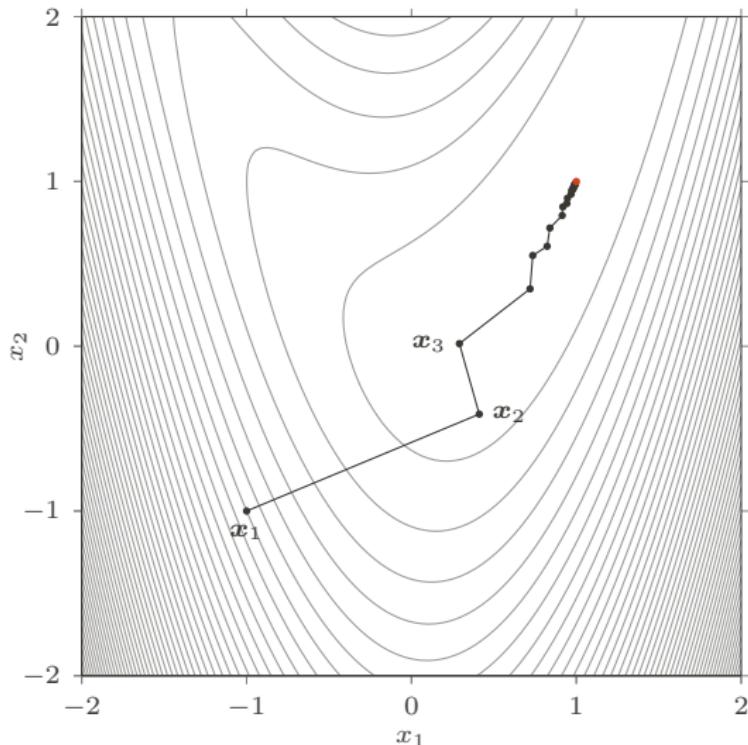
Polak-Ribière method – example

$$f(x_1, x_2) = 2.5(x_1^2 - x_2)^2 + (1 - x_1)^2$$



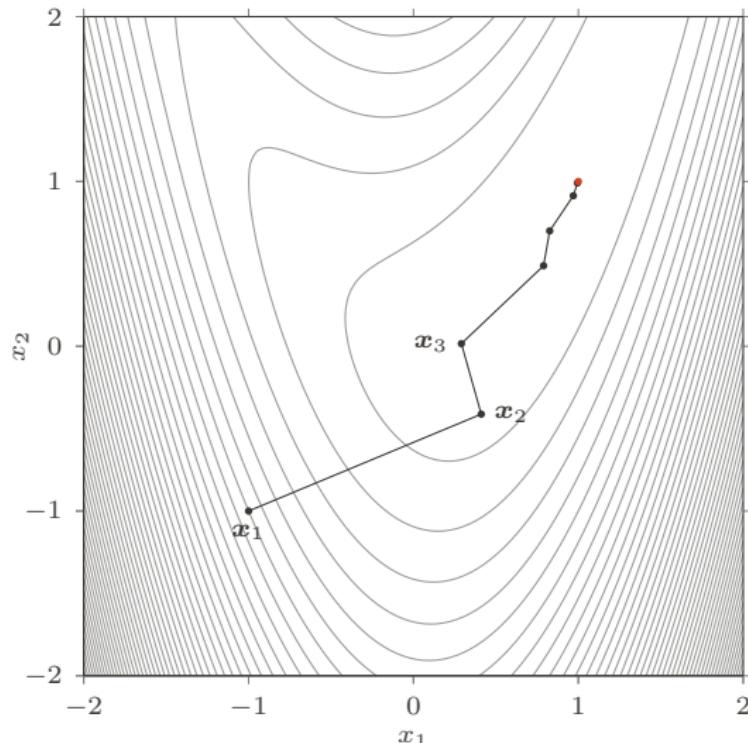
Fletcher-Reeves method – example

$$f(x_1, x_2) = 2.5(x_1^2 - x_2)^2 + (1 - x_1)^2$$



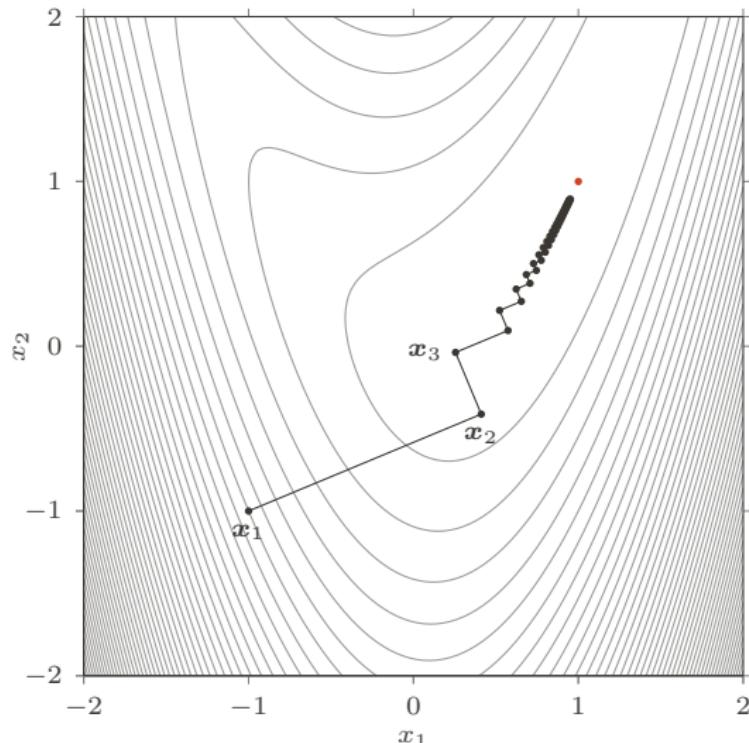
Polak-Ribière method – example

$$f(x_1, x_2) = 2.5(x_1^2 - x_2)^2 + (1 - x_1)^2$$



Compare with the steepest descent

$$f(x_1, x_2) = 2.5(x_1^2 - x_2)^2 + (1 - x_1)^2$$



Optimization Methods for Data Analysis

5. Stochastic gradient descent method

Wojciech Kotłowski

Institute of Computing Science, PUT

<http://www.cs.put.poznan.pl/wkotlowski/>

27.04.2022

Introduction

Unconstrained minimization of **differentiable** function $f(\mathbf{w})$

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

Remark: at this lecture we have changed the notation and use \mathbf{w} to denote the vector of optimization variables rather than x .

In many applications of optimization, the objective has a form of a sum of N elements:

$$f(\mathbf{w}) = \sum_{i=1}^N f_i(\mathbf{w}),$$

This assumption will be used throughout the whole lecture

Example: linear regression

Prediction/explanation of an output ("response") variable (Y) based on the set of input ("explanatory") variables (X_1, \dots, X_n).

Examples

- X – stock prices during the last week, Y – stock price tomorrow.
- X – medical test results, Y – disease advancement level.
- X – size of the computer program, Y – time of the program implementation
- X – road conditions, timestamp, location, Y – average car speed
- X – house features, Y – house price.

Example: linear regression

We model variable Y as a linear function of X_1, \dots, X_n :

$$Y = w_0 + w_1X_1 + \dots + w_nX_n = \mathbf{X}^\top \mathbf{w}$$

where $\mathbf{w} = (w_0, w_1, \dots, w_n)$ is a **weight vector** (a vector of optimization variables), while $\mathbf{X} = (1, X_1, \dots, X_n)$.

Example: linear regression

We model variable Y as a linear function of X_1, \dots, X_n :

$$Y = w_0 + w_1 X_1 + \dots + w_n X_n = \mathbf{X}^\top \mathbf{w}$$

where $\mathbf{w} = (w_0, w_1, \dots, w_n)$ is a **weight vector** (a vector of optimization variables), while $\mathbf{X} = (1, X_1, \dots, X_n)$.

The particular values of weights are selected using the **available data set**

Example: linear regression

We have collected a set of historical data, on which we know the values at the output y :

$(x_{11}, x_{12}, \dots, x_{1n}, y_1)$	(\mathbf{x}_1, y_1)
$(x_{21}, x_{22}, \dots, x_{2n}, y_2)$	(\mathbf{x}_2, y_2)
\dots	or in short
$(x_{N1}, x_{N2}, \dots, x_{Nn}, y_N)$	(\mathbf{x}_N, y_N)

Example: linear regression

We have collected a set of historical data, on which we know the values at the output y :

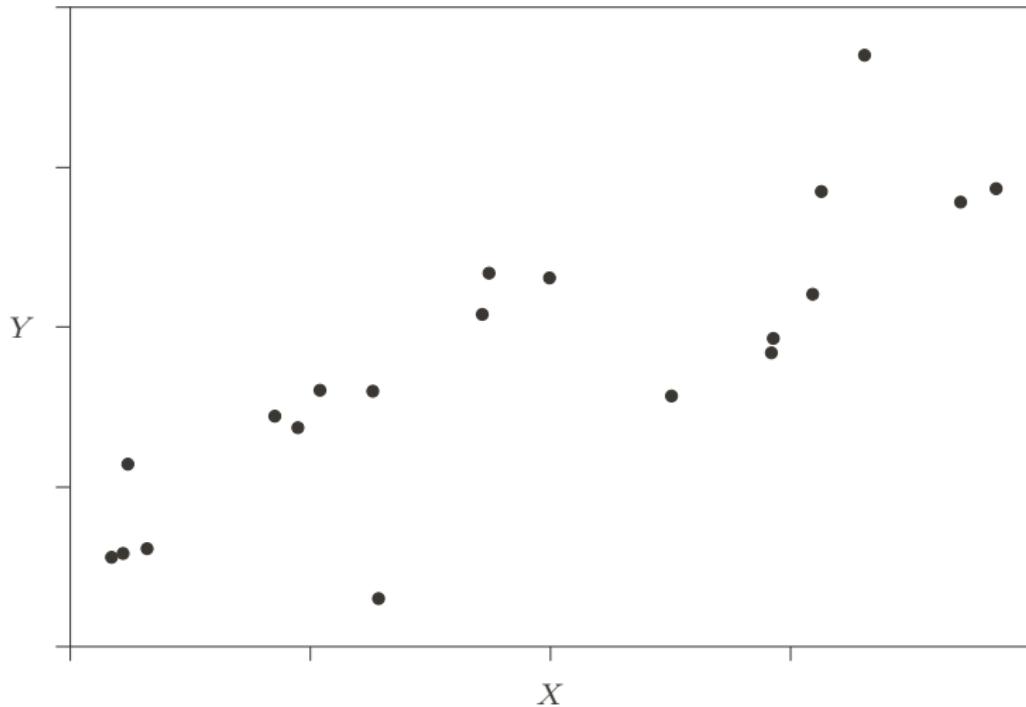
$(x_{11}, x_{12}, \dots, x_{1n}, y_1)$	(\mathbf{x}_1, y_1)
$(x_{21}, x_{22}, \dots, x_{2n}, y_2)$	(\mathbf{x}_2, y_2)
\dots	or in short
$(x_{N1}, x_{N2}, \dots, x_{Nn}, y_N)$	(\mathbf{x}_N, y_N)

We select weight vector \mathbf{w} such that the outputs of a linear function approximate the true outputs as good as possible, i.e. such that:

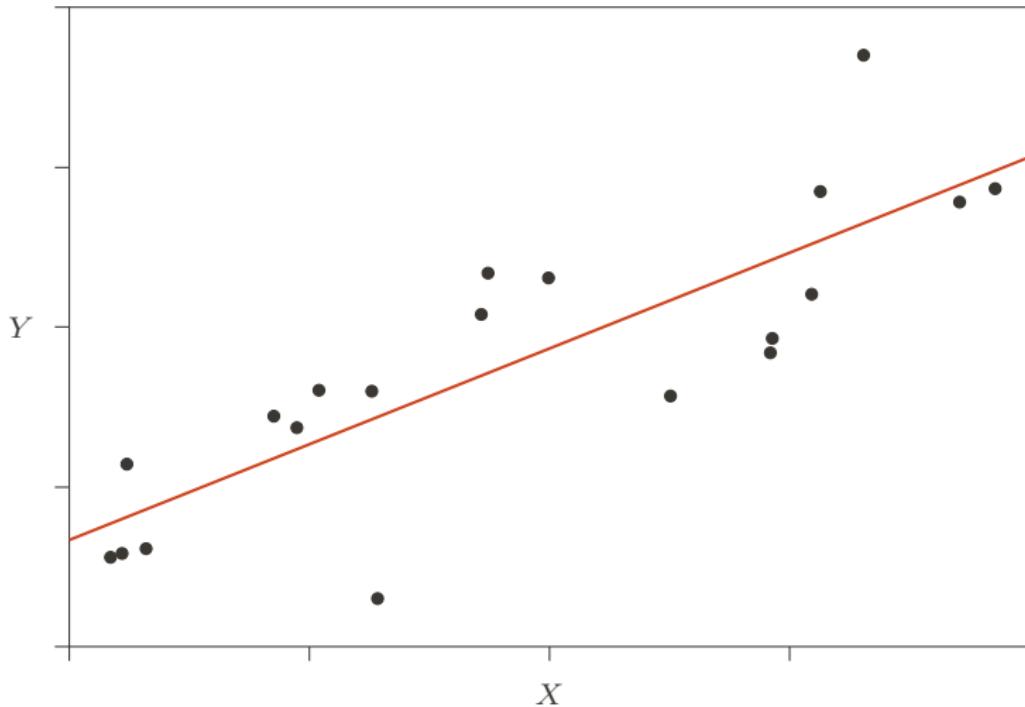
$$\hat{y}_i = \mathbf{x}_i^\top \mathbf{w}$$

is as close as possible to y_i for all $i = 1, \dots, N$

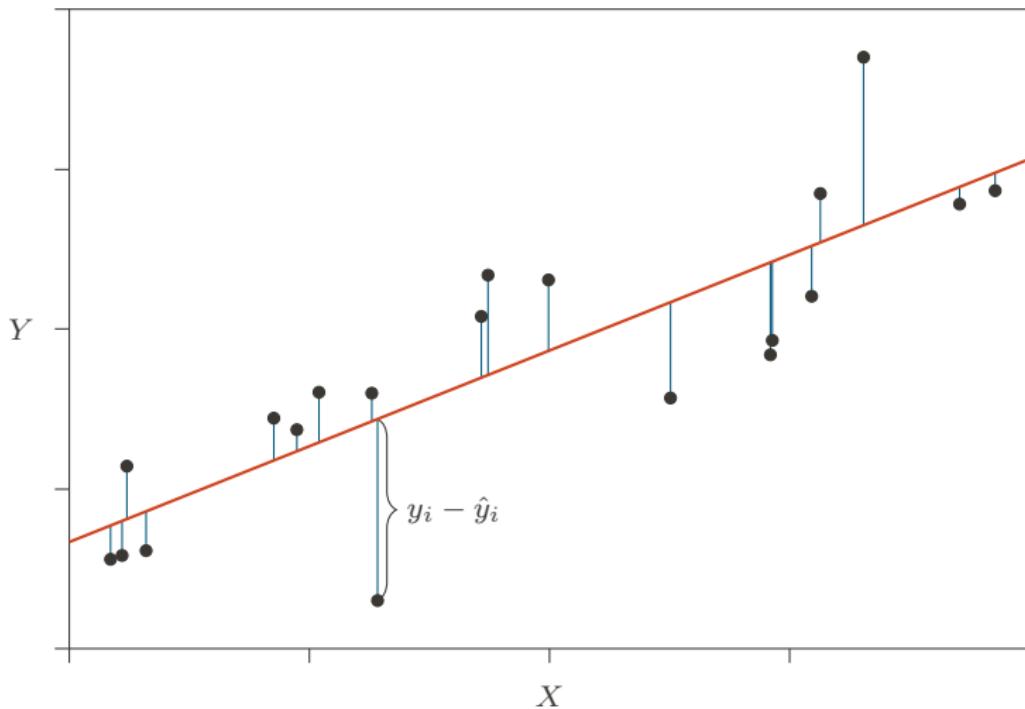
Example: linear regression



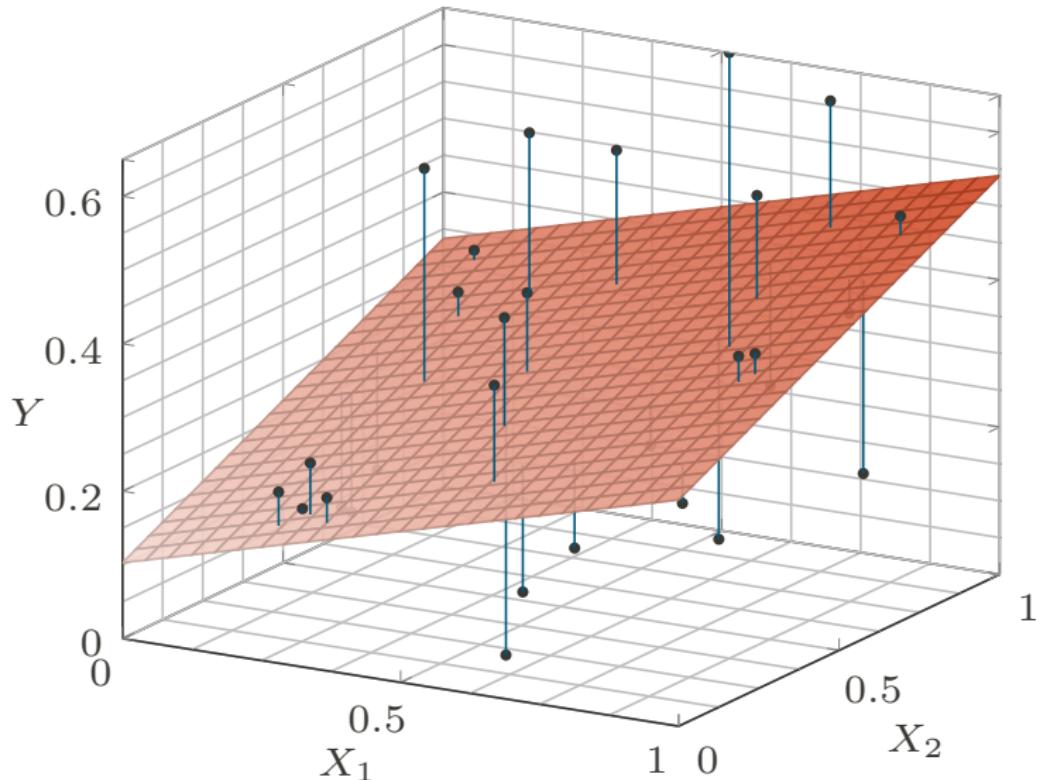
Example: linear regression



Example: linear regression



Example: linear regression



Example: linear regression

The method of Least Squares (LS):

Compute the regression coefficients (weights) w in order to minimize the sum of squares of the deviations of the model from the data:

$$\min_{w \in \mathbb{R}^n} f(w) = \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

Example: linear regression

The method of Least Squares (LS):

Compute the regression coefficients (weights) \mathbf{w} in order to minimize the sum of squares of the deviations of the model from the data:

$$\min_{\mathbf{w} \in \mathbb{R}^n} f(\mathbf{w}) = \sum_{i=1}^N (y_i - \mathbf{x}_i^\top \mathbf{w})^2$$

Example: linear regression

The method of Least Squares (LS):

Compute the regression coefficients (weights) \mathbf{w} in order to minimize the sum of squares of the deviations of the model from the data:

$$\min_{\mathbf{w} \in \mathbb{R}^n} f(\mathbf{w}) = \sum_{i=1}^N \underbrace{(y_i - \mathbf{x}_i^\top \mathbf{w})^2}_{f_i(\mathbf{w})}$$

Example: linear regression

The method of Least Squares (LS):

Compute the regression coefficients (weights) w in order to minimize the sum of squares of the deviations of the model from the data:

$$\min_{w \in \mathbb{R}^n} f(w) = \sum_{i=1}^N \underbrace{(y_i - \mathbf{x}_i^\top w)^2}_{f_i(w)}$$

In other words: we compute the weights by minimizing the **total error**, expressed in terms of the square function, **on the entire data set**

Examples: machine learning

$$f(\mathbf{w}) = \sum_{i=1}^N f_i(\mathbf{w})$$

The majority of methods in **machine learning** is essentially based on minimization of such form of a function.

Having a **data set**, obtain the **parameters of the model \mathbf{w}** by **minimizing the total error on the data**

- \mathbf{w} – model parameters (regression coefficients, weights in the neural network, etc.)
- $f_i(\mathbf{w})$ – error on a single data point from the data set (e.g. quadratic error in the linear regression)
- $f(\mathbf{w})$ – the total error on the data set

Classical approach

Solve:

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

using one of the classical optimization methods, e.g. the gradient descent method or the Newton-Raphson method

Classical approach

Solve:

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

using one of the classical optimization methods, e.g. the gradient descent method or the Newton-Raphson method

Typical problems in practice:

- Computing the gradient/Hessian requires summing N N gradient/Hessian computations for each of the component function in the sum:

$$\nabla f(w) = \sum_{i=1}^N \nabla f_i(w)$$

- The dimension of the problem n (number of parameters = dimension of the weight vector) can be very large, so that the Newton-Raphson method may be too costly to run

Classical approach: gradient descent

Starting from \mathbf{w}_1 , in subsequent iterations $k = 1, 2, \dots$:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha_k \underbrace{\sum_{i=1}^N \nabla f_i(\mathbf{w}_k)}_{\nabla f(\mathbf{w}_k)}$$

Performing a single step requires a pass over all components of the sum in the objective function.

Classical approach: gradient descent

Starting from w_1 , in subsequent iterations $k = 1, 2, \dots$:

$$w_{k+1} = w_k - \alpha_k \underbrace{\sum_{i=1}^N \nabla f_i(w_k)}_{\nabla f(w_k)}$$

Performing a single step requires a pass over all components of the sum in the objective function.

Idea: compute the gradient on a **randomly selected component** f_i instead of on the entire function f !

Stochastic gradient descent (SGD) method

Starting from \mathbf{w}_1 , in each iteration $k = 1, 2, \dots$:

Draw one of the coordinates $i_k \in \{1, \dots, N\}$

Do a step along the negative gradient of function $f_{i_k}(\mathbf{w}_k)$:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha_k \nabla f_{i_k}(\mathbf{w}_k)$$

Stochastic gradient descent (SGD) method

Starting from w_1 , in each iteration $k = 1, 2, \dots$:

Draw one of the coordinates $i_k \in \{1, \dots, N\}$

Do a step along the negative gradient of function $f_{i_k}(w_k)$:

$$w_{k+1} = w_k - \alpha_k \nabla f_{i_k}(w_k)$$

Why should it even work?

Stochastic gradient descent (SGD) method

Starting from \mathbf{w}_1 , in each iteration $k = 1, 2, \dots$:

Draw one of the coordinates $i_k \in \{1, \dots, N\}$

Do a step along the negative gradient of function $f_{i_k}(\mathbf{w}_k)$:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha_k \nabla f_{i_k}(\mathbf{w}_k)$$

Why should it even work?

One can think of a gradient of a single component $\nabla f_i(\mathbf{w}_k)$ as an **estimate** of the true ("full") gradient $\nabla f(\mathbf{w}_k)$

Indeed, **on expectation** the gradient is:

$$\frac{1}{n} \nabla f_1(\mathbf{w}_k) + \frac{1}{n} \nabla f_2(\mathbf{w}_k) + \dots + \frac{1}{n} \nabla f_N(\mathbf{w}_k) = \frac{1}{n} \nabla f(\mathbf{w}_k)$$

Example: linear regression

$$\min_{\mathbf{w} \in \mathbb{R}^n} f(\mathbf{w}) = \sum_{i=1}^N \underbrace{(y_i - \mathbf{x}_i^\top \mathbf{w})^2}_{f_i(\mathbf{w})}$$

Example: linear regression

$$\min_{\mathbf{w} \in \mathbb{R}^n} f(\mathbf{w}) = \sum_{i=1}^N \underbrace{(y_i - \mathbf{x}_i^\top \mathbf{w})^2}_{f_i(\mathbf{w})}$$

Compute the partial derivatives $f_i(\mathbf{w})$:

$$\frac{\partial f_i(\mathbf{w})}{\partial w_j} = -2(y_i - \mathbf{x}_i^\top \mathbf{w})x_{ij}$$

Example: linear regression

$$\min_{\mathbf{w} \in \mathbb{R}^n} f(\mathbf{w}) = \sum_{i=1}^N \underbrace{(y_i - \mathbf{x}_i^\top \mathbf{w})^2}_{f_i(\mathbf{w})}$$

Compute the partial derivatives $f_i(\mathbf{w})$:

$$\frac{\partial f_i(\mathbf{w})}{\partial w_j} = -2(y_i - \mathbf{x}_i^\top \mathbf{w})x_{ij}$$

Therefore the gradient is: $\nabla f_i(\mathbf{w}) = -2(y_i - \mathbf{x}_i^\top \mathbf{w})\mathbf{x}_i$

Example: linear regression

$$\min_{\mathbf{w} \in \mathbb{R}^n} f(\mathbf{w}) = \sum_{i=1}^N \underbrace{(y_i - \mathbf{x}_i^\top \mathbf{w})^2}_{f_i(\mathbf{w})}$$

Compute the partial derivatives $f_i(\mathbf{w})$:

$$\frac{\partial f_i(\mathbf{w})}{\partial w_j} = -2(y_i - \mathbf{x}_i^\top \mathbf{w})x_{ij}$$

Therefore the gradient is: $\nabla f_i(\mathbf{w}) = -2(y_i - \mathbf{x}_i^\top \mathbf{w})\mathbf{x}_i$

Starting from \mathbf{w}_1 , in iterations $k = 1, 2, \dots$:

Draw one of the data points $i_k \in \{1, \dots, N\}$

Make a step towards to negative gradient of $f_{i_k}(\mathbf{w}_k)$:

$$\mathbf{w}_{k+1} = \mathbf{w}_k + 2\alpha_k(y_{i_k} - \mathbf{x}_{i_k}^\top \mathbf{w}_k)\mathbf{x}_{i_k}$$

Example: linear regression

$$\min_{\mathbf{w} \in \mathbb{R}^n} f(\mathbf{w}) = \sum_{i=1}^N \underbrace{(y_i - \mathbf{x}_i^\top \mathbf{w})^2}_{f_i(\mathbf{w})}$$

Compute the partial derivatives $f_i(\mathbf{w})$:

$$\frac{\partial f_i(\mathbf{w})}{\partial w_j} = -2(y_i - \mathbf{x}_i^\top \mathbf{w})x_{ij}$$

Therefore the gradient is: $\nabla f_i(\mathbf{w}) = -2(y_i - \mathbf{x}_i^\top \mathbf{w})\mathbf{x}_i$

Starting from \mathbf{w}_1 , in iterations $k = 1, 2, \dots$:

Draw one of the data points $i_k \in \{1, \dots, N\}$

Make a step towards to negative gradient of $f_{i_k}(\mathbf{w}_k)$:

$$\mathbf{w}_{k+1} = \mathbf{w}_k + 2\alpha_k(y_{i_k} - \mathbf{x}_{i_k}^\top \mathbf{w}_k)\mathbf{x}_{i_k}$$

A step towards the data points \mathbf{x}_{i_k} , with step size proportional to the “over/under-estimation” of the true output $(y_{i_k} - \mathbf{x}_{i_k}^\top \mathbf{w}_k)$.

Pros and cons of SGD

Pros:

- **Speed**: computing the gradient requires taking only a single data point.
- **Scalability**: the entire data set need not even be loaded into the memory (RAM).
- **Simplicity**: the gradient of f_i gives a very simple expression for weight update.

Cons

- **Slow convergence**: the algorithm generally converges slowly and the number of iterations that far exceed the number of data points.
- **Step size α_k** : choosing α_k by a univariate optimization (line search) does not give good results, because we are not optimizing the original function f but only one of its components f_i .

Pros outweigh cons: The SGD algorithm is currently the most commonly used optimization method in machine learning!

SGD in practice

SGD in practice

- Usually we do not randomly draw the data points, but rather go over the entire data set which is first shuffled randomly (worst similarly well, but the implementation is much simpler and more efficient)

SGD in practice

- Usually we do not randomly draw the data points, but rather go over the entire data set which is first shuffled randomly (worst similarly well, but the implementation is much simpler and more efficient)
- Requires doing several passes over the entire data set (a single pass is called an **epoch**).

SGD in practice

- Usually we do not randomly draw the data points, but rather go over the entire data set which is first shuffled randomly (worst similarly well, but the implementation is much simpler and more efficient)
- Requires doing several passes over the entire data set (a single pass is called an **epoch**).
- Methods for choosing the step sizes α_k :

SGD in practice

- Usually we do not randomly draw the data points, but rather go over the entire data set which is first shuffled randomly (worst similarly well, but the implementation is much simpler and more efficient)
- Requires doing several passes over the entire data set (a single pass is called an **epoch**).
- Methods for choosing the step sizes α_k :
 - ▶ A **constant** (small) step size $\alpha_k = \alpha$

SGD in practice

- Usually we do not randomly draw the data points, but rather go over the entire data set which is first shuffled randomly (worst similarly well, but the implementation is much simpler and more efficient)
- Requires doing several passes over the entire data set (a single pass is called an **epoch**).
- Methods for choosing the step sizes α_k :
 - ▶ A **constant** (small) step size $\alpha_k = \alpha$
 \Rightarrow Often used in practice, works good but requires to tune α by trial and error

SGD in practice

- Usually we do not randomly draw the data points, but rather go over the entire data set which is first shuffled randomly (worst similarly well, but the implementation is much simpler and more efficient)
- Requires doing several passes over the entire data set (a single pass is called an **epoch**).
- Methods for choosing the step sizes α_k :
 - ▶ A **constant** (small) step size $\alpha_k = \alpha$
 \implies Often used in practice, works good but requires to tune α by trial and error
 - ▶ A **decreasing** as $\sim \frac{1}{\sqrt{k}}$ step size, i.e., $\alpha_k = \alpha/\sqrt{k}$

SGD in practice

- Usually we do not randomly draw the data points, but rather go over the entire data set which is first shuffled randomly (worst similarly well, but the implementation is much simpler and more efficient)
- Requires doing several passes over the entire data set (a single pass is called an **epoch**).
- Methods for choosing the step sizes α_k :
 - ▶ A **constant** (small) step size $\alpha_k = \alpha$
 \Rightarrow Often used in practice, works good but requires to tune α by trial and error
 - ▶ A **decreasing** as $\sim \frac{1}{\sqrt{k}}$ step size, i.e., $\alpha_k = \alpha/\sqrt{k}$
 \Rightarrow Convergence guaranteed but can work too slow sometimes.

SGD in practice

- Usually we do not randomly draw the data points, but rather go over the entire data set which is first shuffled randomly (worst similarly well, but the implementation is much simpler and more efficient)
- Requires doing several passes over the entire data set (a single pass is called an **epoch**).
- Methods for choosing the step sizes α_k :
 - ▶ A **constant** (small) step size $\alpha_k = \alpha$
 \Rightarrow Often used in practice, works good but requires to tune α by trial and error
 - ▶ A **decreasing** as $\sim \frac{1}{\sqrt{k}}$ step size, i.e., $\alpha_k = \alpha/\sqrt{k}$
 \Rightarrow Convergence guaranteed but can work too slow sometimes.
- In practice, the gradient is calculated not on a single data point, but on a small group of data points (still much smaller than the entire data set), called a **mini-batch**

Convergence of SGD

Theorem: Let $f_i(\mathbf{w})$ ($i = 1, \dots, N$) be convex functions and let \mathbf{w}^* be the (global) minimizer of $f(\mathbf{w})$. After K iterations of the SGD algorithm with a **constant** learning rate α we have

$$\min_{k=1,\dots,K} f(\mathbf{w}_k) - f(\mathbf{w}^*) \leq \frac{\|\mathbf{w}_1 - \mathbf{w}^*\|^2}{2\alpha K} + \frac{\alpha}{2K} \sum_{k=1}^K \|\nabla f_{i_k}(\mathbf{w}_k)\|^2$$

Convergence of SGD

the influence of "initial condition"
(distance from optimum at start)
(decreases with α)

influence of gradients encountered
on the trajectory of the algorithm
(increases with α)

$$\min_{k=1,\dots,K} f(\mathbf{w}_k) - f(\mathbf{w}^*) \leq \frac{\|\mathbf{w}_1 - \mathbf{w}^*\|^2}{2\alpha K} + \frac{\alpha}{2K} \sum_{k=1}^K \|\nabla f_{i_k}(\mathbf{w}_k)\|^2$$

Convergence of SGD

Theorem: Let $f_i(\mathbf{w})$ ($i = 1, \dots, N$) be convex functions and let \mathbf{w}^* be the (global) minimizer of $f(\mathbf{w})$. After K iterations of the SGD algorithm with a **constant** learning rate α we have

$$\min_{k=1,\dots,K} f(\mathbf{w}_k) - f(\mathbf{w}^*) \leq \frac{\|\mathbf{w}_1 - \mathbf{w}^*\|^2}{2\alpha K} + \frac{\alpha}{2K} \sum_{k=1}^K \|\nabla f_{i_k}(\mathbf{w}_k)\|^2$$

Conclusion: it is reasonable to choose α so that both terms on the right-hand side are of similar size:

$$(\dots) \frac{1}{\alpha K} \simeq \frac{\alpha}{K} \sum_{k=1}^K (\dots)$$

Convergence of SGD

Theorem: Let $f_i(\mathbf{w})$ ($i = 1, \dots, N$) be convex functions and let \mathbf{w}^* be the (global) minimizer of $f(\mathbf{w})$. After K iterations of the SGD algorithm with a **constant** learning rate α we have

$$\min_{k=1,\dots,K} f(\mathbf{w}_k) - f(\mathbf{w}^*) \leq \frac{\|\mathbf{w}_1 - \mathbf{w}^*\|^2}{2\alpha K} + \frac{\alpha}{2K} \sum_{k=1}^K \|\nabla f_{i_k}(\mathbf{w}_k)\|^2$$

Conclusion: it is reasonable to choose α so that both terms on the right-hand side are of similar size:

$$(\dots) \frac{1}{\alpha K} \simeq \alpha(\dots)$$

Convergence of SGD

Theorem: Let $f_i(\mathbf{w})$ ($i = 1, \dots, N$) be convex functions and let \mathbf{w}^* be the (global) minimizer of $f(\mathbf{w})$. After K iterations of the SGD algorithm with a **constant** learning rate α we have

$$\min_{k=1,\dots,K} f(\mathbf{w}_k) - f(\mathbf{w}^*) \leq \frac{\|\mathbf{w}_1 - \mathbf{w}^*\|^2}{2\alpha K} + \frac{\alpha}{2K} \sum_{k=1}^K \|\nabla f_{i_k}(\mathbf{w}_k)\|^2$$

Conclusion: it is reasonable to choose α so that both terms on the right-hand side are of similar size:

$$(\dots) \frac{1}{K} \simeq \alpha^2$$

Convergence of SGD

Theorem: Let $f_i(\mathbf{w})$ ($i = 1, \dots, N$) be convex functions and let \mathbf{w}^* be the (global) minimizer of $f(\mathbf{w})$. After K iterations of the SGD algorithm with a **constant** learning rate α we have

$$\min_{k=1,\dots,K} f(\mathbf{w}_k) - f(\mathbf{w}^*) \leq \frac{\|\mathbf{w}_1 - \mathbf{w}^*\|^2}{2\alpha K} + \frac{\alpha}{2K} \sum_{k=1}^K \|\nabla f_{i_k}(\mathbf{w}_k)\|^2$$

Conclusion: it is reasonable to choose α so that both terms on the right-hand side are of similar size:

$$\alpha \simeq \frac{(\dots)}{\sqrt{K}}$$

Convergence of SGD

Theorem: Let $f_i(\mathbf{w})$ ($i = 1, \dots, N$) be convex functions and let \mathbf{w}^* be the (global) minimizer of $f(\mathbf{w})$. After K iterations of the SGD algorithm with a **constant** learning rate α we have

$$\min_{k=1,\dots,K} f(\mathbf{w}_k) - f(\mathbf{w}^*) \leq \frac{\|\mathbf{w}_1 - \mathbf{w}^*\|^2}{2\alpha K} + \frac{\alpha}{2K} \sum_{k=1}^K \|\nabla f_{i_k}(\mathbf{w}_k)\|^2$$

Conclusion: it is reasonable to choose α so that both terms on the right-hand side are of similar size:

$$\alpha \simeq \frac{(\dots)}{\sqrt{K}}$$

Therefore in practice one sometimes take the decreasing step sizes α_k according to $\alpha_k = \frac{\text{const}}{\sqrt{k}}$

Convergence of SGD

Theorem: Let $f_i(\mathbf{w})$ ($i = 1, \dots, N$) be convex functions and let \mathbf{w}^* be the (global) minimizer of $f(\mathbf{w})$. After K iterations of the SGD algorithm with a **constant** learning rate α we have

$$\min_{k=1,\dots,K} f(\mathbf{w}_k) - f(\mathbf{w}^*) \leq \frac{\|\mathbf{w}_1 - \mathbf{w}^*\|^2}{2\alpha K} + \frac{\alpha}{2K} \sum_{k=1}^K \|\nabla f_{i_k}(\mathbf{w}_k)\|^2$$

Conclusion: it is reasonable to choose α so that both terms on the right-hand side are of similar size:

$$\alpha \simeq \frac{(\dots)}{\sqrt{K}}$$

Therefore in practice one sometimes take the decreasing step sizes α_k according to $\alpha_k = \frac{\text{const}}{\sqrt{k}}$

Such tuning of α_k guarantees the suboptimality (optimization error) of order $O\left(\frac{1}{\sqrt{K}}\right)$

Convergence comparison

algorithm	convergence	error after k iterations
Steepest descent	linear	$\propto e^{-ck}$
Newton-Raphson	quadratic	$\propto e^{-c_1 e^{c_2 k}}$
SGD	sublinear	$\propto \frac{c}{\sqrt{k}}$

The SGD algorithm is **very slow** (convergence-wise) in comparison to all previously considered methods

It is applied when we do not need a very good optimization accuracy while we are concerned with the computational time of the optimization procedure.

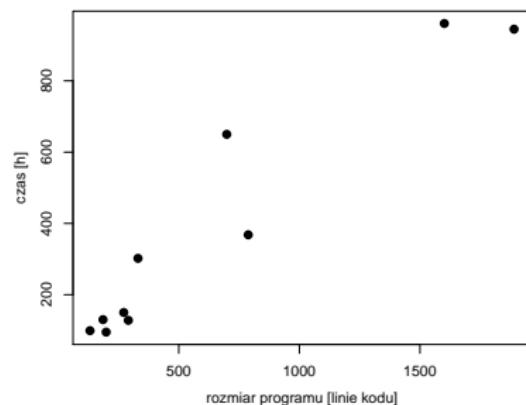
Example: assessment of coding time

X	Y
Program size	Coding time
186	130
699	650
132	99
272	150
291	128
331	302
199	95
1890	945
788	368
1601	961

Example: assessment of coding time

X Y

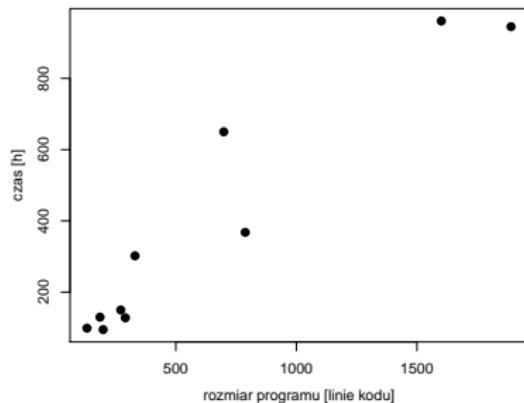
Program size	Coding time
186	130
699	650
132	99
272	150
291	128
331	302
199	95
1890	945
788	368
1601	961



Example: assessment of coding time

X Y

Program size	Coding time
186	130
699	650
132	99
272	150
291	128
331	302
199	95
1890	945
788	368
1601	961



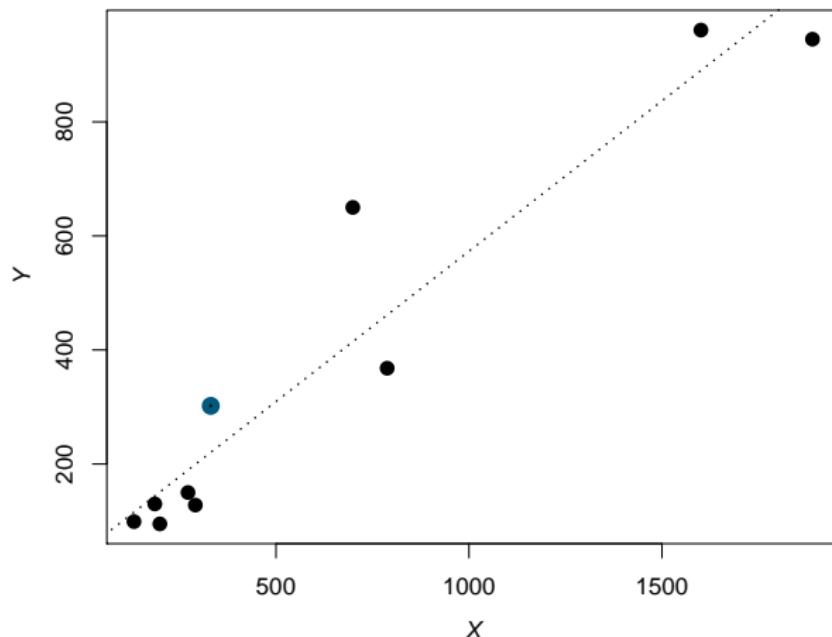
$$w_0 = 45.93, w_1 = 0.5273$$

Example: assessment of coding time

- Start with solution $\mathbf{w} = (w_0, w_1) = \mathbf{0}$.
- Step: $\alpha_k = 0.5/\sqrt{k}$.
- The units were selected to 1000 mins and 1000 code lines, to avoid large numbers.
(normalizing the scales is in practice crucial for the convergence of SGD)

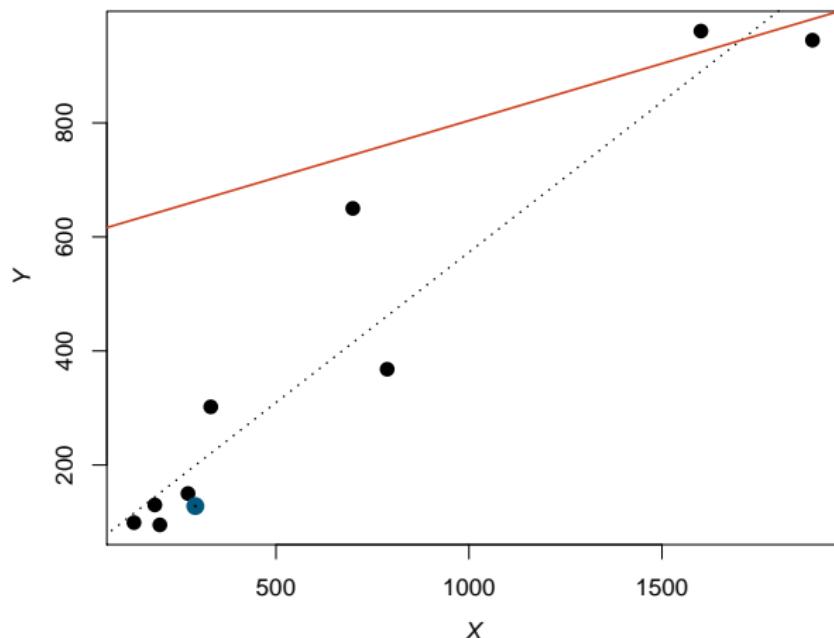
Example: assessment of coding time

iteration 1



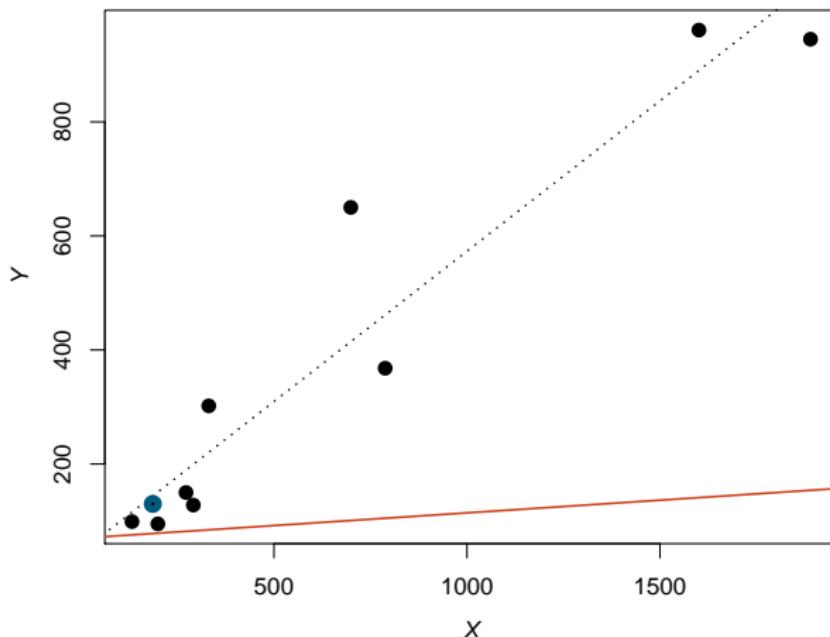
Example: assessment of coding time

iteration 2



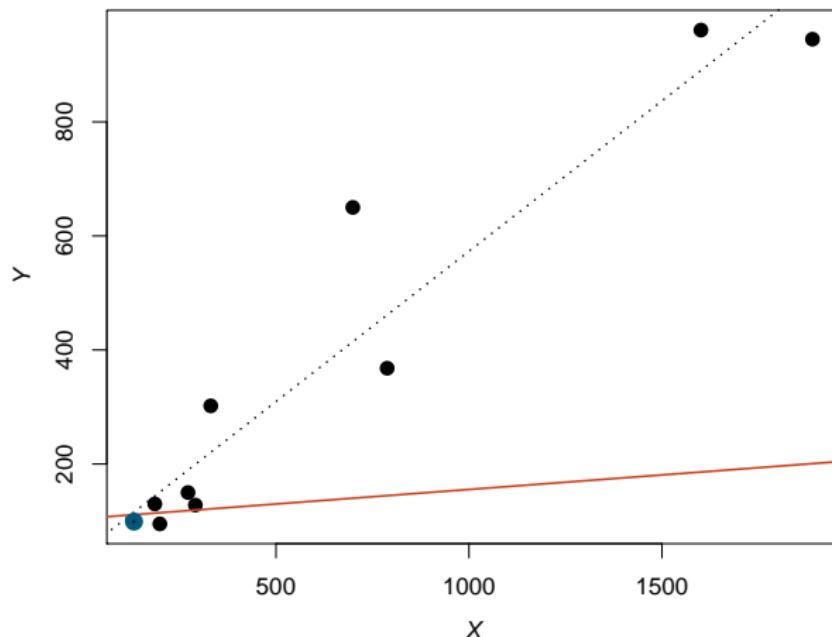
Example: assessment of coding time

iteration 3



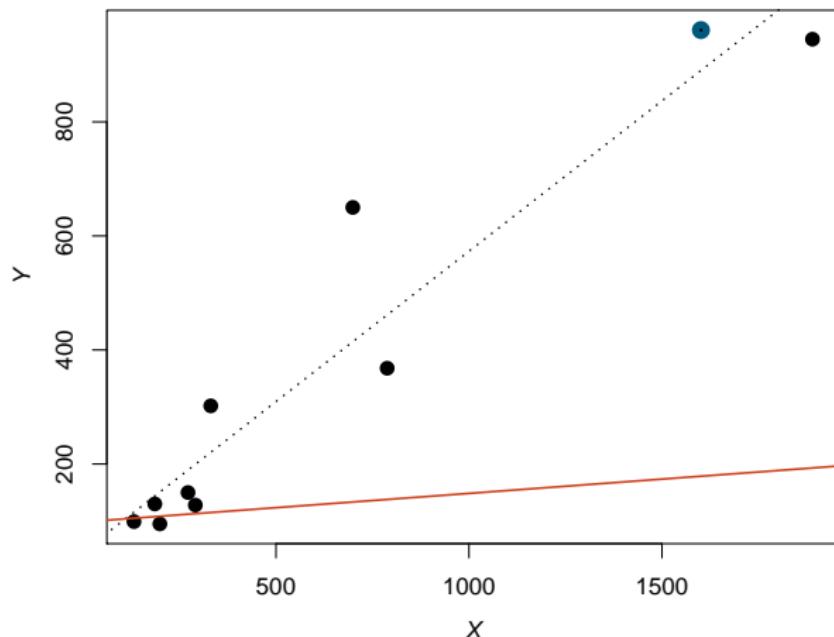
Example: assessment of coding time

iteration 4



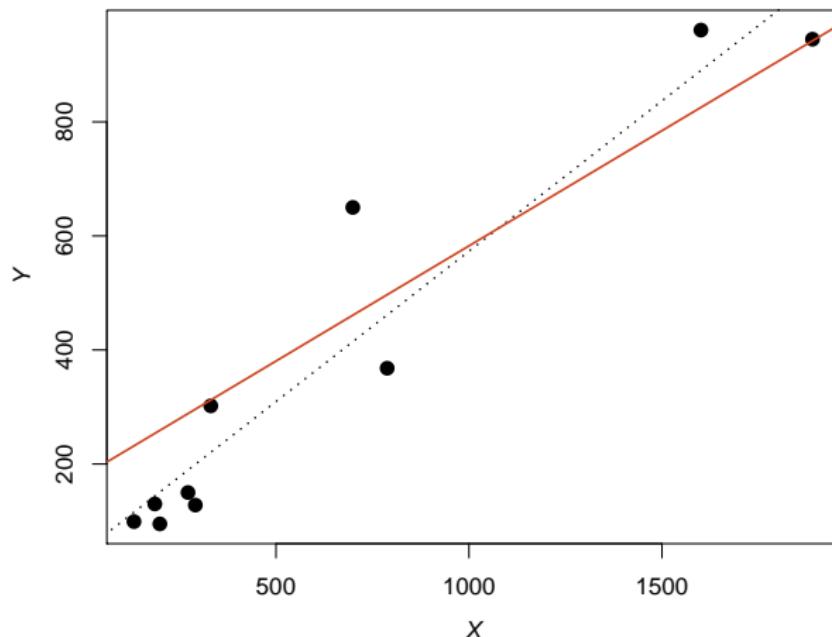
Example: assessment of coding time

iteration 5



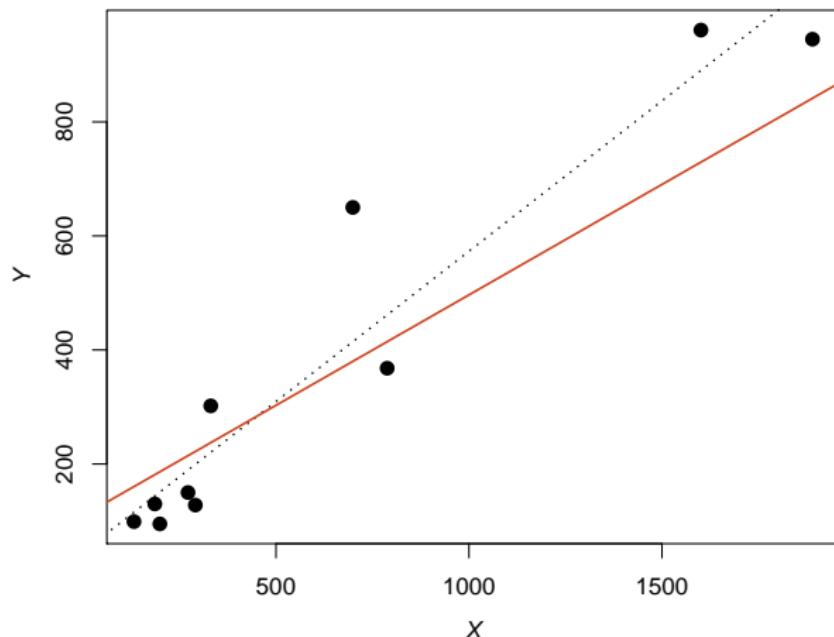
Example: assessment of coding time

iteration 10



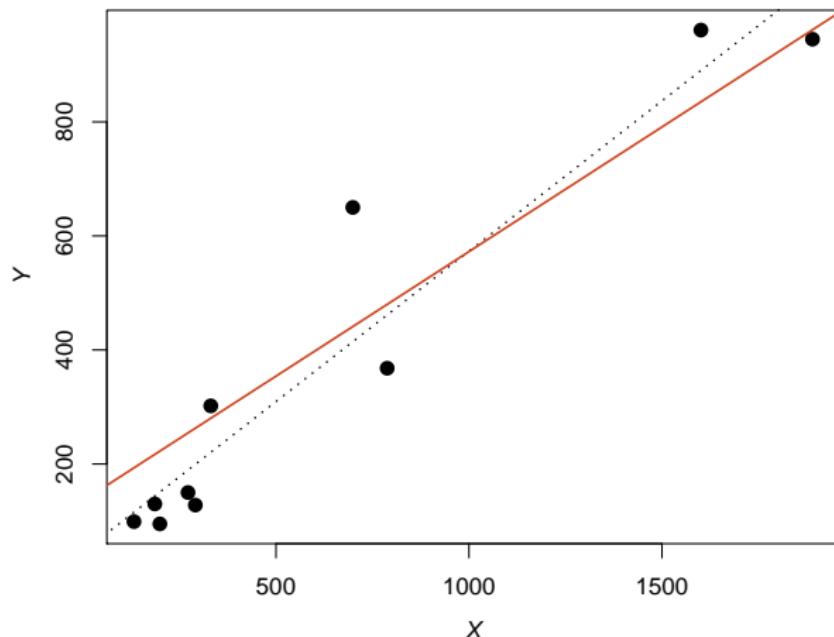
Example: assessment of coding time

iteration 15



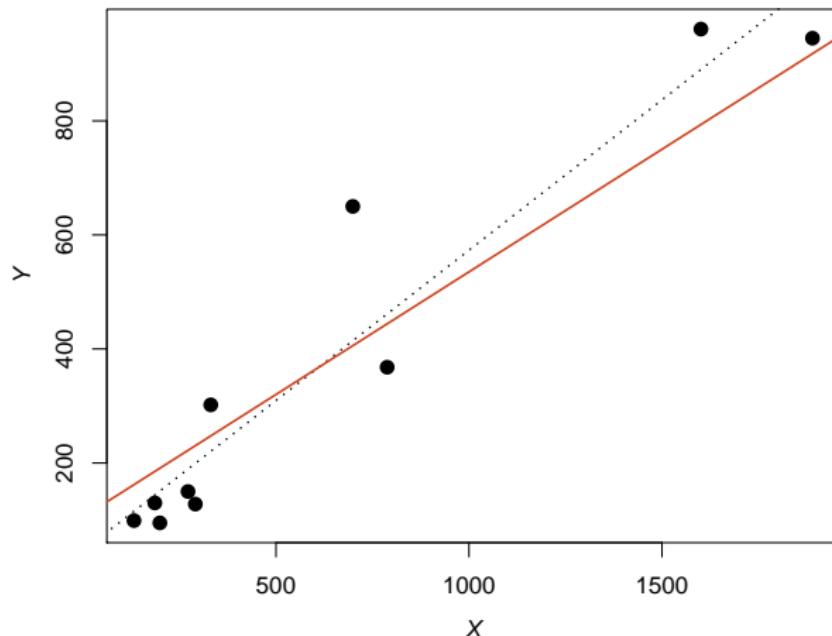
Example: assessment of coding time

iteration 20



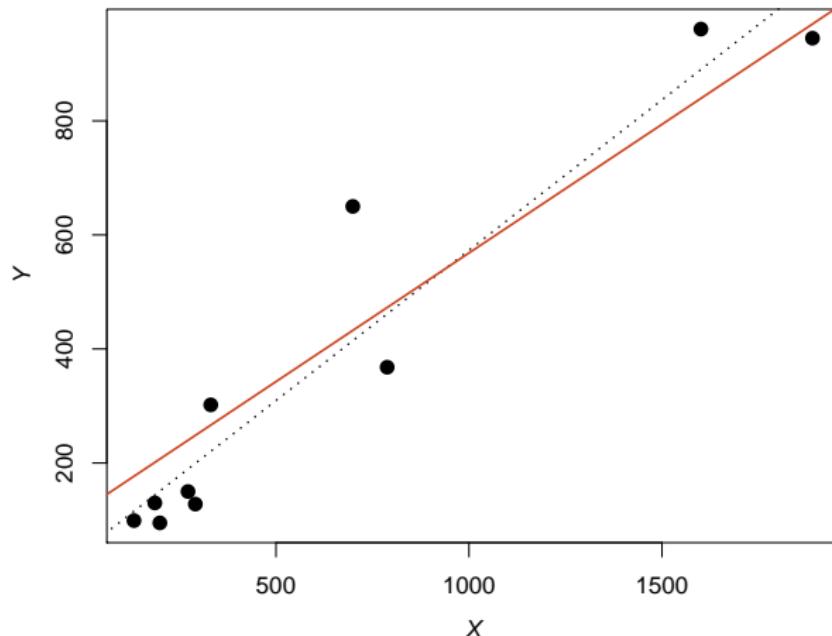
Example: assessment of coding time

iteration 25



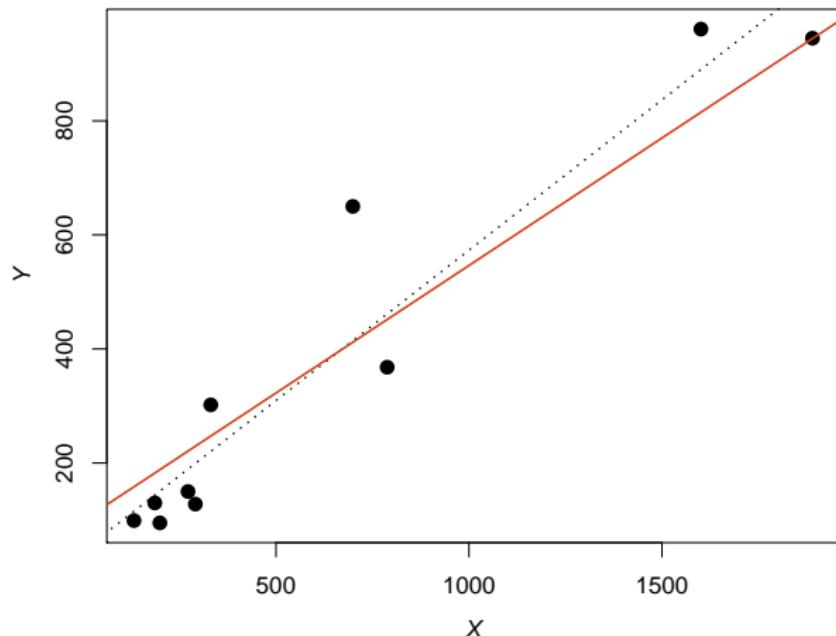
Example: assessment of coding time

iteration 30



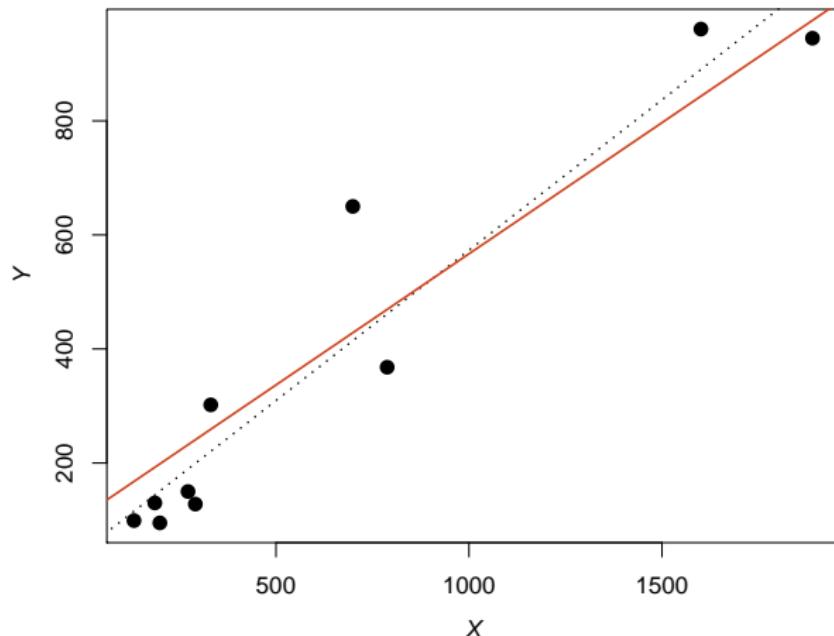
Example: assessment of coding time

iteration 35



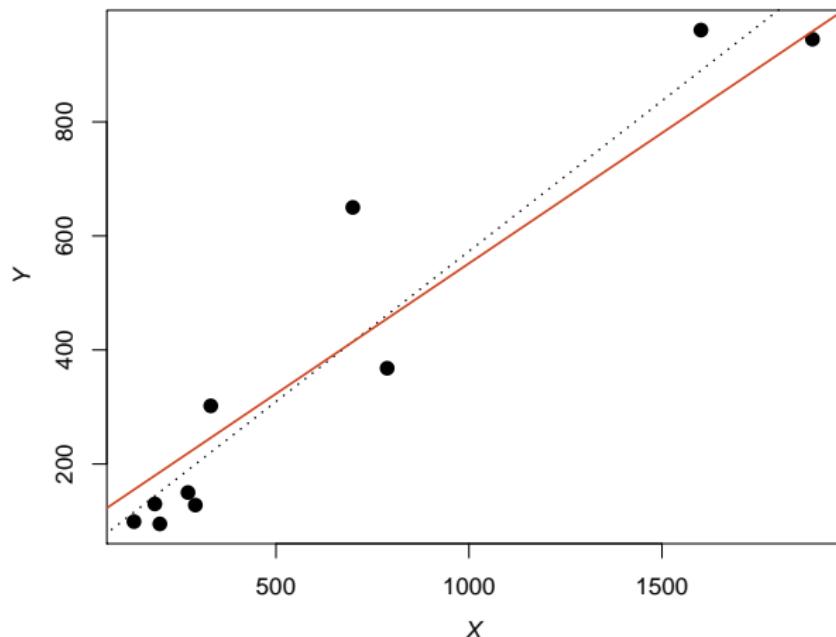
Example: assessment of coding time

iteration 40



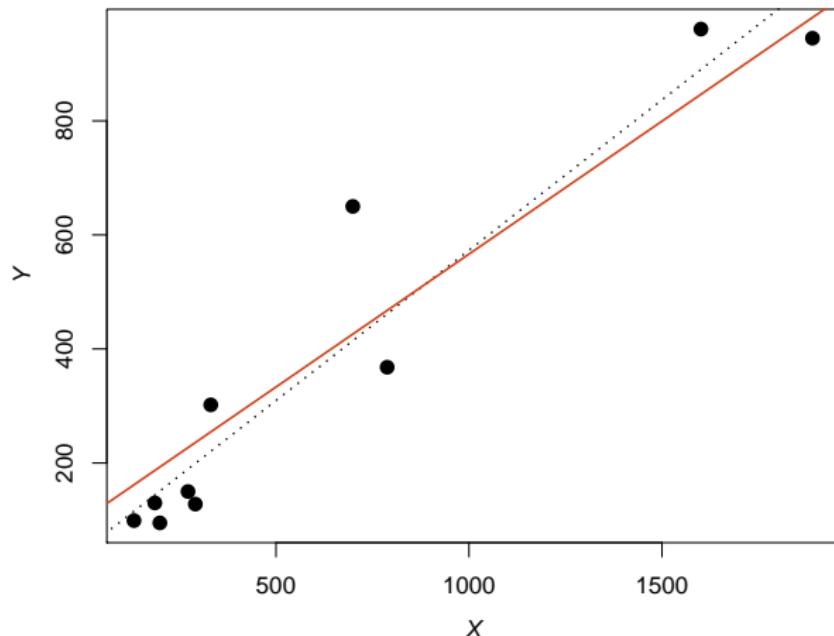
Example: assessment of coding time

iteration 45



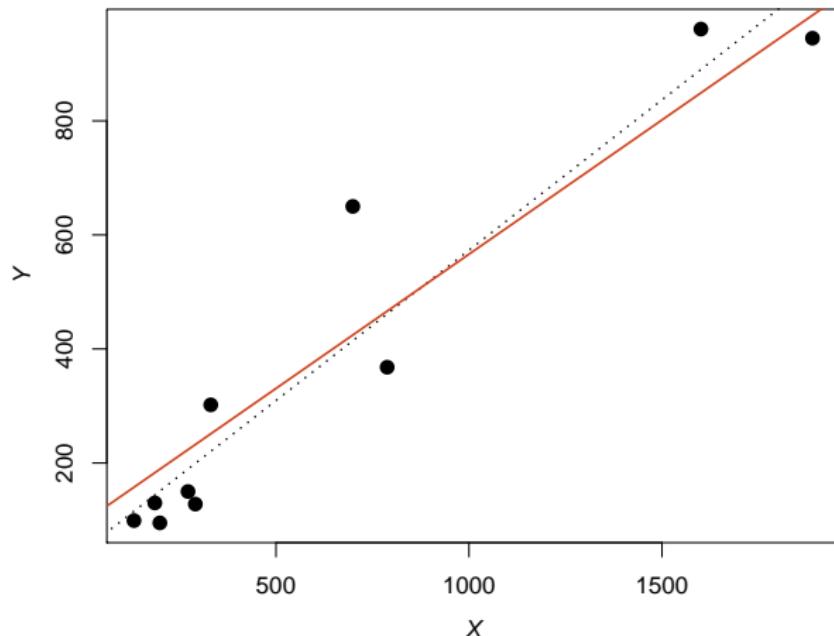
Example: assessment of coding time

iteration 50



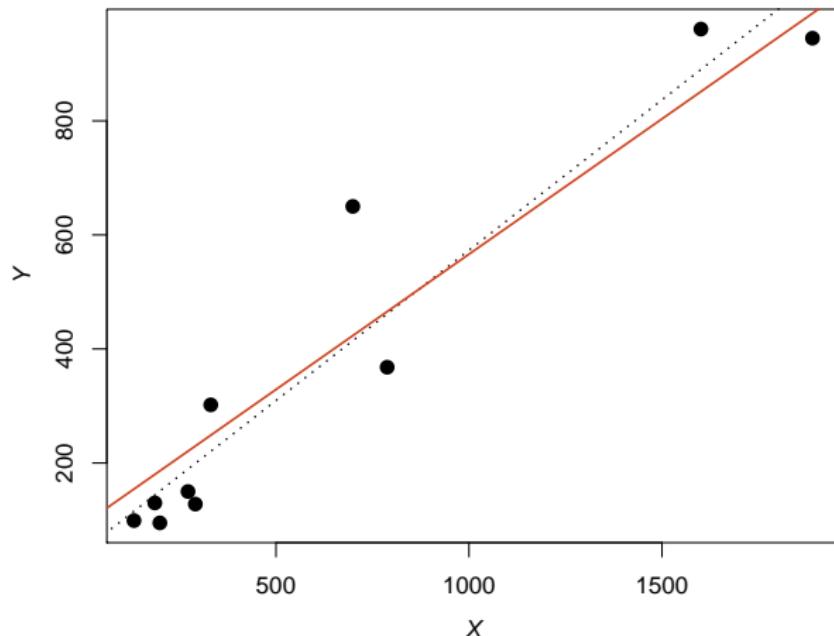
Example: assessment of coding time

iteration 60



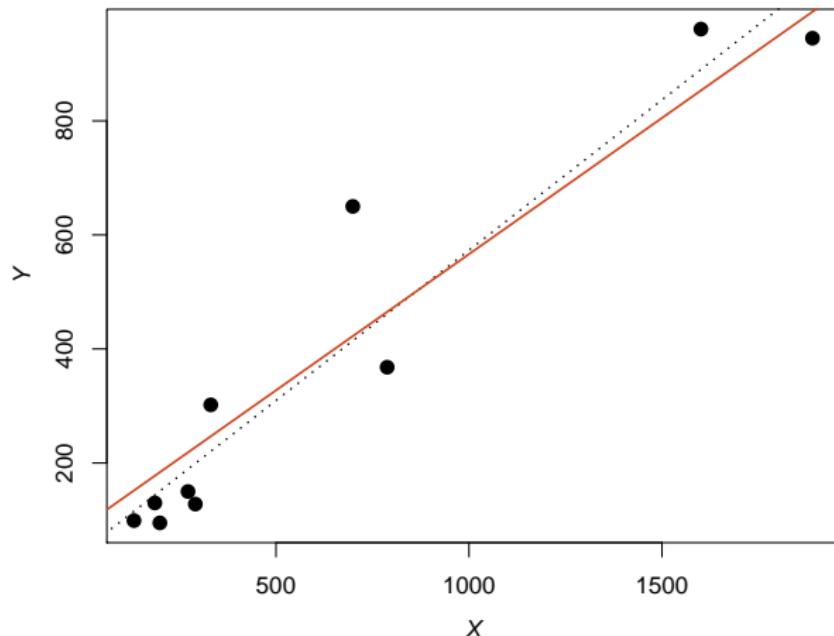
Example: assessment of coding time

iteration 70



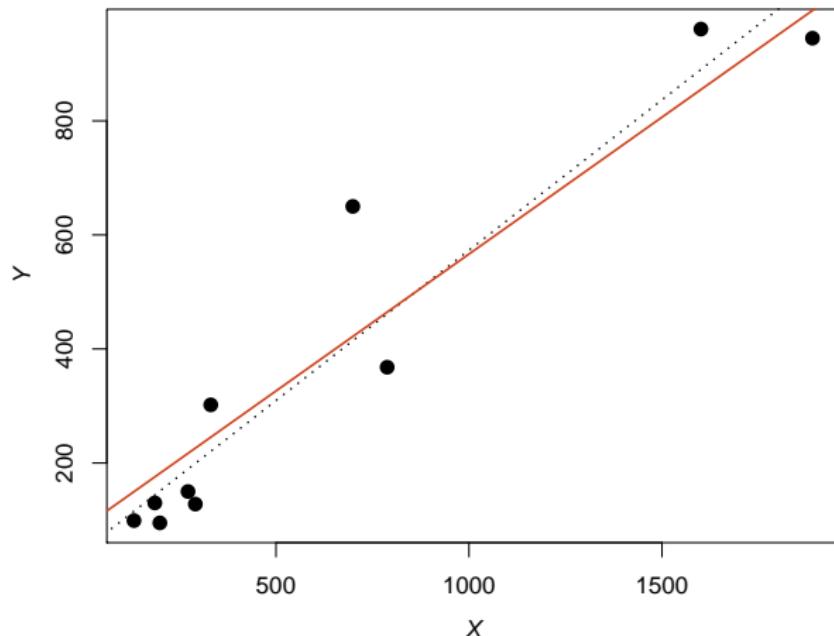
Example: assessment of coding time

iteration 80



Example: assessment of coding time

iteration 90



Example: assessment of coding time

iteration 100

