INFO-F524: Continuous Optimization
– Introduction –

Bernard Fortz, Dimitri Papadimitriou

Office: **2N3 206**

Université Libre de Bruxelles

dimitrios.papadimitriou@ulb.be

Annee Académique: 2024-2025

Course Structure

1. **Unconstrained Optimization**
2. **Nonlinear Programming: first-order iterative algorithms**
3. **Nonlinear Programming: second-order iterative algorithms**
4. **Constrained Optimization, Duality and Lagrangian**
5. **Dual and Lagrangian methods**
6. **Barrier and Interior-point methods**

7a. **Relaxation and decomposition methods for mixed integer (non)linear programming**

7b. **Stochastic programming**

Foreword (Ia)

- **Optimization problem** (OP): problem of finding / selecting the best solution (with regard to some criteria) from the set of all feasible solutions
- **Solving** OP: applying set of mathematical tools to a **model**, i.e., formal representation of the OP

  $\rightarrow$ Mathematical optimization

Foreword (Ia)

- **Optimization problem** (OP): problem of finding / selecting the best solution (with regard to some criteria) from the set of all feasible solutions
- **Solving** OP: applying set of mathematical tools to a **model**, i.e., formal representation of the OP

  $\rightarrow$ Mathematical optimization

Modeling : process of identifying

- **Objective**: a quantitative measure (system performance) that can be represented by a single number
- **Variables** (unknowns): the objective depends on certain characteristics of the system

  $\rightarrow$ **Goal**: find values of variables that optimize the objective function
- **Constraints**: restriction on variables that represent complex relationships between variables

Foreword (Ia)

Modeling : process of identifying

- **Objective**: a quantitative measure (system performance) that can be represented by a single number
- **Variables** (unknowns): the objective depends on certain characteristics of the system

  $\rightarrow$ **Goal**: find values of variables that optimize the objective
- **Constraints**: restriction on variables represent complex relationships among the variables

Foreword (Ia)

Modeling : process of identifying

- **Objective**: a quantitative measure (system performance) that can be represented by a single number
- **Variables** (unknowns): the objective depends on certain characteristics of the system

  $\rightarrow$ **Goal**: find values of variables that optimize the objective
- **Constraints**: restriction on variables represent complex relationships among the variables

Main distinction

- OP that have constraints on the variables: **constrained OP**
- vs. OP that do not have constraints on the variables: **unconstrained OP**

  Unconstrained OP as reformulation of constrained OP (e.g., constraints replaced by penalization terms added to objective function)

Foreword (Ib)

Continuous optimization : variables in the model nominally allowed to take on a **continuous range of values**, usually real numbers

- variables used in the objective function are required to be continuous, that are allowed to take on any (real) values permitted by the constraints
- **key feature** that distinguishes continuous optimization from discrete or combinatorial optimization

Foreword (Ib)

Continuous optimization : variables in the model nominally allowed to take on a **continuous range of values**, usually real numbers

- variables used in the objective function are required to be continuous, that are allowed to take on any (real) values permitted by the constraints
- **key feature** that distinguishes continuous optimization from discrete or combinatorial optimization

Consequence: continuous optimization allows use of **calculus techniques**

- algorithms that generate a sequence of values of the variables, known as **iterates**, that converge to a solution of the problem
- stepping from one iterate to the next, the algorithm makes use of knowledge gained at previous iterates, and information about the model at current iterate
- possibly including information about its sensitivity to perturbations in variables: sensitivities defined in terms of first and second derivatives of functions defining the model

Foreword (Ic)

Continuous optimization

- Stricto sensu, methods to find the minimum or maximum value of a function of one or many real variables, subject to constraints
- Also encompasses **feasibility problems**, i.e., the finding of a point that satisfies all the constraints

Foreword (II)

Tradeoff

- Model is too complex $\rightarrow$ too specific or too high complexity (space/time)
- Model is too simple $\rightarrow$ low utility

Foreword (II)

Tradeoff

- Model is too complex $\rightarrow$ too specific or too high complexity (space/time)
- Model is too simple $\rightarrow$ low utility

...Requires good understanding of model properties !

Foreword (II)

Tradeoff

- Model is too complex $\rightarrow$ too specific or too high complexity (space/time)
- Model is too simple $\rightarrow$ low utility

...Requires good understanding of model properties !

Main criteria for selecting of a solving algorithm

- Robustness: initial conditions, data with (similar) statistical properties, etc.
- Efficiency: in computation time or memory space.
- Accuracy: error in solution due to sensitivity to data, error in computation, etc.

Foreword (III)

Again...

- rapidly converging method for a large unconstrained nonlinear problem may require too much memory space
- a robust method may also be the slowest
- etc.

Foreword (III)

Again...

- rapidly converging method for a large unconstrained nonlinear problem may require too much memory space
- a robust method may also be the slowest
- etc.

Tradeoffs

- between convergence rate and memory space
- between robustness and computation time
- etc.

Foreword (III)

Again...

- rapidly converging method for a large unconstrained nonlinear problem may require too much memory space
- a robust method may also be the slowest
- etc.

Tradeoffs

- between convergence rate and memory space
- between robustness and computation time
- etc.

Verification methods

- Optimality conditions: check if the computed values for set of model variables is OP solution
- Sensitivity analysis: continuous property allows sensitivities to be defined in terms of first and second derivatives of the functions that define the model

Structure of the Course: Main Chapters

1. **Unconstrained Optimization**
2. **Nonlinear Programming: first-order iterative algorithms**
3. **Nonlinear Programming: second-order iterative algorithms**
4. **Constrained Optimization, Duality and Lagrangian**
5. **Dual and Lagrangian methods**
6. **Barrier and Interior-point methods**

7a. **Relaxation and decomposition methods for mixed integer (non)linear programming**

7b. **Stochastic programming**

Ch.1 : Unconstrained Optimization

1.1 **Fundamentals**

1.2 **Global and Local Optimizer**

1.3 **First- and Second-order Conditions**

- 1.3.1 Definitions
- 1.3.2 Necessary conditions
- 1.3.3 Sufficient conditions
- 1.3.4 Convex functions

1.4 **Applications & Examples**

- 1.4.1 Profit maximization
- 1.4.2 Discriminating monopoly
- 1.4.3 Least Squares and Linear regression

Ch.2 : NLP: first-order iterative algorithms

2.1 **Line Search methods**

2.1.1 Conditions: sufficient decrease and curvature

2.1.2 Backtracking Linesearch

2.2 **Gradient methods**

2.2.1 Gradient Descent (GD)

2.2.2 Projected Gradient (PG)

2.2.3 Accelerated Gradient Descent (GD)

2.3 **Projection algorithms**

2.4 **Subgradient methods**

2.4.1 Subgradient and Subdifferential

2.4.2 Optimality conditions

2.4.3 Directional derivative and subgradients

2.4.4 Subgradient method is not a Descent method

2.5 **Proximal methods**

2.5.1 Proximal Point Method (PPM)

2.5.2 Proximal Gradient Method (PGM)

2.5.3 Accelerated PG

2.5.4 Fast Iterative Shrinkage-Thresholding Algorithm (FISTA)

Ch.3 : NLP: second-order iterative algorithms

Part I : **Newton Methods**

3.1 Newton-Raphson

3.2 Gauss-Newton (GN)

3.3 Levenberg–Marquardt: combines GD and GN

3.4 Applications: nonlinear least-squares and parameter fitting

Part II : **Quasi-Newton methods**

Prerequisite: Sequant eq., Sherman-Morrison-Woodbury formula, etc.

3.5 SR1: Symmetric rank-one update

3.6 BFGS: Broyden-Fletcher-Goldfarb-Shanno

3.7 DFP: Davidon-Fletcher-Powell

3.8 Broyden class

3.9 L-BFGS: Limited memory BFGS

Ch.4 : Constrained Optimization, Duality and Lagrangian

4.1 Fundamentals

4.2 Lagrange function, Lagrange dual function and dual problem

4.3 Strong vs. weak duality

4.4 Equality constraints

4.5 Inequality and bound constraints

4.6 Optimality conditions

4.7 Karush-Kuhn-Tucker (KKT) conditions and multipliers

4.8 Constraint Qualification

Ch.5 : Dual and Lagrangian methods

5.1 Lagrange multiplier method

5.2 Dual methods: ascent, decomposition

5.3 Penalty methods

5.4 Augmented Lagrangian method(s)

5.5 Dual Proximal methods: Dual PG, Dual-FISTA

5.6 Variants: Proximal ALM, Inexact ALM, etc.

Ch.6 : Barrier and Interior-point methods

6.1 Fundamentals

6.2 Affine scaling

6.3 Barrier function and problem

6.4 Central path methods (CPM)

6.5 Path following methods: primal, dual, primal-dual

6.6 Potential reduction methods

Ch.7: Open choice

7a **Either** Mixed integer (non)linear programming

7b **Or** Stochastic optimization

... possibility left to follow both

Ch.7a : Relaxation and decomposition methods for mixed integer (non)linear programming

7.1 Mixed-integer (non)linear problems

7.2 Elements of Mixed-integer programming

7.3 Duality and Mixed-integer programming

7.4 Decomposition methods : Lagrangian relaxation

7.5 Other decomposition : variable (Benders), structural, spatial/domain

Ch.7b : Stochastic optimization

7.1 Stochastic convex minimization : risk minimization problem

7.2 Computational methods

- o) Stochastic Approximation (SA)
- o) Sample Average Approximation (SAA)

7.3 Stochastic gradient (SG) and Variance reduction (VR)

7.4 Adaptive SG

7.5 Application : statistical learning

- o) Empirical risk minimization (ERM)
- o) Structural risk minimization (SRM)

References (I)

1. S. Boyd and L. Vandenberghe, **Convex Optimization**, Cambridge University Press

- Chapter 5 (Sections 5.1 to 5.4) cover the material about theoretical results for constrained optimization
- Chapter 9 (Sections 9.2 and 9.5) cover the gradient and Newton methods. (Note: proofs of convergence can -often- be omitted)
- Chapter 11 covers interior point methods that we will consider in the course (Ch.7)

2. A. Beck, **Introduction to Nonlinear Optimization**: Theory, Algorithms, and Applications with MATLAB, SIAM

- Convex optimization (functions, sets, etc.)
- Theory of un/constrained optimization
- Iterative methods (gradient, Newton, etc.), etc.

–Note: A.Beck, First-order methods in Optimization, MOS-SIAM Series on Optimization, 2017.

References (II)

3. D.G. Luenberger and Y. Ye, **Linear and Nonlinear Programming**, Fifth Edition, 2021, Springer.

- Part II (unconstrained optimization)
- Part III (constrained optimization)

cover several sections of the course

4. Dimitris Bertsekas, **Nonlinear Programming**, Third Edition, 2016, Athena Scientific.

Covers in-depth the following sections of the course

- Convex Programming
- Lagrange Multiplier Theory and Algorithms
- Duality, Dual Methods

5. Jorge Nocedal and Stephen J. Wright, **Numerical Optimization**, Second Edition, 2006.

- Covers most chapters of the course (except 4 partly and 5)

Course schedule

- **Week 21** : 1 x 2h
- **Week 22** : 2 x 2h
- **Week 23** : 1 x 2h
- **Week 24** : 2 x 2h
- **Week 25** : 1 x 2h
- **Week 26** : 2 x 2h
- **Week 27** : 1 x 2h
- **Week 28** : 2 x 2h
- **Week 29** : 1 x 2h
- **Week 30** : 2 x 2h
- **Week 31** : 1 x 2h
- Week 32-33 : Easter holiday period
- **Week 34** : 2 x 2h
- **Week 35** : 2 x 2h (including Q&A)

Project (see also UV section)

- MUST be realized by groups of 2 students: groups set by March 15, 2025

Project (see also UV section)

- MUST be realized by groups of 2 students: groups set by March 15, 2025
- Project dedicated to the programming, performance analysis, and numerical experimentation of optimization algorithms (technical description : early March)

Project (see also UV section)

- MUST be realized by groups of 2 students: groups set by March 15, 2025
- Project dedicated to the programming, performance analysis, and numerical experimentation of optimization algorithms (technical description : early March)
  - Code dully commented and explained in the technical report
  - Programming language : C/C++, FORTRAN, Octave, Julia, Python
    Groups selecting either Julia or Python shall also motivate selection of linear algebra and other numerical computation libraries

Project (see also UV section)

- The technical report to be delivered by May 23, 2025
  - Edition: LATEX and compiled in .pdf format (documented code in annex)
  - Language: may be written in French or English

    There is no specific style imposed on the writing of technical report as long as its content is scientifically and technically sound
  - Length: min 14 pages (single column), max number of pages not fixed

    well-balanced report: at most 1/3 of background, at least 1/2 of foreground, the remainder accounts for introductory, concluding section, etc.
  - **Note** : can be delivered for the second session organized in August (dates to be specified)

New initiative : IA

IA = **I**ndustrial **A**fternoon on applied mathematical optimization

- **Why** : mathematical optimization finds broad applicability in various sectors of industry
- **What** : 3 to 4 speakers (45min + 15min Q&A) from different sectors, e.g., energy, transport/logistic, ecology, software, image/signal processing, etc.
- **When** : first half of April 2025
- **Whom** : directed to MSc students (CS/Eng., Applied math's, etc.)
- **Where** : ULB, Campus de la Plaine

INFO-F524: Continuous Optimization
– Section 1 : Unconstrained Optimization –

Bernard Fortz, Dimitri Papadimitriou

Office: **2N3 206**

Université Libre de Bruxelles

dimitrios.papadimitriou@ulb.be

Annee Académique: 2024-2025

Unconstrained Optimization

- **Unconstrained** : minimization or maximization of a function f (called the objective function) with no a-priori restrictions on the domain of the function f ($\mathrm{dom}(f)$)

Unconstrained Optimization

- **Unconstrained** : minimization or maximization of a function f (called the objective function) with no a-priori restrictions on the domain of the function f ($\mathrm{dom}(f)$)
- **Continuous** : $\mathrm{dom}(f)$: $\{x \in \mathcal{C} \,|\, \mathcal{C} \subseteq \mathbb{R}^n\}$

Unconstrained Optimization

- **Unconstrained** : minimization or maximization of a function f (called the objective function) with no a-priori restrictions on the domain of the function f ($\mathrm{dom}(f)$)
- **Continuous** : $\mathrm{dom}(f)$: $\{x \in \mathcal{C} \,|\, \mathcal{C} \subseteq \mathbb{R}^n\}$
- **Note**: except when stated otherwise, the continuous optimization problems considered are finite dimensional ($n < \infty$), i.e., continuous optimization problems with a finite number of continuous decision variables (and a finite number of constraints)

Unconstrained Optimization

Notations

- $x \in \mathbb{R}^n$ $(n \geq 1)$: vector of real variables, also called unknowns
- $\mathcal{C}$ (or X) $\subseteq \mathbb{R}^n$ $(n \geq 1)$: (not necessarily) an open set included in $\mathbb{R}^n$
- $f : \mathbb{R}^n \to \mathbb{R}$: objective function, a scalar function of the variable $x \in \mathbb{R}^n$ to minimize or maximize
- $x^\star \in \mathcal{C}$ (or X): "solution" (minimizer or maximizer) of optimization problem
- $f^\star$: finite value attained by the objective function f at $x^\star$

Unconstrained Optimization

Notations

- $x \in \mathbb{R}^n$ $(n \geq 1)$: vector of real variables, also called unknowns
- $\mathcal{C}$ (or X) $\subseteq \mathbb{R}^n$ $(n \geq 1)$: (not necessarily) an open set included in $\mathbb{R}^n$
- $f : \mathbb{R}^n \to \mathbb{R}$: objective function, a scalar function of the variable $x \in \mathbb{R}^n$ to minimize or maximize
- $x^\star \in \mathcal{C}$ (or X): "solution" (minimizer or maximizer) of optimization problem
- $f^\star$: finite value attained by the objective function f at $x^\star$

Using these notations, unconstrained continuous **minimization** problem

$$\min_{x \in \mathcal{C} \subseteq \mathbb{R}^n} f(x) \tag{1}$$

Unconstrained Optimization

Notations

- $x \in \mathbb{R}^n$ $(n \geq 1)$: vector of real variables, also called unknowns
- $\mathcal{C}$ (or X) $\subseteq \mathbb{R}^n$ $(n \geq 1)$: (not necessarily) an open set included in $\mathbb{R}^n$
- $f : \mathbb{R}^n \to \mathbb{R}$: objective function, a scalar function of the variable $x \in \mathbb{R}^n$ to minimize or maximize
- $x^\star \in \mathcal{C}$ (or X): "solution" (minimizer or maximizer) of optimization problem
- $f^\star$: finite value attained by the objective function f at $x^\star$

Using these notations, unconstrained continuous **minimization** problem

$$\min_{x \in \mathcal{C} \subseteq \mathbb{R}^n} f(x) \tag{1}$$

<u>Note</u>: corresponding maximization problem obtained by using the relation

$$\min_x f(x) = -\max_x(-f(x))$$

Unconstrained Optimization

Using these notations, unconstrained continuous **minimization** problem

$$\min_{x \in \mathcal{C} \subseteq \mathbb{R}^n} f(x) \tag{1}$$

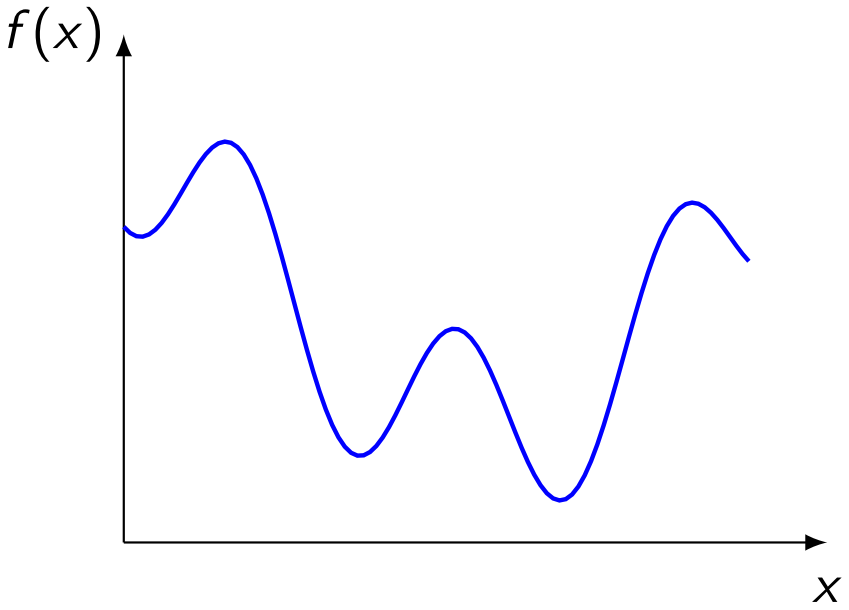
Given a function $f : \mathcal{C} \subseteq \mathbb{R}^n \to \mathbb{R}$, find $x^\star$ i.e. values of the variable $x \in \mathbb{R}^n$ where f attains its smallest value (f is minimized) or biggest value (f is maximized)

Unconstrained Optimization

Using these notations, unconstrained continuous **minimization** problem

$$\min_{x \in \mathcal{C} \subseteq \mathbb{R}^n} f(x) \tag{1}$$

Given a function $f : \mathcal{C} \subseteq \mathbb{R}^n \to \mathbb{R}$, find $x^\star$ i.e. values of the variable $x \in \mathbb{R}^n$ where f attains its smallest value (f is minimized) or biggest value (f is maximized)



Unconstrained Optimization

Def.: open ball

- An open ball is a subset $B_r(x)$ (of a metric space (X, d)) that includes all points of distance less than $r > 0$ from the point $x \in X$

$$B_r(x) = \{y \in X \mid d(x, y) < r\}$$

- **Examples**
  - ◦) In $\mathbb{R}$: open balls are open intervals
  - ◦) In $\mathbb{R}^2$: with the Euclidean metric d, open balls are disks
  - ◦) in $\mathbb{R}^3$: with the Euclidean metric d, open balls are spheres

Unconstrained Optimization

Def.: open ball

- An open ball is a subset $B_r(x)$ (of a metric space (X, d)) that includes all points of distance less than $r > 0$ from the point $x \in X$

$$B_r(x) = \{y \in X \mid d(x, y) < r\}$$

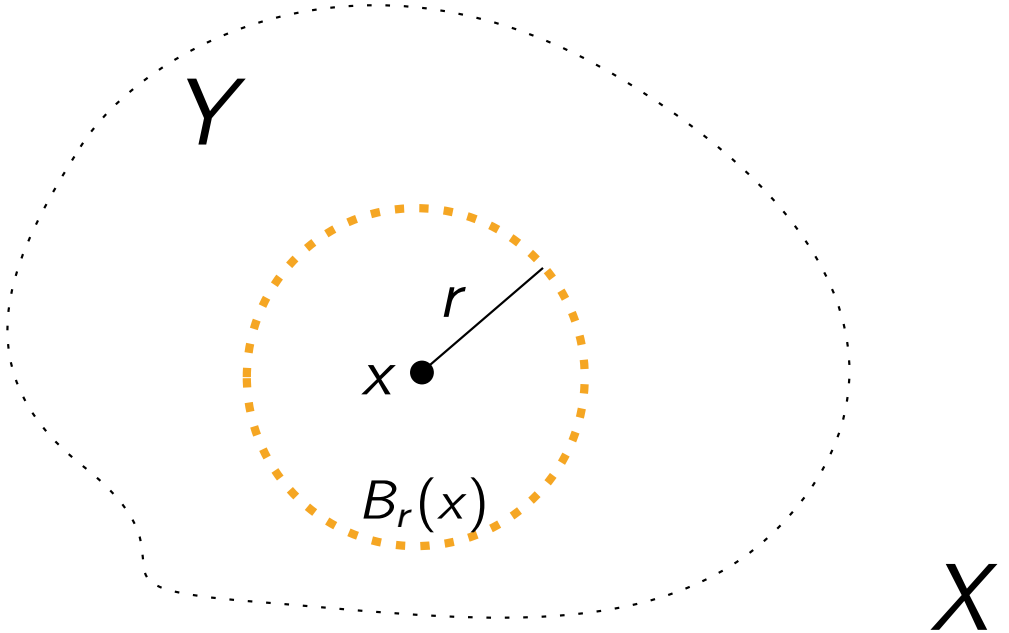
- **Examples**
  - ∘) In $\mathbb{R}$: open balls are open intervals
  - ∘) In $\mathbb{R}^2$: with the Euclidean metric d, open balls are disks
  - ∘) in $\mathbb{R}^3$: with the Euclidean metric d, open balls are spheres

Note : any norm $\|\cdot\|_p, p \geq 1$ defined on a vector space (normed vector space) gives a metric d on the vector space defined by $d(x, y) = \|x - y\|_p$

Unconstrained Optimization

Def.: open set

- The subset Y of X is said open in X; if for every point x of Y, there exists some radius $r > 0$ such that the open ball $B_r(x)$ centered at x is contained in Y, i.e., $\forall x \in Y$, $\exists\, r > 0$, such that $B_r(x) \subseteq Y$
- Note : open balls are themselves open sets but open sets are not necessarily open balls



Unconstrained Optimization: global vs. local optimizer

Let $f : \mathcal{C} \subseteq \mathbb{R}^n \to \mathbb{R} : x \to f(x)$

Def: Global minimum

A point $x^\star \in \mathcal{C}$ is a **global minimum** of function f over $\mathcal{C}$, if

$$f(x^\star) \leq f(x), \forall x \in \mathcal{C} \subseteq \mathbb{R}^n$$

Intuitively, a global minimum of function f is a point $x^\star \in \mathcal{C}$ where f attains its smallest value

Unconstrained Optimization: global vs. local optimizer

Let $f : \mathcal{C} \subseteq \mathbb{R}^n \to \mathbb{R} : x \to f(x)$

Def: Global minimum

A point $x^\star \in \mathcal{C}$ is a **global minimum** of function f over $\mathcal{C}$, if

$$f(x^\star) \leq f(x), \forall x \in \mathcal{C} \subseteq \mathbb{R}^n$$

Intuitively, a global minimum of function f is a point $x^\star \in \mathcal{C}$ where f attains its smallest value

Def: Strict global minimum

A point $x^\star \in \mathcal{C}$ is a **strict global minimum** of function f over $\mathcal{C}$, if

$$f(x^\star) < f(x), \forall x \in \mathcal{C} \subseteq \mathbb{R}^n, x \neq x^\star$$

Unconstrained Optimization: global vs. local optimizer

A global minimum/maximum can be and is often difficult to find because

i) our knowledge of the function f is usually only **local**

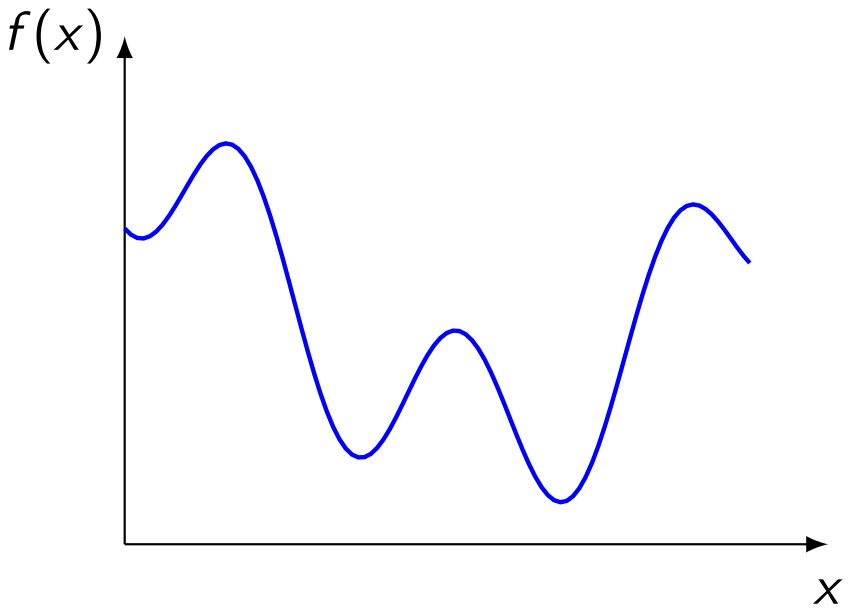
ii) the use of computational methods often leads to very hard challenges : **computational costs** dominated by evaluation of f, possibly its (partial) derivatives wrt to all variables ($n \gg 1$), etc.

Unconstrained Optimization: global vs. local optimizer

A global minimum/maximum can be and is often difficult to find because

i) our knowledge of the function f is usually only **local**

ii) the use of computational methods often leads to very hard challenges : **computational costs** dominated by evaluation of f, possibly its (partial) derivatives wrt to all variables ($n \gg 1$), etc.

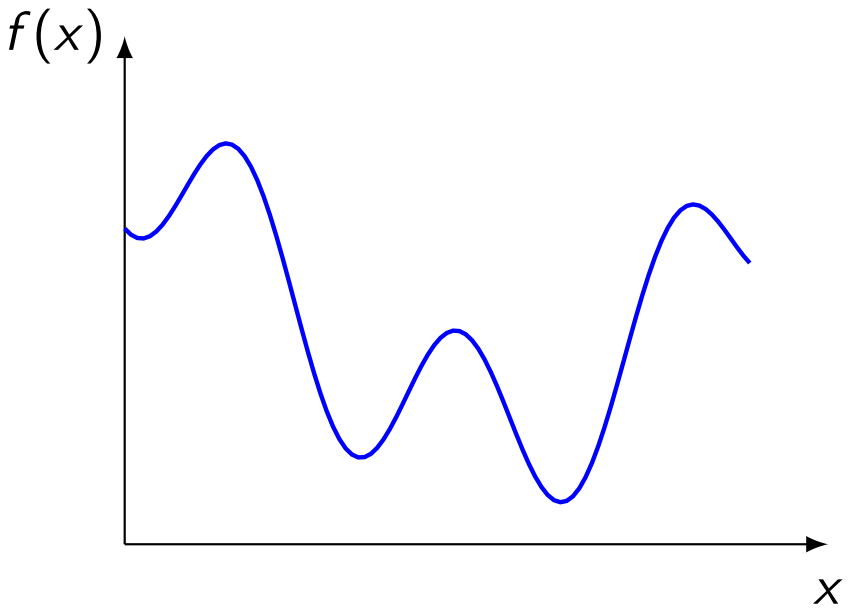


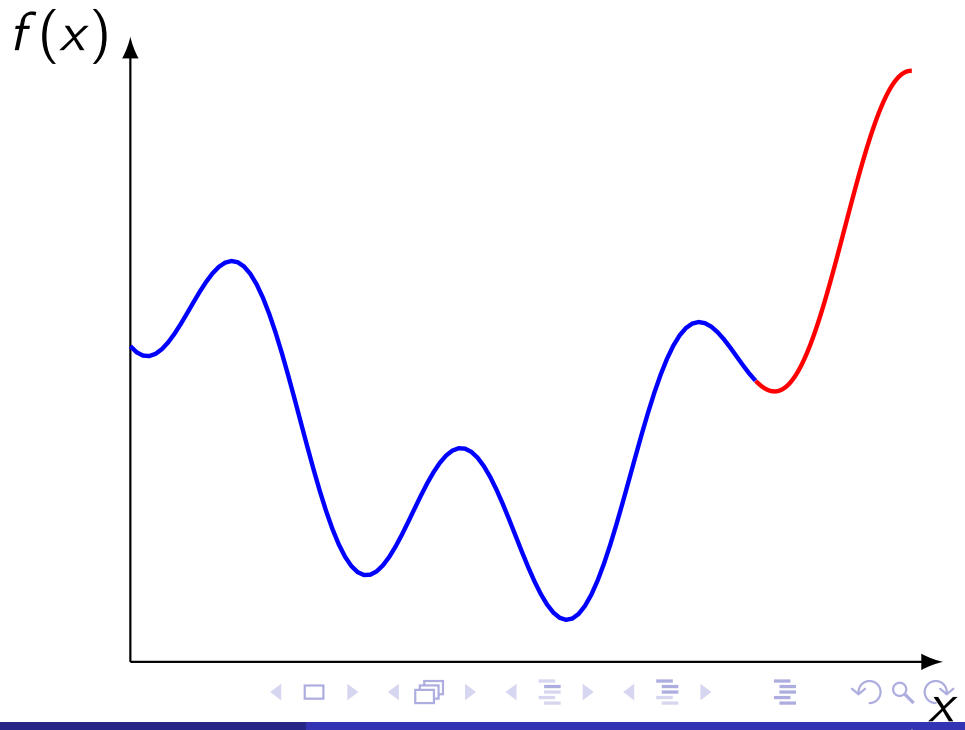
Unconstrained Optimization: global vs. local optimizer

A global minimum/maximum can be and is often difficult to find because

i) our knowledge of the function f is usually only **local**

ii) the use of computational methods often leads to very hard challenges : **computational costs** dominated by evaluation of f, possibly its (partial) derivatives wrt to all variables ($n \gg 1$), etc.





Unconstrained Optimization: global vs. local optimizer

A global minimum/maximum can be and is often difficult to find because

i) our knowledge of the function f is usually only **local**

ii) the use of computational methods often leads to very hard challenges : **computational costs** dominated by evaluation of f, possibly its (partial) derivatives wrt to all variables ($n \gg 1$), etc.

Unconstrained Optimization: global vs. local optimizer

A global minimum/maximum can be and is often difficult to find because

i) our knowledge of the function f is usually only **local**

ii) the use of computational methods often leads to very hard challenges : **computational costs** dominated by evaluation of f, possibly its (partial) derivatives wrt to all variables ($n \gg 1$), etc.

Noticeable particular but **important cases** for which finding of a global minimum remains doable

- Having "global" knowledge about f may help in identifying global minima
- For convex functions f, every local minimum (not necessarily unique) is also a global minimum
- For strictly convex function f, at most one global minimizer, and the only local minimizer is the global minimizer if it exists

Unconstrained Optimization: global vs. local optimizer

Local optimization methods

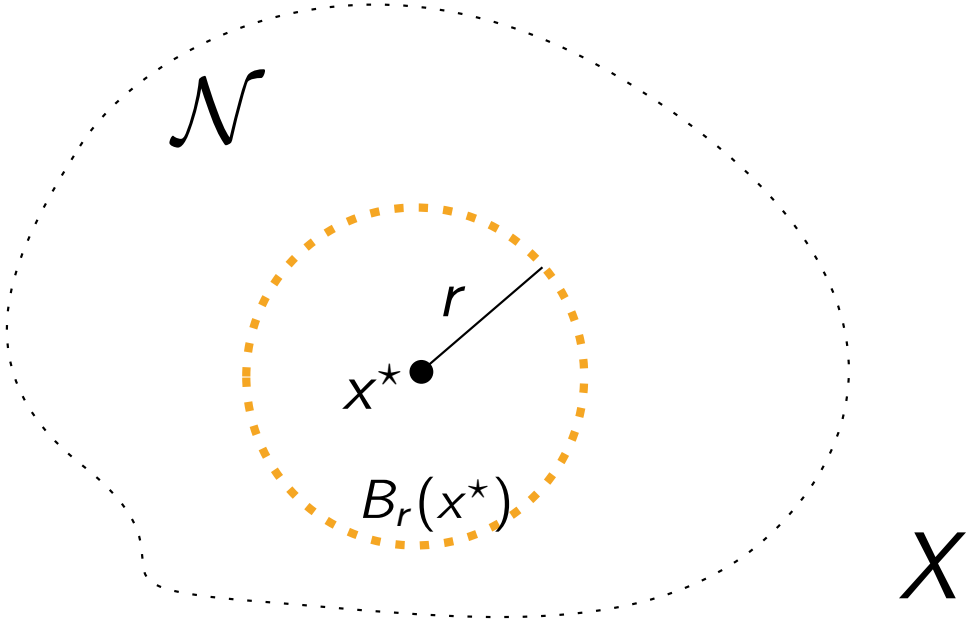
Algorithms able to find only a **local optimum** i.e., local minimum or maximum

- **Local minimum** : point $x^\star$ that achieves the smallest value of f in the neighborhood $\mathcal{N}$ of that point $x^\star$ (informally, an open set that contains that point)
- **Local maximum** : point $x^\star$ that achieves the largest value of f in the neighborhood $\mathcal{N}$ of that point $x^\star$ (informally, an open set that contains that point)

These algorithms often classified as providing **local optimization methods**

Unconstrained Optimization: global vs. local optimizer

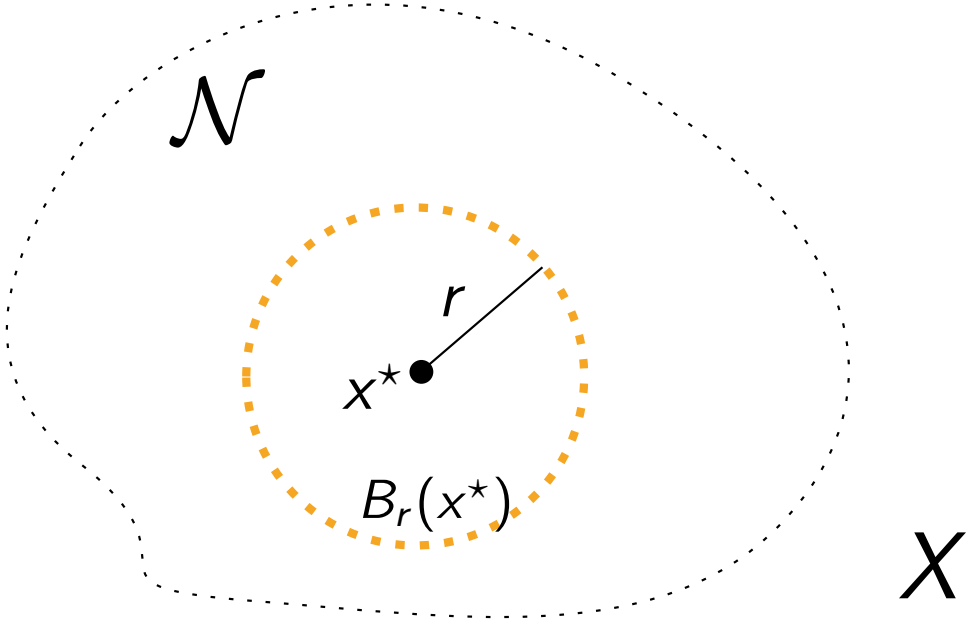
In a metric space (X, d), a subset $\mathcal{N} \subset X$ is a **neighborhood** of the point $x^\star$ if there exist an open ball $B(x^\star; r)$ with center $x^\star$ and radius $r > 0$ such that $B(x^\star; r) \equiv B_r(x^\star) = \{x \in X : d(x^\star, x) < r\}$ is contained in $\mathcal{N}$, i.e., $B_r(x^\star) \subseteq \mathcal{N}$ [1]



[1] Note : this definition restricts open sets to open balls the definition of the neighborhood of a point x as a set that contains an open set containing x.

Unconstrained Optimization: global vs. local optimizer

In a metric space (X, d), a subset $\mathcal{N} \subset X$ is a **neighborhood** of the point $x^\star$ if there exist an open ball $B(x^\star; r)$ with center $x^\star$ and radius $r > 0$ such that $B(x^\star; r) \equiv B_r(x^\star) = \{x \in X : d(x^\star, x) < r\}$ is contained in $\mathcal{N}$, i.e., $B_r(x^\star) \subseteq \mathcal{N}$ [1]



More general: a set N_x is a neighborhood of x if there exists an open set S such that $x \in S \subseteq N_x$. The set of all neighborhoods of x is called the complete system of neighborhoods of x and is denoted $\mathcal{N}_x$

[1] Note : this definition restricts open sets to open balls the definition of the neighborhood of a point x as a set that contains an open set containing x.

Unconstrained Optimization: global vs. local optimizer

Def: Local miminum

A point $x^\star$ is a **local minimum** of f, if there is a ball $B_r(x^\star)$ centered on $x^\star$ such that $f(x^\star) \leq f(x)$ for all $x \in B_r(x^\star) \cap \mathcal{C}$

Unconstrained Optimization: global vs. local optimizer

Def: Local miminum

A point $x^\star$ is a **local minimum** of f, if there is a ball $B_r(x^\star)$ centered on $x^\star$ such that $f(x^\star) \leq f(x)$ for all $x \in B_r(x^\star) \cap \mathcal{C}$

Def: Strict local miminum

A point $x^\star$ is a **strict local minimum** of f, if there is a ball $B_r(x^\star)$ centered on $x^\star$ such that $f(x^\star) < f(x)$ for all $x \in B_r(x^\star) \cap \mathcal{C}, x \neq x^\star$

Unconstrained Optimization: global vs. local optimizer

Def: Local miminum

A point $x^\star$ is a **local minimum** of f, if there is a ball $B_r(x^\star)$ centered on $x^\star$ such that $f(x^\star) \leq f(x)$ for all $x \in B_r(x^\star) \cap \mathcal{C}$

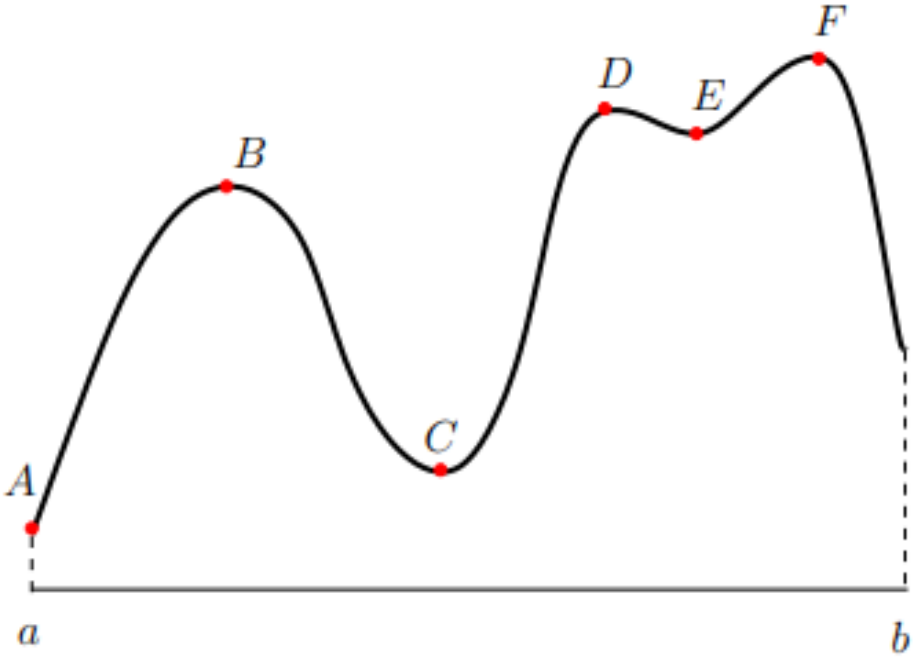
Def: Strict local miminum

A point $x^\star$ is a **strict local minimum** of f, if there is a ball $B_r(x^\star)$ centered on $x^\star$ such that $f(x^\star) < f(x)$ for all $x \in B_r(x^\star) \cap \mathcal{C}, x \neq x^\star$

<u>Note</u>: the definitions of maximum, strict maximum, local maximum, strict local maximum are similar (change $\leq$ in $\geq$ and $<$ in $>$)

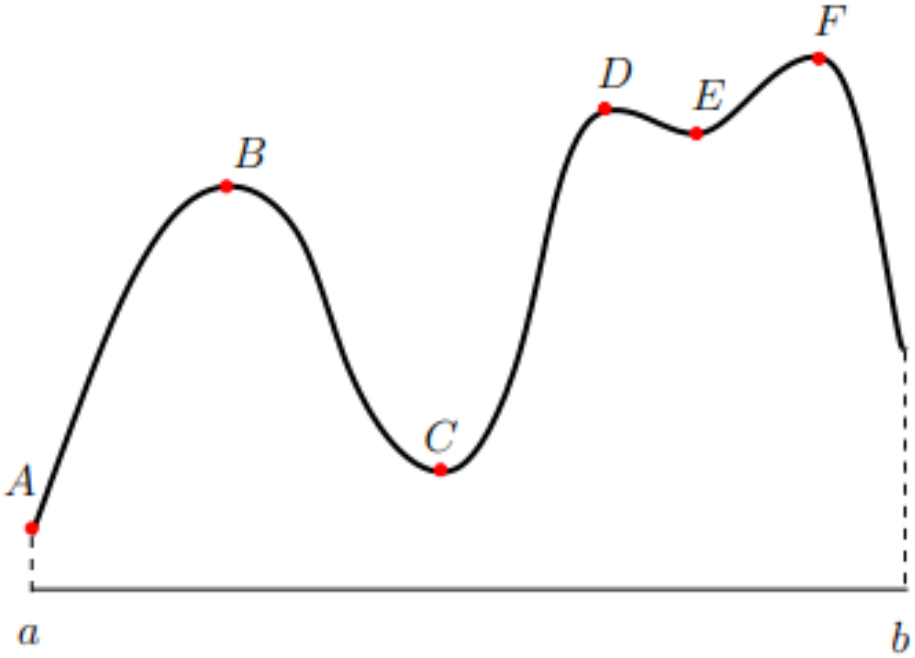
Unconstrained Optimization: global vs. local optimizer

Example: Let $\mathcal{C} = [a, b[\in \mathbb{R}$, the graph of the function f over that interval



Unconstrained Optimization: global vs. local optimizer

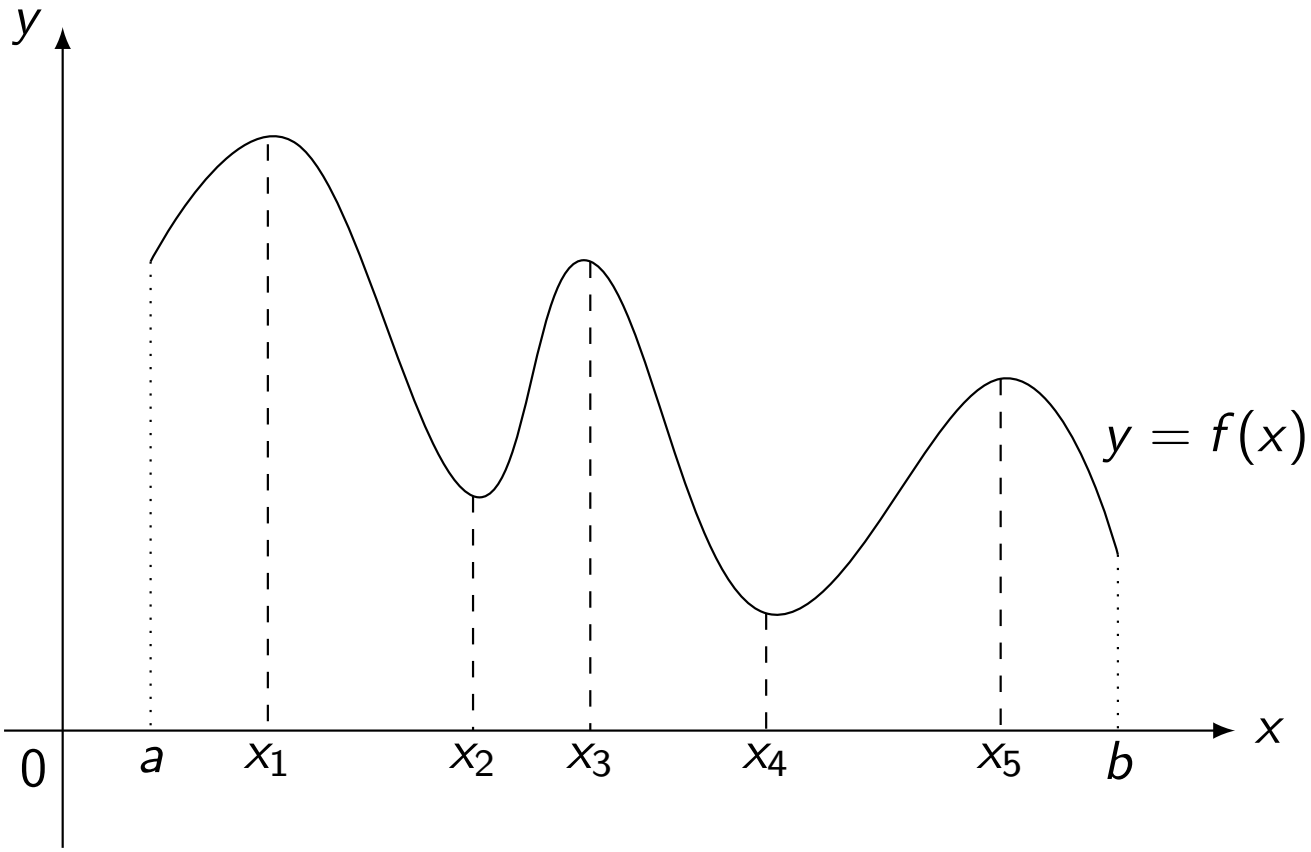
Example: Let $\mathcal{C} = [a, b[\in \mathbb{R}$, the graph of the function f over that interval



- $B = f(x_1)$ and $D = f(x_3)$: x_1 and x_3 are local but not global maxima of f
- $C = f(x_2)$ and $E = f(x_4)$: x_2 and x_4 are local but not global minima of f
- $F = f(x_5)$: x_5 is global maximum of f
- $A = f(a)$: a is global minimum of f

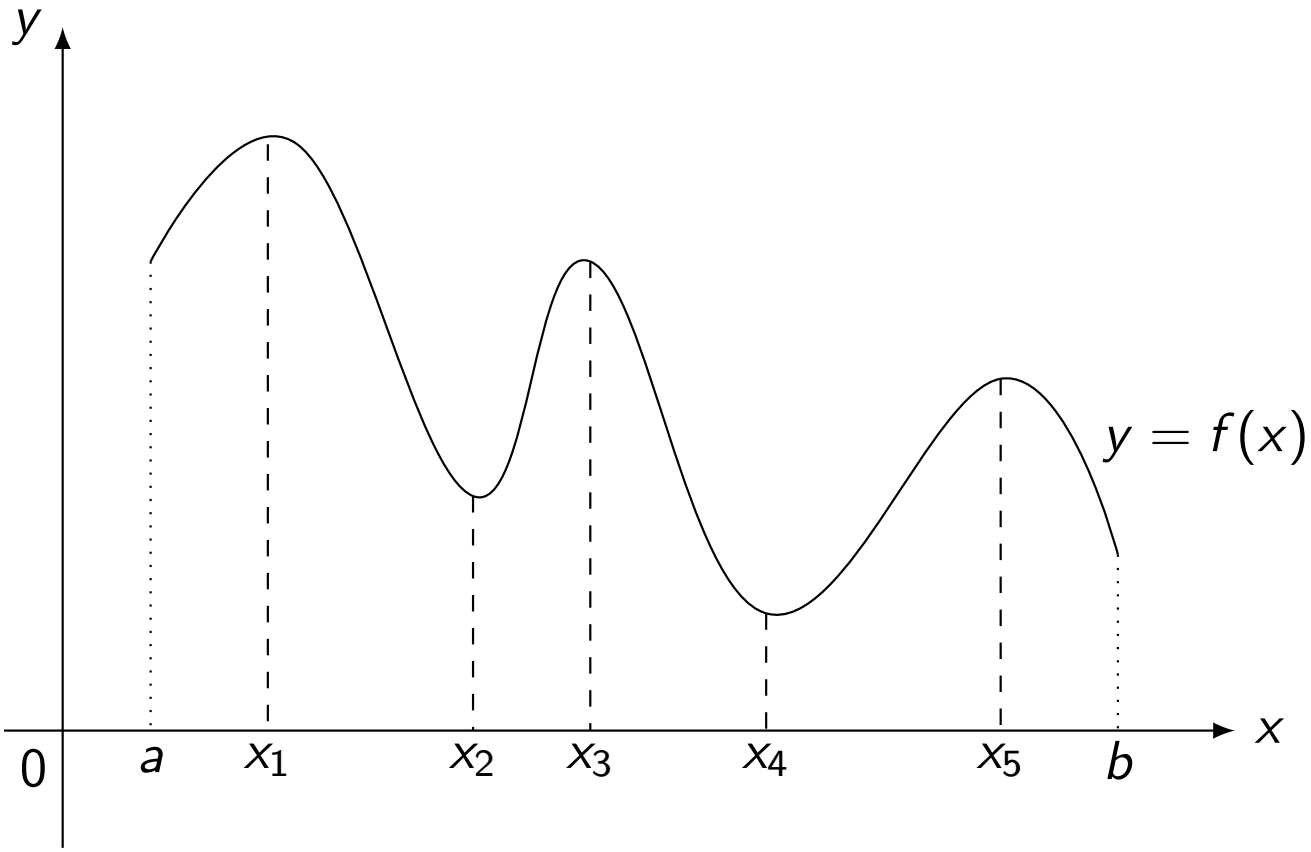
Unconstrained Optimization: global vs. local optimizer

Example function $f : [a, b] \in \mathbb{R} \to \mathbb{R}$



Unconstrained Optimization: global vs. local optimizer

Example function $f : [a, b] \in \mathbb{R} \to \mathbb{R}$



- x_2 and x_4 are local but not global minima of f
- x_3 and x_5 are local but not global maxima of f
- x_1 is global maximum of f
- x_4 is global minimum of f

Unconstrained Optimization: taxonomy

Consequence: in unconstrained optimization

- the (first) main distinction between classes of problems comes from the **convexity** (or not) of the set $\mathcal{C}$ and the **convexity** (or not) of the function f
- the (second) main distinction comes from the (continuity and the) class of **differentiability** of the function f

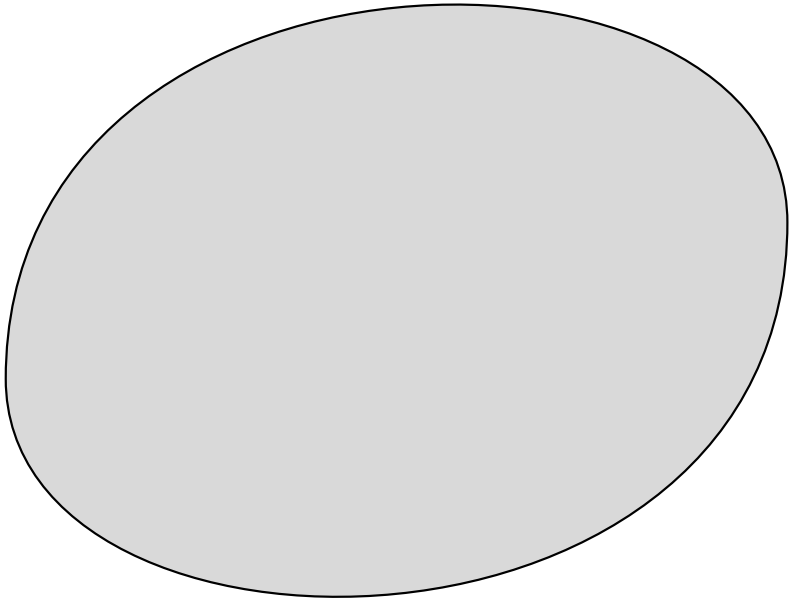
Convex sets

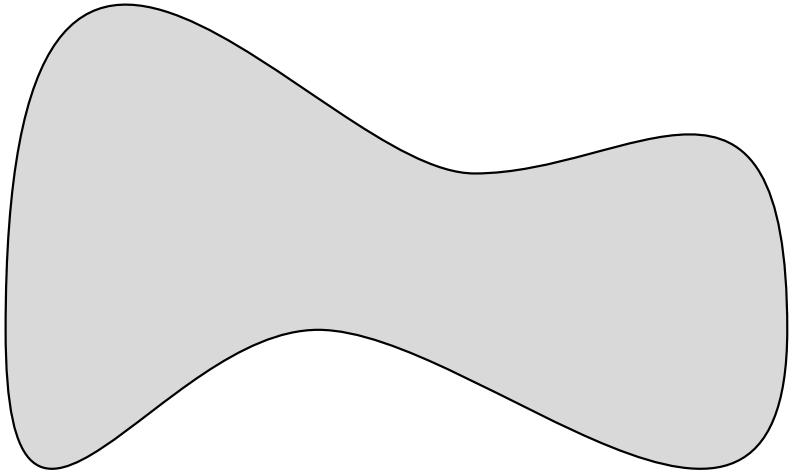
- Recall that the term "convex" can be applied both to sets and to functions

Convex sets

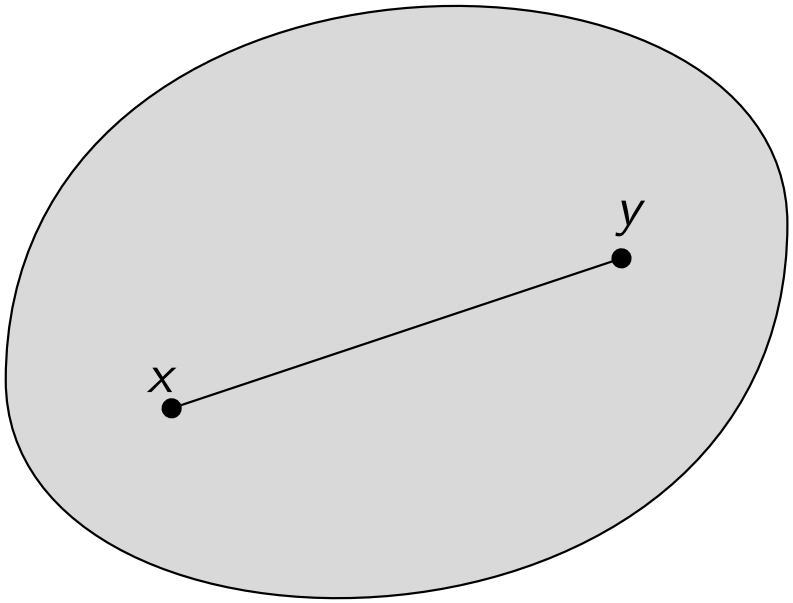
- Recall that the term "convex" can be applied both to sets and to functions
- **Intuitively,** a set $\mathcal{C} \subseteq \mathbb{R}^n$ is convex if the straight line segment connecting any two points in $\mathcal{C}$ lies entirely inside $\mathcal{C}$

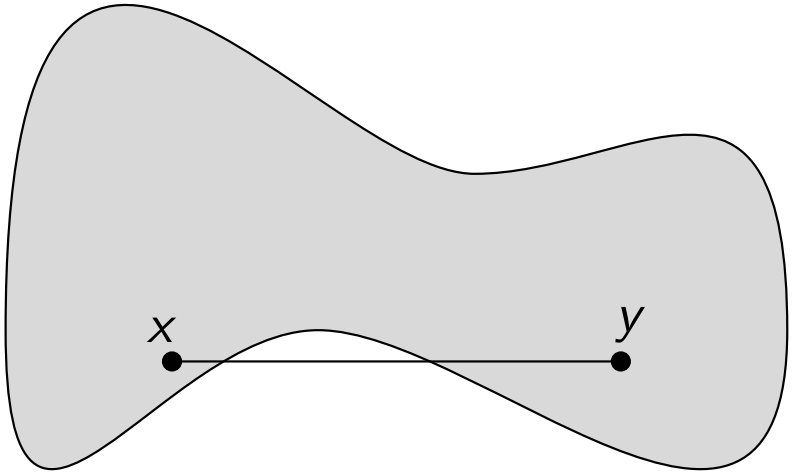
Convex sets



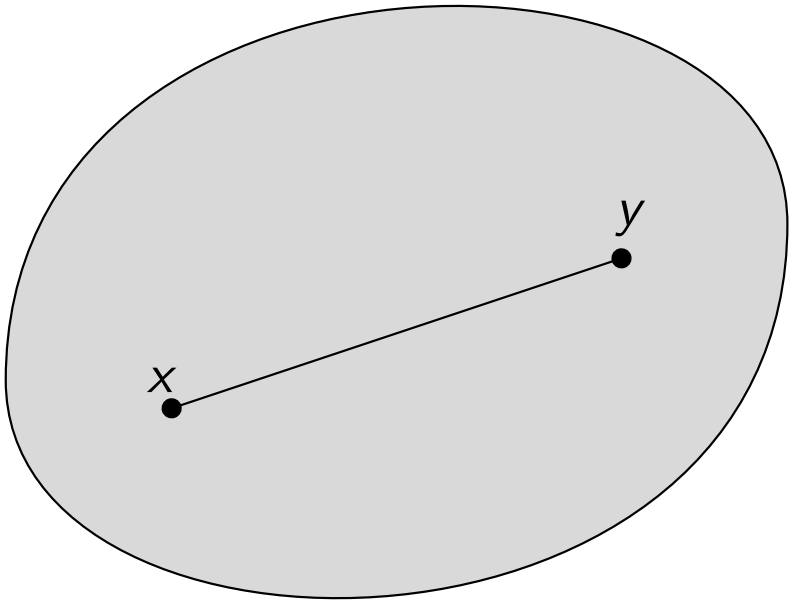


Convex sets

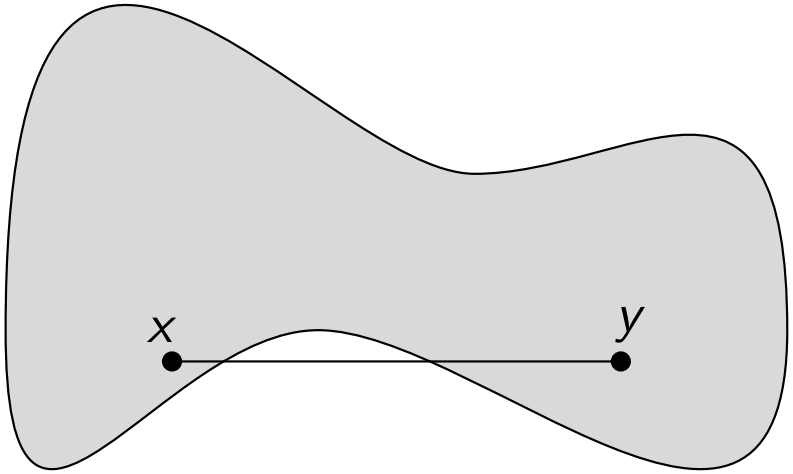




Convex sets



Convex set



Non-convex set

Convex sets: definition

Def: convex set

A set $\mathcal{C} \subseteq \mathbb{R}^n$ is convex iff, for any two points $x, y \in \mathcal{C} \subseteq \mathbb{R}^n$ and any $\alpha \in [0, 1]$, we have $\alpha x + (1 - \alpha) y \in \mathcal{C}$

Convex sets: definition

Def: convex set

A set $\mathcal{C} \subseteq \mathbb{R}^n$ is convex iff, for any two points $x, y \in \mathcal{C} \subseteq \mathbb{R}^n$ and any $\alpha \in [0, 1]$, we have $\alpha x + (1 - \alpha) y \in \mathcal{C}$

Examples of convex sets

- The empty set $\emptyset$, the singleton set $\{x\}$, and the complete space $\mathbb{R}^n$

Convex sets: definition

Def: convex set

A set $\mathcal{C} \subseteq \mathbb{R}^n$ is convex iff, for any two points $x, y \in \mathcal{C} \subseteq \mathbb{R}^n$ and any $\alpha \in [0, 1]$, we have $\alpha x + (1 - \alpha) y \in \mathcal{C}$

Examples of convex sets

- The empty set $\emptyset$, the singleton set $\{x\}$, and the complete space $\mathbb{R}^n$
- A subset $\mathcal{I}$ of $\mathbb{R}$ is an interval if $x, y \in \mathcal{I}$ and $x < z < y$ imply $z \in \mathcal{I}$.

  A subset of $\mathbb{R}$ is convex if and only if it is an interval

Convex sets: definition

Def: convex set

A set $\mathcal{C} \subseteq \mathbb{R}^n$ is convex iff, for any two points $x, y \in \mathcal{C} \subseteq \mathbb{R}^n$ and any $\alpha \in [0, 1]$, we have $\alpha x + (1 - \alpha) y \in \mathcal{C}$

Examples of convex sets

- The empty set $\emptyset$, the singleton set $\{x\}$, and the complete space $\mathbb{R}^n$
- A subset $\mathcal{I}$ of $\mathbb{R}$ is an interval if $x, y \in \mathcal{I}$ and $x < z < y$ imply $z \in \mathcal{I}$.

  A subset of $\mathbb{R}$ is convex if and only if it is an interval
- The unit ball of a normed vector space: $\{x \in \mathbb{R}^n \mid \|x\|_p \leq 1, p \geq 1\}$

Convex sets: definition

Def: convex set

A set $\mathcal{C} \subseteq \mathbb{R}^n$ is convex iff, for any two points $x, y \in \mathcal{C} \subseteq \mathbb{R}^n$ and any $\alpha \in [0, 1]$, we have $\alpha x + (1 - \alpha) y \in \mathcal{C}$

Examples of convex sets

- The empty set $\emptyset$, the singleton set $\{x\}$, and the complete space $\mathbb{R}^n$
- A subset $\mathcal{I}$ of $\mathbb{R}$ is an interval if $x, y \in \mathcal{I}$ and $x < z < y$ imply $z \in \mathcal{I}$.

  A subset of $\mathbb{R}$ is convex if and only if it is an interval
- The unit ball of a normed vector space: $\{x \in \mathbb{R}^n \mid \|x\|_p \leq 1, p \geq 1\}$
- The ellipsoid $\{x \in \mathbb{R}^n \mid (x - x_c)^\top Q (x - x_c) \leq r\}$, centered on $x_c \in \mathbb{R}^n$, $r \in \mathbb{R}$, Q positive definite symmetric matrix $\in \mathbb{R}^{n \times n}$

Convex sets: definition

Def: convex set

A set $\mathcal{C} \subseteq \mathbb{R}^n$ is convex iff, for any two points $x, y \in \mathcal{C} \subseteq \mathbb{R}^n$ and any $\alpha \in [0, 1]$, we have $\alpha x + (1 - \alpha) y \in \mathcal{C}$

Examples of convex sets

- The empty set $\emptyset$, the singleton set $\{x\}$, and the complete space $\mathbb{R}^n$
- A subset $\mathcal{I}$ of $\mathbb{R}$ is an interval if $x, y \in \mathcal{I}$ and $x < z < y$ imply $z \in \mathcal{I}$.

  A subset of $\mathbb{R}$ is convex if and only if it is an interval
- The unit ball of a normed vector space: $\{x \in \mathbb{R}^n \mid \|x\|_p \leq 1, p \geq 1\}$
- The ellipsoid $\{x \in \mathbb{R}^n \mid (x - x_c)^\top Q(x - x_c) \leq r\}$, centered on $x_c \in \mathbb{R}^n$, $r \in \mathbb{R}$, Q positive definite symmetric matrix $\in \mathbb{R}^{n \times n}$
- The set defined by linear inequalities (half-spaces), that is, $\{x \in \mathbb{R}^n \mid Ax \leq b\}$, where the matrix $A \in \mathbb{R}^{m \times n}$, and the vector $b \in \mathbb{R}^m$

Convex vs Strictly convex set

Def: convex set

A set $\mathcal{C} \subseteq \mathbb{R}^n$ is convex iff, for any two points $x, y \in \mathcal{C} \subseteq \mathbb{R}^n$ and any $\alpha \in [0, 1]$, we have $\alpha x + (1 - \alpha)y \in \mathcal{C}$

Def: strictly convex set

A set $\mathcal{C} \subseteq \mathbb{R}^n$ is strictly convex iff, for any two points $x, y \in \mathcal{C} \subseteq \mathbb{R}^n$, $x \neq y$ and any $\alpha \in]0, 1[$, we have $\alpha x + (1 - \alpha)y \in int(\mathcal{C})$

Convex vs Strictly convex set

Def: convex set

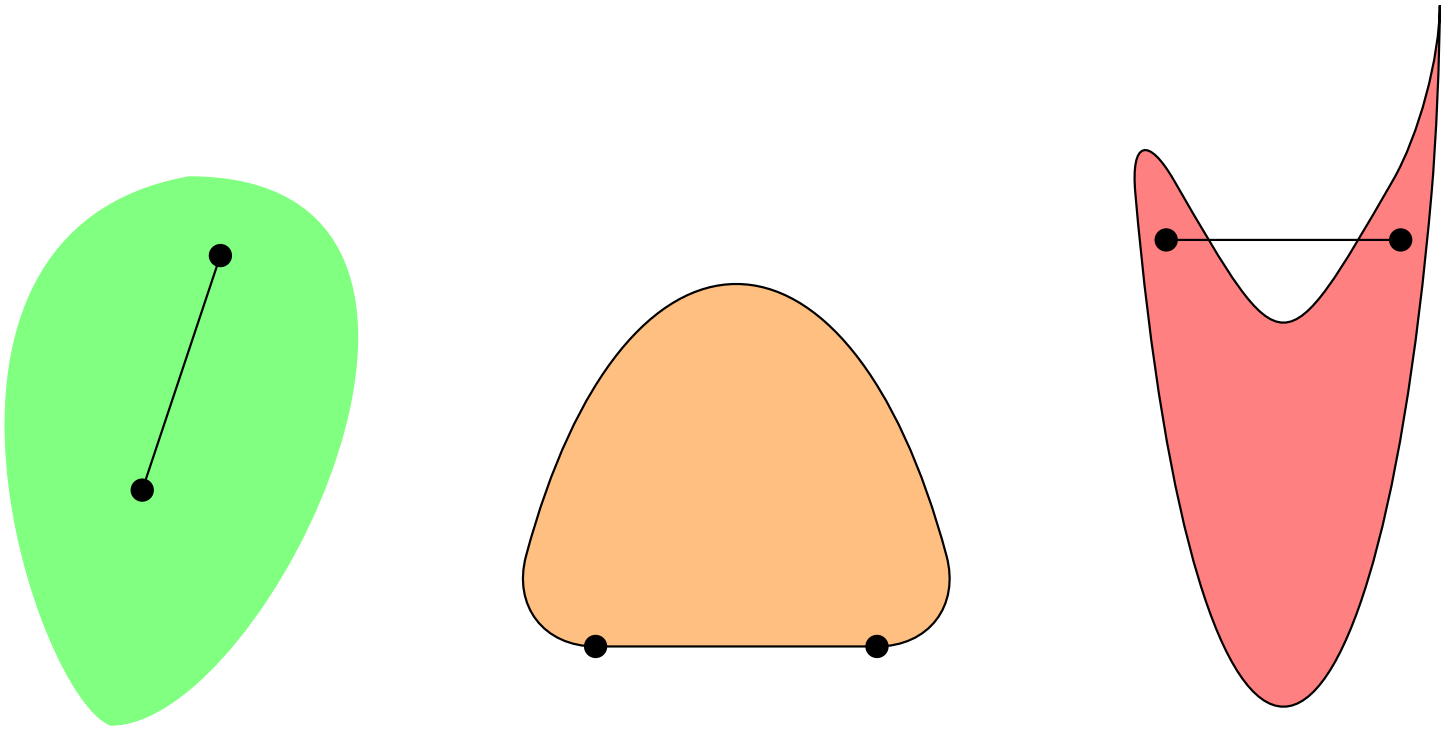
A set $\mathcal{C} \subseteq \mathbb{R}^n$ is convex iff, for any two points $x, y \in \mathcal{C} \subseteq \mathbb{R}^n$ and any $\alpha \in [0, 1]$, we have $\alpha x + (1 - \alpha)y \in \mathcal{C}$

Def: strictly convex set

A set $\mathcal{C} \subseteq \mathbb{R}^n$ is strictly convex iff, for any two points $x, y \in \mathcal{C} \subseteq \mathbb{R}^n$, $x \neq y$ and any $\alpha \in]0, 1[$, we have $\alpha x + (1 - \alpha)y \in int(\mathcal{C})$

- Let D be subset of X. A point $x \in X$ is an interior point of D if there exists $r > 0$ such that the ball $B_r(x, r)$ is completely contained in D ($B_r(x) \subset D$)
- The interior of D ($int(D)$) denotes the set of all of interior points of D

Convex sets: example



Strictly convex Convex Nonconvex

Convex function: definition

Def: Convex function

The function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if the domain of the function f, $\mathrm{dom}(f)$ is a convex set and for every two points $(x, y) \in \mathrm{dom}(f) \times \mathrm{dom}(f)$ and $\alpha \in [0, 1]$, the following inequality is satisfied:

$$f(\alpha x + (1 - \alpha)y) \leq \alpha f(x) + (1 - \alpha)f(y) \tag{2}$$

Generalization

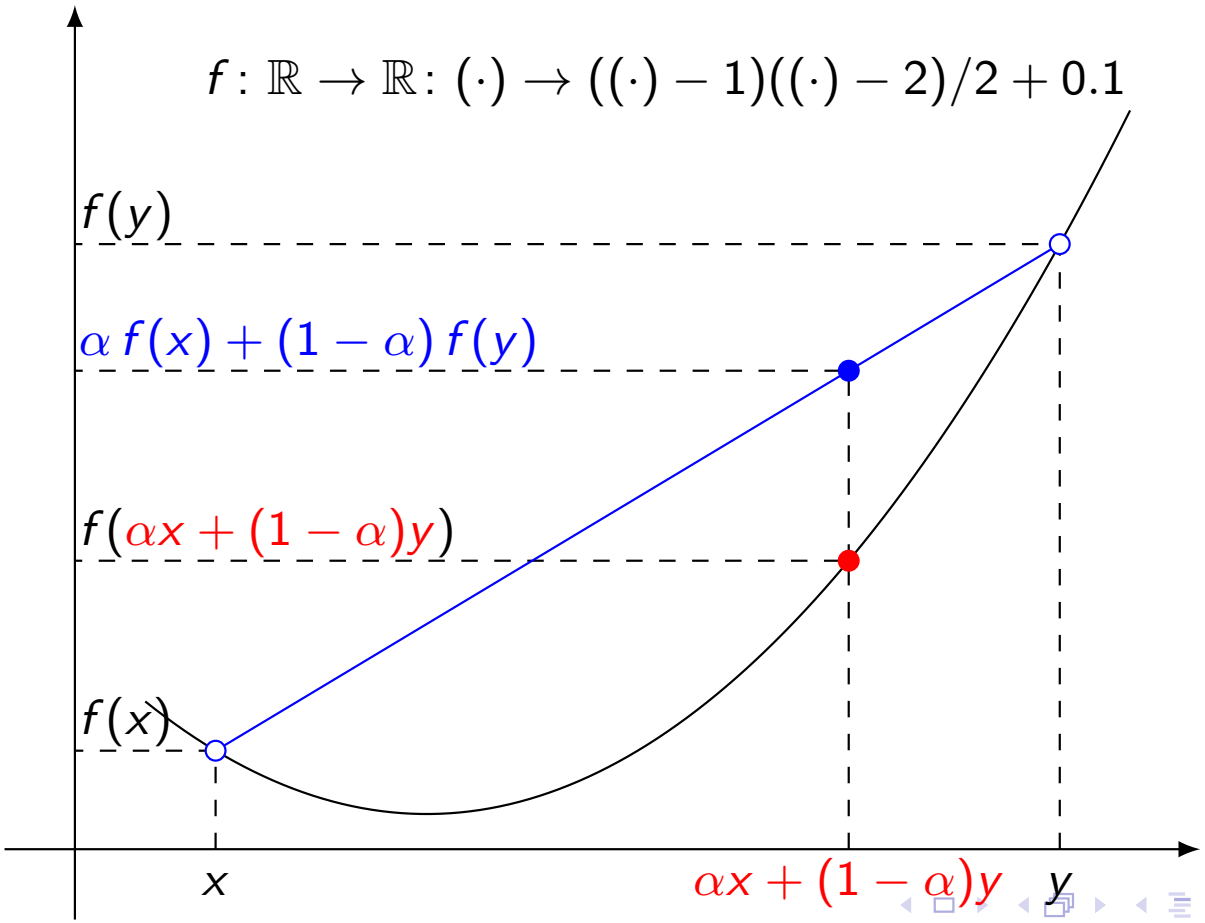
The function $f : \mathcal{C} \subseteq \mathbb{R}^n \to \mathbb{R}$ is convex if $\mathcal{C}$ is a convex set and for every two points $(x, y) \in \mathcal{C} \times \mathcal{C}$ and $\alpha \in [0, 1]$, the following inequality is satisfied:

$$f(\alpha x + (1 - \alpha)y) \leq \alpha f(x) + (1 - \alpha)f(y) \tag{2}$$

Convex function: graphical interpretation

Graphical interpretation: the chord (i.e., line segment) between any two points on the graph of f lies above the graph of f

$$f(\underbrace{\alpha x + (1-\alpha)y}_{\text{convex combination}}) \leq \underbrace{\alpha f(x) + (1-\alpha)f(y)}_{\text{chord}}$$



Convex function: examples

Examples of convex functions

- **Linear** functions $f : \mathbb{R}^n \to \mathbb{R} : x \to c^\top x$, for any constant vector $c \in \mathbb{R}^n$

Convex function: examples

Examples of convex functions

- **Linear** functions $f : \mathbb{R}^n \to \mathbb{R} : x \to c^\top x$, for any constant vector $c \in \mathbb{R}^n$
- **Affine** functions $f : \mathbb{R}^n \to \mathbb{R} : x \to c^\top x + b$, for any $c \in \mathbb{R}^n$ and scalar b; affine (and therefore also linear) functions are both convex and concave
- Affine mappings $f : \mathbb{R}^n \to \mathbb{R}^m : x \to Ax + b$, for any matrix $A \in \mathbb{R}^{m \times n}$ and vector $b \in \mathbb{R}^m$; affine (and therefore also linear) mappings are both convex and concave

Convex function: examples

Examples of convex functions

- **Linear** functions $f : \mathbb{R}^n \to \mathbb{R} : x \to c^\top x$, for any constant vector $c \in \mathbb{R}^n$
- **Affine** functions $f : \mathbb{R}^n \to \mathbb{R} : x \to c^\top x + b$, for any $c \in \mathbb{R}^n$ and scalar b; affine (and therefore also linear) functions are both convex and concave
- Affine mappings $f : \mathbb{R}^n \to \mathbb{R}^m : x \to Ax + b$, for any matrix $A \in \mathbb{R}^{m \times n}$ and vector $b \in \mathbb{R}^m$; affine (and therefore also linear) mappings are both convex and concave
- **Quadratic** functions $f : \mathbb{R}^n \to \mathbb{R} : x \to x^\top Qx$, where $Q \in \mathbb{R}^{n \times n}$ is a (symmetric) positive semi-definite matrix
- **max** function : $f : \mathbb{R}^n \to \mathbb{R} : x \to \max(x) = \max\{x_1, x_2, \ldots, x_n\}$
- **Power** functions of absolute value: $f : \mathbb{R} \to \mathbb{R} : x \to |x|^p$ on $\mathbb{R}$, for $p \geq 1$
- For any $p \geq 1$, the **p-norm** on $\mathbb{R}^n$ defined by

$$\|(x_1, \ldots, x_n)\|_p = \left(|x_1|^p + \ldots + |x_n|^p\right)^{1/p} = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p} \text{ is convex}$$

Convex functions: epigraph

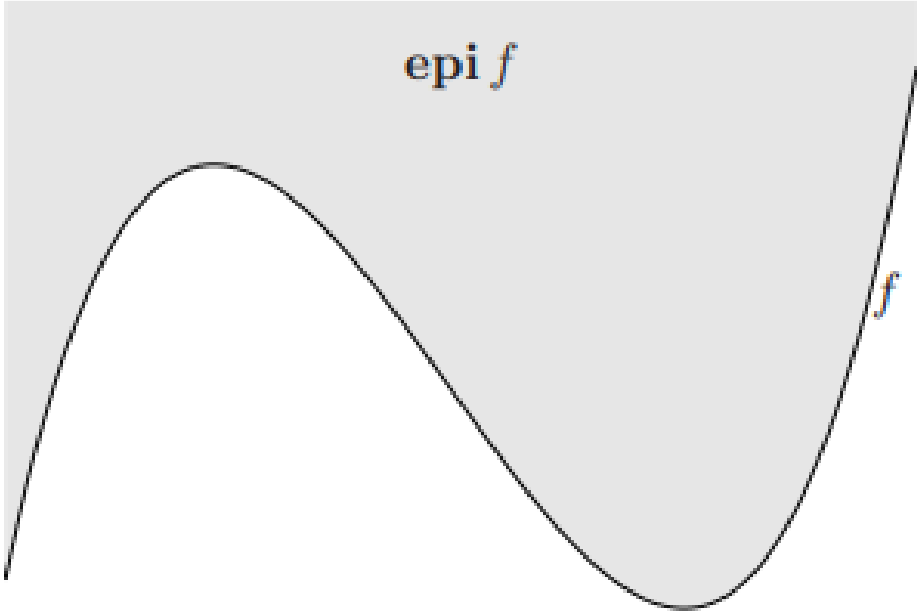
Def: Epigraph

- The **epigraph** of f is the set $epi(f) = \{(x, \mu) \in \mathbb{R}^n \times \mathbb{R} : f(x) \leq \mu\}$

Convex functions: epigraph

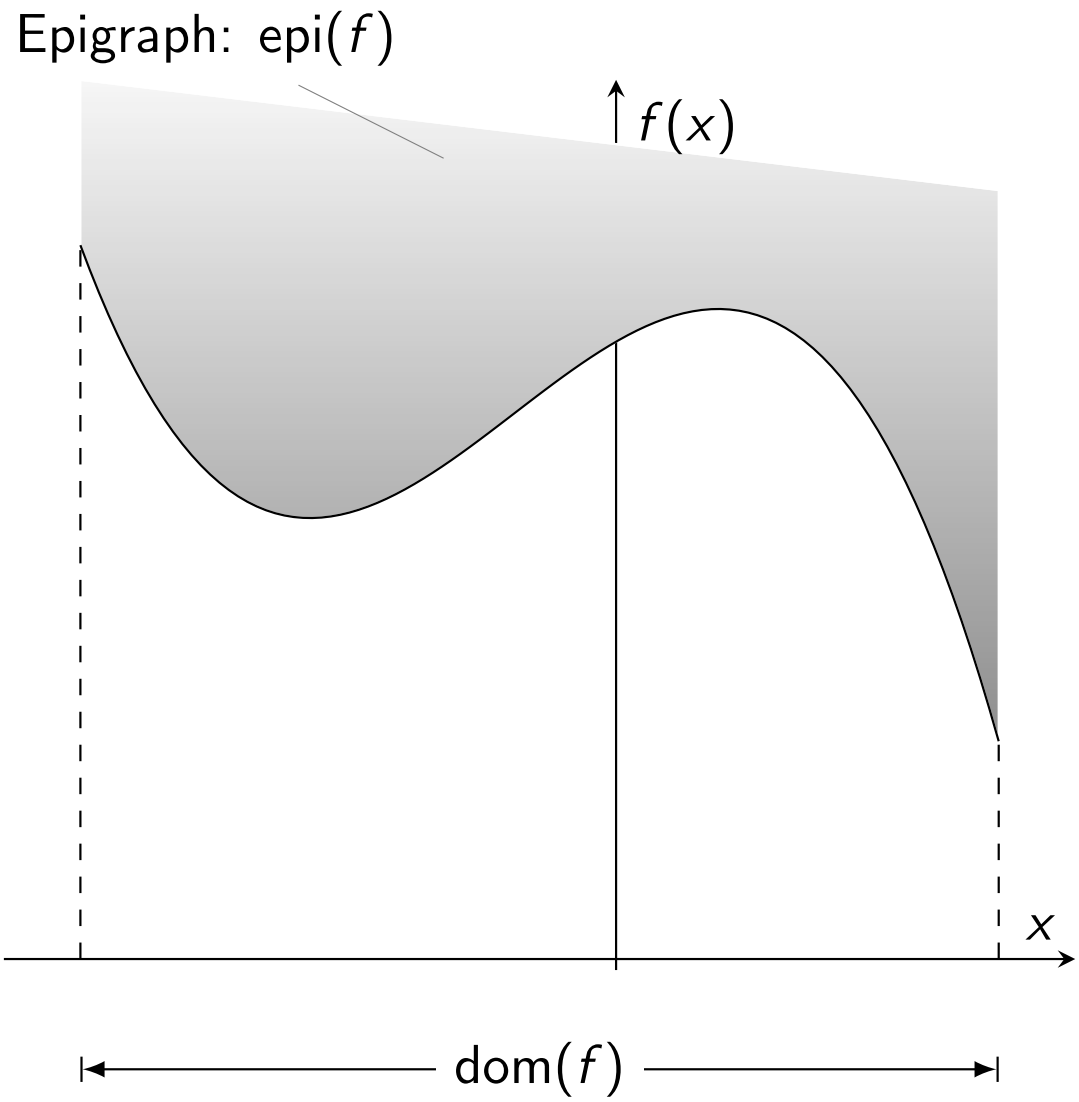
Def: Epigraph

- The **epigraph** of f is the set $epi(f) = \{(x, \mu) \in \mathbb{R}^n \times \mathbb{R} : f(x) \leq \mu\}$
- The convexity of the set $epi(f)$ is equivalent to the convexity of the function f: a function is convex if and only if its epigraph $epi(f)$ is convex

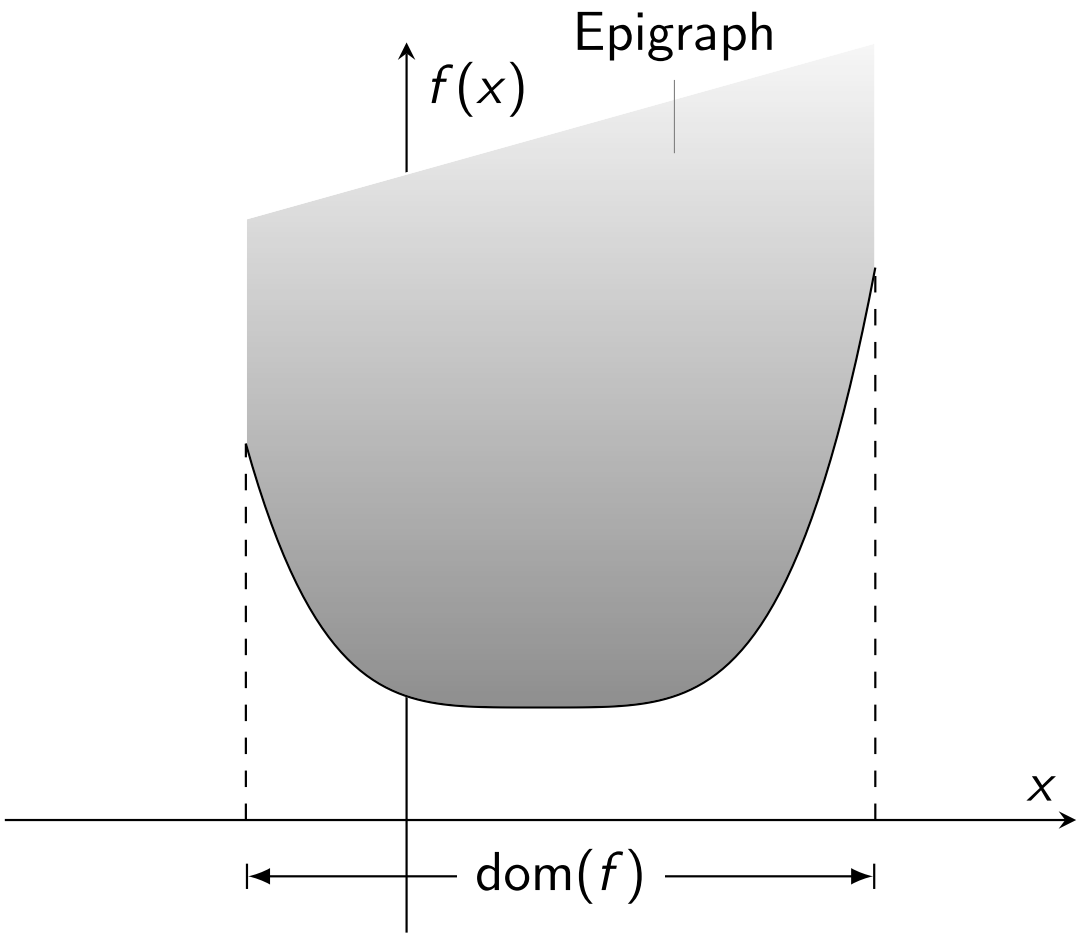


Epigraph of a function f (shaded area). The lower boundary (darker) is the graph of f

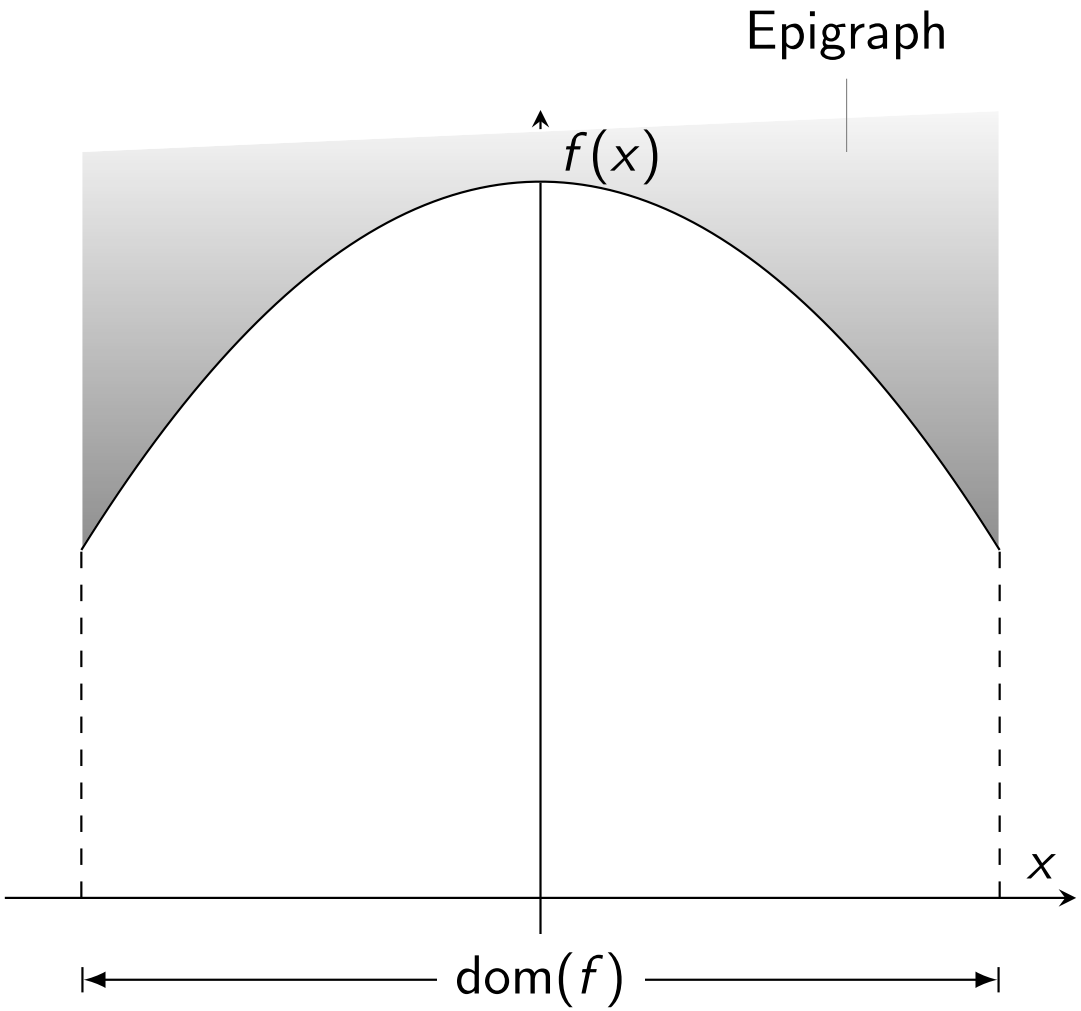
Epigraph of a function $f : -ax^3 + bx^2 - cx + d$



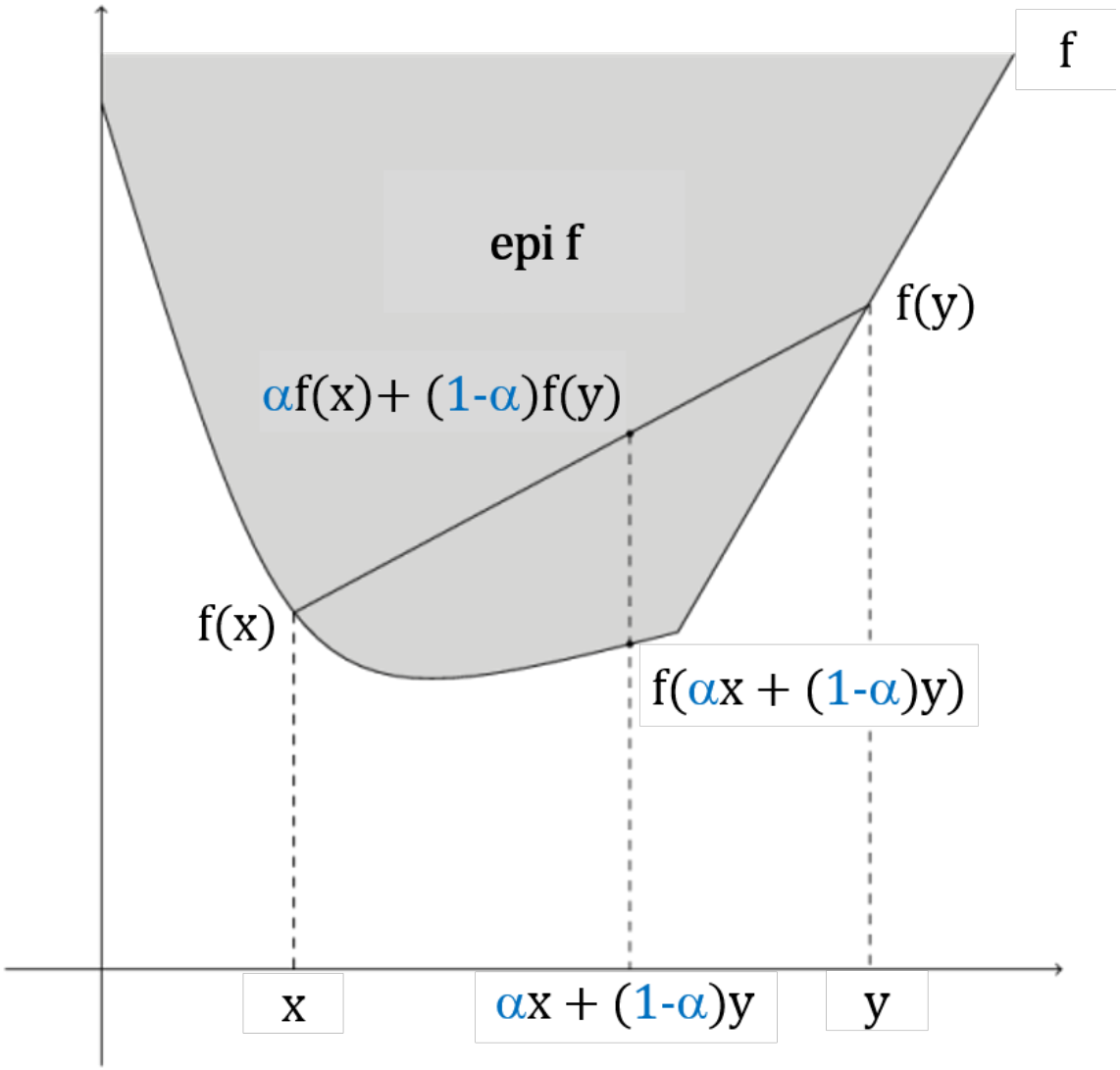
Epigraph of a function $f : (x-a)^2(x-a)^2 + b$



Epigraph of a function $f : -ax^2 + b$



Convex function and Epigraph



Concave function and Hypograph

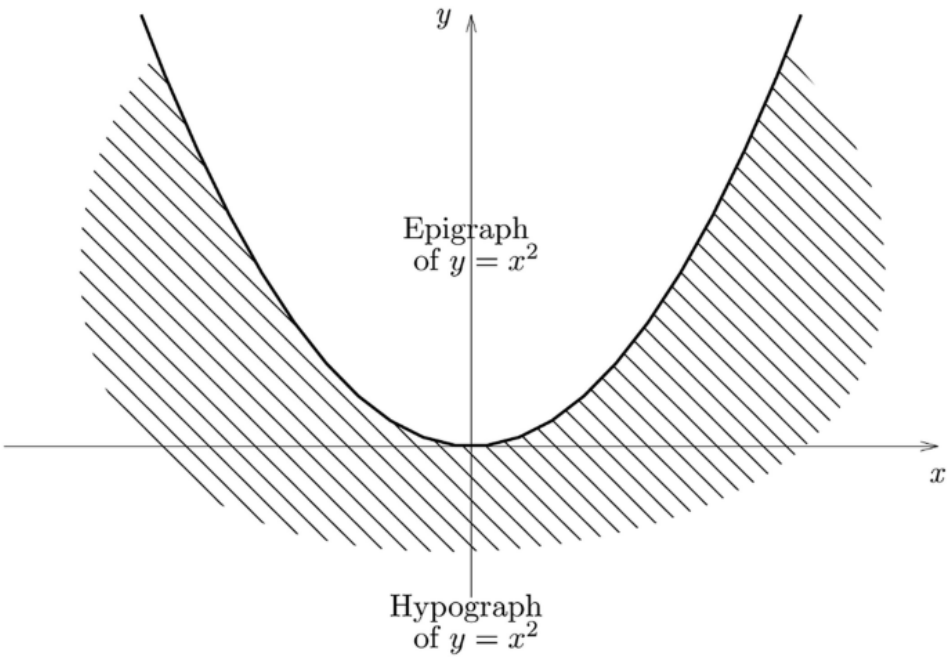
Def: Hypograph

- The **hypograph** of f is the set $hyp(f) = \{(x, \mu) \in \mathbb{R}^n \times \mathbb{R} : \mu \leq f(x)\}$
- The convexity of the set $hyp(f)$ is equivalent to the concavity of the function f: a function is concave if and only if its hypograph $hyp(f)$ is convex

Concave function and Hypograph

Def: Hypograph

- The **hypograph** of f is the set $hyp(f) = \{(x, \mu) \in \mathbb{R}^n \times \mathbb{R} : \mu \leq f(x)\}$
- The convexity of the set $hyp(f)$ is equivalent to the concavity of the function f: a function is concave if and only if its hypograph $hyp(f)$ is convex



Properties

The intersection

$$epi(f) \cap hyp(f) = \{(x, y) \in \mathbb{R}^n \times \mathbb{R} | y = f(x)\}$$

is the **graph of the function** f

Properties

The intersection

$$epi(f) \cap hyp(f) = \{(x, y) \in \mathbb{R}^n \times \mathbb{R} | y = f(x)\}$$

is the **graph of the function** f

- A function f is convex over a convex set **iff** the function $-f$ (reflection) is a concave function over the set
- A function f is concave over a convex set **iff** the function $-f$ (reflection) is a convex function over the set

Convex vs. Strictly convex

Def: Strictly convex

The function $f : \mathcal{C} \subseteq \mathbb{R}^n \to \mathbb{R}$ is **strictly** convex if $\mathcal{C}$ is a convex set and for every two distinct points $x, y \in \mathcal{C}$ and $\alpha \in]0, 1[$, the following strict inequality is satisfied:

$$f(\alpha x + (1 - \alpha)y) < \alpha f(x) + (1 - \alpha)f(y) \tag{3}$$

Convex vs. Strictly convex

Def: Strictly convex

The function $f : \mathcal{C} \subseteq \mathbb{R}^n \to \mathbb{R}$ is **strictly** convex if $\mathcal{C}$ is a convex set and for every two distinct points $x, y \in \mathcal{C}$ and $\alpha \in]0, 1[$, the following strict inequality is satisfied:

$$f(\alpha x + (1 - \alpha)y) < \alpha f(x) + (1 - \alpha)f(y) \tag{3}$$

Geometrically: a strictly convex function f is a function for which the straight line segment between any pair of points on $gra(f)$ lies above $gra(f)$ except for intersection points between straight line and $gra(f)$

... **intuitively** f is convex and has greater curvature than a linear function

Convex vs. Strictly convex

Def: Strictly convex

The function $f : \mathcal{C} \subseteq \mathbb{R}^n \to \mathbb{R}$ is **strictly** convex if $\mathcal{C}$ is a convex set and for every two distinct points $x, y \in \mathcal{C}$ and $\alpha \in]0, 1[$, the following strict inequality is satisfied:

$$f(\alpha x + (1 - \alpha)y) < \alpha f(x) + (1 - \alpha)f(y) \tag{3}$$

Geometrically: a strictly convex function f is a function for which the straight line segment between any pair of points on $gra(f)$ lies above $gra(f)$ except for intersection points between straight line and $gra(f)$

... **intuitively** f is convex and has greater curvature than a linear function

If the function f is strictly convex, then f is convex (this is obvious from the definition) but the converse is not true (e.g., see below example $f(x) = x, x \in \mathbb{R}$)

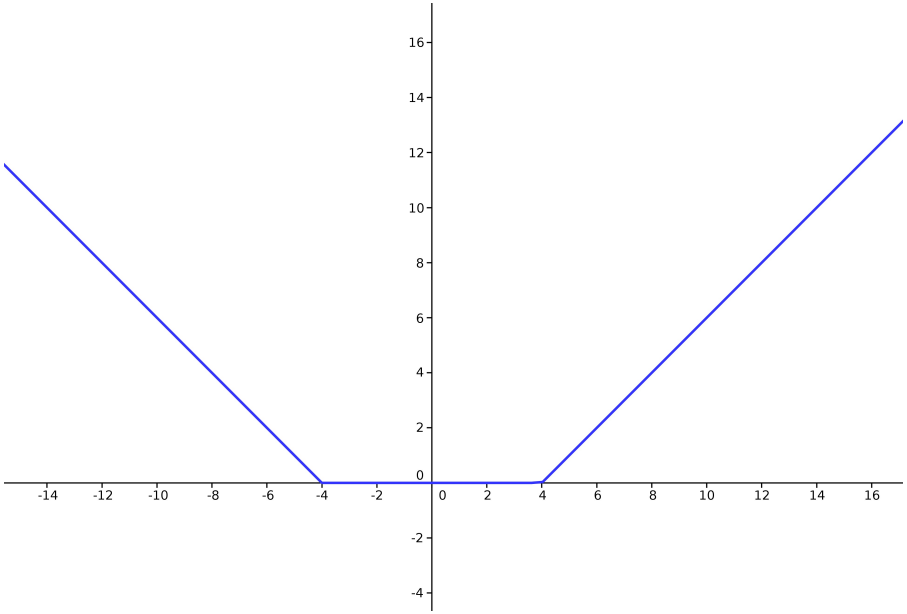
Convex vs. Strictly convex: example

Geometrically: a strictly convex function f is a function for which the straight line segment between any pair of points on $gra(f)$ lies above $gra(f)$ except for intersection points between straight line and $gra(f)$

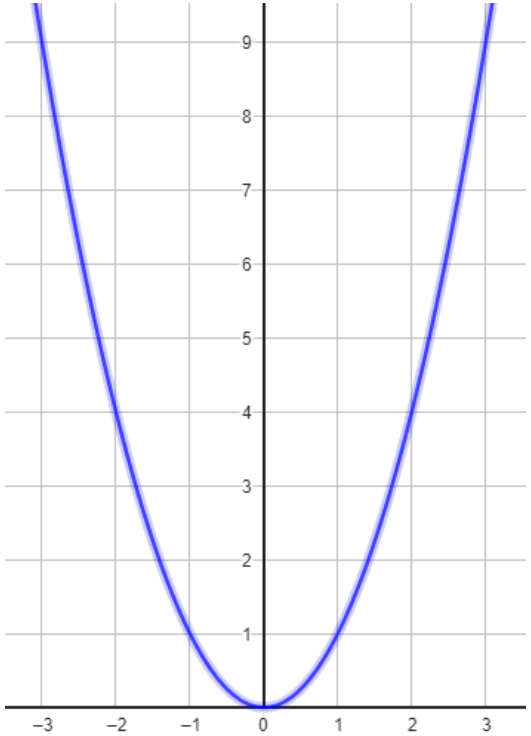
Convex vs. Strictly convex: example

Geometrically: a strictly convex function f is a function for which the straight line segment between any pair of points on $gra(f)$ lies above $gra(f)$ except for intersection points between straight line and $gra(f)$

$$f(x) = \begin{cases} -x-4, & \text{if } x \leq -4 \\ 0, & \text{if } -4 < x < 4 \\ x-4, & \text{if } x \geq 4 \end{cases}$$



$$f(x) = x^2$$



Convex vs. Strictly convex

- Every affine function $f : \mathbb{R} \to \mathbb{R} \colon x \to ax + b$ is convex –not strictly convex
  - Proof:

$$\begin{aligned} f((1-\alpha)x_1 + \alpha x_2) &= a((1-\alpha)x_1 + \alpha x_2) + b \\ &= a((1-\alpha)x_1 + \alpha x_2) + ((1-\alpha) + \alpha)b \\ &= (1-\alpha)(ax_1 + b) + \alpha(ax_2 + b) \\ &= (1-\alpha)f(x_1) + \alpha f(x_2) \end{aligned} \tag{4}$$

  - Implication: impossible to replace inequality $\leq$ with strict inequality $<$
- Function $f : \mathbb{R} \to \mathbb{R} \colon x \to |x|$ is convex –not strictly convex
  - Proof:

$$\begin{aligned} f((1-\alpha)x_1 + \alpha x_2) &= |(1-\alpha)x_1 + \alpha x_2| \\ &\leq |(1-\alpha)x_1| + |\alpha x_2| \\ &= (1-\alpha)\,|x_1| + \alpha\,|x_2| \\ &= (1-\alpha)f(x_1) + \alpha f(x_2) \end{aligned} \tag{5}$$

  - Implication: impossible to replace inequality $\leq$ with strict inequality $<$

Convex vs. Strictly convex

Function $f : \mathbb{R} \to \mathbb{R} \colon x \to x^2$ is strictly convex

- Proof:
  - Expand expression of f

$$\begin{aligned} f((1-\alpha)x_1 + \alpha x_2) &= \big((1-\alpha)x_1 + \alpha x_2\big)^2 \\ &= (1-\alpha)^2 x_1^2 + 2\alpha(1-\alpha)x_1 x_2 + \alpha^2 x_2^2 \end{aligned} \tag{6}$$

  - By definition of strict convexity, $x_1 \neq x_2$; thus, $(x_1 - x_2)^2 > 0$
    Implying in turn that $x_1^2 + x_2^2 > 2x_1x_2$
  - Then, by substitution in the RHS of (6)

$$\begin{aligned} f((1-\alpha)x_1 + \alpha x_2) &< (1-\alpha)^2 x_1^2 + \alpha(1-\alpha)(x_1^2 + x_2^2) + \alpha^2 x_2^2 \\ &= (1-\alpha)x_1^2 + \alpha x_2^2 \\ &= (1-\alpha)f(x_1) + \alpha f(x_2) \end{aligned} \tag{7}$$

- Implication: strict inequality $<$, thus strict convexity

Convex vs. Strictly convex

Consequences: consider function $f : \mathbb{R}^n \to \mathbb{R}$ defined on a convex set $\mathcal{C} \subseteq \mathbb{R}^n$

- If f is **strictly convex**, it has at most one global minimizer, and the only local minimizer of f in $\mathcal{C}$ is the global minimizer of f (**if it exists**)
- If f is **convex** but not strictly convex, its local minimizer needs not to be unique but any local minimizer of f in $\mathcal{C}$ is also a global minimizer of f

Convex vs. Strictly convex

Consequences: consider function $f : \mathbb{R}^n \to \mathbb{R}$ defined on a convex set $\mathcal{C} \subseteq \mathbb{R}^n$

- If f is **strictly convex**, it has at most one global minimizer, and the only local minimizer of f in $\mathcal{C}$ is the global minimizer of f (**if it exists**)
- If f is **convex** but not strictly convex, its local minimizer needs not to be unique but any local minimizer of f in $\mathcal{C}$ is also a global minimizer of f

These are known as the fundamental theorems of convex analysis

Convex vs. Strictly convex

Proposition 1

- Let $f : \mathbb{R}^n \to \mathbb{R}$ be a **convex** function defined on a convex set $\mathcal{C} \subseteq \mathbb{R}^n$
- Then, any local minimum of f over $\mathcal{C}$ is also a global minimum of f over $\mathcal{C}$

Convex vs. Strictly convex

Proposition 1

- Let $f : \mathbb{R}^n \to \mathbb{R}$ be a **convex** function defined on a convex set $\mathcal{C} \subseteq \mathbb{R}^n$
- Then, any local minimum of f over $\mathcal{C}$ is also a global minimum of f over $\mathcal{C}$

Proof (by contradiction): Let $x^\star$ be a local minimum of f convex,

- then $x^\star \in \mathcal{C}$ and $\exists\, \varepsilon > 0$ such that $f(x^\star) \leq f(x), \forall x \in B(x^\star, \varepsilon)$
- Suppose $x^\star$ is a local minimum that is not global, i.e., $\exists\, y \in \mathcal{C}$ such that $f(y) < f(x^\star)$
- By convexity of $\mathcal{C}$, we have $\alpha x^\star + (1-\alpha)y \in \mathcal{C}$, $\forall\, \alpha \in [0,1]$
- By convexity of f, we have

$$\begin{aligned} f(\alpha x^\star + (1-\alpha)y) &\leq \alpha f(x^\star) + (1-\alpha) f(y) \\ &< \alpha f(x^\star) + (1-\alpha) f(x^\star) = f(x^\star) \end{aligned}$$

- Since f has smaller value along line segment from y to $x^\star$ (as $\alpha \to 1$, $(\alpha x^\star + (1-\alpha)y) \to x^\star$), this inequality contradicts local minimality of $x^\star$

Convex vs. Strictly convex

Proposition 2

- Let $f : \mathbb{R}^n \to \mathbb{R}$ be a **strictly convex** function defined on a convex set $\mathcal{C} \subseteq \mathbb{R}^n$
- Then, the local minimizer of f in $\mathcal{C}$ is unique that is (**if it exists**) the global minimizer of f

Proof:

- Suppose f has two local minimizers x_1, x_2, $x_1 \neq x_2$ such that $f(x_1) \leq f(x_2)$
- Since f is strictly convex, for any α, $0 < \alpha < 1$:

$$f\big(\alpha x_1 + (1-\alpha)x_2\big) < \alpha f(x_1) + (1-\alpha) f(x_2)$$

- Since $\alpha > 0$, $f(x_1) \leq f(x_2) \Rightarrow \alpha f(x_1) \leq \alpha f(x_2)$; thus,

$$\alpha f(x_1) + (1-\alpha) f(x_2) \leq \alpha f(x_2) + (1-\alpha) f(x_2) = f(x_2)$$

- Thus, following strict convexity, $f\big(\alpha x_1 + (1-\alpha)x_2\big) < f(x_2)$ which contradicts that x_2 is a local minimizer of f
- To satisfy both conditions, $x_1 = x_2$ showing that f has at most one local minimum in $\mathcal{C}$ (which is global min. following Proposition 1)

Convex vs. Strictly convex

In Proposition 2, the **if condition** is essential: there exist strictly convex functions that have no global minimizers

Convex vs. Strictly convex

In Proposition 2, the **if condition** is essential: there exist strictly convex functions that have no global minimizers

Examples:

- Every exponential function $f : x \to a^x$, with $a > 0$ and $a \neq 1$ is strictly convex, but admits no local (and thus no global) minimizer on $\mathbb{R}$

  Indeed, since $f'(x) = a^x \ln(a)$

$$\begin{aligned} f(x) = a^x &\Leftrightarrow \ln(f(x)) = \ln(a^x) \\ &\Leftrightarrow \exp(\ln(f(x))) = \exp(\ln(a^x)) \\ &\Rightarrow f(x) = \exp(\ln(a^x)) = \exp(x \ln(a)) \\ &\Rightarrow f'(x) = \ln(a) \exp(x \ln(a)) = \ln(a) \exp(\ln(a)^x)) = a^x \ln(a) \end{aligned}$$

  $\Rightarrow f$ is strictly increasing for $a > 1$ and strictly decreasing for $0 < a < 1$

Convex vs. Strictly convex

In Proposition 2, the **if condition** is essential: there exist strictly convex functions that have no global minimizers

Examples:

- Every exponential function $f : x \to a^x$, with $a > 0$ and $a \neq 1$ is strictly convex, but admits no local (and thus no global) minimizer on $\mathbb{R}$

  Indeed, since $f'(x) = a^x \ln(a)$

$$\begin{aligned} f(x) = a^x &\Leftrightarrow \ln(f(x)) = \ln(a^x) \\ &\Leftrightarrow \exp(\ln(f(x))) = \exp(\ln(a^x)) \\ &\Rightarrow f(x) = \exp(\ln(a^x)) = \exp(x \ln(a)) \\ &\Rightarrow f'(x) = \ln(a) \exp(x \ln(a)) = \ln(a) \exp(\ln(a)^x)) = a^x \ln(a) \end{aligned}$$

  $\Rightarrow f$ is strictly increasing for $a > 1$ and strictly decreasing for $0 < a < 1$

- Function $f(x) = 1/x$ on $[1, \infty[$

  Infimum of f (value 0) is nowhere achieved; thus, f has no minimizer

Convex function: examples

Examples of convex functions

- **Linear** functions $f : \mathbb{R}^n \to \mathbb{R} : x \to c^\top x$, for any constant vector $c \in \mathbb{R}^n$

Convex function: examples

Examples of convex functions

- **Linear** functions $f : \mathbb{R}^n \to \mathbb{R} : x \to c^\top x$, for any constant vector $c \in \mathbb{R}^n$
- **Affine** functions $f : \mathbb{R}^n \to \mathbb{R} : x \to c^\top x + b$, for any $c \in \mathbb{R}^n$ and scalar b; affine (and therefore also linear) functions are both convex and concave
- Affine mappings $f : \mathbb{R}^n \to \mathbb{R}^m : x \to Ax + b$, for any matrix $A \in \mathbb{R}^{m \times n}$ and vector $b \in \mathbb{R}^m$; affine (and therefore also linear) mappings are both convex and concave

Convex function: examples

Examples of convex functions

- **Linear** functions $f : \mathbb{R}^n \to \mathbb{R} : x \to c^\top x$, for any constant vector $c \in \mathbb{R}^n$
- **Affine** functions $f : \mathbb{R}^n \to \mathbb{R} : x \to c^\top x + b$, for any $c \in \mathbb{R}^n$ and scalar b; affine (and therefore also linear) functions are both convex and concave
- Affine mappings $f : \mathbb{R}^n \to \mathbb{R}^m : x \to Ax + b$, for any matrix $A \in \mathbb{R}^{m \times n}$ and vector $b \in \mathbb{R}^m$; affine (and therefore also linear) mappings are both convex and concave
- **Quadratic** functions $f : \mathbb{R}^n \to \mathbb{R} : x \to x^\top Qx$, where $Q \in \mathbb{R}^{n \times n}$ is a (symmetric) positive semi-definite matrix
- **max** function : $f : \mathbb{R}^n \to \mathbb{R} : x \to \max(x) = \max\{x_1, x_2, \ldots, x_n\}$
- **Power** functions of absolute value: $f : \mathbb{R} \to \mathbb{R} : x \to |x|^p$ on $\mathbb{R}$, for $p \geq 1$
- For any $p \geq 1$, the **p-norm** on $\mathbb{R}^n$ defined by

$$\|(x_1, \ldots, x_n)\|_p = \left(|x_1|^p + \ldots + |x_n|^p\right)^{1/p} = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p} \text{ is convex}$$

Convex function : max

max function $f : \mathbb{R}^n \to \mathbb{R} \colon x \to \max(x) = \max\{x_1, x_2, \ldots, x_n\}$ is convex

Convex function : max

max function $f : \mathbb{R}^n \to \mathbb{R} \colon x \to \max(x) = \max\{x_1, x_2, \ldots, x_n\}$ is convex

- Fix $k \in \{1, \ldots, n\}$
  Since $x_k \leq \max_i x_i$, $y_k \leq \max_i y_i$ and $\alpha \in [0, 1]$, thus $1 - \alpha \geq 0$, we have

$$\alpha x_k + (1 - \alpha) y_k \leq \alpha \max_i x_i + (1 - \alpha) \max_i y_i$$

- Therefore, since this inequality is verified for any $k \in \{1, \ldots, n\}$

$$\max_k (\alpha x_k + (1 - \alpha) y_k) \leq \alpha \max_i x_i + (1 - \alpha) \max_i y_i$$

- Consequently, the following inequality is verified

$$\max_i (\alpha x_i + (1 - \alpha) y_i) \leq \alpha \max_i x_i + (1 - \alpha) \max_i y_i$$

Convex function : preservation

Several operations preserve convexity that allow constructing new convex functions

Theorem (...)

Let $f : \mathbb{R}^n \to \mathbb{R}$ and $f_\nu : \mathbb{R}^n \to \mathbb{R}$ be convex functions for $\nu \in \mathcal{I}$ where $\mathcal{I}$ is an arbitrary index set, and let $\nu_i \in \mathcal{I}$ and $\lambda_i \geq 0, i = 1, \ldots, m$

Then following functions are also convex

(0) **Precomposition with affine mapping** (affine transformation of function domain):

$$g \colon \mathbb{R}^m \to \mathbb{R} \colon x \to g(x) = f(Ax + b)$$

where $A \colon \mathbb{R}^n \to \mathbb{R}^m$ and $b \in \mathbb{R}^n$ with $\mathrm{dom}(g) = \{x \mid Ax + b \in \mathrm{dom}(f)\}$

(1) **Composition with non-decreasing function**:

$$g(x) := (\varphi \circ f)(x) = \varphi(f(x))$$

where $\varphi : \mathbb{R} \to \mathbb{R}$ is any non-decreasing function on $\mathbb{R}$, with $\mathrm{dom}(g) = \{x \in \mathrm{dom}(f) \mid f(x) \in \mathrm{dom}(\varphi)\}$

Convex function : preservation

(...) Theorem

(2) **Non-negative linear combinations** (weighted sum): $g(x) := \sum_{i=1}^{m} \lambda_i f_{\nu_i}(x)$

Particular cases:

- Non-negative sum of two convex functions (corresponds to intersection of epigraphs) :
$$g(x) := \lambda_1 f_1(x) + \lambda_2 f_2(x),\ \lambda_1 \geq 0,\ \lambda_2 \geq 0$$
- Positive multiple ($\lambda \geq 0$) of a convex function: $g : x \to \lambda f(x)$ is convex

(3) **Pointwise maximum** of finite number of functions:

$$g(x) := \max_{\nu \in \mathcal{I}} f_\nu(x) = \max\{f_1(x), f_2(x), \ldots, f_m(x)\}$$

(4) **Pointwise supremum** of finite number of convex functions :

$$g(x) := \sup_{\nu \in \mathcal{I}} f_\nu(x)$$

Convex function : preservation

Particular cases and **Examples**:

o) Pointwise maximum of two convex functions f_1 and f_2: pointwise maximum $f(x) = \max\{f_1(x), f_2(x)\}$ with $\mathrm{dom}(f) = \mathrm{dom}(f_1) \cap \mathrm{dom}(f_2)$ is also convex

$$\begin{aligned} g(\alpha x + (1-\alpha)y) &= \max\{f_1(\alpha x + (1-\alpha)y), f_2(\alpha x + (1-\alpha)y)\} \\ &\leq \max\{\alpha f_1(x) + (1-\alpha)f_1(y), \alpha f_2(x) + (1-\alpha)f_2(y)\} \\ &\leq \alpha \max\{f_1(x), f_2(x)\} + (1-\alpha)\max\{f_1(y), f_2(y)\} \\ &= \alpha g(x) + (1-\alpha)g(y) \end{aligned}$$

o) Example: piecewise-linear functions defines as

$$f(x) = \max\{a_1^\top x + b_1, a_2^\top x + b_2, \ldots, a_L^\top x + b_L\}$$

with L or fewer regions/segments. This function is convex since pointwise maximum of affine functions

Important: maximum (largest member of set) vs. supremum (smallest upper bound of the set)

Convex function : preservation

Particular cases and **Examples**:

o) Let $f(y),\ y \in \mathbb{R}^m$ be a convex function. Then, the function $g(x) = f(Ax + b)$, where $A : \mathbb{R}^n \to \mathbb{R}^m$ and $b \in \mathbb{R}^n$ is convex with $\mathrm{dom}(g) = \{x \in \mathbb{R}^n \mid Ax + b \in \mathrm{dom}(f)\}$

Convex function : preservation

Particular cases and **Examples**:

- Let $f(y),\ y \in \mathbb{R}^m$ be a convex function. Then, the function $g(x) = f(Ax + b)$, where $A : \mathbb{R}^n \to \mathbb{R}^m$ and $b \in \mathbb{R}^n$ is convex with $\mathrm{dom}(g) = \{x \in \mathbb{R}^n \mid Ax + b \in \mathrm{dom}(f)\}$

- **Proof**:

  Let $(x_1, x_2) \in \mathrm{dom}(g) \times \mathrm{dom}(g)$

  Denote $y_1 = Ax_1 + b$ et $y_2 = Ax_2 + b$

  Then, for all $\alpha \in [0, 1]$, we have

$$\begin{aligned} g(\alpha x_1 + (1-\alpha)x_2) &= f(\alpha(Ax_1 + b) + (1-\alpha)(Ax_2 + b)) \\ &\leq \alpha f(Ax_1 + b) + (1-\alpha) f(Ax_2 + b) \\ &\leq \alpha g(x_1) + (1-\alpha) g(x_2) \end{aligned} \tag{8}$$

Convex function : p-norm

The $p-$norm, $p \geq 1$, is convex

Def: Norm over vector space X

Given a vector space X, a norm is a real-valued function $\|\cdot\| : X \to \mathbb{R}$ with the following properties

(1) **positivity**: $\forall x \in X, \|x\| \geq 0$ and **definite**: $\forall x \in X, \|x\| = 0 \Leftrightarrow x = 0$

Convex function : p-norm

The $p-$norm, $p \geq 1$, is convex

Def: Norm over vector space X

Given a vector space X, a norm is a real-valued function $\|\cdot\| : X \to \mathbb{R}$ with the following properties

(1) **positivity**: $\forall x \in X, \|x\| \geq 0$ and **definite**: $\forall x \in X, \|x\| = 0 \Leftrightarrow x = 0$

(2) **absolutely scalable**, a.k.a. absolute homogeneity:

$$\forall x \in X, \lambda \in \mathbb{R}\colon \ \|\lambda x\| = |\lambda| \, \|x\|$$

Convex function : p-norm

The $p-$norm, $p \geq 1$, is convex

Def: Norm over vector space X

Given a vector space X, a norm is a real-valued function $\|\cdot\| : X \to \mathbb{R}$ with the following properties

(1) **positivity**: $\forall x \in X, \|x\| \geq 0$ and **definite**: $\forall x \in X, \|x\| = 0 \Leftrightarrow x = 0$

(2) **absolutely scalable**, a.k.a. absolute homogeneity:

$$\forall x \in X, \lambda \in \mathbb{R}\colon \ \|\lambda x\| = |\lambda| \, \|x\|$$

(3) **triangle inequality**/sub-additivity:

$$\forall x, y \in X : \|x + y\| \leq \|x\| + \|y\|$$

Convex function : p-norm

- By the triangle inequality property (3) and the fact that the norm is absolutely scalable (2), the **norm** is convex:

$$\|\lambda x + (1-\lambda)y\| \leq \|\lambda x\| + \|(1-\lambda)y\| \tag{9}$$

$$= \lambda \|x\| + (1-\lambda)\|y\| \tag{10}$$

Convex function : p-norm

- By the triangle inequality property (3) and the fact that the norm is absolutely scalable (2), the **norm** is convex:

$$\|\lambda x + (1-\lambda)y\| \leq \|\lambda x\| + \|(1-\lambda)y\| \tag{9}$$
$$= \lambda \|x\| + (1-\lambda)\|y\| \tag{10}$$

- The **p-norm** (for any $p \geq 1$) on $\mathbb{R}^n$ is defined by

$$\|(x_1, \ldots, x_n)\|_p = \left(|x_1|^p + \ldots + |x_n|^p\right)^{1/p} \tag{11}$$
$$\Leftrightarrow \|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p} \tag{12}$$

Convex function : p-norm

- By the triangle inequality property (3) and the fact that the norm is absolutely scalable (2), the **norm** is convex:

$$\|\lambda x + (1-\lambda)y\| \leq \|\lambda x\| + \|(1-\lambda)y\| \tag{9}$$
$$= \lambda \|x\| + (1-\lambda)\|y\| \tag{10}$$

- The **p-norm** (for any $p \geq 1$) on $\mathbb{R}^n$ is defined by

$$\|(x_1, \ldots, x_n)\|_p = \left(|x_1|^p + \ldots + |x_n|^p\right)^{1/p} \tag{11}$$
$$\Leftrightarrow \|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p} \tag{12}$$

$\Rightarrow$ The $p-$norm satisfies properties (1) and (2)

Convex function : p-norm

- The third property (3) follows from the **Minkowski inequality**

Convex function : p-norm

- The third property (3) follows from the **Minkowski inequality**

Minkowski inequality

- Let $x, y \in \mathbb{R}^n$, if $p \geq 1$ then $\|x + y\|_p \leq \|x\|_p + \|y\|_p$; else if $p < 1, p \neq 0$, then the inequality is reversed (for $p < 0$, one must have $x_i, y_i > 0$)
- Generalizes the triangle inequality for the 2-norm (often called the Euclidean norm) to the p-norm, $p \geq 1$

Convex function : p-norm

- The third property (3) follows from the **Minkowski inequality**

Minkowski inequality

- Let $x, y \in \mathbb{R}^n$, if $p \geq 1$ then $\|x + y\|_p \leq \|x\|_p + \|y\|_p$; else if $p < 1, p \neq 0$, then the inequality is reversed (for $p < 0$, one must have $x_i, y_i > 0$)
- Generalizes the triangle inequality for the 2-norm (often called the Euclidean norm) to the p-norm, $p \geq 1$

$\Rightarrow$ Consequently, the $p-$norm being a norm, the $p-$norm is convex

Convex function : p-norm

Illustration: Figure 1 shows the unit ball (in $\mathbb{R}^2$) $B_X(1) = \{x \in X \mid \|x\|_p \leq 1\}$ with respect to various p-norms

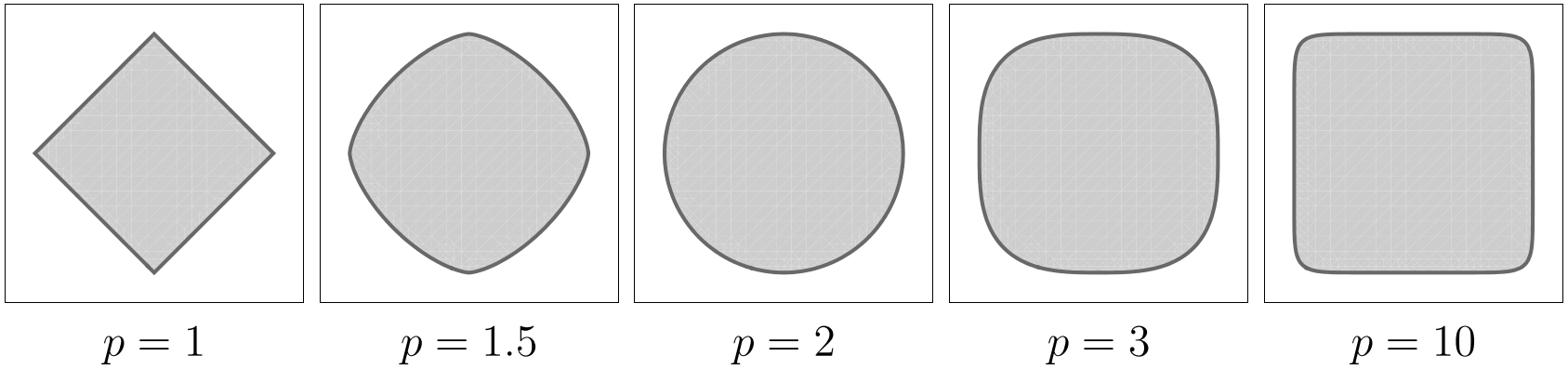
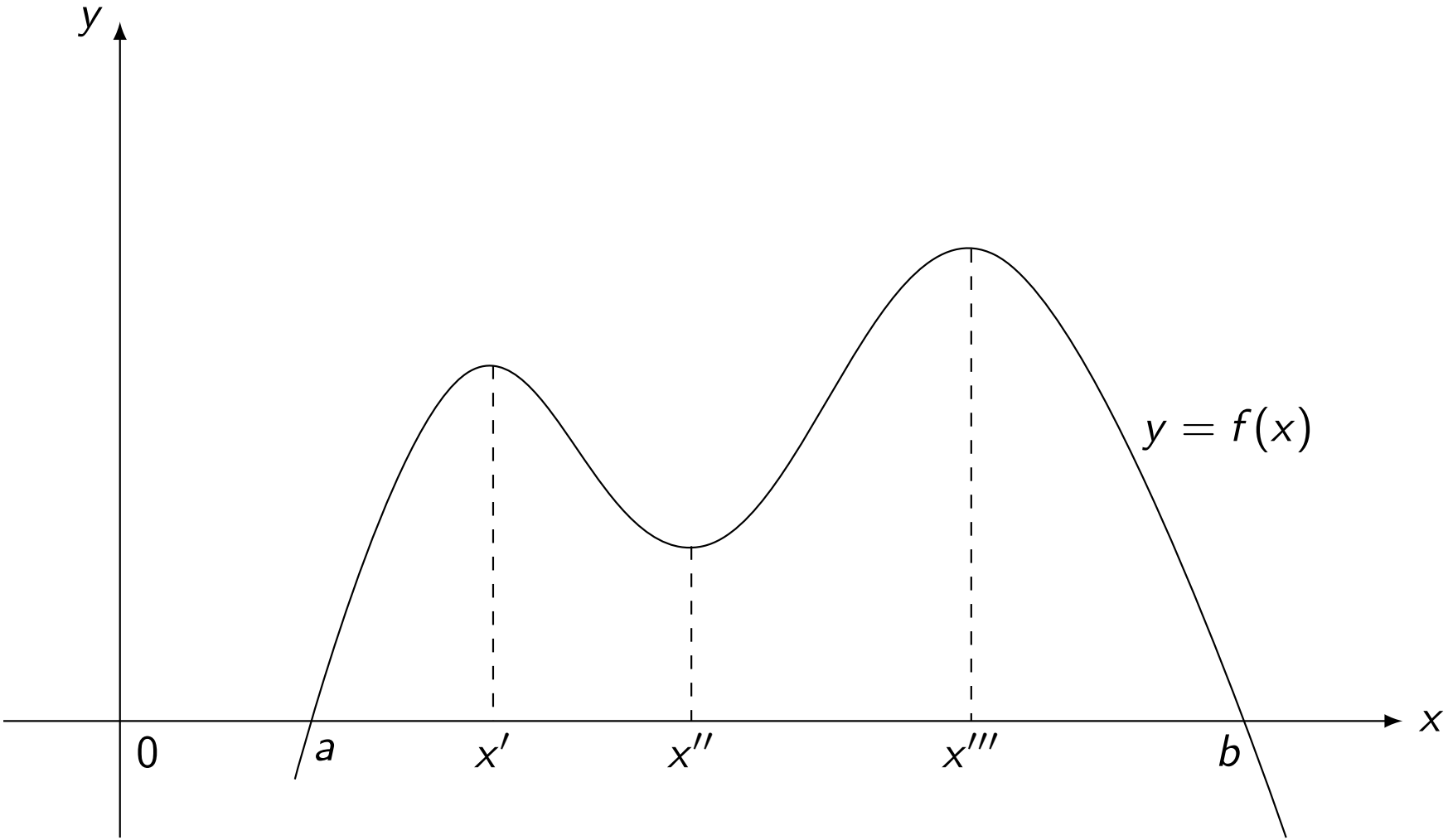


Figure: Shape of unit ball for various p-norms

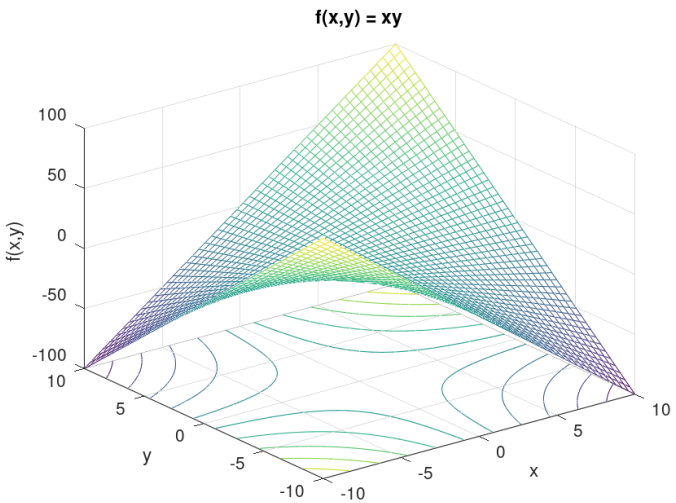
<u>Note</u>: any norm $\|-\|$ defined on a vector space gives a metric d on the vector space defined by the expression $d(x, y) = \|x - y\|$

Nonconvex function

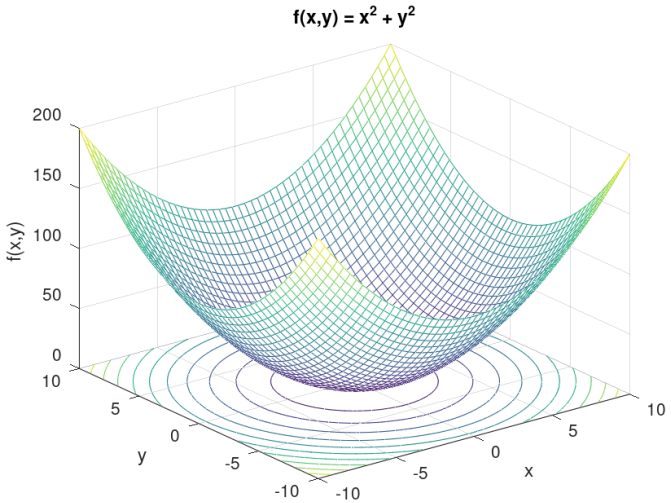
Example of nonconvex function



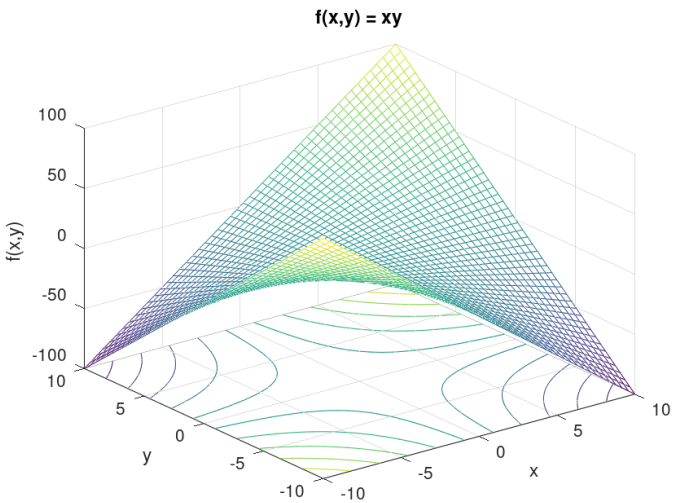
Nonconvex function



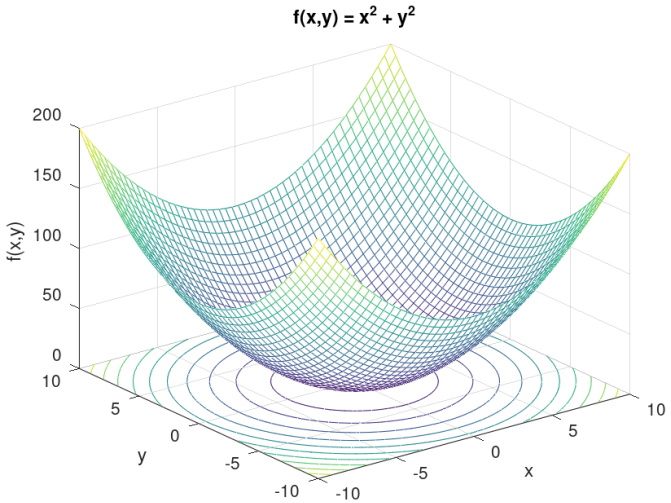
+



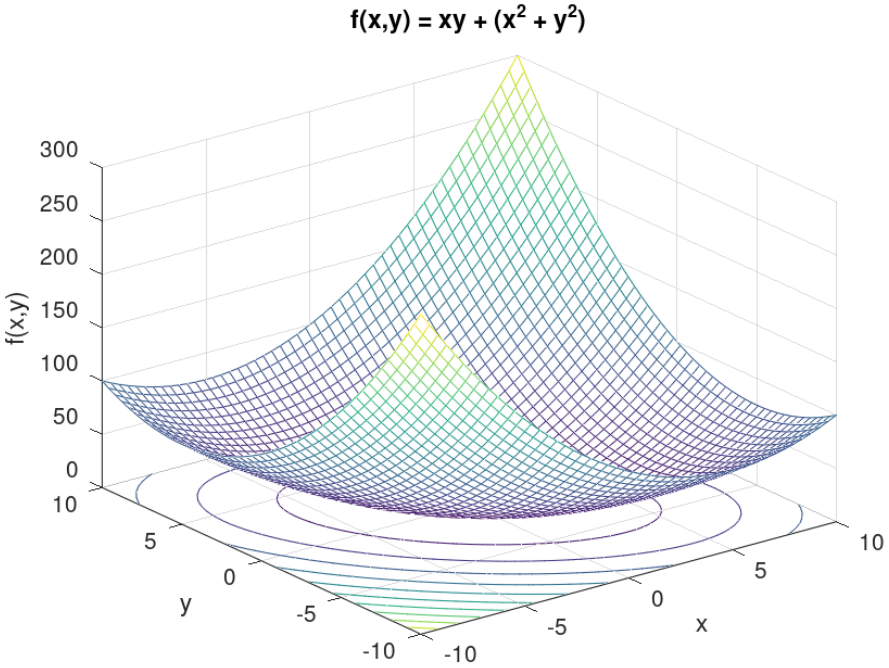
Nonconvex function



+



$\Downarrow$



First- and Second-order Conditions

First- and Second-order Conditions: Gradient

Def.: Gradient

The gradient of a function $f : \mathbb{R}^n \to \mathbb{R} : x \to f(x)$ is defined at the point $x = (x_1, \ldots, x_n)$ as the vector

$$\nabla f(x) = \left(\frac{\partial f}{\partial x_1}(x), \ldots, \frac{\partial f}{\partial x_n}(x) \right)^\top,$$

where

$$\frac{\partial f}{\partial x_i}, i \in 1, \ldots, n$$

denotes the partial derivative of f with respect to x_i

First- and Second-order Conditions: Gradient

Example: $f(x_1, x_2) = x_1^2 x_2$

- The gradient (vector of partial derivatives) of f is given by

$$\left.\begin{aligned} \frac{\partial f}{\partial x_1}(x_1, x_2) &= 2x_1 x_2 \\ \frac{\partial f}{\partial x_2}(x_1, x_2) &= x_1^2 \end{aligned}\right\} \longrightarrow \nabla f(x_1, x_2) = (2x_1 x_2, x_1^2)^T$$

First- and Second-order Conditions: Gradient

Example: $f(x_1, x_2) = x_1^2 x_2$

- The gradient (vector of partial derivatives) of f is given by

$$\left.\begin{aligned} \frac{\partial f}{\partial x_1}(x_1, x_2) &= 2x_1x_2 \\ \frac{\partial f}{\partial x_2}(x_1, x_2) &= x_1^2 \end{aligned}\right\} \longrightarrow \nabla f(x_1, x_2) = (2x_1x_2, x_1^2)^T$$

- At point (-3,2)

$$\left.\begin{aligned} \frac{\partial f}{\partial x_1}(-3, 2) &= -12 \\ \frac{\partial f}{\partial x_2}(-3, 2) &= 9 \end{aligned}\right\} \longrightarrow \nabla f(-3, 2) = (-12, 9)^T$$

First- and Second-order Conditions: Gradient

Example: $f(x_1, x_2) = x_1^2 x_2$

- The gradient (vector of partial derivatives) of f is given by

$$\left.\begin{aligned} \frac{\partial f}{\partial x_1}(x_1, x_2) &= 2x_1x_2 \\ \frac{\partial f}{\partial x_2}(x_1, x_2) &= x_1^2 \end{aligned}\right\} \longrightarrow \nabla f(x_1, x_2) = (2x_1x_2, x_1^2)^T$$

- At point (-3,2)

$$\left.\begin{aligned} \frac{\partial f}{\partial x_1}(-3, 2) &= -12 \\ \frac{\partial f}{\partial x_2}(-3, 2) &= 9 \end{aligned}\right\} \longrightarrow \nabla f(-3, 2) = (-12, 9)^T$$

- **Interpretation**

  Each component of gradient (vector) gives slope of tangent plane in one dimension only: at (-3,2) has slope of -12 in x_1-direction and $+9$ in x_2-direction

  Gradient vector points in the **direction** of steepest increase of f and its **magnitude** gives the (steepest possible) slope of the tangent plane in that direction

First- and Second-order Conditions: Gradient

Examples: $f(x_1, x_2) = 2x_1^2 x_2 - 4x_1$

- The gradient (vector of partial derivatives) of f is given by

$$\left.\begin{aligned} \frac{\partial f}{\partial x_1}(x_1, x_2) &= 4x_1x_2 - 4 \\ \frac{\partial f}{\partial x_2}(x_1, x_2) &= 2x_1^2 \end{aligned}\right\} \longrightarrow \nabla f(x_1, x_2) = (4x_1x_2 - 4, 2x_1^2)^T$$

First- and Second-order Conditions: Gradient

Examples: $f(x_1, x_2) = 2x_1^2 x_2 - 4x_1$

- The gradient (vector of partial derivatives) of f is given by

$$\left.\begin{aligned} \frac{\partial f}{\partial x_1}(x_1, x_2) &= 4x_1x_2 - 4 \\ \frac{\partial f}{\partial x_2}(x_1, x_2) &= 2x_1^2 \end{aligned}\right\} \longrightarrow \nabla f(x_1, x_2) = (4x_1x_2 - 4, 2x_1^2)^T$$

- At point (1,1)

$$\left.\begin{aligned} \frac{\partial f}{\partial x_1}(1, 1) &= 0 \\ \frac{\partial f}{\partial x_2}(1, 1) &= 1 \end{aligned}\right\} \longrightarrow \nabla f(1, 1) = (0, 1)^T$$

First- and Second-order Conditions: Gradient

Exercises:

- Find the gradient of $f(x_1, x_2) = 2x_1x_2 - 2x_2$

  and the magnitude of the gradient vector (using the Pythagorean theorem) at the points a) (0, 0), b) (5, -3), c) (10, 5), d) (-5, 4)

- Find the gradient of $f(x_1, x_2) = 4x_1 + 2x_2 - x_1^2 - 3x_2^2$

  and the magnitude of the gradient vector (using the Pythagorean theorem) at the points a) (1, 5), b) (3, -2), c) (2, 0), d) (-2, -3)

- Find the gradient of $f(x_1, x_2, x_3) = 3x_1 + 2x_2^2 - x_3^3$

  and the magnitude of the gradient vector (using the Pythagorean theorem) at the point (1,2,3)

First-order Conditions

By language abuse, we say that

- the function f is **differentiable** on $(\mathcal{C} \subseteq)\ \mathbb{R}^n$ if $\nabla f(x)$ exists for all $x \in (\mathcal{C} \subseteq)\ \mathbb{R}^n$
- the function f is **continuously differentiable** if $\nabla f(x)$ is a continuous function of x

First-order Conditions

By language abuse, we say that

- the function f is **differentiable** on $(\mathcal{C} \subseteq)\ \mathbb{R}^n$ if $\nabla f(x)$ exists for all $x \in (\mathcal{C} \subseteq)\ \mathbb{R}^n$
- the function f is **continuously differentiable** if $\nabla f(x)$ is a continuous function of x

Def.: Interior point

The point $x^\star$ is an interior point of $\mathcal{C}$ if there exists a ball $B_r(x^\star)$ with center $x^\star$ and radius $r > 0$ such that $B_r(x^\star) \subset \mathcal{C}$

First-order Necessary Conditions

Definition of **stationary point** of function $f : \mathbb{R}^n \to \mathbb{R}$

Def: Stationary point

Let the function $f : \mathbb{R}^n \to \mathbb{R}$ be differentiable. Any point $\overline{x} \in \mathbb{R}^n$ satisfying $\nabla f(\overline{x}) = 0$ is said to be a **stationary point** of f

First-order Necessary Conditions

Definition of **stationary point** of function $f : \mathbb{R}^n \to \mathbb{R}$

Def: Stationary point

Let the function $f : \mathbb{R}^n \to \mathbb{R}$ be differentiable. Any point $\overline{x} \in \mathbb{R}^n$ satisfying $\nabla f(\overline{x}) = 0$ is said to be a **stationary point** of f

First-order Necessary Conditions

Definition of **stationary point** of function $f : \mathbb{R}^n \to \mathbb{R}$

Def: Stationary point

Let the function $f : \mathbb{R}^n \to \mathbb{R}$ be differentiable. Any point $\overline{x} \in \mathbb{R}^n$ satisfying $\nabla f(\overline{x}) = 0$ is said to be a **stationary point** of f

Note : for a differentiable function, critical point equivalent to stationary point

First-order Necessary Conditions

Definition of **stationary point** of function $f : \mathbb{R}^n \to \mathbb{R}$

Def: Stationary point

Let the function $f : \mathbb{R}^n \to \mathbb{R}$ be differentiable. Any point $\overline{x} \in \mathbb{R}^n$ satisfying $\nabla f(\overline{x}) = 0$ is said to be a **stationary point** of f

Note : for a differentiable function, critical point equivalent to stationary point

When $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable, any point $\overline{x} \in \mathbb{R}^n$ satisfying $\nabla f(\overline{x}) = 0$ is said to be a **stationary point** of f or equivalently a critical point of f

First-order Necessary Conditions

Definition of **stationary point** of function $f : \mathbb{R}^n \to \mathbb{R}$

Def: Stationary point

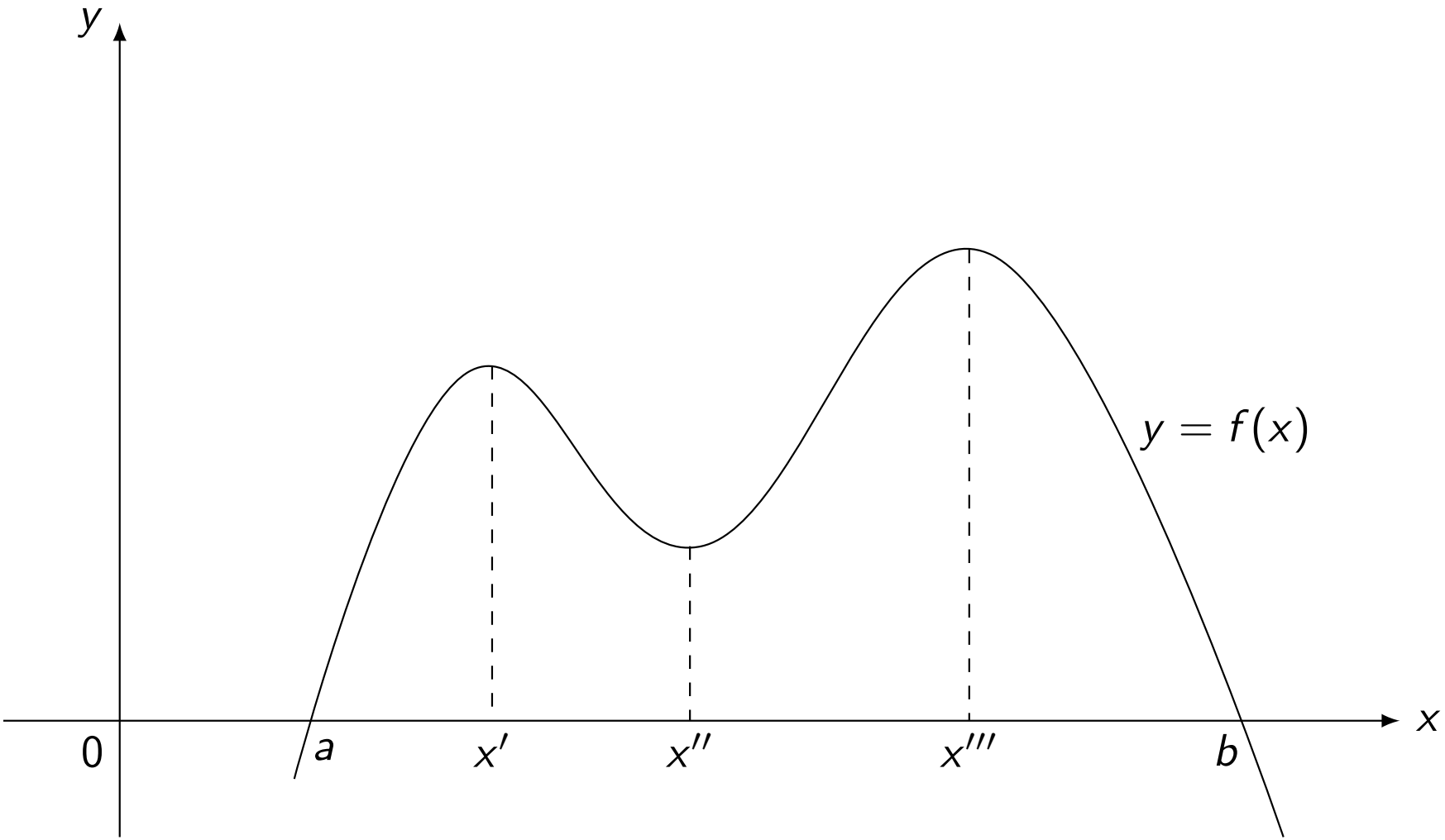
Let the function $f : \mathbb{R}^n \to \mathbb{R}$ be differentiable. Any point $\overline{x} \in \mathbb{R}^n$ satisfying $\nabla f(\overline{x}) = 0$ is said to be a **stationary point** of f

Note : for a differentiable function, critical point equivalent to stationary point

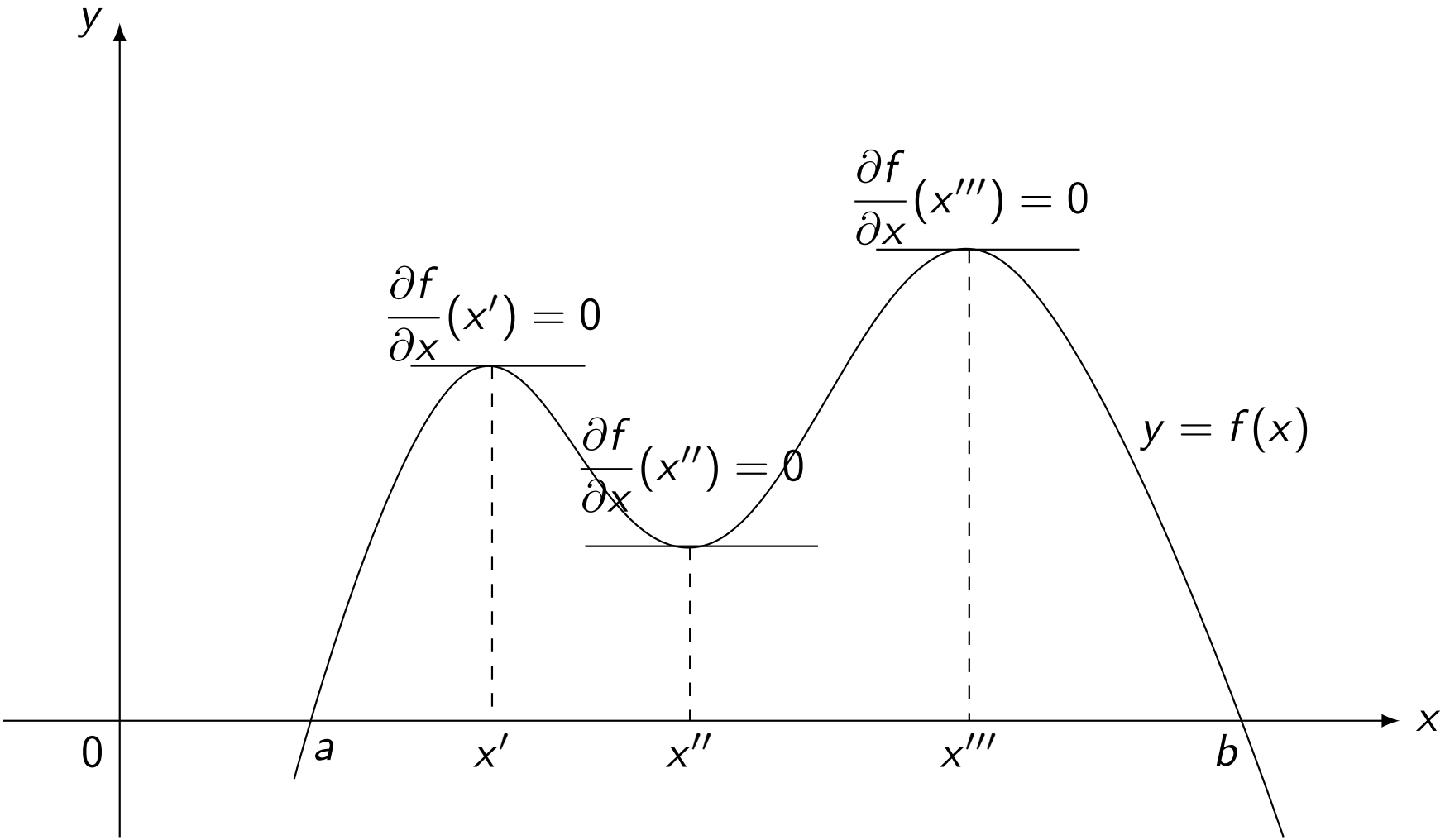
When $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable, any point $\overline{x} \in \mathbb{R}^n$ satisfying $\nabla f(\overline{x}) = 0$ is said to be a **stationary point** of f or equivalently a critical point of f

Important: at this point, no assumption about convexity properties of f

Stationary points: Example



Stationary points: Example



First-order Necessary Conditions

Theorem: First-order necessary conditions

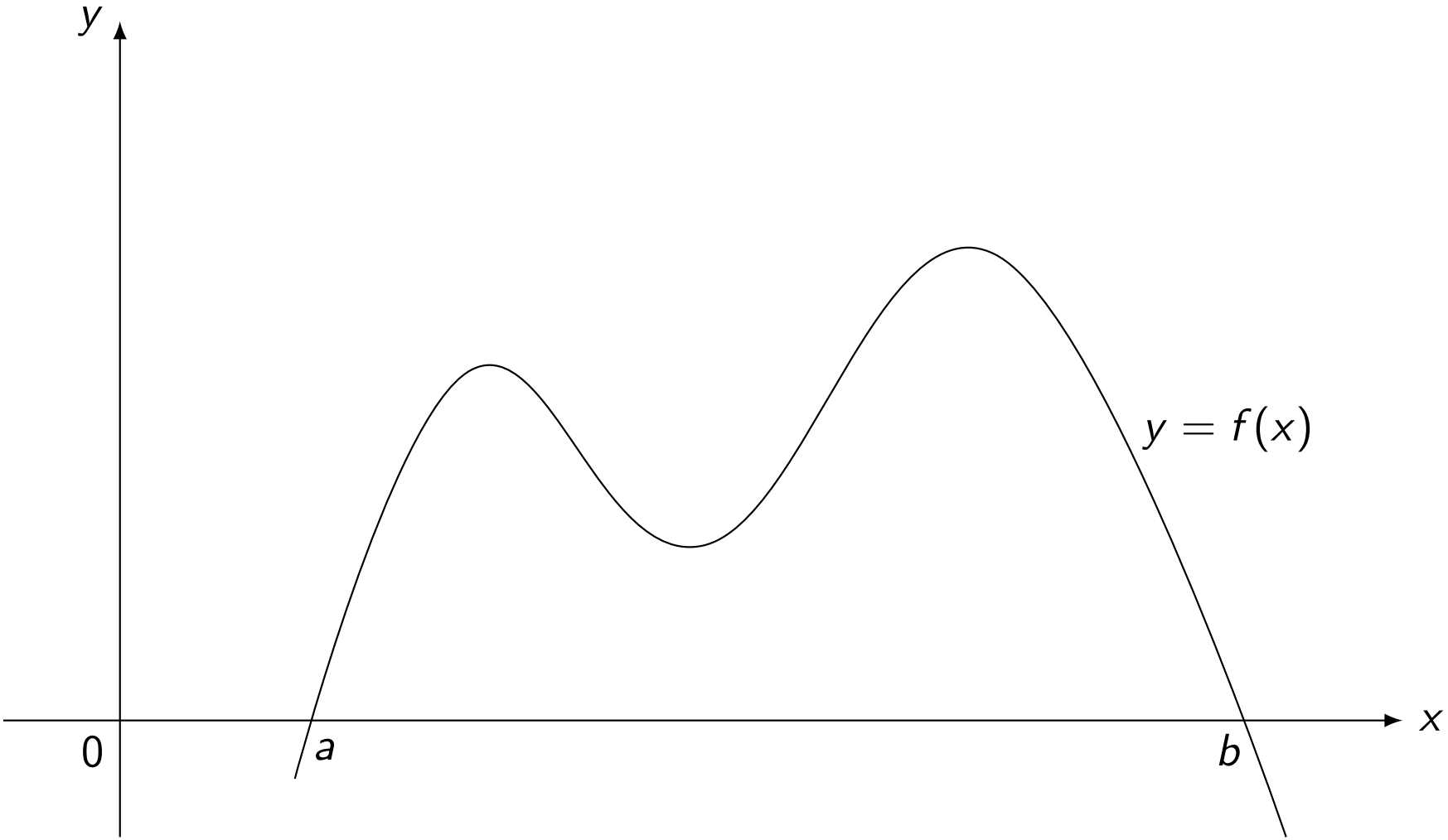
- Let function $f : \mathbb{R}^n \to \mathbb{R}$ be differentiable at a point $x^\star \in \mathbb{R}^n$
- **If** $x^\star$ is a local minimum or a local maximum of f

  Then $x^\star$ is a stationary point of f, formally

$$\nabla f(x^\star) = \left(\frac{\partial f}{\partial x_1}(x^\star), \ldots, \frac{\partial f}{\partial x_n}(x^\star) \right)^\top = 0 \Leftrightarrow \left(\frac{\partial f}{\partial x_i} \right)(x^\star) = 0, \ \forall i$$

First-order Necessary Conditions

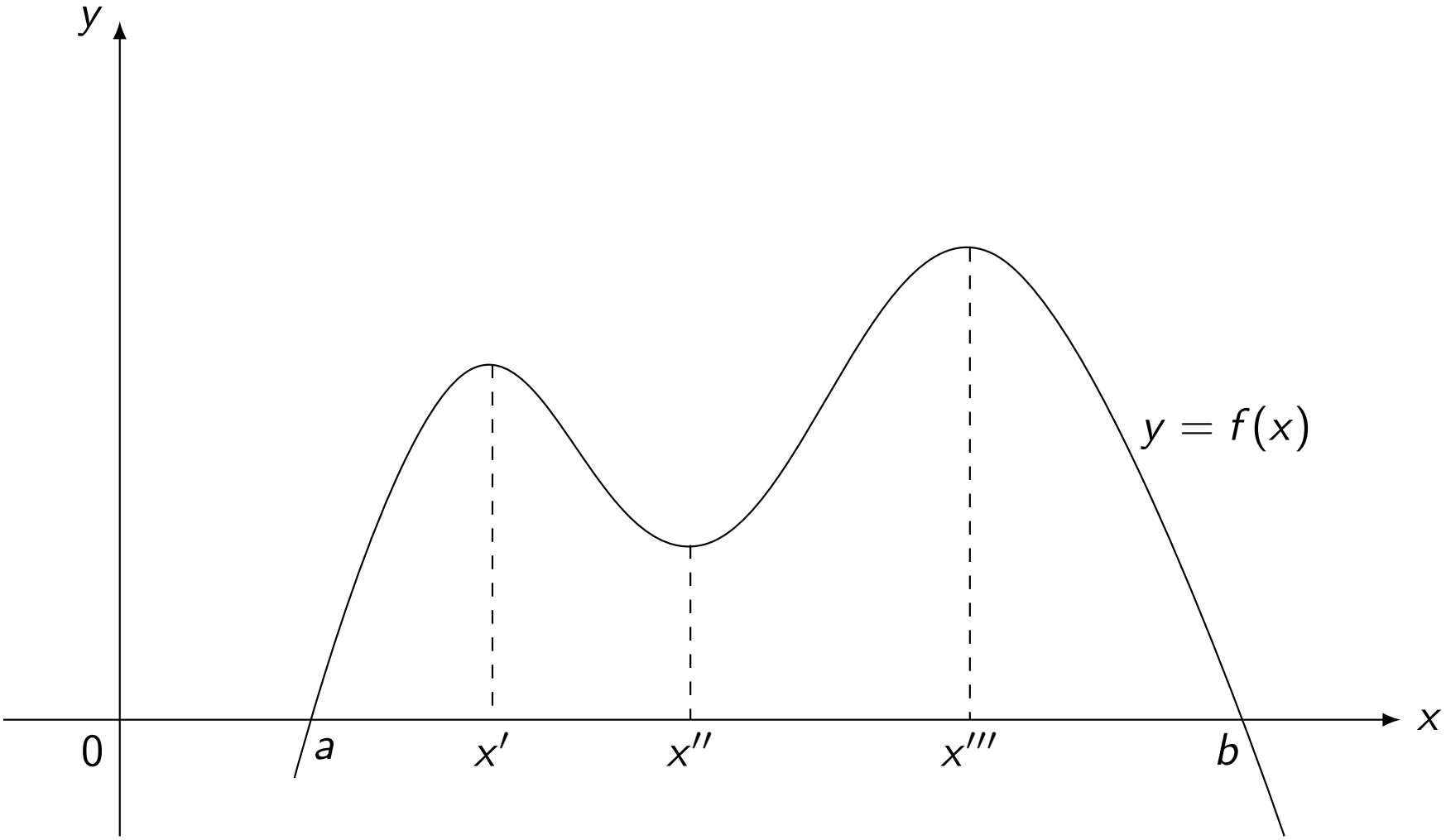
This theorem tells us ...



... **if** $x^\star$ is a local extremum of f **then** $x^\star$ is a stationary point of f (only)

First-order Necessary Conditions

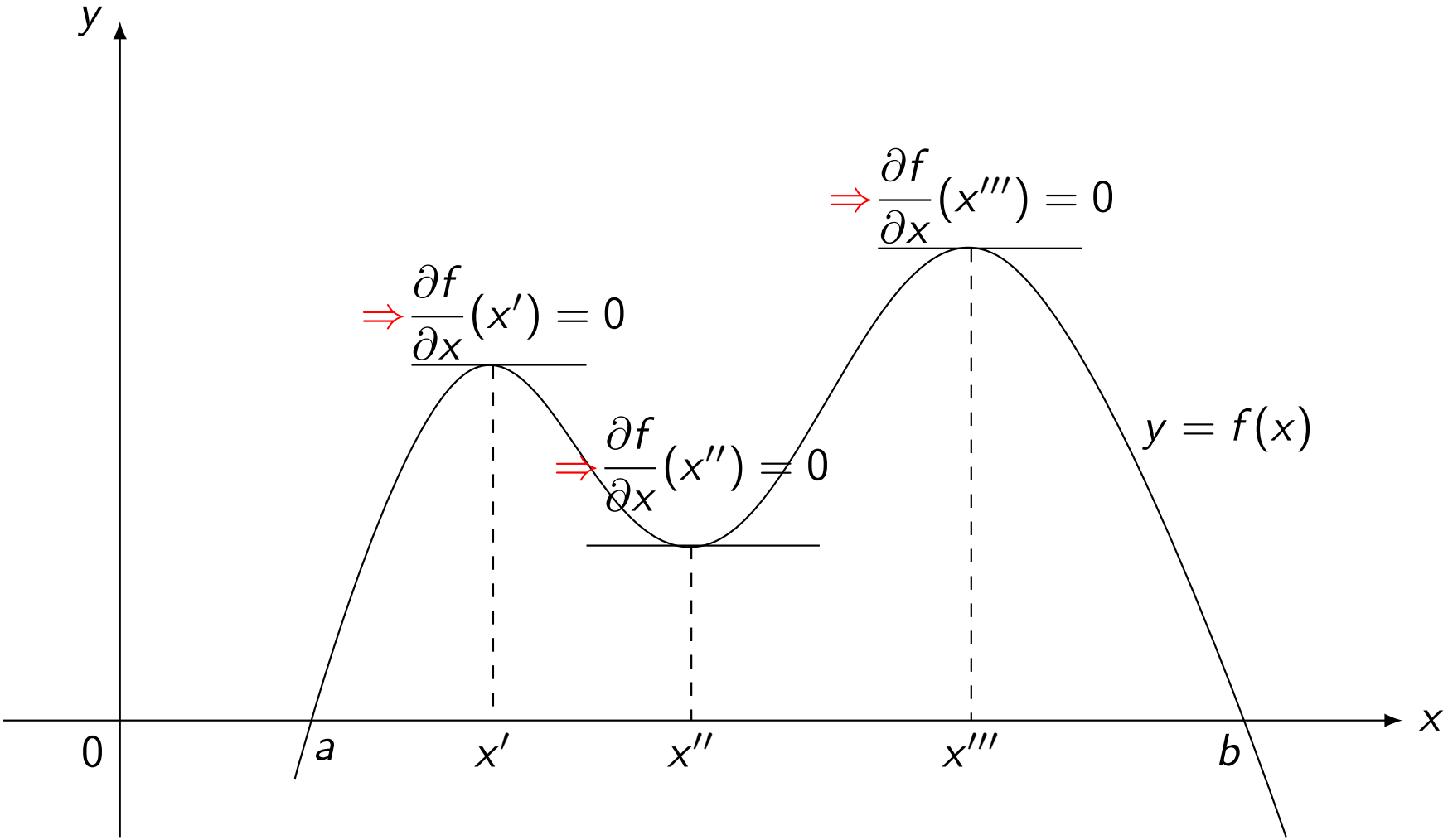
This theorem tells us ...



... **if** $x^\star$ is a local extremum of f **then** $x^\star$ is a stationary point of f (only)

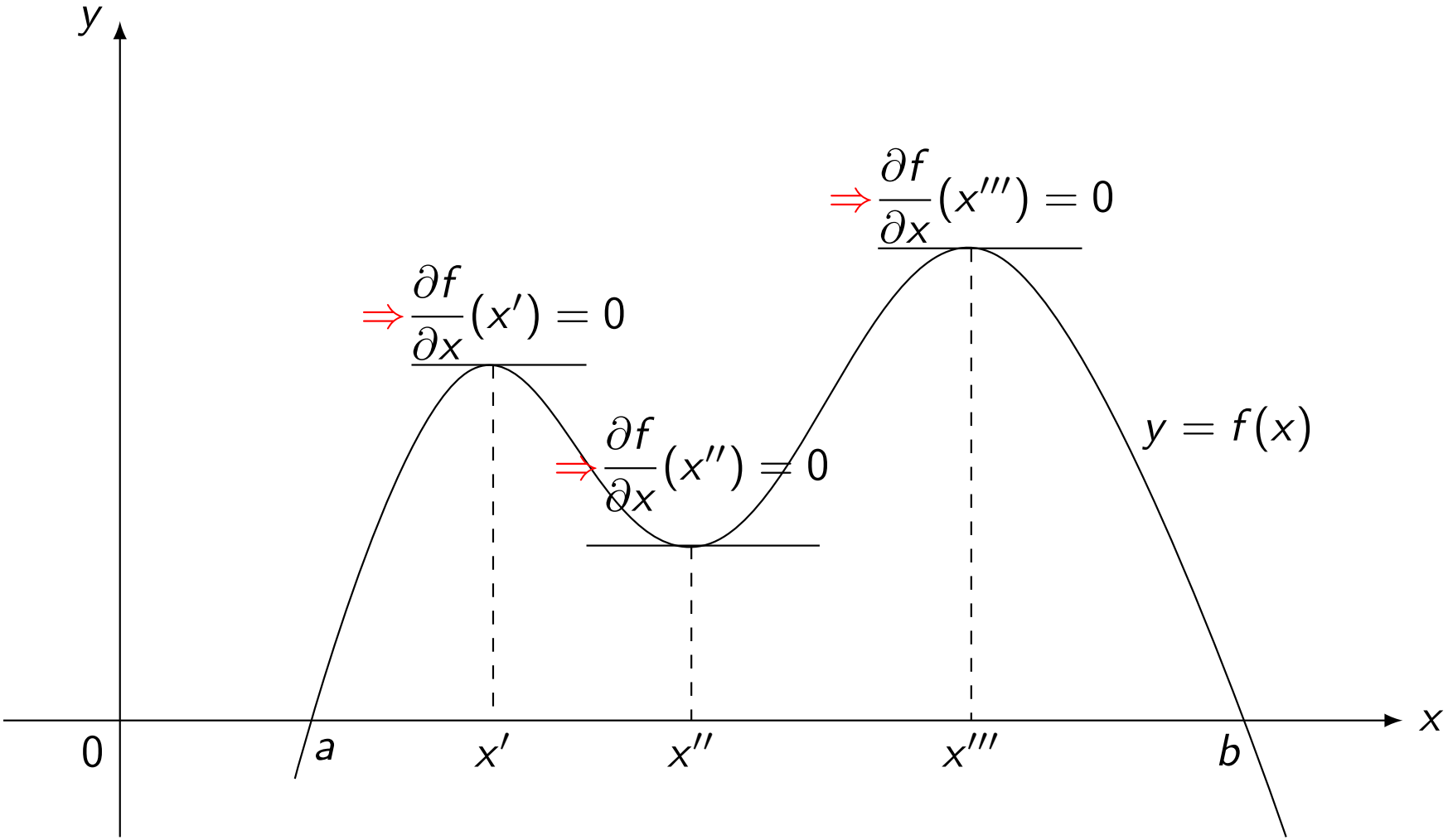
First-order Necessary Conditions

This theorem tells us ...



First-order Necessary Conditions

This theorem tells us ...



... **if** $x^\star$ is a local extremum of f **then** $x^\star$ is a stationary point of f (only)

First-order Necessary Conditions: Directional derivative

Def: Directional derivative

For general function $f : \mathbb{R}^n \to \mathbb{R}$, the directional derivative of f at $x \in \mathbb{R}^n$ in the **direction** d is

$$Df(x; d) := \lim_{t \to 0} \frac{f(x + td) - f(x)}{t} \tag{13}$$

First-order Necessary Conditions: Directional derivative

Def: Directional derivative

For general function $f : \mathbb{R}^n \to \mathbb{R}$, the directional derivative of f at $x \in \mathbb{R}^n$ in the **direction** d is

$$Df(x; d) := \lim_{t \to 0} \frac{f(x + td) - f(x)}{t} \tag{13}$$

Intuitively: the directional derivative of $f : \mathbb{R}^n \to \mathbb{R}$ along given vector d (direction) at given point x represents instantaneous rate of change of f moving through x with direction specified by d

First-order Necessary Conditions: Directional derivative

Def: Directional derivative

For general function $f : \mathbb{R}^n \to \mathbb{R}$, the directional derivative of f at $x \in \mathbb{R}^n$ in the **direction** d is

$$Df(x; d) := \lim_{t \to 0} \frac{f(x + td) - f(x)}{t} \tag{13}$$

Intuitively: the directional derivative of $f : \mathbb{R}^n \to \mathbb{R}$ along given vector d (direction) at given point x represents instantaneous rate of change of f moving through x with direction specified by d

Properties

- If f is **convex** and finite near $x \in int(\mathrm{dom}(f))$, then $Df(x; d)$ exists for all d
- If f is **differentiable** at x, then

$$Df(x; d) = \nabla f(x)^T d \tag{14}$$

thus, $Df(x; d)$ is **linear function** of d

First-order Necessary Conditions

Proof

- Let $x^\star \in \mathbb{R}^n$ be a local minimum for $\min_{x \in \mathbb{R}^n} f(x)$
- Then, the **directional derivative** $Df(x^\star; d)$ of f at $x^\star$,

$$Df(x^\star; d) := \lim_{t \to 0} \frac{f(x^\star + td) - f(x^\star)}{t} \geq 0$$

for every direction $d \in \mathbb{R}^n$ for which $Df(x^\star; d)$ exists

<u>Reason</u>: if there is a direction d such that $Df(x^\star; d)$ exists with $Df(x^\star; d) < 0$, then x cannot be a local solution

First-order Necessary Conditions

Proof

- Let $x^\star \in \mathbb{R}^n$ be a local minimum for $\min_{x\in\mathbb{R}^n} f(x)$
- Then, the **directional derivative** $Df(x^\star; d)$ of f at $x^\star$,

$$Df(x^\star; d) := \lim_{t\to 0} \frac{f(x^\star + td) - f(x^\star)}{t} \geq 0$$

  for every direction $d \in \mathbb{R}^n$ for which $Df(x^\star; d)$ exists

  Reason: if there is a direction d such that $Df(x^\star; d)$ exists with $Df(x^\star; d) < 0$, then x cannot be a local solution

- Since f is (continuously) differentiable in a neighborhood of $x^\star$, we have

$$0 \leq Df(x^\star; d) = \nabla f(x^\star)^\top d, \ \forall d \in \mathbb{R}^n$$

- Taking $d = -\nabla f(x^\star)$, we obtain

$$0 \leq -\nabla f(x^\star)^\top \nabla f(x^\star) = -\left\|\nabla f(x^\star)\right\|^2 \leq 0$$

- Therefore, $\nabla f(x^\star) = 0$

First-order Necessary Conditions

This Theorem also applies to functions defined on a set $\mathcal{C} \subseteq \mathbb{R}^n$

First-order Necessary Conditions

This Theorem also applies to functions defined on a set $\mathcal{C} \subseteq \mathbb{R}^n$

Def.: Interior point

The point $x^\star$ is an interior point of $\mathcal{C}$ if there exists a ball $B_r(x^\star)$ with center $x^\star$ and radius $r > 0$ such that $B_r(x^\star) \subset \mathcal{C}$

First-order Necessary Conditions

This Theorem also applies to functions defined on a set $\mathcal{C} \subseteq \mathbb{R}^n$

Def.: Interior point

The point $x^\star$ is an interior point of $\mathcal{C}$ if there exists a ball $B_r(x^\star)$ with center $x^\star$ and radius $r > 0$ such that $B_r(x^\star) \subset \mathcal{C}$

Theorem: First-order necessary conditions

- Let $f : \mathcal{C} \subseteq \mathbb{R}^n \to \mathbb{R}$ be differentiable at an **interior point** $x^\star$ of $\mathcal{C}$
- If $x^\star$ is a local minimum or a local maximum of f (over $\mathcal{C}$),

  Then $x^\star$ is a stationary point of f

First-order Necessary Conditions

This Theorem also applies to functions defined on a set $\mathcal{C} \subseteq \mathbb{R}^n$

Def.: Interior point

The point $x^\star$ is an interior point of $\mathcal{C}$ if there exists a ball $B_r(x^\star)$ with center $x^\star$ and radius $r > 0$ such that $B_r(x^\star) \subset \mathcal{C}$

Theorem: First-order necessary conditions

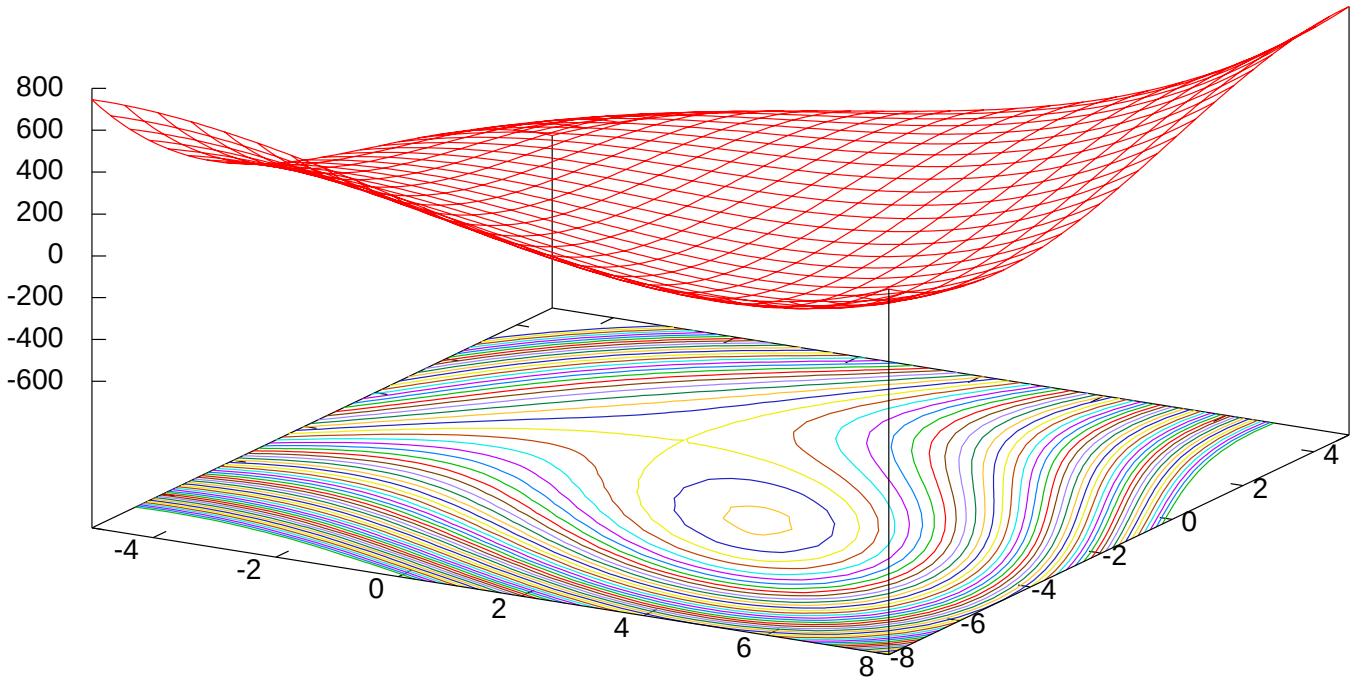
- Let $f : \mathcal{C} \subseteq \mathbb{R}^n \to \mathbb{R}$ be differentiable at an **interior point** $x^\star$ of $\mathcal{C}$
- If $x^\star$ is a local minimum or a local maximum of f (over $\mathcal{C}$),

  Then $x^\star$ is a stationary point of f

Consequence: according to these Theorems, any local extremum (i.e. local minimum or local maximum) of a function must be a stationary point

First-order Necessary conditions: example

Example: Let $f : \mathbb{R}^2 \to \mathbb{R}$: $f(x_1, x_2) = x_1^3 - x_2^3 + 9x_1x_2$



The first-order necessary conditions yield $\begin{cases} \dfrac{\partial f}{\partial x_1} & = 3x_1^2 + 9x_2 = 0 \\ \dfrac{\partial f}{\partial x_2} & = -3x_2^2 + 9x_1 = 0 \end{cases}$

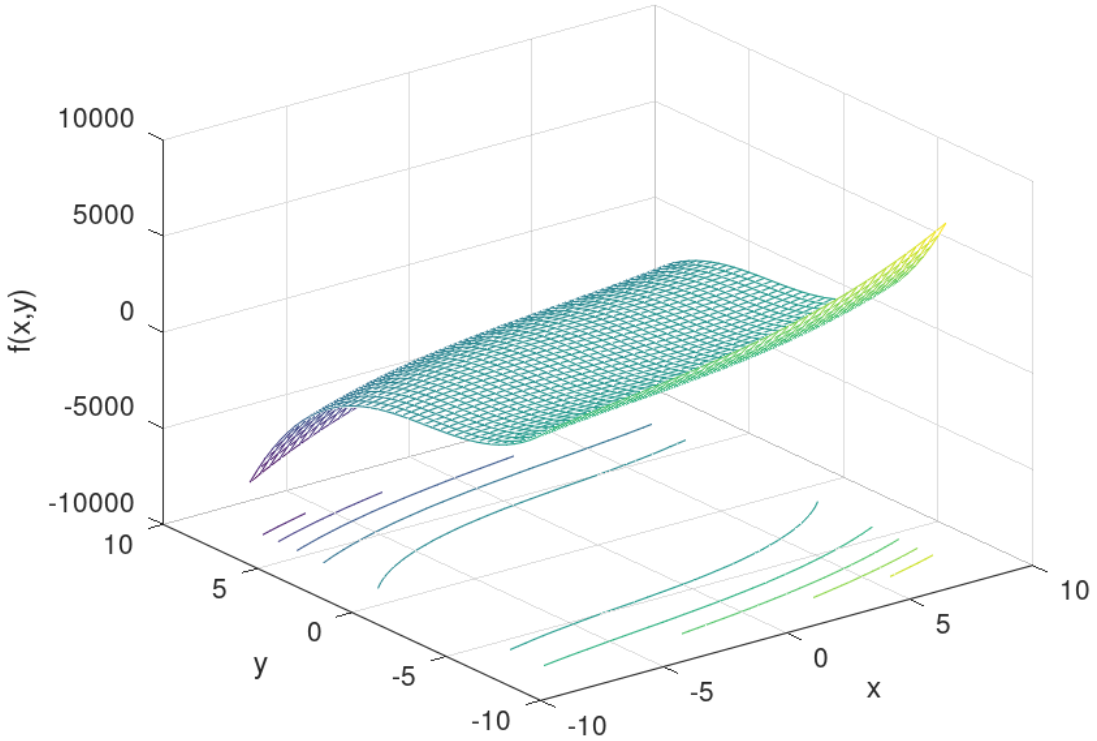
Stationary points: (0;0) and (3;-3)

First-order Necessary conditions: example

Example: Let function $f(x_1, x_2) = x_1^3 - 8x_2^3 - 2x_1^2 x_2 + 4x_1 x_2^2 - 4x_1 + 8x_2$

The first-order (necessary) conditions yield $\begin{cases} \dfrac{\partial f}{\partial x_1} & = 3x_1^2 - 4x_1 x_2 + 4x_2^2 - 4 = 0 \\ \dfrac{\partial f}{\partial x_2} & = -24x_2^2 - 2x_1^2 + 8x_1 x_2 + 8 = 0 \end{cases}$

Stationary points (4): $\pm(\sqrt{2}, 1/\sqrt{2})$ and $\pm(\sqrt{2/3}, -1/\sqrt{6})$



Hessian: definition

- **Hessian (matrix)**: square matrix of second-order partial derivatives of $f : \mathbb{R}^n \to \mathbb{R}$ denoted $\nabla^2 f$ (also denoted H_f, or $H(f)$)

$$\nabla^2 f := \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix} = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}$$

Hessian: definition

- **Hessian (matrix)**: square matrix of second-order partial derivatives of $f : \mathbb{R}^n \to \mathbb{R}$ denoted $\nabla^2 f$ (also denoted H_f, or $H(f)$)

$$\nabla^2 f := \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix} = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}$$

- **If** f is twice continuously differentiable at $x \in \mathbb{R}^n$
  then Hessian $\nabla^2 f(x)$ (also $H_f(x)$ or $Hf(x)$) is a symmetric matrix

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

Hessian: definition

- **Hessian (matrix)**: square matrix of second-order partial derivatives of $f : \mathbb{R}^n \to \mathbb{R}$ denoted $\nabla^2 f$ (also denoted H_f, or $H(f)$)

$$\nabla^2 f := \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix} = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}$$

- **If** f is twice continuously differentiable at $x \in \mathbb{R}^n$
  then Hessian $\nabla^2 f(x)$ (also $H_f(x)$ or $Hf(x)$) is a symmetric matrix

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

- **Note**: remember that symmetric matrices are orthogonally diagonalizable

Hessian: example

Example: Let $f : \mathbb{R}^2 \to \mathbb{R}$: $f(x_1, x_2) = 2x_1 - x_2 - x_1^2 + x_1x_2 - x_2^2$. The Hessian matrix H_f is

$$\nabla^2 f := \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} \end{pmatrix} = \begin{pmatrix} -2 & 1 \\ 1 & -2 \end{pmatrix}$$

Example: Let $f : \mathbb{R}^2 \to \mathbb{R}$: $f(x_1, x_2) = x_1^3 - x_2^3 + 9x_1x_2$. The Hessian matrix at $\tilde{x} = (\tilde{x_1}, \tilde{x_2})$, $H_f(\tilde{x})$ is

$$\nabla^2 f := \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} \end{pmatrix} = \begin{pmatrix} 6x_1 & 9 \\ 9 & -6x_2 \end{pmatrix}$$

$$\to \nabla^2 f(\tilde{x}) = \begin{pmatrix} 6\tilde{x}_1 & 9 \\ 9 & -6\tilde{x}_2 \end{pmatrix} \overset{(\tilde{x_1}, \tilde{x_2}) = (3, -3)}{=} \begin{pmatrix} 18 & 9 \\ 9 & 18 \end{pmatrix}$$

Hessian: example

Example: Let $f(x_1, x_2, x_3) = x_1 + x_2^2 + x_3^3 - x_1x_2 - 3x_3$. The Hessian matrix H_f at $x \in \mathbb{R}^n$, $H_f(x)$ is

$$\nabla^2 f(x) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_1 \partial x_3} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \frac{\partial^2 f}{\partial x_2 \partial x_3} \\ \frac{\partial^2 f}{\partial x_3 \partial x_1} & \frac{\partial^2 f}{\partial x_3 \partial x_2} & \frac{\partial^2 f}{\partial x_3^2} \end{pmatrix}(x)$$

$$= \begin{pmatrix} \frac{\partial(1 - x_2)}{\partial x_1} & \frac{\partial(2x_2 - x_1)}{\partial x_1} & \frac{\partial(3x_3^2 - 3)}{\partial x_1} \\ \frac{\partial(1 - x_2)}{\partial x_2} & \frac{\partial(2x_2 - x_1)}{\partial x_2} & \frac{\partial(3x_3^2 - 3)}{\partial x_2} \\ \frac{\partial(1 - x_2)}{\partial x_3} & \frac{\partial(2x_2 - x_1)}{\partial x_3} & \frac{\partial(3x_3^2 - 3)}{\partial x_3} \end{pmatrix}(x) = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 2 & 0 \\ 0 & 0 & 6x_3 \end{pmatrix}$$

Hessian: properties

Second-order necessary (and sufficient) conditions for optimality rely on the notions of **positive (semi-)definiteness** along with second-order Taylor series expansion for f at given point $x \in \mathbb{R}^n$

Hessian: properties

Second-order necessary (and sufficient) conditions for optimality rely on the notions of **positive (semi-)definiteness** along with second-order Taylor series expansion for f at given point $x \in \mathbb{R}^n$

The Hessian matrix $H_f(x) = \nabla^2 f(x)$ is

- positive (negative) semi-definite, i.e., $x^T Hx \geq 0$ $(x^T Hx \leq 0), \forall x \in \mathbb{R}^n$ **if and only if** its eigenvalues $\lambda_i \geq 0$ $(\lambda_i \leq 0), i = 1, 2, \ldots, n$

Hessian: properties

Second-order necessary (and sufficient) conditions for optimality rely on the notions of **positive (semi-)definiteness** along with second-order Taylor series expansion for f at given point $x \in \mathbb{R}^n$

The Hessian matrix $H_f(x) = \nabla^2 f(x)$ is

- positive (negative) semi-definite, i.e., $x^T Hx \geq 0$ $(x^T Hx \leq 0), \forall x \in \mathbb{R}^n$ **if and only if** its eigenvalues $\lambda_i \geq 0$ $(\lambda_i \leq 0), i = 1, 2, \ldots, n$
- positive (negative) definite, i.e., $x^T Hx > 0$ $(x^T Hx < 0), \forall x \in \mathbb{R}^n \setminus \{0\}$ **if and only if** its eigenvalue $\lambda_i > 0$ $(\lambda_i < 0), i = 1, 2, \ldots, n$

Hessian: properties

Second-order necessary (and sufficient) conditions for optimality rely on the notions of **positive (semi-)definiteness** along with second-order Taylor series expansion for f at given point $x \in \mathbb{R}^n$

The Hessian matrix $H_f(x) = \nabla^2 f(x)$ is

- positive (negative) semi-definite, i.e., $x^T Hx \geq 0$ $(x^T Hx \leq 0), \forall x \in \mathbb{R}^n$
  if and only if its eigenvalues $\lambda_i \geq 0$ $(\lambda_i \leq 0), i = 1, 2, \ldots, n$
- positive (negative) definite, i.e., $x^T Hx > 0$ $(x^T Hx < 0), \forall x \in \mathbb{R}^n \setminus \{0\}$
  if and only if its eigenvalue $\lambda_i > 0$ $(\lambda_i < 0), i = 1, 2, \ldots, n$

Note: a positive semidefinite (PSD) matrix is positive definite (PD) if and only if it is nonsingular (invertible)

Hessian: exercises

Exercises:

- Consider the function $f(x_1, x_2) = 2x_1x_2^3 - 2x_1^3x_2$
  - ◦) Find its stationary points
  - ◦) Determine expression of the Hessian (and its evaluation at these points)
  - ◦) Compute its leading principal minors (see below)
- Consider the function $f(x_1, x_2) = 4x_1^3 - 3x_2^3 - 2x_2^2 - x_1^2$
  - ◦) Find its stationary points
  - ◦) Determine expression of the Hessian (and its evaluation at these points)
  - ◦) Compute its leading principal minors (see below)
- Consider the function $f(x_1, x_2, x_3) = 3x_1^3 + 2x_2^2 - x_3^3$
  - ◦) Find its stationary points
  - ◦) Determine expression of the Hessian (and its evaluation at these points)
  - ◦) Compute its leading principal minors (see below)

Second-Order conditions

Theorem: Second-order necessary conditions

Let $f : \mathbb{R}^n \to \mathbb{R}$ be twice continuously differentiable at a point $x^\star \in \mathbb{R}^n$

- If $x^\star$ is a local minimum of f,

  then $\nabla f(x^\star) = 0$ and $\nabla^2 f(x^\star)$ is positive semi-definite ($\nabla^2 f(x^\star) \succeq 0$)
- If $x^\star$ is a local maximum of f,

  then $\nabla f(x^\star) = 0$ and $\nabla^2 f(x^\star)$ is negative semi-definite ($\nabla^2 f(x^\star) \preceq 0$)

Second-Order conditions

Theorem: Second-order necessary conditions

Let $f : \mathbb{R}^n \to \mathbb{R}$ be twice continuously differentiable at a point $x^\star \in \mathbb{R}^n$

- If $x^\star$ is a local minimum of f,

  then $\nabla f(x^\star) = 0$ and $\nabla^2 f(x^\star)$ is positive semi-definite ($\nabla^2 f(x^\star) \succeq 0$)
- If $x^\star$ is a local maximum of f,

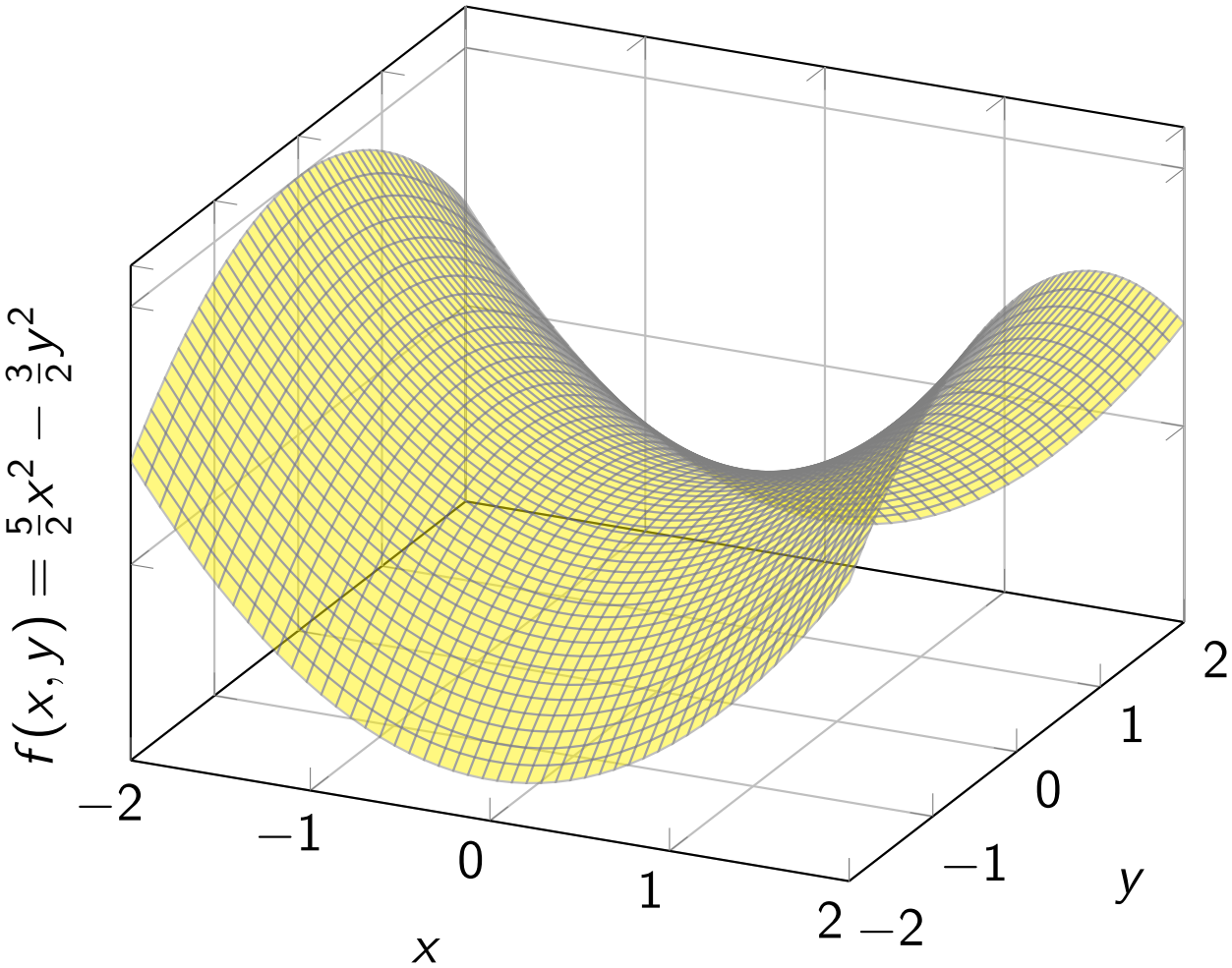
  then $\nabla f(x^\star) = 0$ and $\nabla^2 f(x^\star)$ is negative semi-definite ($\nabla^2 f(x^\star) \preceq 0$)

Theorem: Second-Order sufficient conditions

Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable at a point $x^\star \in \mathbb{R}^n$ and that $\nabla f(x^\star) = 0$ ($x^\star$ is a stationary point of f)

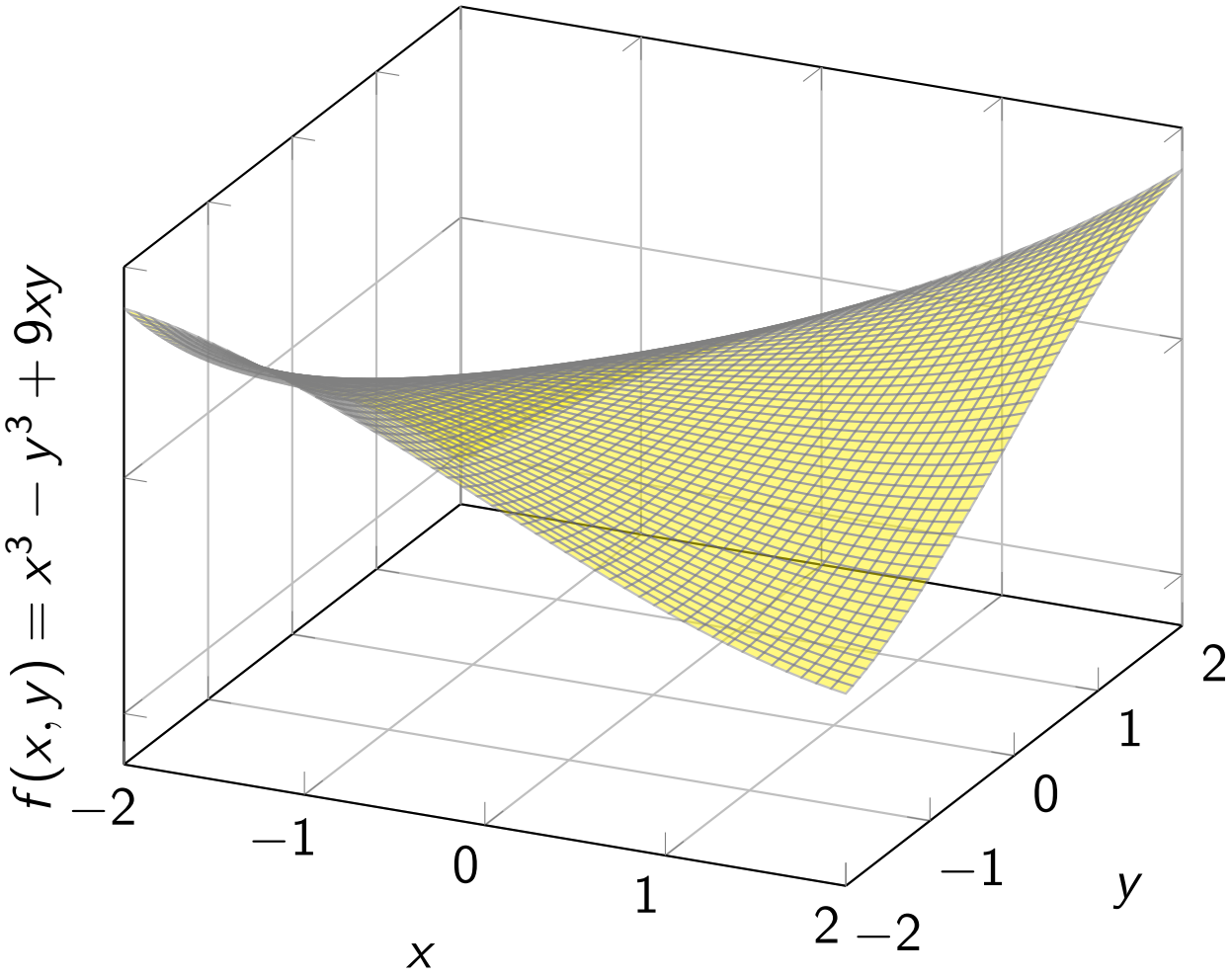
1) **If** $\nabla^2 f(x^\star)$ is positive definite, **then** $x^\star$ is a **strict local minimum** of f
2) **If** $\nabla^2 f(x^\star)$ is negative definite, **then** $x^\star$ is a **strict local maximum** of f
3) **If** $\nabla^2 f(x^\star)$ has both positive and negative eigenvalues, **then** $x^\star$ is neither a local maximum or a local minimum but referred as a **saddle-point** of f

Saddle point



Saddle Point: a point of a multi-variate function which is a stationary point but not an extremum (note: a saddle point is both a stationary point and an inflection point, whereas an inflection point is not a saddle point)

Saddle point



Second-Order conditions: Sufficient conditions

- Sufficient conditions are conditions on the (partial) derivatives of f that guarantee that a point $x^\star$ is a local minimum or maximum

Second-Order conditions: Sufficient conditions

- Sufficient conditions are conditions on the (partial) derivatives of f that guarantee that a point $x^\star$ is a local minimum or maximum
- Sufficient conditions are **not necessary**: a point $x^\star$ may be a strict local minimum, and yet may fail to satisfy the (second-order) sufficient conditions

Second-Order conditions: Sufficient conditions

- Sufficient conditions are conditions on the (partial) derivatives of f that guarantee that a point $x^\star$ is a local minimum or maximum
- Sufficient conditions are **not necessary**: a point $x^\star$ may be a strict local minimum, and yet may fail to satisfy the (second-order) sufficient conditions
- In order to apply the second-order sufficient condition one must be able to verify that a symmetric matrix (Hessian) is positive/negative definite

  1) By computing the **eigenvalues** of Hessian matrix and checking that they are all positive/negative

  2) By using the **principal minors** of Hessian matrix

Principal and Leading principal minor

Def: Leading principal minor

Let matrix $H \in \mathbb{R}^{n \times n}$ be symmetric. The kth **leading principal minor** of H, denoted $D_k(H)$ is the determinant of the upper-left $k \times k$ submatrix of H.

Principal and Leading principal minor

Def: Leading principal minor

Let matrix $H \in \mathbb{R}^{n \times n}$ be symmetric. The kth **leading principal minor** of H, denoted $D_k(H)$ is the determinant of the upper-left $k \times k$ submatrix of H.

Theorem: Let $H \in \mathbb{R}^{n \times n}$ be symmetric.

(1) H is positive definite **if and only if** $D_k(H) > 0, k = 1, 2, \ldots, n$.

(2) H is negative definite **if and only if** $(-1)^k D_k(H) > 0, k = 1, 2, \ldots, n$.

Thus, it is enough to check the inequality for all the leading principal minors, i.e., for $1 \leq k \leq n$

Principal and Leading principal minor

- **Example**: Consider the matrix

$$H = \begin{pmatrix} 1 & 3 & 2 \\ 1 & 5 & -2 \\ -1 & 5 & 1 \end{pmatrix}$$

Then, $D_1(H) = 1$, $D_2(H) = 5 - 3 = 2$, $D_3(H) = \det(H) = 21 - (-17) = 38$

Principal and Leading principal minor

- **Example**: Consider the matrix

$$H = \begin{pmatrix} 1 & 3 & 2 \\ 1 & 5 & -2 \\ -1 & 5 & 1 \end{pmatrix}$$

Then, $D_1(H) = 1$, $D_2(H) = 5 - 3 = 2$, $D_3(H) = \det(H) = 21 - (-17) = 38$

- **Example**: Determine the definiteness of the quadratic form

$$f(x_1, x_2, x_3) = 3x_1^2 + 6x_1x_3 + x_2^2 - 4x_2x_3 + 8x_3^2$$

The symmetric matrix associated with the quadratic form is

$$Q = \begin{pmatrix} 3 & 0 & 3 \\ 0 & 1 & -2 \\ 3 & -2 & 8 \end{pmatrix}$$

Since the leading principal minors are positive, Q is positive definite $Q \succ 0$

$$D_1(Q) = 3,\ D_2(Q) = 3 - 0 = 3,\ D_3(Q) = \det(Q) = 24 - 21 = 3$$

Principal and Leading principal minor: examples

Example: Let $f(x_1, x_2) = 2x_1 - x_2 - x_1^2 + x_1x_2 - x_2^2$

- Compute the Hessian H_f : since $\dfrac{\partial f}{\partial x_1} = 2 - 2x_1 + x_2$, and $\dfrac{\partial f}{\partial x_2} = -1 + x_1 - 2x_2$, we have

$$H_f = \begin{pmatrix} -2 & 1 \\ 1 & -2 \end{pmatrix}$$

- Since the leading principal minors are $D_1 = -2$ and $D_2 = 3$, the Hessian is negative definite (thus, f is strictly concave)

Principal and Leading principal minor: examples

Example: Let $f(x_1, x_2, x_3) = x_1 + x_2^2 + x_3^3 - x_1x_2 - 3x_3$

- Compute the Hessian H_f : since $\dfrac{\partial f}{\partial x_1} = 1 - x_2$, $\dfrac{\partial f}{\partial x_2} = 2x_2 - x_1$, and $\dfrac{\partial f}{\partial x_3} = 3x_3^2 - 3$, we have

$$H_f = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 2 & 0 \\ 0 & 0 & 6x_3 \end{pmatrix}$$

- Since the leading principal minors are $D_1 = 0$, $D_2 = -1$ and $D_3 = -6x_3$, the Hessian H_f is indefinite for all x

Principal and Leading principal minor: example

Example: Let $f : \mathbb{R}^2 \to \mathbb{R}$: $f(x_1, x_2) = x_1^3 - x_2^3 + 9x_1x_2$. The Hessian matrix H_f is

$$\begin{pmatrix} \dfrac{\partial^2 f}{\partial x_1 \partial x_1} & \dfrac{\partial^2 f}{\partial x_1 \partial x_2} \\ \dfrac{\partial^2 f}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f}{\partial x_2 \partial x_2} \end{pmatrix} = \begin{pmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1 \partial x_2} \\ \dfrac{\partial^2 f}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} \end{pmatrix} = \begin{pmatrix} 6x_1 & 9 \\ 9 & -6x_2 \end{pmatrix}$$

The principal minors of H_f are

$$\begin{aligned} D_1(H_f(x_1, x_2)) = 6x_1 \quad &\text{and} \quad D_2(H_f(x_1, x_2)) = \det(H_f(x_1, x_2)) = -36x_1x_2 - 81 \\ D_1(H_f(0, 0)) = 0 \quad &\text{and} \quad D_2(H_f(0, 0)) = -81 \quad (\Rightarrow H_f \text{ indefinite}) \\ D_1(H_f(3, -3)) = 18 \quad &\text{and} \quad D_2(H_f(3, -3)) = 243 \quad (\Rightarrow H_f \succ 0)) \end{aligned}$$

- Hence, $(0; 0)$ is a **saddle-point** of f, and $(3; -3)$ is a **local minimum** of f

Principal and Leading principal minor: example

Example: Let $f : \mathbb{R}^2 \to \mathbb{R}$: $f(x_1, x_2) = x_1^3 - x_2^3 + 9x_1x_2$. The Hessian matrix H_f is

$$\begin{pmatrix} \dfrac{\partial^2 f}{\partial x_1 \partial x_1} & \dfrac{\partial^2 f}{\partial x_1 \partial x_2} \\ \dfrac{\partial^2 f}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f}{\partial x_2 \partial x_2} \end{pmatrix} = \begin{pmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1 \partial x_2} \\ \dfrac{\partial^2 f}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} \end{pmatrix} = \begin{pmatrix} 6x_1 & 9 \\ 9 & -6x_2 \end{pmatrix}$$

The principal minors of H_f are

$$\begin{aligned} D_1(H_f(x_1, x_2)) = 6x_1 \quad &\text{and} \quad D_2(H_f(x_1, x_2)) = \det(H_f(x_1, x_2)) = -36x_1x_2 - 81 \\ D_1(H_f(0, 0)) = 0 \quad &\text{and} \quad D_2(H_f(0, 0)) = -81 \quad (\Rightarrow H_f \text{ indefinite}) \\ D_1(H_f(3, -3)) = 18 \quad &\text{and} \quad D_2(H_f(3, -3)) = 243 \quad (\Rightarrow H_f \succ 0)) \end{aligned}$$

- Hence, $(0; 0)$ is a **saddle-point** of f, and $(3; -3)$ is a **local minimum** of f
- Note also that the point $(3; -3)$ is not a global minimum of f because at the point $(0; n)$ the function $f(0; n) = -n^3$; thus, $f \to -\infty$ with $n \to +\infty$

Principal and Leading principal minor

Def: Principal minor

Let the matrix $H \in \mathbb{R}^{n \times n}$ be symmetric. The kth minor of H, denoted $\Delta_k(H)$ is a **principal minor** if it is obtained by deleting $n-k$ rows and the $n-k$ columns with the same numbers. Thus, there are $\binom{n}{k}$ principal minors of order k.

Principal and Leading principal minor

Def: Principal minor

Let the matrix $H \in \mathbb{R}^{n \times n}$ be symmetric. The kth minor of H, denoted $\Delta_k(H)$ is a **principal minor** if it is obtained by deleting $n-k$ rows and the $n-k$ columns with the same numbers. Thus, there are $\binom{n}{k}$ principal minors of order k.

Theorem: Let $H \in \mathbb{R}^{n \times n}$ be symmetric.

(1) H is positive semi-definite **if and only if** $\Delta_k(H) \geq 0$ for all principal minors

(2) H is negative semi-definite **if and only if** $(-1)^k \Delta_k(H) \geq 0$ for all principal minors

Principal and Leading principal minor

Def: Principal minor

Let the matrix $H \in \mathbb{R}^{n \times n}$ be symmetric. The kth minor of H, denoted $\Delta_k(H)$ is a **principal minor** if it is obtained by deleting $n-k$ rows and the $n-k$ columns with the same numbers. Thus, there are $\binom{n}{k}$ principal minors of order k.

Theorem: Let $H \in \mathbb{R}^{n \times n}$ be symmetric.

(1) H is positive semi-definite **if and only if** $\Delta_k(H) \geq 0$ for all principal minors

(2) H is negative semi-definite **if and only if** $(-1)^k \Delta_k(H) \geq 0$ for all principal minors

$\Rightarrow$ Warning: to apply this method, we must check for all principal minors, i.e., for each k with $1 \leq k \leq n$ and for each of the $\binom{n}{k}$ principal minors of order k

Principal and Leading principal minor

Def: Indefinite matrix

Let $H \in \mathbb{R}^{n \times n}$ be symmetric

- If H is neither positive definite or negative definite -**and**- *all of its principal minors are non-zero*
- Then H has both positive and negative eigenvalues. In this case, the matrix H is indefinite

- **Sufficient condition** for indefiniteness: if matrix has zero diagonal element that has a nonzero element in its row (rules out matrix being PSD or NSD)

Principal and Leading principal minor

Def: Indefinite matrix

Let $H \in \mathbb{R}^{n \times n}$ be symmetric

- If H is neither positive definite or negative definite -**and**- *all of its principal minors are non-zero*
- Then H has both positive and negative eigenvalues. In this case, the matrix H is indefinite

- **Sufficient condition** for indefiniteness: if matrix has zero diagonal element that has a nonzero element in its row (rules out matrix being PSD or NSD)
- **Example**: saddle point matrix, block 2×2 form

$$C = \begin{bmatrix} B & A^T \\ A & 0 \end{bmatrix}$$

  where B is symmetric PD and $A \neq 0$. When B identity matrix, C augmented system matrix associated with least squares problem
- **Example**: reverse identity matrix which has eigenvalues ± 1

Optimality conditions: Convex functions

- Since so far : determined first- and second-order conditions for optimality based on **local information** (only) $\rightarrow$ deduce properties of **local extrema**
- Goal : exploit convexity of sets and functions to provide (sufficient) conditions for **global optimality**

Optimality conditions: Convex functions

Theorem 1

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function

(i) Any local minimum $x^\star$ of f is a global minimum of f

(ii) If in addition f is differentiable, then any stationary point $x^\star$ of f (i.e., $\nabla f(x^\star) = 0$) is a global minimum of f

Optimality conditions: Convex functions

Theorem 1

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function

(i) Any local minimum $x^\star$ of f is a global minimum of f

(ii) If in addition f is differentiable, then any stationary point $x^\star$ of f (i.e., $\nabla f(x^\star) = 0$) is a global minimum of f

Proof: 2 parts

(i) Suppose $x^\star$ local but not global minimum; then, $\exists\, y \in \mathbb{R}^n$ s.t. $f(y) < f(x^\star)$

Consider line segment that joins $x^\star$ to y, $x_\alpha = \alpha y + (1-\alpha)x^\star$, for $\alpha \in]0,1]$

By convexity property of f, we can deduce (since $f(y) < f(x^\star)$)

$$f(x_\alpha) = f(\alpha y + (1-\alpha)x^\star) \leq \alpha f(y) + (1-\alpha) f(x^\star) < f(x^\star) \tag{15}$$

Any neighborhood $\mathcal{N}$ of $x^\star$ contains a piece of the line segment x_α

Thus, there will always be points $x \in \mathrm{dom}(f) \cap \mathcal{N}$ at which (15) is satisfied

$\Rightarrow$ $x^\star$ is **not** a local minimum

Optimality conditions: Convex functions

Part (ii)

- Suppose that $x^\star$ is **not** a global minimum and choose y as in Part (i)
- Since f differentiable, the **directional derivative** of f at $x^\star$, $Df(x^\star; d) = \nabla f(x^\star)^\top d$; thus, by choosing direction $d = y - x^\star$, we obtain

$$\nabla f(x^\star)^\top d = \nabla f(x^\star)^\top (y - x^\star) \overset{\text{By def.}}{=} \lim_{\alpha \to 0} \frac{f(x^\star + \alpha(y - x^\star)) - f(x^\star)}{\alpha} \tag{16}$$

- Since f is assumed convex

$$\begin{aligned}
\nabla f(x^\star)(y - x^\star) &\overset{\text{By def.}}{=} \lim_{\alpha \to 0} \frac{f(x^\star + \alpha(y - x^\star)) - f(x^\star)}{\alpha} && (16)\\
&= \lim_{\alpha \to 0} \frac{f((1-\alpha)x^\star + \alpha y) - f(x^\star)}{\alpha} && (17)\\
&\leq \lim_{\alpha \to 0} \frac{\alpha f(y) + (1-\alpha) f(x^\star) - f(x^\star)}{\alpha} && (18)\\
&= f(y) - f(x^\star) < 0 && (19)
\end{aligned}$$

- Therefore, $\nabla f(x^\star) < 0$, and so $x^\star$ is **not** a stationary point of f

Optimality conditions: Convex functions

Consequence: whenever f is convex

- if $x^\star$ is a local solution (optimizer) to the minimization problem

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

- then $x^\star$ is a global solution (optimizer) to this problem

Optimality conditions: Convex functions

Consequence: whenever f is convex

- if $x^\star$ is a local solution (optimizer) to the minimization problem

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

- then $x^\star$ is a global solution (optimizer) to this problem

Moreover, **strict convexity** of f implies uniqueness of the solution

Optimality conditions: Convex functions

Consequence: whenever f is convex

- if $x^\star$ is a local solution (optimizer) to the minimization problem

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

- then $x^\star$ is a global solution (optimizer) to this problem

Moreover, **strict convexity** of f implies uniqueness of the solution

Theorem 2

Let $f : \mathbb{R}^n \to \mathbb{R}$ be strictly convex. If f has a global minimizer, then it is unique.

Optimality conditions: Strictly convex functions

Theorem 2

Let $f : \mathbb{R}^n \to \mathbb{R}$ be strictly convex. If f has a global minimizer, then it is unique.

Proof

- Let x_1 and x_2 be distinct global minimizers of f

  Then, for $\alpha \in]0, 1[$, we have, following definition of strict convexity,

$$f(\alpha x_1 + (1-\alpha)x_2) < \alpha f(x_1) + (1-\alpha)f(x_2) \quad (i)$$

- Since $\alpha \in]0, 1[$, the following is verified $f(x_1) \leq f(x_2) \Rightarrow \alpha f(x_1) \leq \alpha f(x_2)$
  Thus, adding by $(1-\alpha)f(x_2)$ on both sides, we obtain

$$\alpha f(x_1) + (1-\alpha)f(x_2) \leq \alpha f(x_2) + (1-\alpha)f(x_2) = f(x_2) \quad (ii)$$

- Thus, bringing (i) and (ii) together, we have

$$f(\alpha x_1 + (1-\alpha)x_2) < f(x_2)$$

  which contradicts the assumption that x_2 is a global minimizer of f

Optimality conditions: Convex functions

Theorem 3

Let function $f : \mathbb{R}^n \to \mathbb{R}$

1. If f is differentiable on $\mathbb{R}^n$, then the following statements are equivalent

   (a) f **is convex**, Jensen inequality (2) hold for $x, y \in \mathrm{dom}(f)$

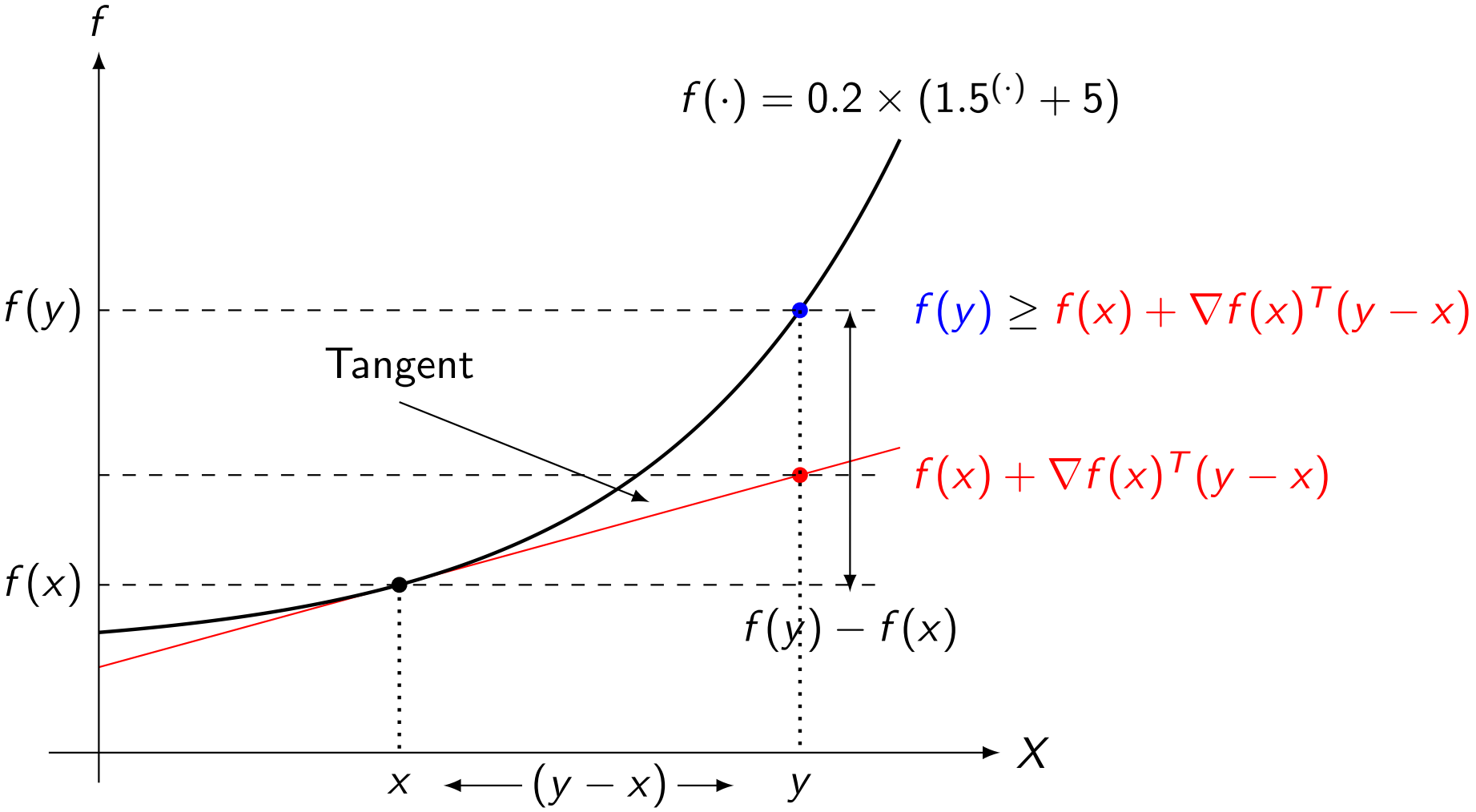
   (b) $f(y) \geq f(x) + \nabla f(x)^\intercal (y - x)$ **for all** $x, y \in \mathbb{R}^n$ (sub-differential inequality)

   (c) $(\nabla f(y) - \nabla f(x))^\intercal (y - x) \geq 0$ for all $x, y \in \mathbb{R}^n$ (gradient monotonicity)

2. If f is twice differentiable, then f is convex if and only if $\nabla^2 f(x)$ is positive semi-definite for all $x \in \mathbb{R}^n$

Optimality conditions: Convex functions

Geometric interpretation of sub-differential inequality



Optimality conditions: Convex functions

This equivalence also applies to functions defined on open convex subsets $\mathcal{C}$ of $\mathbb{R}^n$

Theorem 4

Let function $f : \mathcal{C} \subseteq \mathbb{R}^n \to \mathbb{R}$ and $\mathcal{C}$ open convex set

1. If f is differentiable on $\mathcal{C} \subseteq \mathbb{R}^n$, then the following statements are equivalent
   (a) f **is convex on** $\mathcal{C}$,
   (b) $f(y) \geq f(x) + \nabla f(x)^\intercal (y - x)$ **for all** $x, y \in \mathcal{C}^2$ (sub-differential inequality)
   (c) $(\nabla f(y) - \nabla f(x))^\intercal (y - x) \geq 0$ for all $x, y \in \mathcal{C}^2$ (gradient monotonicity)
2. If f is twice differentiable on $\mathcal{C} \subseteq \mathbb{R}^n$, then f is convex if and only if $\nabla^2 f(x)$ is positive semi-definite for all $x \in \mathcal{C}$

Optimality conditions: Convex functions

Proof (a) $\rightarrow$ (b)

- Suppose f is convex, then for any $x, y\,(\in \mathcal{C} \subset)\,\mathbb{R}^n$ and $\alpha \in]0, 1]$, we have

$$f((1-\alpha)x + \alpha y) \leq (1-\alpha)f(x) + \alpha f(y)$$

- Rearrange terms to obtain

$$f(x + \alpha(y-x)) \leq f(x) + \alpha(f(y) - f(x))$$

Thus,

$$\frac{f(x + \alpha(y-x)) - f(x)}{\alpha} \leq f(y) - f(x)$$

- Since f differentiable, taking the limit as $\alpha \rightarrow 0$ yields following def. of **directional derivative** at x in direction $d = y - x$

$$\nabla f(x)^T (y-x) \leq f(y) - f(x)$$

Thus,

$$f(y) \geq f(x) + \nabla f(x)^T (y-x)$$

Optimality conditions: Convex functions

Proof (b) $\rightarrow$ (c)

- If f is differentiable and convex, then (subdifferential inequality)

$$\begin{cases} f(y) & \geq f(x) + \nabla f(x)^\top (y - x) \\ f(x) & \geq f(y) + \nabla f(y)^\top (x - y) \end{cases}$$

- Adding these inequalities, we obtain

$$\begin{aligned} f(y) + f(x) &\geq f(x) + f(y) + \nabla f(x)^\top (y - x) + \nabla f(y)^\top (x - y) \\ \iff 0 &\geq \nabla f(x)^\top (y - x) + \nabla f(y)^\top (x - y) \end{aligned}$$

- Thus, by re-arranging the terms, the gradient of f, $\nabla f : \mathbb{R}^n \rightarrow \mathbb{R}$ defines a monotone mapping

$$\big(\nabla f(y) - \nabla f(x)\big)^\top (y - x) \geq 0$$

- Note: observe (c) is equivalent to

$$\nabla f(y)^\top (y - x) \geq \nabla f(x)^\top (y - x)$$

Optimality conditions: Convex functions

Proof (c) $\rightarrow$ (b)

- Let $g(t) = f(x + t(y - x))$. Thus, $\dfrac{dg}{dt} = g'(t) = \nabla f(x + t(y - x))^\top (y - x)$.
- If ∇f is monotone, then $g'(t) \geq g'(0) = \nabla f(x)^\top (y - x)$ for $t \geq 0, t \in \mathrm{dom}(g)$
- Since,

$$\begin{cases} g(1) & = f(x + (y - x)) = f(y) \\ g(0) & = f(x) \end{cases}$$

  we can write

$$f(y) = g(1) = g(0) + \int_0^1 g'(t)dt \geq g(0) + g'(0)$$
$$= f(x) + \nabla f(x)^\top (y - x)$$

- Consequently, we obtain the first-order condition for convexity

$$f(y) \geq f(x) + \nabla f(x)^\top (y - x)$$

INFO-F524: Continuous Optimization
–Section 1 : Application–

Bernard Fortz, Dimitri Papadimitriou

Bureau: **2N3 206**

Université Libre de Bruxelles

dimitrios.papadimitriou@ulb.be

Annee Académique: 2024-2025

Least Squares (LS)

- Consider the system of linear equations $Ax = b$, where $A \in \mathbb{R}^{m \times n}$, $x \in \mathbb{R}^n$, and $b \in \mathbb{R}^m$
- When $m > n$, this problem has generally no solution because the linear system of equations is overdetermined

Least Squares (LS)

- Consider the system of linear equations $Ax = b$, where $A \in \mathbb{R}^{m \times n}$, $x \in \mathbb{R}^n$, and $b \in \mathbb{R}^m$
- When $m > n$, this problem has generally no solution because the linear system of equations is overdetermined

For $m > n$, instead of solving this problem exactly, the goal is instead to approximately solve this overdetermined system of linear equations

Least Squares (LS)

Least squares (LS): least squares approximation of functions to data points

→ **Linear** LS (LLS): least squares approximation of linear functions to data points

Least Squares (LS)

Least squares (LS): least squares approximation of functions to data points

→ **Linear** LS (LLS): least squares approximation of linear functions to data points **Goal**: find (value of) variable x that minimizes the error $\|Ax - b\|^2$,

where $\|\cdot\|$ denotes the Euclidean norm induced by standard scalar product in $\mathbb{R}^n$

Least Squares (LS)

Least squares (LS): least squares approximation of functions to data points

→ **Linear** LS (LLS): least squares approximation of linear functions to data points **Goal**: find (value of) variable x that minimizes the error $\|Ax - b\|^2$,

where $\|\cdot\|$ denotes the Euclidean norm induced by standard scalar product in $\mathbb{R}^n$

- Nonlinear LS (NLLS): Section 3

Least Squares (LS)

Least squares (LS): least squares approximation of functions to data points

→ **Linear** LS (LLS): least squares approximation of linear functions to data points **Goal**: find (value of) variable x that minimizes the error $\|Ax - b\|^2$,

where $\|\cdot\|$ denotes the Euclidean norm induced by standard scalar product in $\mathbb{R}^n$

- Nonlinear LS (NLLS): Section 3

Least Squares Method (1)

- Consider $Ax = b$, where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ For $m > n$, find value of x that minimizes error $\|Ax - b\|^2$: least squares problem

$$\underset{x \in \mathbb{R}^m}{\text{minimize}} \ \frac{1}{2}\|Ax - b\|^2 \tag{1}$$

Least Squares Method (1)

- Consider $Ax = b$, where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ For $m > n$, find value of x that minimizes error $\|Ax - b\|^2$: least squares problem

$$\underset{x \in \mathbb{R}^m}{\text{minimize}} \ \frac{1}{2}\|Ax - b\|^2 \tag{1}$$

- Equivalent to **quadratic programming (QP)** problem

$$\min_x \ \frac{1}{2}x^\top Qx + c^\top x, \text{ with } Q = A^\top A \text{ and } c = -A^\top b,$$

since $\|Ax - b\|^2 = (Ax - b)^\top (Ax - b) = x^\top A^\top Ax - 2b^\top Ax + b^\top b$

Convex QP because Q is **by definition** positive semi-definite: $Q = A^T A \succeq 0$

Least Squares Method and QP

Proposition

Function $f : \mathbb{R}^n \to \mathbb{R} \colon x \to \frac{1}{2}x^\top Qx + c^\top x$ is convex if and only if Q is positive semi-definite (PSD) $Q \succeq 0$.

Proof ($\Leftarrow$): Suppose that Q is PSD. For all $\alpha \in [0, 1]$, and for all x, y we have

$$
\begin{aligned}
f(\alpha x + (1-\alpha)y) &= f(y + \alpha(x-y)) \\
&= \frac{1}{2}(y + \alpha(x-y))^\top Q(y + \alpha(x-y)) + c^\top(y + \alpha(x-y)) \\
&= \frac{1}{2}y^\top Qy + \alpha(x-y)^\top Qy + \frac{1}{2}\alpha^2(x-y)^\top Q(x-y) + \alpha c^\top x + (1-\alpha)c^\top y \\
&\leq \frac{1}{2}y^\top Qy + \alpha(x-y)^\top Qy + \frac{1}{2}\alpha(x-y)^\top Q(x-y) + \alpha c^\top x + (1-\alpha)c^\top y \\
&= \frac{1}{2}\alpha x^\top Qx + \alpha c^\top x + \frac{1}{2}(1-\alpha)y^\top Qy + (1-\alpha)c^\top y \\
&= \alpha\left(\frac{1}{2}x^\top Qx + c^\top x\right) + (1-\alpha)\left(\frac{1}{2}y^\top Qy + c^\top y\right) \\
&= \alpha f(x) + (1-\alpha)f(y) \quad \text{thus, the function } f(x) \text{ is a convex function}
\end{aligned}
$$

Least Squares Method and QP

Proof ($\Rightarrow$): Suppose that f is convex. Using the definition of convex function and focusing on quadratic part, for all $\alpha \in [0, 1]$

$$\begin{aligned}\alpha x^\top Qx + (1-\alpha)y^\top Qy &\geq (\alpha x + (1-\alpha)y)^\top Q(\alpha x + (1-\alpha)y) \\ &= \alpha^2 x^\top Qx + (1-\alpha)^2 y^\top Qy + \alpha(1-\alpha)x^\top Qy + \alpha(1-\alpha)y^\top Qx\end{aligned}$$

By rearranging the terms and simplifying

$$\begin{aligned}&\alpha(1-\alpha)x^\top Qx + \alpha(1-\alpha)y^\top Qy - \alpha(1-\alpha)x^\top Qy - \alpha(1-\alpha)y^\top Qx \geq 0 \\ &\Rightarrow x^\top Qx + y^\top Qy - x^\top Qy - y^\top Qx \geq 0 \\ &\Rightarrow (x-y)^\top Q(x-y) \geq 0\end{aligned}$$

which is true for positive semi-definite $Q \succeq 0$ $\square$

Least Squares Method and QP

Proposition

Function $f : \mathbb{R}^n \to \mathbb{R}$: $x \to \frac{1}{2}x^\top Qx + c^\top x$ is strictly convex if and only if Q is positive definite (PD) $Q \succ 0$

[1] The range (also called the column space or image) of a $m \times n$ matrix A is the set of all possible linear combinations of its m-dim column vectors

Least Squares Method and QP

Proposition

Function $f : \mathbb{R}^n \to \mathbb{R}: x \to \frac{1}{2}x^\top Qx + c^\top x$ is strictly convex if and only if Q is positive definite (PD) $Q \succ 0$

Note: If $Q \succeq 0$ but not positive definite, the problem can be

- unbounded (e.g., take $Q = 0, c \neq 0$)
- bounded below and have infinitely many solutions (e.g., $f(x = (x_1, x_2)) = (x_1 - x_2)^2$)

$\to$ Distinguish between the two cases by testing if $c \in range(Q)$[1]

[1] The range (also called the column space or image) of a $m \times n$ matrix A is the set of all possible linear combinations of its m-dim column vectors

Least Squares Method and QP

Important: general QP model, one always presume that Q is a symmetric matrix (i.e., $Q^\top = Q$), because

$$\frac{1}{2}x^\top \tilde{Q} x = x^\top (Q + Q^\top) x$$

thus, Q can be replaced by the symmetric matrix $\tilde{Q} := \frac{1}{2}(Q + Q^\top)$

Least Squares Method: Description

Input : finite set of **m data points** $(a_1, b_1), \ldots, (a_m, b_m)$ also denoted

$$\{(a_i, b_i)\}_{i=1}^{m}, \; m < \infty$$

- Each observation i is represented by a data point $(a_i, b_i) \in \mathbb{R} \times \mathbb{R}$

Least Squares Method: Description

Input : finite set of **m data points** $(a_1, b_1), \ldots, (a_m, b_m)$ also denoted

$$\{(a_i, b_i)\}_{i=1}^{m}, \; m < \infty$$

- Each observation i is represented by a data point $(a_i, b_i) \in \mathbb{R} \times \mathbb{R}$

Description

- With **two** data points ($m = 2$): fit unique line segment

Least Squares Method: Description

Input : finite set of **m data points** $(a_1, b_1), \ldots, (a_m, b_m)$ also denoted

$$\{(a_i, b_i)\}_{i=1}^{m}, \; m < \infty$$

- Each observation i is represented by a data point $(a_i, b_i) \in \mathbb{R} \times \mathbb{R}$

Description

- With **two** data points ($m = 2$): fit unique line segment
- With **three** data points **or more** ($m \geq 3$): it is unlikely that all data points will fit exactly on a single straight line segment

Least Squares Method: Description

Input : finite set of **m data points** $(a_1, b_1), \ldots, (a_m, b_m)$ also denoted

$$\{(a_i, b_i)\}_{i=1}^{m}, \ m < \infty$$

- Each observation i is represented by a data point $(a_i, b_i) \in \mathbb{R} \times \mathbb{R}$

Description

- With **two** data points ($m = 2$): fit unique line segment
- With **three** data points **or more** ($m \geq 3$): it is unlikely that all data points will fit exactly on a single straight line segment

$\Rightarrow$ **Goal**: Find a straight line segment (linear function) that would best fit m data points $(a_1, b_1), \ldots, (a_m, b_m)$, i.e., $\{(a_i, b_i)\}_{i=1}^{m}, \ m < \infty$

a.k.a. fit a linear function graph that best adjusts (and thus may represent) these data points

Least Squares Method (1)

Let linear function $\mathbb{R} \to \mathbb{R}$: $\hat{b} = ax + x_0$ (x slope and x_0 intercept with the vertical axis)

Least Squares Method (1)

Let linear function $\mathbb{R} \to \mathbb{R}$: $\hat{b} = ax + x_0$ (x slope and x_0 intercept with the vertical axis)

- Compute distance along **vertical** axis between each data point and graph of the linear function

Least Squares Method (1)

Let linear function $\mathbb{R} \to \mathbb{R}$: $\hat{b} = ax + x_0$ (x slope and x_0 intercept with the vertical axis)

- Compute distance along **vertical** axis between each data point and graph of the linear function
- **Distance** d_i (vertical) between the data point (a_i, b_i) and the straight line at $(a_i, \hat{b}_i)$

$$\begin{aligned} d_i &= \sqrt{(a_i - a_i)^2 + (b_i - \hat{b}_i)^2} \\ &= \sqrt{(a_i - a_i)^2 + (b_i - (a_i x + x_0))^2} = |b_i - (xa_i + x_0)| \end{aligned} \tag{2}$$

Least Squares Method (1)

Let linear function $\mathbb{R} \to \mathbb{R}$: $\hat{b} = ax + x_0$ (x slope and x_0 intercept with the vertical axis)

- Compute distance along **vertical** axis between each data point and graph of the linear function
- **Distance** d_i (vertical) between the data point (a_i, b_i) and the straight line at $(a_i, \hat{b}_i)$

$$\begin{aligned} d_i &= \sqrt{(a_i - a_i)^2 + (b_i - \hat{b}_i)^2} \\ &= \sqrt{(a_i - a_i)^2 + (b_i - (a_i x + x_0))^2} = |b_i - (xa_i + x_0)| \end{aligned} \tag{2}$$

- Find a straight line that minimizes the square of distance for all data points, i.e., the sum over i (for $i = 1, \ldots, m$) of the squared distances d_i^2 defined by

$$\begin{aligned} S(x, x_0) &= \sum_{i=1}^{m} d_i^2 \\ &= \sum_{i=1}^{m} \left((a_i x + x_0) - b_i\right)^2 \end{aligned} \tag{3}$$

Least Squares Method (2)

- Now, we aim at finding a straight line that minimizes the sum of the squared distances d_i^2 over the vector (x, x_0)

Least Squares Method (2)

- Now, we aim at finding a straight line that minimizes the sum of the squared distances d_i^2 over the vector (x, x_0)
- Thus, the optimization problem consists of solving:

$$\underset{x,x_0}{\operatorname{argmin}}\ S(x, x_0) = \sum_{i=1}^{m} \left((a_i x + x_0) - b_i\right)^2 \tag{4}$$

- First-order conditions:

$$\begin{cases} \dfrac{\partial S}{\partial x} = 0 & \Leftrightarrow \displaystyle\sum_{i=1}^{m} 2((a_i x + x_0) - b_i)a_i = 0 \\ \dfrac{\partial S}{\partial x_0} = 0 & \Leftrightarrow \displaystyle\sum_{i=1}^{m} 2((a_i x + x_0) - b_i) = 0 \end{cases}$$

Least Squares Method (3)

- First-order conditions:

$$\begin{cases} \dfrac{\partial S}{\partial x} = 0 & \Leftrightarrow \displaystyle\sum_{i=1}^{m} 2((a_i x + x_0) - b_i)a_i = 0 \\ \dfrac{\partial S}{\partial x_0} = 0 & \Leftrightarrow \displaystyle\sum_{i=1}^{m} 2((a_i x + x_0) - b_i) = 0 \end{cases}$$

- This system can be rewritten as follows

$$\begin{cases} \left(\displaystyle\sum_{i=1}^{m} a_i^2\right) x + & \left(\displaystyle\sum_{i=1}^{m} a_i\right) x_0 & = \displaystyle\sum_{i=1}^{n} a_i b_i \\ \left(\displaystyle\sum_{i=1}^{m} a_i\right) x + & m x_0 & = \displaystyle\sum_{i=1}^{m} b_i \end{cases}$$

Least Squares Method (4)

- This system can be rewritten as follows

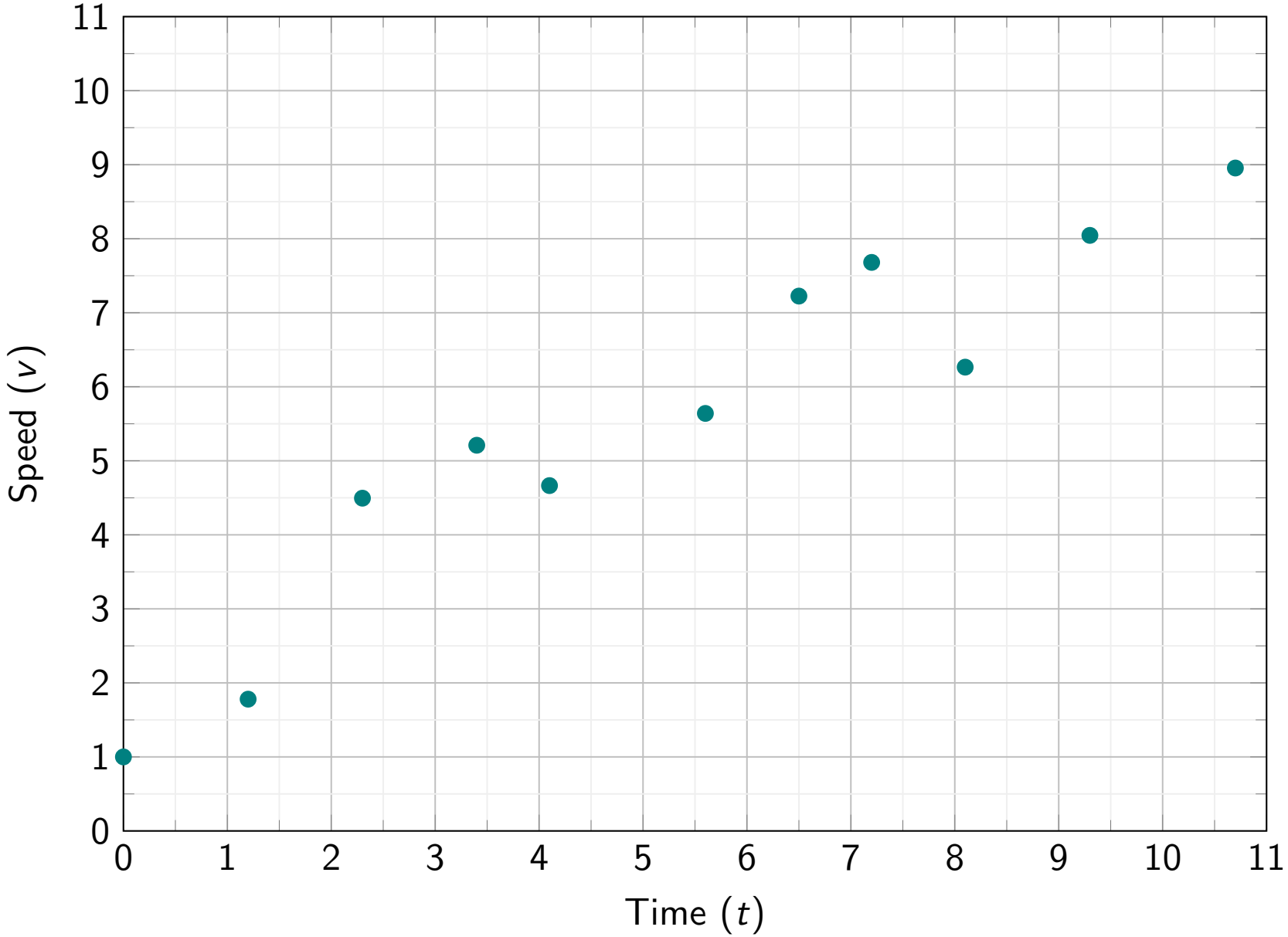
$$\begin{cases} \left(\sum_{i=1}^{m} a_i^2\right) x + \left(\sum_{i=1}^{m} a_i\right) x_0 & = \sum_{i=1}^{n} a_i b_i \\ \left(\sum_{i=1}^{m} a_i\right) x + m x_0 & = \sum_{i=1}^{m} b_i \end{cases}$$

- The solution is given by

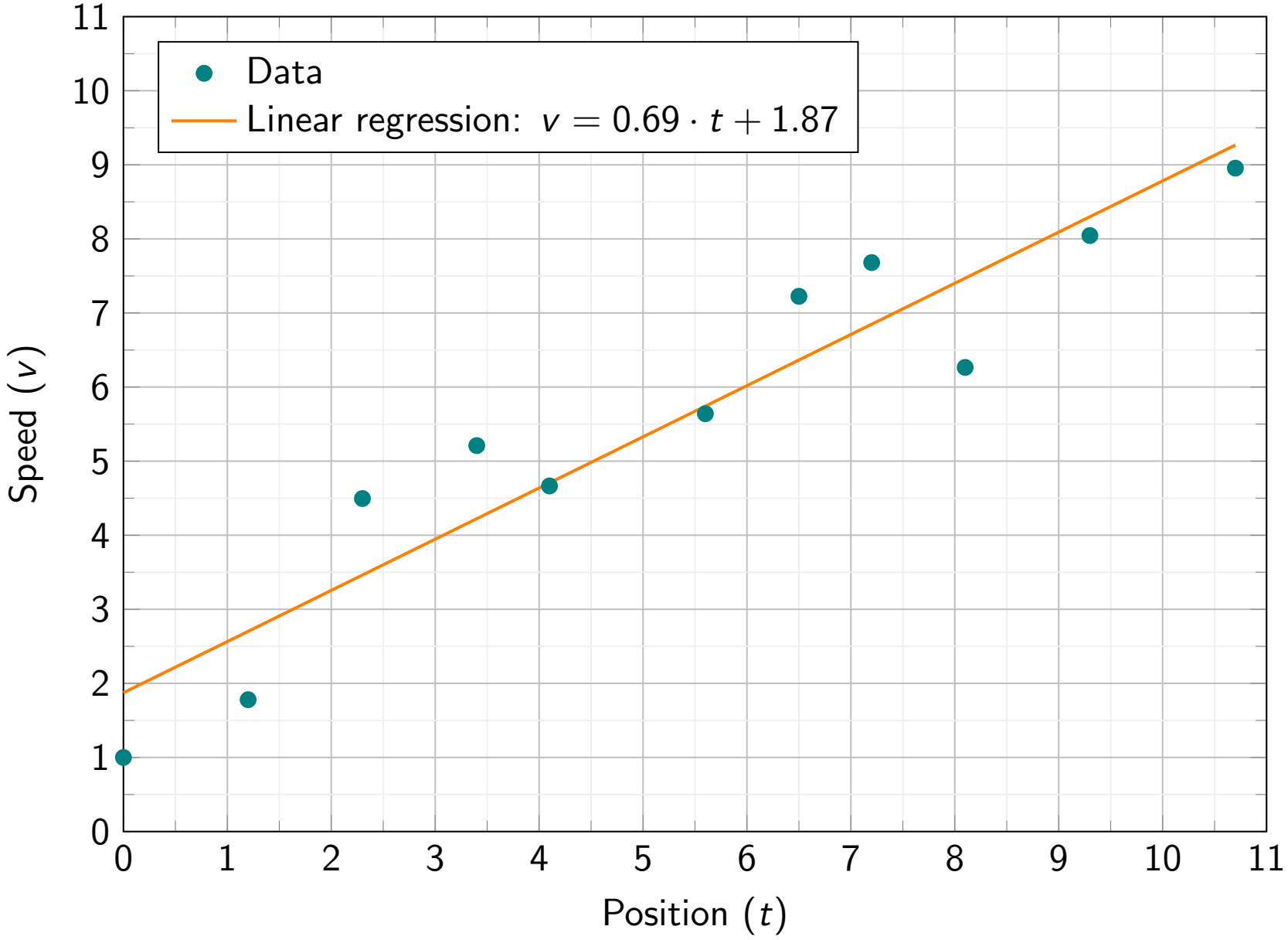
$$\begin{cases} x^\star & = \dfrac{m \sum_{i=1}^{m} a_i b_i - (\sum_{i=1}^{m} a_i)(\sum_{i=1}^{m} b_i)}{m \sum_{i=1}^{m} a_i^2 - (\sum_{i=1}^{m} a_i)^2} \\ x_0^\star & = \dfrac{(\sum_{i=1}^{m} a_i^2)(\sum_{i=1}^{m} b_i) - (\sum_{i=1}^{m} a_i)(\sum_{i=1}^{m} a_i b_i)}{m \sum_{i=1}^{m} a_i^2 - (\sum_{i=1}^{m} a_i)^2} \end{cases}$$

- Note : other first-order iterative methods for solving (1) and (4) will be detailed throughout this course.

Least Squares: Example (2)



Least Squares (2)



LS method and Linear regression

Link between LS method and **Regression** :

- The straight line $\hat{b} = ax^\star + x_0^\star$ is called the **regression line**

LS method and Linear regression

Link between LS method and **Regression** :

- The straight line $\hat{b} = ax^\star + x_0^\star$ is called the **regression line**
- The solution $x^\star$ and $x_0^\star$ are referred to as the **regression parameters** (a.k.a. **regression coefficients**) and $x_0^\star$ as the intercept term (if one is included in the regression model)

LS method and Linear regression

Link between LS method and **Regression** :

- The straight line $\hat{b} = ax^\star + x_0^\star$ is called the **regression line**
- The solution $x^\star$ and $x_0^\star$ are referred to as the **regression parameters** (a.k.a. **regression coefficients**) and $x_0^\star$ as the intercept term (if one is included in the regression model)

Linear regression (LR)

- **Regression line** : straight line that describes how scalar response b varies as the input a changes

LS method and Linear regression

Link between LS method and **Regression** :

- The straight line $\hat{b} = ax^\star + x_0^\star$ is called the **regression line**
- The solution $x^\star$ and $x_0^\star$ are referred to as the **regression parameters** (a.k.a. **regression coefficients**) and $x_0^\star$ as the intercept term (if one is included in the regression model)

Linear regression (LR)

- **Regression line** : straight line that describes how scalar response b varies as the input a changes
- Estimates the **linear relationship** between a **scalar response** b (a.k.a. dependent variable, response variable or regressand in regression analysis) and the **input** a (a.k.a. independent variable(s), explanatory variable(s) or regressor(s) in regression analysis)

LS method and Linear regression

Link between LS method and **Regression** :

- The straight line $\hat{b} = ax^\star + x_0^\star$ is called the **regression line**
- The solution $x^\star$ and $x_0^\star$ are referred to as the **regression parameters** (a.k.a. **regression coefficients**) and $x_0^\star$ as the intercept term (if one is included in the regression model)

Linear regression (LR)

- **Regression line** : straight line that describes how scalar response b varies as the input a changes
- Estimates the **linear relationship** between a **scalar response** b (a.k.a. dependent variable, response variable or regressand in regression analysis) and the **input** a (a.k.a. independent variable(s), explanatory variable(s) or regressor(s) in regression analysis)
- Note: variable x of the corresponding LS minimization problem belongs to $\mathbb{R}$ (single dimensional)

Multi-Linear Regression

- **Input** : finite set of **m data points**
  $\big((a_{11}, \ldots, a_{1n}), b_1\big), \ldots, \big((a_{m1}, \ldots, a_{mn}), b_m\big) \in (\mathbb{R}^n \times \mathbb{R})^m$

Multi-Linear Regression

- **Input** : finite set of **m data points** $\big((a_{11}, \ldots, a_{1n}), b_1\big), \ldots, \big((a_{m1}, \ldots, a_{mn}), b_m\big) \in (\mathbb{R}^n \times \mathbb{R})^m$
- Vector $x \in \mathbb{R}^n$ (one component per input dimension); thus, we have a vector x with components x_j, $j = 1, \ldots, n$ in the least squares minimization problem

Multi-Linear Regression

- **Input** : finite set of **m data points** $\big((a_{11}, \ldots, a_{1n}), b_1\big), \ldots, \big((a_{m1}, \ldots, a_{mn}), b_m\big) \in (\mathbb{R}^n \times \mathbb{R})^m$
- Vector $x \in \mathbb{R}^n$ (one component per input dimension); thus, we have a vector x with components x_j, $j = 1, \ldots, n$ in the least squares minimization problem
- Consequently, the distance d_i for each data point $i = 1, \ldots, m$ writes

$$d_i = |\big(\sum_{j=1}^{n} a_{ij} x_j\big) - b_i|, \; i = 1, \ldots, m \tag{5}$$

Thus, the least squares minimization problem consists of solving

$$\underset{x \in \mathbb{R}^n}{\operatorname{argmin}} \sum_{i=1}^{m} d_i^2 = \sum_{i=1}^{m} \Big(\big(\sum_{j=1}^{n} a_{ij} x_j\big) - b_i\Big)^2 \tag{6}$$

which corresponds to the indexed form of (1)

Multi-Linear Regression

- **Input** : finite set of **m data points** $\big((a_{11}, \ldots, a_{1n}), b_1\big), \ldots, \big((a_{m1}, \ldots, a_{mn}), b_m\big) \in (\mathbb{R}^n \times \mathbb{R})^m$
- Vector $x \in \mathbb{R}^n$ (one component per input dimension); thus, we have a vector x with components x_j, $j = 1, \ldots, n$ in the least squares minimization problem
- Consequently, the distance d_i for each data point $i = 1, \ldots, m$ writes

$$d_i = |\big(\sum_{j=1}^{n} a_{ij} x_j\big) - b_i|, \; i = 1, \ldots, m \tag{5}$$

Thus, the least squares minimization problem consists of solving

$$\underset{x \in \mathbb{R}^n}{\operatorname{argmin}} \sum_{i=1}^{m} d_i^2 = \sum_{i=1}^{m} \Big(\big(\sum_{j=1}^{n} a_{ij} x_j\big) - b_i\Big)^2 \tag{6}$$

which corresponds to the indexed form of (1)

- Conversely, (1) $\equiv$ matrix form of (6), where matrix A is the **design matrix**

Regression Analysis (1)

A data set includes

i) vector $\mathbf{y}$ comprising m observed values $y_1, \ldots, y_m$ of the dependent variable y called the regressand or response variable, i.e., $\mathbf{y} = (y_1, \ldots, y_m)^\top$

Regression Analysis (1)

A data set includes

i) vector $\mathbf{y}$ comprising m observed values $y_1, \ldots, y_m$ of the dependent variable y called the regressand or response variable, i.e., $\mathbf{y} = (y_1, \ldots, y_m)^\top$

ii) model/design matrix $\mathbf{X} = (\mathbf{x}_1 \mid \ldots \mid \mathbf{x}_n)$ where the n independent variables $\mathbf{x}_j = (x_{1j}, \ldots, x_{mj})^\top$, $j = 1, \ldots, n$ are called the regressors or explanatory variables (a.k.a., predictor variables or predictors, covariates)

Regression Analysis (1)

A data set includes

i) vector $\mathbf{y}$ comprising m observed values $y_1, \ldots, y_m$ of the dependent variable y called the regressand or response variable, i.e., $\mathbf{y} = (y_1, \ldots, y_m)^\top$

ii) model/design matrix $\mathbf{X} = (\mathbf{x}_1 \mid \ldots \mid \mathbf{x}_n)$ where the n independent variables $\mathbf{x}_j = (x_{1j}, \ldots, x_{mj})^\top$, $j = 1, \ldots, n$ are called the regressors or explanatory variables (a.k.a., predictor variables or predictors, covariates)

iii) usually a constant included as one of the regressors, $x_{0j} = 1$ for $j = 1, \ldots, n$, corresponding to the regression parameter β_0 called the intercept

Regression Analysis (1)

A data set includes

i) vector $\mathbf{y}$ comprising m observed values $y_1, \ldots, y_m$ of the dependent variable y called the regressand or response variable, i.e., $\mathbf{y} = (y_1, \ldots, y_m)^\top$

ii) model/design matrix $\mathbf{X} = (\mathbf{x}_1 \mid \ldots \mid \mathbf{x}_n)$ where the n independent variables $\mathbf{x}_j = (x_{1j}, \ldots, x_{mj})^\top$, $j = 1, \ldots, n$ are called the regressors or explanatory variables (a.k.a., predictor variables or predictors, covariates)

iii) usually a constant included as one of the regressors, $x_{0j} = 1$ for $j = 1, \ldots, n$, corresponding to the regression parameter β_0 called the intercept

Linear regression model : characterizes linear relationship between the regressand (dependent variable) and regressors (independent variables)

$$\mathbf{y} = \beta_0 + \mathbf{x}_1\beta_1 + \cdots + \mathbf{x}_n\beta_n$$

Matrix notation: let $\boldsymbol{\beta}$ denote $(n+1)$-dimensional vector of regression parameters $(\beta_0, \beta_1, \ldots, \beta_n)$

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta}$$

Regression Analysis(2)

Matrix formulation of **linear regression model**

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta}, \tag{7}$$

where

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}, \qquad \boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{bmatrix}$$
$$\mathbf{X} = (\mathbf{x}_0 \,|\, \mathbf{x}_1 \,|\, \ldots \,|\, \mathbf{x}_n) = \begin{bmatrix} 1 & x_{11} & x_{12} & \ldots & x_{1n} \\ 1 & x_{21} & x_{22} & \ldots & x_{2n} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{m1} & x_{m2} & \ldots & x_{mn} \end{bmatrix} \tag{8}$$

Regression Analysis (2)

- Fitting a linear model to a given data set requires estimating the vector $\boldsymbol{\beta} \in \mathbb{R}^{n+1}$ of regression parameters such that (**residual**) **error** is minimized

$$\mathbf{y} - \mathbf{X}\boldsymbol{\beta}$$

Regression Analysis (2)

- Fitting a linear model to a given data set requires estimating the vector $\boldsymbol{\beta} \in \mathbb{R}^{n+1}$ of regression parameters such that (**residual**) **error** is minimized

$$\mathbf{y} - \mathbf{X}\boldsymbol{\beta}$$

- For this purpose, minimize the sum of squared errors

Regression Analysis (2)

- Fitting a linear model to a given data set requires estimating the vector $\boldsymbol{\beta} \in \mathbb{R}^{n+1}$ of regression parameters such that (**residual**) **error** is minimized

$$\mathbf{y} - \mathbf{X}\boldsymbol{\beta}$$

- For this purpose, minimize the sum of squared errors (convex function)

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 \tag{9}$$

More precisely, the estimator $\hat{\boldsymbol{\beta}}$ of the vector $\boldsymbol{\beta}$ of regression parameters is the minimizer to the optimization problem

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 \tag{10}$$

To find a minimizer, solve the so-called normal equation (why ?)

$$\mathbf{X}^\top (y - \mathbf{X}\hat{\boldsymbol{\beta}}) = 0$$

Regression quiz

Assume m data points $\{(x_i, y_i)\}_{i=1}^{m}$, $(x_i, y_i) \in \mathbb{R} \times \mathbb{R}$, $\forall i$.
Which of the following regression models are linear/non-linear ?

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 \tag{11}$$

$$y = \beta_0 + \beta_1^2 x \tag{12}$$

$$y = \beta_0 + \exp(\beta_1 x) \tag{13}$$

$$y = \beta_0 + \beta_1 \ln(x) \tag{14}$$

$$y = \beta_0 + \sin(\beta_1 x) + \cos(\beta_2 x) \tag{15}$$

$$y = \beta_0 + \beta_1 \sin(x) + \beta_2 \cos(x) \tag{16}$$

Regression Analysis (3)

- A regression model remains linear as long as it is linear in its regression parameter vector $\boldsymbol{\beta}$

  ◦) **Linearity means** : the response variable is a linear combination of the regression parameters and the input variables (regressors), and only imposes linearity restriction on the regression parameters $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_n)$

Regression Analysis (3)

- A regression model remains linear as long as it is linear in its regression parameter vector $\boldsymbol{\beta}$

  ◦) **Linearity means** : the response variable is a linear combination of the regression parameters and the input variables (regressors), and only imposes linearity restriction on the regression parameters $\boldsymbol{\beta} = (\beta_0, \beta_1, \ldots, \beta_n)$

- **Example**: assume m data points $\{(y_i, t_i)\}_{i=1}^m$. The relationship between dependent variable y and independent variable t modeled as

$$y_i = \beta_1 t_i + \beta_2 t_i^2, \; i = 1, \ldots, m$$

  ◦) Linear regression can be used to estimate the regression parameters β_1 and β_2 from observed data : this model is non-linear in the variable t but linear in the parameters β_1 and β_2

  ◦) Taking as regressors $\mathbf{x} = (x_{i1}, x_{i2}) = (t_i, t_i^2)$, the regression model takes on standard linear form

$$y_i = \beta_1 x_{i1} + \beta_2 x_{i2}$$

INFO-F524: Continuous Optimization
– Section 2 : Part 1 –

Bernard Fortz, Dimitri Papadimitriou

Bureau: **2N3 206**

Université Libre de Bruxelles

dimitrios.papadimitriou@ulb.be

Annee Académique: 2024-2025

Line search method

Line search method: given search direction (step vector) $d \in \mathbb{R}^n$, procedure to find step size $t^\star \in \mathbb{R}$ so that

$$x_{\textbf{new}} = x_{\textbf{current}} + t^\star d$$

approximately optimizes f along the line $\{x + t\,d \mid \in \mathbb{R}\}$.

Line search method

Line search method: given search direction (step vector) $d \in \mathbb{R}^n$, procedure to find step size $t^\star \in \mathbb{R}$ so that

$$x_{\textbf{new}} = x_{\textbf{current}} + t^\star d$$

approximately optimizes f along the line $\{x + t\,d \mid \in \mathbb{R}\}$.

Curry step size : if $t^\star$ is the global solution to the problem

$$\min_{t \in \mathbb{R}} \varphi(t) = f(x_{\textbf{current}} + t\,d)$$

Line search method

Line search method: given search direction (step vector) $d \in \mathbb{R}^n$, procedure to find step size $t^\star \in \mathbb{R}$ so that

$$x_{\textbf{new}} = x_{\textbf{current}} + t^\star d$$

approximately optimizes f along the line $\{x + t\,d \mid \in \mathbb{R}\}$.

Curry step size : if $t^\star$ is the global solution to the problem

$$\min_{t \in \mathbb{R}} \varphi(t) = f(x_{\textbf{current}} + t\,d)$$

Iterative method: compute

$$x_{k+1} = x_k + t_k d_k, \;\; t_k > 0$$

such that

$$f(x_{k+1}) = f(x_k + t_k d_k) < f(x_k)$$

along descent direction $d_k \rightarrow$ Compute exact minimum in each iteration !

Line search method: tradeoff

1) Tradeoff when computing step length t_k at each iteration k between
   - ◦) substantial reduction of f
   - ◦) computational cost for finding t_k

Line search method: tradeoff

1) Tradeoff when computing step length t_k at each iteration k between
   - ◦) substantial reduction of f
   - ◦) computational cost for finding t_k
2) Tradeoff in step size value t_k itself
   - ◦) if t_k is too large, then the iteration will overshoot
   - ◦) if t_k is too small, then $(x_k)_{k\in\mathbb{N}}$ converges slowly

Line search method: tradeoff

1) Tradeoff when computing step length t_k at each iteration k between
   - ◦) substantial reduction of f
   - ◦) computational cost for finding t_k
2) Tradeoff in step size value t_k itself
   - ◦) if t_k is too large, then the iteration will overshoot
   - ◦) if t_k is too small, then $(x_k)_{k\in\mathbb{N}}$ converges slowly

$\Rightarrow$ **Inexact line search** to find t_k that satisfies $f(x_k + t_k d_k) < f(x_k)$ at minimal computational cost

Line search method: tradeoff

- Convergence to $f^\star$ not guaranteed by only imposing decrease of f

  ... even when objective function is convex

Line search method: tradeoff

- Convergence to $f^\star$ not guaranteed by only imposing decrease of f

  ... even when objective function is convex

- Insufficient reduction (at each k) of $f \;\Rightarrow\; (x_k)_{k\in\mathbb{N}} \not\to x^\star$ (minimizer of f)

Line search method: tradeoff

- Convergence to $f^\star$ not guaranteed by only imposing decrease of f

  ... even when objective function is convex

- Insufficient reduction (at each k) of $f \;\Rightarrow\; (x_k)_{k\in\mathbb{N}} \not\to x^\star$ (minimizer of f)

- Example: $f(x_k) = a/k$.

Line search method: sufficient decrease (SD) condition

Armijo condition

Choose the step size $t^* = t_k$ such that

$$\begin{aligned} f(x_k + t_k d_k) &\leq f(x_k) + c\, t_k Df(x_k; d_k) \\ \Leftrightarrow f(x_k + t_k d_k) &\leq f(x_k) + c\, t_k \nabla f(x_k)^\top d_k \end{aligned} \tag{1}$$

where

- coefficient $c \in]0, 1[$
- directional derivative $Df(x_k; d_k) = \nabla f(x_k)^\top d_k$

Line search method: sufficient decrease (SD) condition

Armijo condition

Choose the step size $t^* = t_k$ such that

$$\begin{aligned} f(x_k + t_k d_k) &\leq f(x_k) + c\, t_k Df(x_k; d_k) \\ \Leftrightarrow f(x_k + t_k d_k) &\leq f(x_k) + c\, t_k \nabla f(x_k)^\top d_k \end{aligned} \tag{1}$$

where

- coefficient $c \in]0, 1[$
- directional derivative $Df(x_k; d_k) = \nabla f(x_k)^\top d_k$

Interpretation

- Guarantees sufficient decrease of **objective function** $f(x_k + t^* d_k) < f(x_k)$

Line search method: sufficient decrease (SD) condition

Armijo condition

Choose the step size $t^* = t_k$ such that

$$\begin{aligned} f(x_k + t_k d_k) &\leq f(x_k) + c\, t_k Df(x_k; d_k) \\ \Leftrightarrow f(x_k + t_k d_k) &\leq f(x_k) + c\, t_k \nabla f(x_k)^\top d_k \end{aligned} \tag{1}$$

where

- coefficient $c \in]0, 1[$
- directional derivative $Df(x_k; d_k) = \nabla f(x_k)^\top d_k$

Interpretation

- Guarantees sufficient decrease of **objective function** $f(x_k + t^* d_k) < f(x_k)$
- Reduction of f value : **at least** fixed fraction c of reduction obtained by first-order Taylor approximation of f at x_k

Line search method: sufficient decrease (SD) condition

Armijo condition

Choose the step size $t^* = t_k$ such that

$$\begin{aligned} f(x_k + t_k d_k) &\leq f(x_k) + c\, t_k Df(x_k; d_k) \\ \Leftrightarrow f(x_k + t_k d_k) &\leq f(x_k) + c\, t_k \nabla f(x_k)^\top d_k \end{aligned} \tag{1}$$

where

- coefficient $c \in]0, 1[$
- directional derivative $Df(x_k; d_k) = \nabla f(x_k)^\top d_k$

Interpretation

- Guarantees sufficient decrease of **objective function** $f(x_k + t^* d_k) < f(x_k)$
- Reduction of f value : **at least** fixed fraction c of reduction obtained by first-order Taylor approximation of f at x_k
- Does not guarantee that x_k converges in reasonable number of iterations (condition always satisfied when step length remains small enough $\rightarrow$ thus, prevents step length from being too large)

Line search method: sufficient decrease (SD) condition

Armijo condition

Choose the step size $t^* = t_k$ such that

$$\begin{aligned} f(x_k + t_k d_k) &\leq f(x_k) + c\, t_k Df(x_k; d_k) \\ \Leftrightarrow f(x_k + t_k d_k) &\leq f(x_k) + c\, t_k \nabla f(x_k)^\top d_k \end{aligned} \tag{2}$$

where

- coefficient $c \in]0, 1[$
- directional derivative $Df(x_k; d_k) = \nabla f(x_k)^\top d_k$

Line search method: sufficient decrease (SD) condition

Armijo condition

Choose the step size $t^* = t_k$ such that

$$\begin{aligned} f(x_k + t_k d_k) &\leq f(x_k) + c\, t_k Df(x_k; d_k) \\ \Leftrightarrow f(x_k + t_k d_k) &\leq f(x_k) + c\, t_k \nabla f(x_k)^\top d_k \end{aligned} \tag{2}$$

where

- coefficient $c \in]0, 1[$
- directional derivative $Df(x_k; d_k) = \nabla f(x_k)^\top d_k$

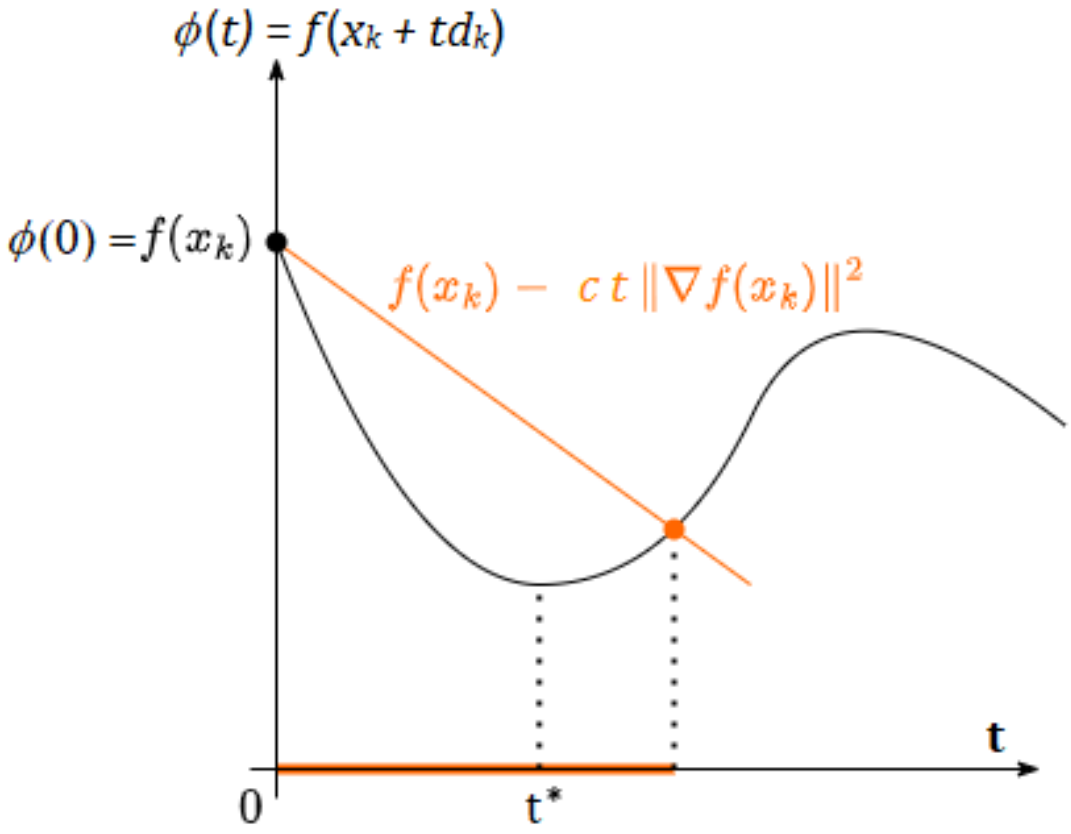
Examples:

- in gradient descent, $d_k = -\nabla f(x_k)$
- in normalized gradient descent, $d_k = -\nabla f(x_k) / \|\nabla f(x_k)\|$
- in Newton method, $d_k = -\nabla^2 f(x_k)^{-1} \nabla f(x_k)$

Line search method: sufficient decrease (SD) condition

In gradient descent, $d_k = -\nabla f(x_k)$; the condition becomes

$$\begin{aligned} f(x_k + t_k d_k) &\leq f(x_k) + c\, t_k \nabla f(x_k)^\top d_k \\ \Leftrightarrow f(x_k + t_k d_k) &\leq f(x_k) - c\, t_k \left\|\nabla f(x_k)\right\|^2 \\ \Leftrightarrow f(x_{k+1}) &\leq f(x_k) - c\, t_k \left\|\nabla f(x_k)\right\|^2 \end{aligned} \tag{3}$$



Line search method: curvature condition

- Armijo/SD condition **alone** does not guarantee x_k converges in a reasonable number of iterations

$\Rightarrow$ Curvature condition (combined with the Armijo/SD condition) to keep t_k from being too short

Line search method: curvature condition

- Armijo/SD condition **alone** does not guarantee x_k converges in a reasonable number of iterations

$\Rightarrow$ Curvature condition (combined with the Armijo/SD condition) to keep t_k from being too short

Curvature condition

Let $\tilde{c} \in]c, 1[$

$$\tilde{c}\, \nabla f(x_k)^\top d_k \leq \nabla f(x_k + t_k d_k)^\top d_k$$

where, RHS: derivative of $\varphi(t)$ at t_k ($\varphi(t) = f(x + td)$)

Line search method: curvature condition

- Armijo/SD condition **alone** does not guarantee x_k converges in a reasonable number of iterations

$\Rightarrow$ Curvature condition (combined with the Armijo/SD condition) to keep t_k from being too short

Curvature condition

Let $\tilde{c} \in]c, 1[$

$$\tilde{c}\, \nabla f(x_k)^\top d_k \leq \nabla f(x_k + t_k d_k)^\top d_k$$

where, RHS: derivative of $\varphi(t)$ at t_k ($\varphi(t) = f(x + td)$)

- The RHS of this condition is the derivative $\varphi'(t)$ of $\varphi(t) = f(x + td)$ at t_k to ensure that t_k remains in neighborhood of stationary point of $\varphi(t)$
- **Intuitively**: this condition ensures a sufficient increase of the instantaneous rate of change of f (ensures the step length not too small)

Line search method: curvature condition

Curvature condition

Let $\tilde{c} \in]c, 1[$

$$\begin{aligned} \tilde{c}\, \nabla f(x_k)^\top d_k &\leq \nabla f(x_k + t_k d_k)^\top d_k \\ \Leftrightarrow \tilde{c}\, Df(x_k; d_k) &\leq Df(x_k + t_k d_k; d_k) \end{aligned} \tag{4}$$

where, RHS: derivative of $\varphi(t)$ at t_k $(\varphi(t) = f(x + td))$

Line search method: curvature condition

Curvature condition

Let $\tilde{c} \in]c, 1[$

$$\begin{aligned} \tilde{c}\, \nabla f(x_k)^\top d_k &\leq \nabla f(x_k + t_k d_k)^\top d_k \\ \Leftrightarrow \tilde{c}\, Df(x_k; d_k) &\leq Df(x_k + t_k d_k; d_k) \end{aligned} \tag{4}$$

where, RHS: derivative of $\varphi(t)$ at t_k ($\varphi(t) = f(x + td)$)

Interpretation: the slope of φ at t_k, i.e., $\varphi'(t_k)$, is greater than $\tilde{c}$ times its initial slope at $t = 0$, i.e., the derivative of $\varphi(t)$ at $t = 0$, $\varphi'(0) = \nabla f(x_k)^\top d_k$

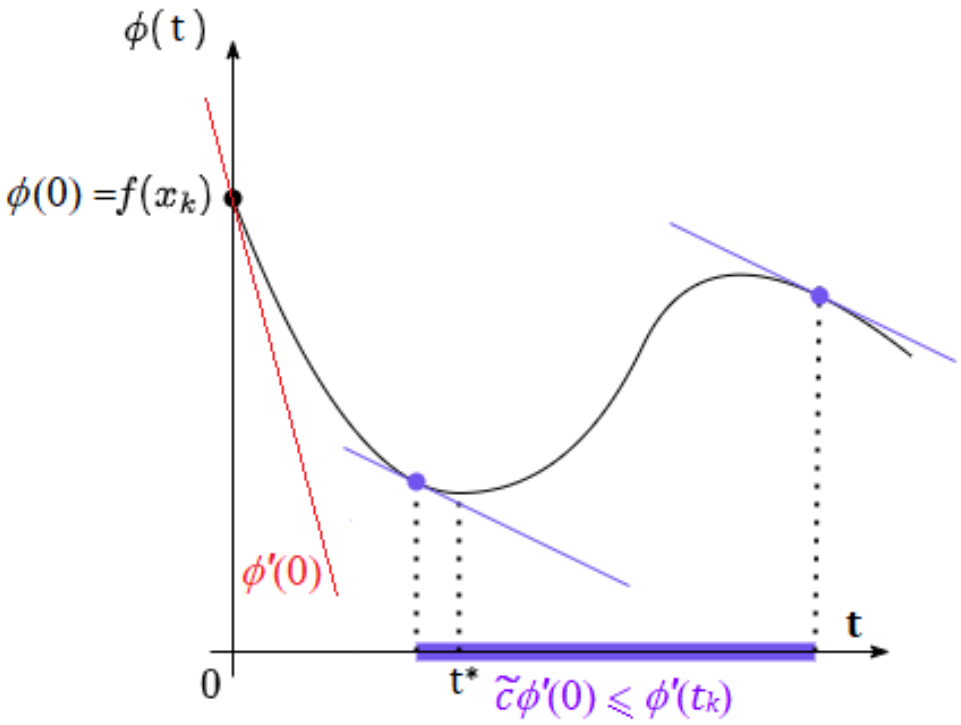
$$\tilde{c}\varphi'(0) \leq \varphi'(t_k) \tag{5}$$

- If $\varphi'(t_k)$ strongly negative: f can be significantly reduced (by moving further along chosen direction d_k)
- If $\varphi'(t_k)$ only slightly negative (or even positive): no more decrease in f in this direction

Line search method: curvature condition

In gradient descent, $d_k = -\nabla f(x_k)$; thus, assuming $\tilde{c} \in]c, 1[$

$$\begin{aligned} \tilde{c}\varphi'(0) &\leq \varphi'(t_k) \\ \tilde{c}\,\nabla f(x_k)^\top d_k &\leq \nabla f(x_k + t_k d_k)^\top d_k \\ \Leftrightarrow \tilde{c}\,\nabla f(x_k)^\top(-\nabla f(x_k)) &\leq \nabla f(x_k - t_k \nabla f(x_k))^\top(-\nabla f(x_k)) \\ \Leftrightarrow -\tilde{c}\,\|\nabla f(x_k)\|^2 &\leq \nabla f(x_k - t_k \nabla f(x_k))^\top(-\nabla f(x_k)) \end{aligned} \tag{6}$$



Wolfe conditions

Stepsize t_k is chosen to satisfy

$$f(x_k + t_k\, d_k) = \min_{t \in \mathbb{R}} f(x_k + t d_k)$$

First-order optimality conditions

$$\nabla f(x_k + t_k d_k)^\top d_k = 0$$

Wolfe conditions

Stepsize t_k is chosen to satisfy

$$f(x_k + t_k\, d_k) = \min_{t \in \mathbb{R}} f(x_k + t d_k)$$

First-order optimality conditions

$$\nabla f(x_k + t_k d_k)^\top d_k = 0$$

Wolfe conditions: combine SD condition (Armijo) with condition steering towards a new point such that

- **either** $\nabla f(x_k + t_k d_k)^\top d_k$ (in direction d_k) closer to zero (strong Wolfe)
- **or** search direction d_k less of descent direction (weak Wolfe)

Line search method: Wolfe conditions

Weak Wolfe conditions

Let c_1 and c_2 so that $0 < c_1 < c_2 < 1$.

$$\begin{cases} f(x_k + t_k d_k) & \leq & f(x_k) + c_1 t_k Df(x_k; d_k) \\ c_2 Df(x_k; d_k) & \leq & Df(x_k + t_k d_k; d_k) \end{cases} \tag{7}$$

Line search method: Wolfe conditions

Weak Wolfe conditions

Let c_1 and c_2 so that $0 < c_1 < c_2 < 1$.

$$\begin{cases} f(x_k + t_k d_k) & \leq & f(x_k) + c_1 t_k Df(x_k; d_k) \\ c_2 Df(x_k; d_k) & \leq & Df(x_k + t_k d_k; d_k) \end{cases} \tag{7}$$

Strong Wolfe conditions

Let c_1 and c_2 so that $0 < c_1 < c_2 < 1$.

$$\begin{cases} f(x_k + t_k d_k) & \leq & f(x_k) + c_1 t_k Df(x_k; d_k) \\ Df(x_k + t_k d_k; d_k) & \leq & c_2 \left| Df(x_k; d_k) \right| \end{cases} \tag{8}$$

Line search method: Goldstein conditions

Goldstein conditions

Let c_1 and c_2 so that $0 < c_1 \leq c_2 < 1$.

$$c_2\, t_k Df(x_k; d_k) \leq f(x_k + t_k d_k) - f(x_k) \leq c_1\, t_k Df(x_k; d_k) \tag{9}$$

Line search method: Goldstein conditions

Goldstein conditions

Let c_1 and c_2 so that $0 < c_1 \leq c_2 < 1$.

$$c_2\, t_k Df(x_k; d_k) \leq f(x_k + t_k d_k) - f(x_k) \leq c_1\, t_k Df(x_k; d_k) \tag{9}$$

Typical setting: $c_1 = c, c_2 = 1 - c$ with $c \in]0, 1/2]$

$$(1 - c)\, t_k Df(x_k; d_k) \leq f(x_k + t_k d_k) - f(x_k) \leq c\, t_k Df(x_k; d_k) \tag{10}$$

$$\Updownarrow$$

$$f(x_k) + (1 - c)\, t_k Df(x_k; d_k) \leq f(x_k + t_k d_k) \leq f(x_k) + c\, t_k Df(x_k; d_k) \tag{11}$$

Line search method: Goldstein conditions

Goldstein conditions

Let c_1 and c_2 so that $0 < c_1 \leq c_2 < 1$.

$$c_2\, t_k Df(x_k; d_k) \leq f(x_k + t_k d_k) - f(x_k) \leq c_1\, t_k Df(x_k; d_k) \tag{9}$$

Typical setting: $c_1 = c, c_2 = 1 - c$ with $c \in]0, 1/2]$

$$(1 - c)\, t_k Df(x_k; d_k) \leq f(x_k + t_k d_k) - f(x_k) \leq c\, t_k Df(x_k; d_k) \tag{10}$$

$$\Updownarrow$$

$$f(x_k) + (1 - c)\, t_k Df(x_k; d_k) \leq f(x_k + t_k d_k) \leq f(x_k) + c\, t_k Df(x_k; d_k) \tag{11}$$

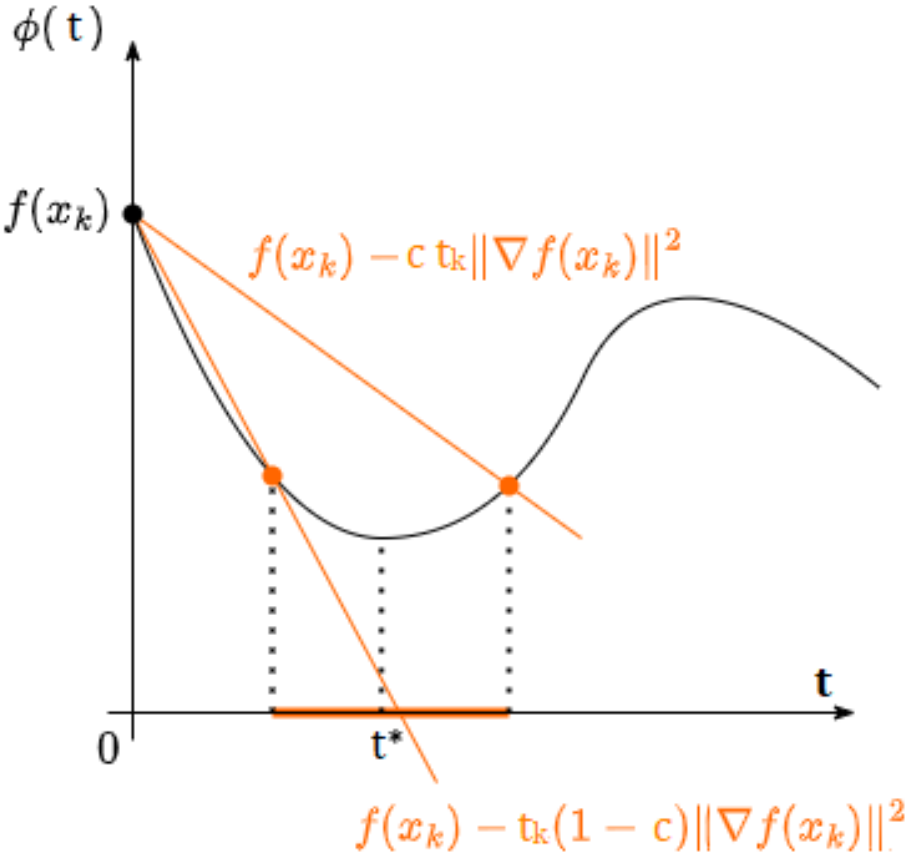
In gradient descent, $d_k = -\nabla f(x_k)$; thus, we have

$$f(x_k) - (1 - c)\, t_k \left\|\nabla f(x_k)\right\|^2 \leq f(x_k + t_k d_k) \leq f(x_k) - c\, t_k \left\|\nabla f(x_k)\right\|^2 \tag{12}$$

Line search method: Goldstein conditions

In gradient descent, $d_k = -\nabla f(x_k)$; thus, we have with $c \in]0, 1/2]$

$$f(x_k) - (1-c)\, t_k \left\|\nabla f(x_k)\right\|^2 \leq f(x_k + t_k d_k) \leq f(x_k) - c\, t_k \left\|\nabla f(x_k)\right\|^2 \qquad (12)$$



Backtracking Line search method

Backtracking method

- `Start`: relatively large initial step size (e.g., $t_{\text{init}} = 1$ for Newton method),
- `Until`: iterative decrease by contraction factor (α) that satisfies SD condition (Armijo)
- `End`: step size short enough to satisfy SD condition but large enough to allow sufficient decrease

Backtracking Line search method

Backtracking method

- `Start`: relatively large initial step size (e.g., $t_{\text{init}} = 1$ for Newton method),
- `Until`: iterative decrease by contraction factor (α) that satisfies SD condition (Armijo)
- `End`: step size short enough to satisfy SD condition but large enough to allow sufficient decrease

Backtracking line search algorithm:

```
input: c ∈ ]0,1[, contraction factor α ∈ ]0,1[ and t_init > 0
init: t ← t_init
repeat until: f(x_k + t d_k) ≤ f(x_k) + c t ∇f(x_k)^⊤ d_k
    t ← α t
end repeat
return t_k ← t
```

Backtracking Line search method

Backtracking method:

- Generic (desired output after **finite number of iterations**)
- Contraction factor $\alpha_k \in [\alpha_L, \alpha_U]$ with $0 < \alpha_L \leq \alpha_k \leq \alpha_U < 1$
- Tuning may depend on f, d_k computation algorithm, etc.

Backtracking Line search method

Backtracking method:

- Generic (desired output after **finite number of iterations**)
- Contraction factor $\alpha_k \in [\alpha_L, \alpha_U]$ with $0 < \alpha_L \leq \alpha_k \leq \alpha_U < 1$
- Tuning may depend on f, d_k computation algorithm, etc.

Example: for gradient descent ($d_k = -\nabla f(x_k)$)

- Condition:

$$\begin{aligned} f(x_k + td_k) &\leq f(x_k) + c\, t\, \nabla f(x_k)^\top d_k \\ &= f(x_k) - c\, t\, \|\nabla f(x_k)\|^2 \end{aligned}$$

Gradient (Descent) method

- Unconstrained optimization problem

$$\min_{x \in \mathbb{R}^n} f(x) \tag{13}$$

- **Assumptions**

  ◦) $f : \mathbb{R}^n \to \mathbb{R}$ is (continuously) differentiable, i.e., $(\forall x \in \mathbb{R}^n)\ \nabla f(x)$ exists

  ◦) optimal value $f^\star = \inf_x f(x)$ finite and attained at $x^\star$

Gradient (Descent) method

- Unconstrained optimization problem

$$\min_{x \in \mathbb{R}^n} f(x) \tag{13}$$

- **Assumptions**

  ◦) $f : \mathbb{R}^n \to \mathbb{R}$ is (continuously) differentiable, i.e., $(\forall x \in \mathbb{R}^n)\ \nabla f(x)$ exists

  ◦) optimal value $f^\star = \inf_x f(x)$ finite and attained at $x^\star$

- First-order approximation (Taylor theorem) of f at iterate $\overline{x}$

$$\begin{aligned} f(x) &\approx f(\overline{x}) + \nabla f(\overline{x})^T (x - \overline{x}) \\ &= (f(\overline{x}) - \nabla f(\overline{x})^T \overline{x}) + \nabla f(\overline{x})^T x \end{aligned}$$

- Vector d is **search direction** for f at $x = \overline{x}$ if

$$f(\overline{x} + td) < f(\overline{x}),\ \forall t > 0, t \in]0, \overline{t}[$$

.

Gradient (Descent) method

Def: Directional derivative

The directional derivative of $f : \mathbb{R}^n \to \mathbb{R}$ at x in the direction d is

$$Df(x; d) := \lim_{t \to 0} \frac{f(x + td) - f(x)}{t} \tag{14}$$

Gradient (Descent) method

Def: Directional derivative

The directional derivative of $f : \mathbb{R}^n \to \mathbb{R}$ at x in the direction d is

$$Df(x; d) := \lim_{t \to 0} \frac{f(x + td) - f(x)}{t} \tag{14}$$

Properties

- If f **convex** and finite near $x \in \mathit{int}\,\mathrm{dom}(f)$, then $Df(x; d)$ exists for all d
- If $Df(x; d) < 0$, then $\exists\, \overline{t}$ such that $\forall\, t \in]0, \overline{t}[, f(x + td) < f(x)$
- If f **differentiable** at x, then $Df(x; d)$ is a linear function of d

$$Df(x; d) = \nabla f(x)^T d \tag{15}$$

Gradient (Descent) method

Proof:

- Define $\varphi(\alpha) = f(x + \alpha d) = f(y(\alpha))$ with $y(\alpha) = x + \alpha d$
- Using this definition for φ, and following the def. of directional derivative (14)

$$Df(x; d) := \lim_{t \to 0} \frac{f(x + td) - f(x)}{t} = \lim_{t \to 0} \frac{\varphi(t) - \varphi(0)}{t} = \varphi'(0) \quad (1)$$

- At any α, since $y(\alpha) = x + \alpha d$

$$\begin{aligned} \varphi'(\alpha) &= \sum_{i=1}^{n} \frac{\partial f(y(\alpha))}{\partial y_i} \nabla y_i(\alpha) \\ &= \sum_{i=1}^{n} \frac{\partial f(y(\alpha))}{\partial y_i} d_i \\ &= \nabla f(y(\alpha))^\top d = \nabla f(x + \alpha d)^\top d \quad (2) \end{aligned}$$

  If $\alpha = 0$, then from (2) we obtain $\varphi'(0) = \nabla f(x)^\top d \quad (3)$
- Hence, by equating (1) and (3), we deduce the desired result

$$\varphi'(0) \overset{(1)}{=} \lim_{t \to 0} \frac{f(x + td) - f(x)}{t} = \nabla f(x)^\top d \overset{(3)}{=} \varphi'(0)$$

Gradient (Descent) method

Implications

- For f differentiable, d defines descent direction for f at x if

$$Df(x; d) = \nabla f(x)^T d < 0$$

$\Rightarrow d = -\nabla f(x)$ defines a **descent direction** for f (differentiable) at x except when $\nabla f(x) = 0$

Gradient (Descent) method

Implications

- For f differentiable, d defines descent direction for f at x if

$$Df(x;d) = \nabla f(x)^T d < 0$$

  $\Rightarrow d = -\nabla f(x)$ defines a **descent direction** for f (differentiable) at x except when $\nabla f(x) = 0$

- For f convex (not necessarilty differentiable), assume for every $x, d \in \mathbb{R}^n$, $Df(x;d)$ (14) always exists

  $\rightarrow$ Substituting $t = 1$ and $d = y - x$ into RHS of (14) where y is any other point in $\mathbb{R}^n$: **subdifferential inequality**

$$f(y) \geq f(x) + Df(x; y - x) \; \forall y \in \mathbb{R}^n, x \in \mathrm{dom}(f)$$

Gradient (Descent) method

Implications

- For f differentiable, d defines descent direction for f at x if

$$Df(x;d) = \nabla f(x)^T d < 0$$

$\Rightarrow d = -\nabla f(x)$ defines a **descent direction** for f (differentiable) at x except when $\nabla f(x) = 0$

- For f convex (not necessarilty differentiable), assume for every $x, d \in \mathbb{R}^n$, $Df(x;d)$ (14) always exists

$\rightarrow$ Substituting $t = 1$ and $d = y - x$ into RHS of (14) where y is any other point in $\mathbb{R}^n$: **subdifferential inequality**

$$f(y) \geq f(x) + Df(x; y - x) \; \forall y \in \mathbb{R}^n, x \in \mathrm{dom}(f)$$

$\Rightarrow$ For f convex **and** f differentiable, the subdifferential inequality becomes

$$f(y) \geq f(x) + Df(x; y - x) = f(x) + \nabla f(x)^T (y - x), \; \forall y \in \mathbb{R}^n, x \in \mathrm{dom}(f)$$

Gradient (Descent): algorithm

In order to have a **descent method**, the **search** direction (d) must satisfy -**at each iteration** k- the properties of a **descent** direction:

$$Df(x_k; d_k) = \nabla f(x_k)^\top d_k < 0 \tag{16}$$

Gradient (Descent): algorithm

In order to have a **descent method**, the **search** direction (d) must satisfy -**at each iteration** k- the properties of a **descent** direction:

$$Df(x_k; d_k) = \nabla f(x_k)^\top d_k < 0 \tag{16}$$

Examples:

- in **gradient descent**:

$$d_k = -\nabla f(x_k) \Rightarrow Df(x_k; d_k) = \nabla f(x_k)^\top d_k = -\left\|\nabla f(x_k)\right\|^2 < 0$$

- in **normalized gradient descent**:

$$d_k = -\nabla f(x_k)/\left\|\nabla f(x_k)\right\| \Rightarrow Df(x_k; d_k) = \nabla f(x_k)^\top d_k = -\left\|\nabla f(x_k)\right\| < 0$$

- in Newton method:

$$d_k = -\nabla^2 f(x_k)^{-1} \nabla f(x_k)$$

  Condition (16) satisfied if $\nabla f(x_k)^\top B \nabla f(x_k) > 0$, where matrix B approximates $-\nabla^2 f(x_k)^{-1}$

Gradient (Descent): algorithm

Gradient (descent) algorithm:

`input`: initial point $x_0 \in \mathrm{dom}(f)$
`repeat`:

1. Determine a **step**/search direction d_k (vector starting at x_k) $:= -\nabla f(x_k)$
2. Compute **step size** $t_k \in \mathbb{R}^+$: fixed $t_k = t$ or via inexact line search (e.g. backtracking)
3. Update: $x_{k+1} = x_k + t_k d_k = x_k - t_k \nabla f(x_k)$

`until`: stopping criterion is satisfied

Gradient (Descent): algorithm

Gradient (descent) algorithm:

`input`: initial point $x_0 \in \mathrm{dom}(f)$
`repeat`:

1. Determine a **step**/search direction d_k (vector starting at x_k) $:= -\nabla f(x_k)$
2. Compute **step size** $t_k \in \mathbb{R}^+$: fixed $t_k = t$ or via inexact line search (e.g. backtracking)
3. Update: $x_{k+1} = x_k + t_k d_k = x_k - t_k \nabla f(x_k)$

`until`: stopping criterion is satisfied

Stopping criterion

- Most common: $\|\nabla f(x)\|_2 \leq \varepsilon$
- Saturated improvements in (scaled) function value: $\dfrac{|f(x_{k+1}) - f(x_k)|}{\max\{1, |f(x_k)|\}}$
- etc.

Gradient (Descent): descent lemma

Descent Lemma

Assume

- $f : \mathbb{R}^n \to \mathbb{R}$ **convex** and continuously **differentiable** function
- $\nabla f(x)$ Lipschitz continuous with constant $L > 0$,

$$\|\nabla f(x) - \nabla f(y)\| \leq L \|x - y\| \ \ \forall x, y \tag{17}$$

Starting from $x_0 \in \mathrm{dom}(f)$, $x_{k+1} = x_k - t_k \nabla f(x_k)$ with **fixed step size** $t_k = \dfrac{1}{L}$, the following inequality is verified $\forall k$

$$f(x_{k+1}) \leq f(x_k) - \frac{1}{2L} \|\nabla f(x_k)\|_2^2 \tag{18}$$

Gradient (Descent): descent lemma

Descent Lemma

Assume

- $f : \mathbb{R}^n \to \mathbb{R}$ **convex** and continuously **differentiable** function
- $\nabla f(x)$ Lipschitz continuous with constant $L > 0$,

$$\|\nabla f(x) - \nabla f(y)\| \leq L \|x - y\| \ \ \forall x, y \tag{17}$$

Starting from $x_0 \in \mathrm{dom}(f)$, $x_{k+1} = x_k - t_k \nabla f(x_k)$ with **fixed step size** $t_k = \dfrac{1}{L}$, the following inequality is verified $\forall k$

$$f(x_{k+1}) \leq f(x_k) - \frac{1}{2L} \|\nabla f(x_k)\|_2^2 \tag{18}$$

Interpretation:

- At each k, f decreases by at least $\dfrac{1}{2L} \|\nabla f(x_k)\|_2^2$
- At $x_k = x^\star$, $\nabla f(x^\star) = 0$ ($x^\star$ stationary point of f); from (18) $f(x_{k+1}) = f(x_k)$

Gradient (Descent): descent lemma

- Since f is continuously differentiable, by the Taylor theorem (first-order expansion), we have for the vector $a_t = x + t(y - x)$ with $t \in [0, 1]$

$$f(y) = f(x) + \langle \nabla f(a_t), (y - x) \rangle$$

Gradient (Descent): descent lemma

- Since f is continuously differentiable, by the Taylor theorem (first-order expansion), we have for the vector $a_t = x + t(y - x)$ with $t \in [0, 1]$

$$f(y) = f(x) + \langle \nabla f(a_t), (y - x) \rangle$$

- In the RHS, add and subtract $\langle \nabla f(x), y - x \rangle$ to obtain

$$f(y) - f(x) - \langle \nabla f(x), y - x \rangle = \langle \nabla f(a_t) - \nabla f(x), y - x \rangle$$

Gradient (Descent): descent lemma

- Since f is continuously differentiable, by the Taylor theorem (first-order expansion), we have for the vector $a_t = x + t(y - x)$ with $t \in [0, 1]$

$$f(y) = f(x) + \langle \nabla f(a_t), (y - x) \rangle$$

- In the RHS, add and subtract $\langle \nabla f(x), y - x \rangle$ to obtain

$$f(y) - f(x) - \langle \nabla f(x), y - x \rangle = \langle \nabla f(a_t) - \nabla f(x), y - x \rangle$$

- Since $|f(y) - f(x) - \langle \nabla f(x), y - x \rangle| = |\langle \nabla f(a_t) - \nabla f(x), y - x \rangle|$, we deduce from the Cauchy-Schwarz inequality ($|\langle \mathbf{u}, \mathbf{v} \rangle| \leq \|\mathbf{u}\| \|\mathbf{v}\|$)

$$|\langle \nabla f(a_t) - \nabla f(x), y - x \rangle| \leq \|\nabla f(a_t) - \nabla f(x)\| \, \|y - x\|$$

Gradient (Descent): descent lemma

- Since f is continuously differentiable, by the Taylor theorem (first-order expansion), we have for the vector $a_t = x + t(y - x)$ with $t \in [0, 1]$

$$f(y) = f(x) + \langle \nabla f(a_t), (y - x) \rangle$$

- In the RHS, add and subtract $\langle \nabla f(x), y - x \rangle$ to obtain

$$f(y) - f(x) - \langle \nabla f(x), y - x \rangle = \langle \nabla f(a_t) - \nabla f(x), y - x \rangle$$

- Since $|f(y) - f(x) - \langle \nabla f(x), y - x \rangle| = |\langle \nabla f(a_t) - \nabla f(x), y - x \rangle|$, we deduce from the Cauchy-Schwarz inequality ($|\langle \mathbf{u}, \mathbf{v} \rangle| \leq \|\mathbf{u}\| \|\mathbf{v}\|$)

$$|\langle \nabla f(a_t) - \nabla f(x), y - x \rangle| \leq \|\nabla f(a_t) - \nabla f(x)\| \, \|y - x\|$$

- The gradient ∇f being Lipschitz continuous with constant $L > 0$, the RHS verifies the inequality $\|\nabla f(a_t) - \nabla f(x)\| \, \|y - x\| \leq \frac{L}{2} \|y - x\|^2$. Hence,

$$|f(y) - f(x) - \langle \nabla f(x), y - x \rangle| \leq \frac{L}{2} \|y - x\|^2$$

Gradient (Descent): descent lemma

- Thus, we get an **quadratic upper bound** of f

$$f(y) \leq f(x) + \nabla f(x)^{\intercal}(y - x) + \frac{L}{2}\|y - x\|_2^2, \ \forall x, y \in \mathrm{dom}(f) \tag{19}$$

Gradient (Descent): descent lemma

- Thus, we get an **quadratic upper bound** of f

$$f(y) \leq f(x) + \nabla f(x)^\intercal (y - x) + \frac{L}{2} \|y - x\|_2^2, \ \forall x, y \in \mathrm{dom}(f) \tag{19}$$

- In (19), set $y = x_{k+1}$ and $x = x_k$

$$f(x_{k+1}) \leq f(x_k) + \nabla f(x_k)^\intercal (x_{k+1} - x_k) + \frac{L}{2} \|x_{k+1} - x_k\|_2^2 \tag{20}$$

Gradient (Descent): descent lemma

- Thus, we get an **quadratic upper bound** of f

$$f(y) \leq f(x) + \nabla f(x)^\intercal (y - x) + \frac{L}{2} \|y - x\|_2^2, \ \forall x, y \in \mathrm{dom}(f) \tag{19}$$

- In (19), set $y = x_{k+1}$ and $x = x_k$

$$f(x_{k+1}) \leq f(x_k) + \nabla f(x_k)^\intercal (x_{k+1} - x_k) + \frac{L}{2} \|x_{k+1} - x_k\|_2^2 \tag{20}$$

- In the RHS of (20), replace the expression $x_{k+1} - x_k$ by the update $x_{k+1} - x_k = -\frac{1}{L} \nabla f(x_k)$ to obtain

$$f(x_{k+1}) \leq f(x_k) + \nabla f(x_k)^\intercal (-\frac{1}{L} \nabla f(x_k)) + \frac{L}{2} \|\frac{1}{L} \nabla f(x_k)\|_2^2 \tag{21}$$

- Thus,

$$f(x_{k+1}) \leq f(x_k) - \frac{1}{2L} \|\nabla f(x_k)\|_2^2 \tag{22}$$

Gradient (Descent): fixed step size

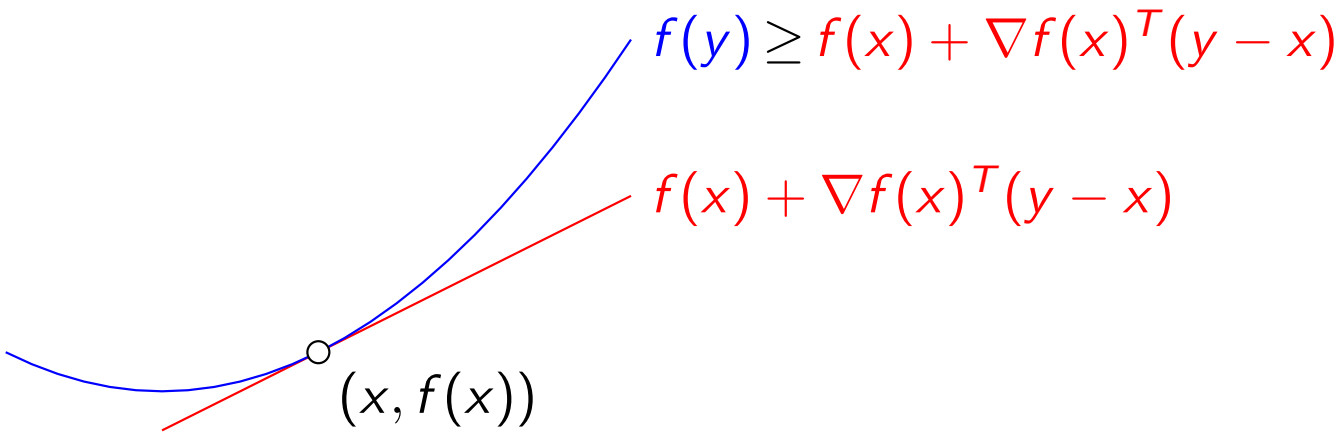
Step size t_k selection

1. **Fixed** step-size: $t_k = t, \ \forall k \geq 0$
   - ◦) Case 1: if t too large, then $(x_k)_{k \in \mathbb{N}}$ can diverge
   - ◦) Case 2: if t too small, then $(x_k)_{k \in \mathbb{N}}$ slow to converge
   - ◦) Case 3: if t "just right", i.e., when $t \in]0, 1/L]$

Gradient (Descent): lower bound

Recall: if function f **convex** and differentiable, then sub-differential inequality

$$f(y) \geq f(x) + \nabla f(x)^T (y - x), \ \forall y \in \mathrm{dom}(f) \tag{23}$$



Thus, the first-order approximation of f at x is a global underestimator, i.e., a global **lower bound**, of f at x

Gradient (Descent): quadratic upper bound

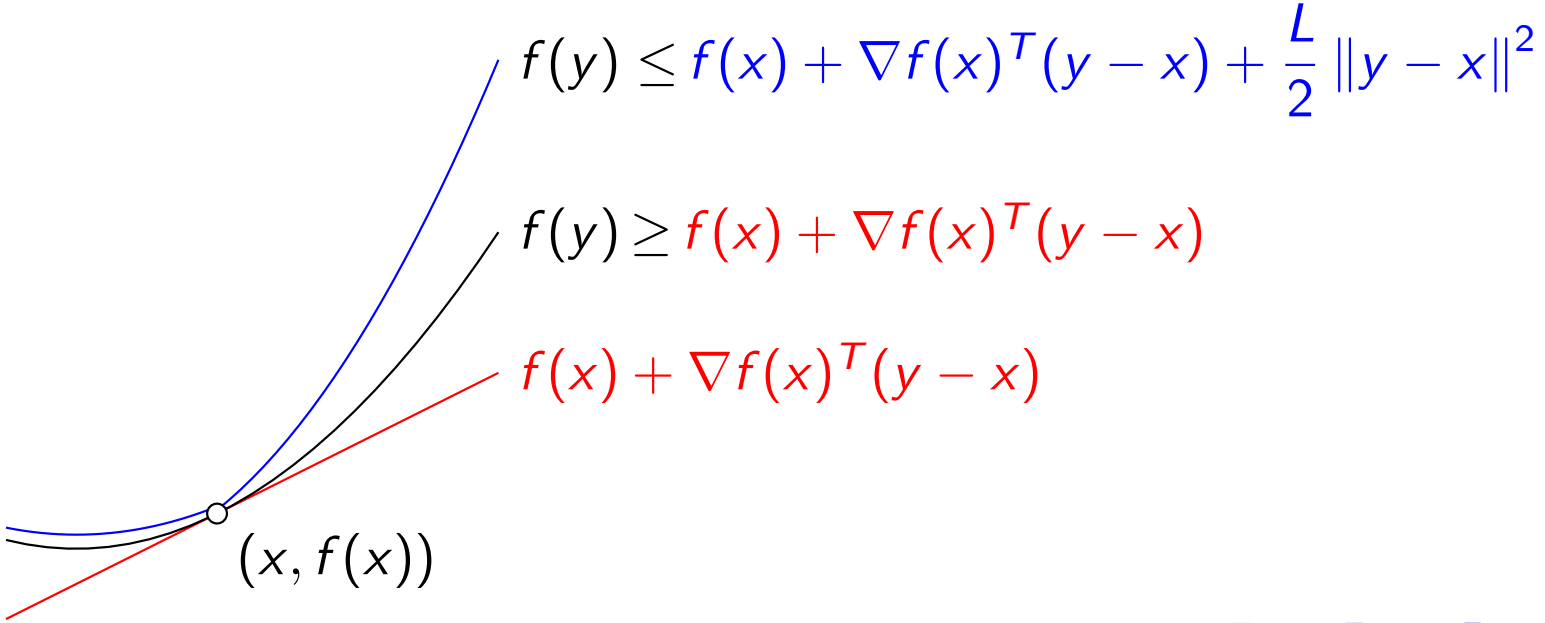
Suppose ∇f is Lipschitz continuous with constant $L > 0$ (see (17))

- From the Cauchy–Schwarz inequality, Lipschitz continuity of ∇f implies

$$(\nabla f(x) - \nabla f(y))^T (x - y) \leq L \|x - y\|^2 \ \ \forall x, y \in \mathrm{dom}(f) \tag{24}$$

- If $\mathrm{dom}(f)$ is convex, (24) is equivalent to **quadratic upper bound**

$$f(y) \ \leq \ f(x) + \nabla f(x)^\intercal (y - x) + \frac{L}{2} \|y - x\|^2 \ \ \forall x, y \in \mathrm{dom}(f) \tag{25}$$



Gradient (Descent): quadratic upper bound

Proof: equivalence (24) $\Leftrightarrow$ (25)

- Define $g(t) = f(x + t(y - x))$ for $x, y \in \mathrm{dom}(f)$; thus, it holds

$$g'(t) = \nabla f(x + t(y - x))^T (y - x)$$

- Since f is convex, $g(t)$ defined for $t \in [0, 1]$ and

$$\begin{cases} g(1) & = f(x + (y - x)) = f(y) \\ g(0) & = f(x) \end{cases} \text{ and } \begin{cases} g'(1) & = \nabla f(y)^T (y - x) \\ g'(0) & = \nabla f(x)^T (y - x) \end{cases}$$

- If (24) holds, then

$$g'(t) - g'(0) = (\nabla f(x + t(y - x)) - \nabla f(x))^T (y - x) \leq tL \|x - y\|^2$$

- Integrating from $t = 0$ to $t = 1$ gives (since $g'(t) \leq g'(0) + tL \|x - y\|^2$)

$$f(y) = g(1) = g(0) + \int_0^1 g'(t) dt \leq g(0) + g'(0) + \frac{L}{2} \|x - y\|^2$$
$$= f(x) + \nabla f(x)^T (y - x) + \frac{L}{2} \|y - x\|^2$$

Gradient (Descent): quadratic upper bound

Proof:

- If (25) holds, then since verified for all $x, y \in \mathrm{dom}(f)$, we can write

$$f(x) \leq f(y) + \nabla f(y)^{\intercal}(x-y) + \frac{L}{2}\|x-y\|^2 \ \ \forall x,\, y \in \mathrm{dom}(f) \tag{26}$$

- Its combination (addition) with

$$f(y) \leq f(x) + \nabla f(x)^{\intercal}(y-x) + \frac{L}{2}\|y-x\|^2 \ \ \forall x,\, y \in \mathrm{dom}(f) \tag{25}$$

gives

$$\left(\nabla f(x) - \nabla f(y)\right)^T (x-y) \leq L\|x-y\|^2 \ \ \forall x,\, y \in \mathrm{dom}(f) \tag{24}$$

Gradient (Descent): Consequence of quadratic UB

If $\mathrm{dom}(f) = \mathbb{R}$ and f has a minimizer at $x = x^\star$, $y = z$ then (25) becomes

$$f(z) \leq f(x) + \nabla f(x)^\intercal (z - x) + \frac{L}{2} \|z - x\|^2, \ \forall x, z \in \mathrm{dom}(f) \tag{27}$$

Then, consequence of quadratic upper bound (note: $\nabla f(x = x^\star) = 0$)

$$f(z) - f(x^\star) \leq \frac{L}{2} \|z - x^\star\|^2, \ \forall z \in \mathrm{dom}(f) \tag{28}$$

How to obtain a lower bound ? By minimizing quadratic upper bound for $x = z$. That is

$$\begin{aligned}
\inf_y f(y) &\leq \inf_y \left\{ f(z) + \nabla f(z)^\intercal (y - z) + \frac{L}{2} \|y - z\|^2 \right\} \\
&= \inf_{\|v\|_p = 1} \inf_t \left\{ f(z) + t \nabla f(z)^\intercal v + \frac{L t^2}{2} \right\} \\
&= \inf_{\|v\|_p = 1} \left\{ f(z) - \frac{1}{2L} (\nabla f(z)^\intercal v)^2 \right\} \\
&= f(z) - \frac{1}{2L} \|\nabla f(z)\|_*^2
\end{aligned} \tag{29}$$

Gradient (Descent): Consequence of quadratic UB

Hence, if $\mathrm{dom}(f) = \mathbb{R}^n$ and f has a minimizer $x^\star$, then for all z, it holds that

$$\frac{1}{2L} \|\nabla f(z)\|_*^2 \leq f(z) - f(x^\star) \leq \frac{L}{2} \|z - x^\star\|^2, \ \forall z \in \mathrm{dom}(f) \tag{30}$$

Gradient (Descent): step size 1/L

- **Subdifferential inequality** (lower bound): f convex

$$f(y) \geq f(x) + Df(x; y-x) = f(x) + \nabla f(x)^T (y-x), \ \forall y \in \mathbb{R}^n, x \in \mathrm{dom}(f)$$

- **Quadratic upper approximation of f (quadratic upper bound)**: f convex, differentiable and ∇f Lipschitz continuous with constant $L > 0$

$$f(y) \leq f(x) + \nabla f(x)^\intercal (y-x) + \frac{L}{2} \|y-x\|^2, \ \forall x, y \in \mathrm{dom}(f) \tag{25}$$

- **Gradient method iterate**:

$$x_{k+1} = x_k + t_k d_k = x_k - t_k \nabla f(x_k)$$

verifies **descent lemma** with **fixed step size** $t_k = 1/L$

- $\forall k$, f decreases by at least $\frac{1}{2L} \|\nabla f(x_k)\|_2^2$
- At stationary point $x_k = x^\star$, $\nabla f(x^\star) = 0$; thus, from (18) $f(x_{k+1}) = f(x_k)$

Gradient (Descent): fixed step size in $]0, 1/L]$

- In the quadratic upper bound (25)

$$f(y) \leq f(x) + \nabla f(x)^\intercal (y - x) + \frac{L}{2} \|y - x\|_2^2 \ \ \forall x,\, y \in \mathrm{dom}(f) \tag{31}$$

set $y = x - t\nabla f(x)$

$$f(x - t\nabla f(x)) \leq f(x) - t(1 - \frac{Lt}{2}) \|\nabla f(x)\|_2^2 \tag{32}$$

- Therefore, if the iterate $x^+ = x - t\nabla f(x)$ and $t \in]0, 1/L]$

$$f(x^+) \leq f(x) - \frac{t}{2} \|\nabla f(x)\|_2^2 \tag{33}$$

- Thus, in the gradient method, assuming $\nabla f(x) \neq 0$, the function value decreases

$$f(x^+) < f(x)$$

Gradient (Descent): fixed step size in $]0, 1/L]$

- Let $f^\star = \inf_x f(x)$ finite and attained at $x^\star$
- From (33) and convexity of f

$$\begin{aligned} f(x^+) - f^\star &\leq \nabla f(x)^T (x - x^\star) - \frac{t}{2} \|\nabla f(x)\|_2^2 \\ &= \frac{1}{2t} \left(\|x - x^\star\|_2^2 - \|(x - x^\star) - t\nabla f(x)\|_2^2 \right) \\ &= \frac{1}{2t} \left(\|x - x^\star\|_2^2 - \left\|x^+ - x^\star\right\|_2^2 \right) \end{aligned} \tag{34}$$

- Thus, in the gradient method, assuming $\nabla f(x) \neq 0$, the distance to the optimal set decreases

$$\left\|x^+ - x^\star\right\|_2^2 < \|x - x^\star\|_2^2 \tag{35}$$

Gradient (Descent): fixed step size in $]0, 1/L]$

- Let $f^\star = \inf_x f(x)$ finite and attained at $x^\star$
- From (33) and convexity of f

$$\begin{aligned} f(x^+) - f^\star &\leq \nabla f(x)^T (x - x^\star) - \frac{t}{2} \|\nabla f(x)\|_2^2 \\ &= \frac{1}{2t} \left(\|x - x^\star\|_2^2 - \|(x - x^\star) - t\nabla f(x)\|_2^2 \right) \\ &= \frac{1}{2t} \left(\|x - x^\star\|_2^2 - \left\|x^+ - x^\star\right\|_2^2 \right) \end{aligned} \tag{34}$$

- Thus, in the gradient method, assuming $\nabla f(x) \neq 0$, the distance to the optimal set decreases

$$\left\|x^+ - x^\star\right\|_2^2 < \|x - x^\star\|_2^2 \tag{35}$$

Descent properties

Consequently, in the gradient method, when f convex with L-Lipschitz continuous gradient, both function value and distance to the optimal set decrease

Gradient (Descent): fixed step size in $]0, 1/L]$

- Assume for all $k = 0, 1, \ldots,$ that the step size $t \in]0, 1/L]$ is constant and

$$x_{k+1} = x_k - t\nabla f(x_k) \tag{36}$$

- Set $x = x_{i-1}$ and $x^+ = x_i$ in (34) and assume $f^\star = \inf_x f(x)$ attained at $x^\star$

$$f(x_i) - f^\star \leq \frac{1}{2t}\left(\|x_{i-1} - x^\star\|_2^2 - \|x_i - x^\star\|_2^2\right) \tag{37}$$

- Sum over $i, i = 1, \ldots, k$

$$\begin{aligned}\sum_{i=1}^{k}(f(x_i) - f^\star) &\leq \frac{1}{2t}\sum_{i=1}^{k}\left(\|x_{i-1} - x^\star\|_2^2 - \|x_i - x^\star\|_2^2\right) \\ &= \frac{1}{2t}\left(\|x_0 - x^\star\|_2^2 - \|x_k - x^\star\|_2^2\right) \\ &\leq \frac{1}{2t}\|x_0 - x^\star\|_2^2\end{aligned} \tag{38}$$

- Since following (33) (Descent lemma), f_i is non-increasing

$$f(x_k) - f^\star \leq \frac{1}{k}\sum_{i=1}^{k}(f(x_i) - f^\star) \leq \frac{1}{2kt}\|x_0 - x^\star\|_2^2 \tag{39}$$

Gradient (Descent): step size

Step size t_k selection

1. **Fixed** step-size: $t_k = t, \ \forall k \geq 0$
   - ◦) Case 1: if t too large, then $(x_k)_{k\in\mathbb{N}}$ can diverge
   - ◦) Case 2: if t too small, then $(x_k)_{k\in\mathbb{N}}$ slow to converge
   - ◦) Case 3: if $t \in]0, 1/L]$ then $(x_k)_{k\in\mathbb{N}}$ converges at rate $O(1/k)$

$$f(x_k) - f^\star \leq \frac{1}{k}\sum_{i=1}^{k}\Big(f(x_i) - f^\star\Big) \leq \frac{1}{2kt}\,\|x_0 - x^\star\|_2^2$$

Gradient (Descent): step size

Step size t_k selection

1. **Fixed** step-size: $t_k = t, \ \forall k \geq 0$
   - ◦) Case 1: if t too large, then $(x_k)_{k\in\mathbb{N}}$ can diverge
   - ◦) Case 2: if t too small, then $(x_k)_{k\in\mathbb{N}}$ slow to converge
   - ◦) Case 3: if $t \in]0, 1/L]$ then $(x_k)_{k\in\mathbb{N}}$ converges at rate $O(1/k)$

$$f(x_k) - f^\star \leq \frac{1}{k}\sum_{i=1}^{k}\Big(f(x_i) - f^\star\Big) \leq \frac{1}{2kt}\,\|x_0 - x^\star\|_2^2$$

2. **Adaptive** step-size
   - ◦) Exact LS: rarely used
   - ◦) Inexact LS: backtracking LS

Gradient (Descent): adaptive step size

Gradient method with backtracking line search (LS)

- **Iterate**:

$$x_{k+1} = x_k + t_k d_k = x_k - t_k \nabla f(x_k)$$

- **Assumptions**:
  1. f convex and differentiable with $\mathrm{dom}(f) = \mathbb{R}^n$. Implication:
  $$f(y) \geq f(x) + \nabla f(x)^\top (y - x), \ \forall x, y \in \mathrm{dom}(f)$$
  2. ∇f L-Lipschitz continuous w.r.t. Euclidean norm, with $L > 0$
  Implication: (if $\mathrm{dom}(f)$ is convex)
  $$f(y) \leq f(x) + \nabla f(x)^T (y - x) + \frac{L}{2} \|y - x\|^2, \ \forall x, y \in \mathrm{dom}(f)$$
  3. Optimal value $f^\star = \inf_x f(x)$ is finite and attained at $x^\star$

Gradient (Descent): adaptive step size

Gradient method with backtracking line search (LS)

- **Iterate**:

$$x_{k+1} = x_k + t_k d_k = x_k - t_k \nabla f(x_k)$$

- **Assumptions**:
  1. f convex and differentiable with $\mathrm{dom}(f) = \mathbb{R}^n$. Implication:
  $$f(y) \geq f(x) + \nabla f(x)^\top (y - x), \ \forall x, y \in \mathrm{dom}(f)$$
  2. ∇f L-Lipschitz continuous w.r.t. Euclidean norm, with $L > 0$
  Implication: (if $\mathrm{dom}(f)$ is convex)
  $$f(y) \leq f(x) + \nabla f(x)^T (y - x) + \frac{L}{2} \|y - x\|^2, \ \forall x, y \in \mathrm{dom}(f)$$
  3. Optimal value $f^\star = \inf_x f(x)$ is finite and attained at $x^\star$

- Provided $t_k \geq t_{\min} = \min\{t_{\mathrm{init}}, \alpha/L\}$, **convergence rate**: $O(1/k)$

$$f(x_k) - f^\star \leq \frac{1}{k} \sum_{i=1}^{k} \left(f(x_i) - f^\star \right) \leq \frac{1}{2kt_{\min}} \|x_0 - x^\star\|_2^2$$

Gradient (Descent): step size

Step size t_k selection

1. **Fixed** step-size: $t_k = t, \ \forall k \geq 0$
   - ◦) Case 1: if t too large, then $(x_k)_{k\in\mathbb{N}}$ can diverge
   - ◦) Case 2: if t too small, then $(x_k)_{k\in\mathbb{N}}$ slow to converge
   - ◦) Case 3: if $t \in]0, 1/L]$ then $(x_k)_{k\in\mathbb{N}}$ converges at rate $O(1/k)$

$$f(x_k) - f^\star \leq \frac{1}{k}\sum_{i=1}^{k}\Big(f(x_i) - f^\star\Big) \leq \frac{1}{2kt}\,\|x_0 - x^\star\|_2^2$$

2. **Adaptive** step-size
   - ◦) Exact LS: rarely used
   - ◦) Inexact LS: backtracking LS ($t_k \geq t_{\min} = \min\{t_{\text{init}}, \alpha/L\}$)

$$f(x_k) - f^\star \leq \frac{1}{k}\sum_{i=1}^{k}\Big(f(x_i) - f^\star\Big) \leq \frac{1}{2kt_{\min}}\,\|x_0 - x^\star\|_2^2$$

Gradient (Descent): step size

Step size t_k selection

1. **Fixed** step-size: $t_k = t, \ \forall k \geq 0$
   - ◦) Case 1: if t too large, then $(x_k)_{k\in\mathbb{N}}$ can diverge
   - ◦) Case 2: if t too small, then $(x_k)_{k\in\mathbb{N}}$ slow to converge
   - ◦) Case 3: if $t \in]0, 1/L]$ then $(x_k)_{k\in\mathbb{N}}$ converges at rate $O(1/k)$

$$f(x_k) - f^\star \leq \frac{1}{k}\sum_{i=1}^{k}\Big(f(x_i) - f^\star\Big) \leq \frac{1}{2kt}\,\|x_0 - x^\star\|_2^2$$

2. **Adaptive** step-size
   - ◦) Exact LS: rarely used
   - ◦) Inexact LS: backtracking LS ($t_k \geq t_{\min} = \min\{t_{\text{init}}, \alpha/L\}$)

$$f(x_k) - f^\star \leq \frac{1}{k}\sum_{i=1}^{k}\Big(f(x_i) - f^\star\Big) \leq \frac{1}{2kt_{\min}}\,\|x_0 - x^\star\|_2^2$$

$\Rightarrow$ If α is not too small, close to fixed step size rate (α/L vs. $1/L$)

Gradient (descent) method: convergence rate

f is **convex**, differentiable and ∇f is Lipschitz continuous with $L > 0$

- GD with fixed step size $t \leq 1/L$: convergence rate $O(1/k)$

  $\Rightarrow$ $(f(x_k) - f^\star)$ decreases at least as fast as $1/k$
- GD with backtracking LS to adaptively select step size $t_k \geq t_{\min} = \min\{t_{\text{init}}, \alpha/L\}$: convergence rate $O(1/k)$
- To reach (precision) bound of $f(x_k) - f^\star \leq \epsilon$: run $O(1/\epsilon)$ iterations of GD

  $\Rightarrow$ GD algorithm finds $\epsilon-$suboptimal point in $O(1/\epsilon)$ iterations

f is **nonconvex**, differentiable and ∇f is Lipschitz continuous with $L > 0$

- GD with fixed step size $t \leq 1/L$: convergence rate $O(1/\sqrt{k})$ in the finding of $\varepsilon-$stationary point x, i.e., $\|\nabla f(x)\|_2 \leq \varepsilon$

  $\Rightarrow$ GD finds $\varepsilon-$stationary point x, i.e., $\|\nabla f(x)\|_2 \leq \varepsilon$, in $O(1/\varepsilon^2)$ iterations

Projected Gradient

Projected gradient method: solve (constrained) optimization problem

$$\min_x f(x) \quad \text{subject to } x \in \mathcal{C}$$

- $f : \mathbb{R}^n \to \mathbb{R}$ differentiable
- Set $\mathcal{C} \subseteq \mathbb{R}^n$ is non-empty, **convex** and compact (closed and bounded)

Projected Gradient

Projected gradient method: solve (constrained) optimization problem

$$\min_x f(x) \quad \text{subject to } x \in \mathcal{C}$$

- $f : \mathbb{R}^n \to \mathbb{R}$ differentiable
- Set $\mathcal{C} \subseteq \mathbb{R}^n$ is non-empty, **convex** and compact (closed and bounded)

(Euclidean) Projection operator P

- Projection of x_0 onto convex set $\mathcal{C}$: point x' satisfying constraints specified by $\mathcal{C}$ that is the closest to x_0 (closeness: Euclidean distance $\|x - x_0\|_2$)
- **Projection (operator)** P of x_0 onto $\mathcal{C}$

$$\mathrm{P}_{\mathcal{C}}(x_0) = \operatorname*{argmin}_{x \in \mathcal{C}} \|x - x_0\|_2$$

- For convex $\mathcal{C}$, projection of x_0 onto $\mathcal{C}$ is unique

Projected Gradient

Projected gradient method: solve (constrained) optimization problem

$$\min_x f(x) \quad \text{subject to } x \in \mathcal{C}$$

- $f : \mathbb{R}^n \to \mathbb{R}$ differentiable
- Set $\mathcal{C} \subseteq \mathbb{R}^n$ is non-empty, **convex** and compact (closed and bounded)

(Euclidean) Projection operator P

- Projection of x_0 onto convex set $\mathcal{C}$: point x' satisfying constraints specified by $\mathcal{C}$ that is the closest to x_0 (closeness: Euclidean distance $\|x - x_0\|_2$)
- **Projection (operator)** P of x_0 onto $\mathcal{C}$

$$\mathrm{P}_{\mathcal{C}}(x_0) = \operatorname*{argmin}_{x \in \mathcal{C}} \|x - x_0\|_2$$

- For convex $\mathcal{C}$, projection of x_0 onto $\mathcal{C}$ is unique

Euclidean projection onto a polyhedron

The projection of x_0 onto a polyhedron described by linear inequalities $Ax \preceq b$ can be computed by solving the QP

$$\begin{aligned} \min_x \ & \|x - x_0\|_2^2 \\ s.t. \ & Ax \preceq b \end{aligned} \tag{40}$$

Some special cases admit analytical solutions

- Euclidean projection of x_0 onto **hyperplane** $\mathcal{C} = \{x \mid a^T x = b, a \neq 0\}$

$$P_{\mathcal{C}}(x_0) = x_0 + \frac{(b - a^T x_0)}{\|a\|_2^2} a \tag{41}$$

Euclidean projection onto a polyhedron

The projection of x_0 onto a polyhedron described by linear inequalities $Ax \preceq b$ can be computed by solving the QP

$$\begin{aligned} \min_x \quad & \|x - x_0\|_2^2 \\ s.t. \quad & Ax \preceq b \end{aligned} \tag{40}$$

Some special cases admit analytical solutions

- Euclidean projection of x_0 onto **hyperplane** $\mathcal{C} = \{x \mid a^T x = b, a \neq 0\}$

$$P_{\mathcal{C}}(x_0) = x_0 + \frac{(b - a^T x_0)}{\|a\|_2^2} a \tag{41}$$

- Euclidean projection of x_0 onto **affine set** $\mathcal{C} = \{x \mid A^T x = b, A \in \mathbb{R}^{m \times n}, rank(A) = m\}$

$$P_{\mathcal{C}}(x_0) = x_0 + A^T \left(A^T A\right)^{-1} \left(b - A^T x_0\right) \tag{42}$$

Euclidean projection onto a polyhedron

Some special cases admit analytical solutions

- Euclidean projection of x_0 on **halfspace** $\mathcal{C} = \{x \mid a^T x \leq b, a \neq 0\}$

$$P_{\mathcal{C}}(x_0) = \begin{cases} x_0 + \dfrac{(b - a^T x_0)}{\|a\|_2^2} a & a^T x_0 > b \\ x_0 & a^T x_0 \leq b \end{cases} \tag{43}$$

Euclidean projection onto a polyhedron

Some special cases admit analytical solutions

- Euclidean projection of x_0 on **halfspace** $\mathcal{C} = \{x \mid a^T x \leq b, a \neq 0\}$

$$P_{\mathcal{C}}(x_0) = \begin{cases} x_0 + \dfrac{(b - a^T x_0)}{\|a\|_2^2} a & a^T x_0 > b \\ x_0 & a^T x_0 \leq b \end{cases} \tag{43}$$

- Euclidean projection of x_0 on **rectangle** $\mathcal{C} = \{x \in \mathbb{R}^n \mid \ell \leq x \leq u, \ell < u\}$

$$P_{\mathcal{C}}(x_0)_k = \begin{cases} \ell_k & (x_0)_k \leq \ell_k \\ (x_0)_k & \ell_k \leq (x_0)_k \leq u_k \\ u_k & (x_0)_k \geq u_k \end{cases} \tag{44}$$

Euclidean projection onto a polyhedron

Some special cases admit analytical solutions

- Euclidean projection of x_0 on **halfspace** $\mathcal{C} = \{x \mid a^T x \leq b, a \neq 0\}$

$$P_{\mathcal{C}}(x_0) = \begin{cases} x_0 + \dfrac{(b - a^T x_0)}{\|a\|_2^2} a & a^T x_0 > b \\ x_0 & a^T x_0 \leq b \end{cases} \tag{43}$$

- Euclidean projection of x_0 on **rectangle** $\mathcal{C} = \{x \in \mathbb{R}^n \mid \ell \leq x \leq u, \ell < u\}$

$$P_{\mathcal{C}}(x_0)_k = \begin{cases} \ell_k & (x_0)_k \leq \ell_k \\ (x_0)_k & \ell_k \leq (x_0)_k \leq u_k \\ u_k & (x_0)_k \geq u_k \end{cases} \tag{44}$$

- Euclidean projection of x_0 on **nonnegative orthant** $\mathcal{C} = \{x \in \mathbb{R}^n \mid x \geq 0\}$

$$P_{\mathcal{C}}(x_0) = \Big(\max(0, x_{01}), \max(0, x_{02}), \ldots, \max(0, x_{0n}) \Big) \tag{45}$$

Projected Gradient

Iteration: $x_{k+1} = \mathrm{P}_{\mathcal{C}}(x_k - t_k \nabla f(x_k))$

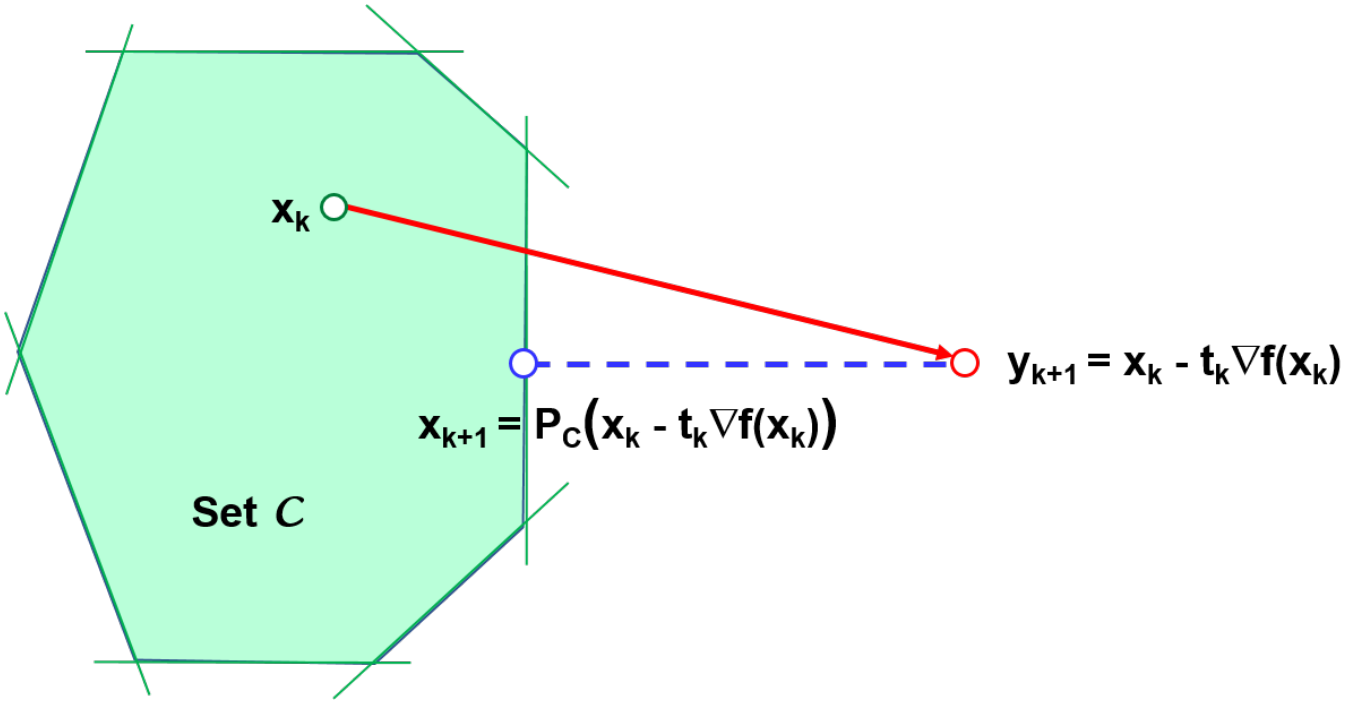


Figure: Projected Gradient Iteration

Projected Gradient

Two steps (in addition to step vector/descent direction and step size)

1. Perform unconstrained gradient descent step: $y_{k+1} = x_k - t_k \nabla f(x_k)$
2. Compute Euclidean projection of y_{k+1} onto $\mathcal{C}$:

$$x_{k+1} = \mathrm{P}_{\mathcal{C}}(y_{k+1}) = \mathrm{P}_{\mathcal{C}}\big(x_k - t_k \nabla f(x_k)\big)$$

Projected Gradient

Two steps (in addition to step vector/descent direction and step size)

1. Perform unconstrained gradient descent step: $y_{k+1} = x_k - t_k \nabla f(x_k)$
2. Compute Euclidean projection of y_{k+1} onto $\mathcal{C}$:

$$x_{k+1} = \mathrm{P}_{\mathcal{C}}(y_{k+1}) = \mathrm{P}_{\mathcal{C}}\big(x_k - t_k \nabla f(x_k)\big)$$

Projected Gradient algorithm:

`input`: choose initial point $x_0 \in dom(f)$
`repeat`:

1. Determine a **step**/search direction d_k (vector starting at x_k) $:= -\nabla f(x_k)$
2. Compute **step size** $t_k \in \mathbb{R}^+$: fixed $t_k = t$ or via inexact (e.g. backtracking) line search
3. Update: $y_{k+1} = x_k - t_k \nabla f(x_k)$
4. Project: $x_{k+1} = \mathrm{P}_{\mathcal{C}}(y_{k+1}) = \mathrm{P}_{\mathcal{C}}\big(x_k - t_k \nabla f(x_k)\big)$

`until`: stopping criterion is satisfied

Projected Gradient

- Line Search for Projected Gradient: (mainly) two methods: backtracking
  - **along feasible direction**: backtrack along line between x+ and x (search interior): costs 1 projection per iteration
  - **along projection arc**: backtrack by decreasing α and re-projecting (search boundary): costs 1 projection per backtrack

Projected Gradient

- Line Search for Projected Gradient: (mainly) two methods: backtracking
  - **along feasible direction**: backtrack along line between x+ and x (search interior): costs 1 projection per iteration
  - **along projection arc**: backtrack by decreasing α and re-projecting (search boundary): costs 1 projection per backtrack
- Projected-gradient is only efficient if the projection (computational) cost is relatively limited

Projected Gradient

- Line Search for Projected Gradient: (mainly) two methods: backtracking
  - **along feasible direction**: backtrack along line between x+ and x (search interior): costs 1 projection per iteration
  - **along projection arc**: backtrack by decreasing α and re-projecting (search boundary): costs 1 projection per backtrack

- Projected-gradient is only efficient if the projection (computational) cost is relatively limited

- **If** $\mathcal{C}$ = intersection of simple convex sets, i.e.,

$$\mathcal{C} := \bigcap_i \mathcal{C}_i \text{ with } \mathcal{C}_i = \{x \mid a_i^T x \leq b_i\}$$

  then feasibility problem corresponds to existence of intersection (Dykstra's algorithm, 1983)

  Dykstra algorithm : **iteratively** computes the nearest point projection (a.k.a. orthogonal projection) of any given point onto the intersection of finite number of general closed and convex sets

Projection algorithm: Dykstra

Best approximation problem (BAP)

- Given nonempty closed, convex sets $C_1, \ldots, C_p \subseteq \mathbb{R}^n$ with nonempty intersection, i.e., $\bigcap_{i=1}^p C_i \neq \emptyset$, and any point $y \in \mathbb{R}^n$
- Find the point in $C := \bigcap_{i=1}^p C_i = C_1 \cap \ldots \cap C_p$ that is closest to y

$$
\begin{aligned}
(BAP) \quad & \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} \ \|y - x\|_2^2 \\
& s.t. \ x \in C := \bigcap_{i=1}^{p} C_i = C_1 \cap \ldots \cap C_p
\end{aligned}
\tag{46}
$$

Projection algorithm: Dykstra

Best approximation problem (BAP)

- Given nonempty closed, convex sets $C_1, \ldots, C_p \subseteq \mathbb{R}^n$ with nonempty intersection, i.e., $\bigcap_{i=1}^p C_i \neq \emptyset$, and any point $y \in \mathbb{R}^n$
- Find the point in $C := \bigcap_{i=1}^p C_i = C_1 \cap \ldots \cap C_p$ that is closest to y

$$
\begin{aligned}
(BAP) \quad & \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} \ \|y - x\|_2^2 \\
& s.t. \ x \in C := \bigcap_{i=1}^{p} C_i = C_1 \cap \ldots \cap C_p
\end{aligned}
\tag{46}
$$

Note well

- There exists a unique solution x^* to (46), denoted $\mathrm{P}_C(y)$, i.e., the projection of y onto $C := \bigcap_{i=1}^p C_i$
- Assumes that for any $y \in \mathbb{R}^n$, computation of $\mathrm{P}_C(y)$ not trivial, whereas for each C_i, $P_{C_i}(y)$ is easy to obtain

Projection algorithm: Dykstra

Dykstra algorithm solves the (BAP) by

- Adding correction vectors before projecting onto each convex set individually to complete a cycle repeated iteratively
- Starting from initial point $y \in \mathbb{R}^n$, the Dykstra **cyclic** projections algorithm

$\triangleright$ `Initialize` $(k = 0)$

$$x^{(0)} = y$$

$$z^{(-p+1)} = \ldots = z^{(0)} = 0$$

`repeat` for $k = 1, 2, 3, \ldots$

$$x^{(k)} = \mathrm{P}_{C_{[k]}}\left(x^{(k-1)} + z^{(k-p)}\right)$$

$$z^{(k)} = x^{(k-1)} + z^{(k-p)} - x^{(k)}$$

Projection algorithm: Dykstra

Dykstra algorithm solves the (BAP) by

- Adding correction vectors before projecting onto each convex set individually to complete a cycle repeated iteratively
- Starting from initial point $y \in \mathbb{R}^n$, the Dykstra **cyclic** projections algorithm

$\triangleright$ `Initialize` $(k = 0)$

$$x^{(0)} = y$$

$$z^{(-p+1)} = \ldots = z^{(0)} = 0$$

`repeat` for $k = 1, 2, 3, \ldots$

$$x^{(k)} = \mathrm{P}_{C_{[k]}}\left(x^{(k-1)} + z^{(k-p)}\right)$$

$$z^{(k)} = x^{(k-1)} + z^{(k-p)} - x^{(k)}$$

In this algorithm

- $\mathrm{P}_{C_{[\cdot]}}(y) = \underset{x \in C_{[\cdot]}}{\operatorname{argmin}} \|y - x\|_2^2$ denotes the (Euclidean) projection of y onto a closed, convex set $C_{[\cdot]}$
- $[\cdot]$ denotes the modulo operator taking values in $\{1, \ldots, p\}$

Projection algorithm: Dykstra

- When convex sets C_i, $i = 1, \ldots, p$ are closed affine subspaces : projection is a linear map and Dykstra algorithm reduces to classical alternating projections method (MAP)

Projection algorithm: Dykstra

- When convex sets C_i, $i = 1, \ldots, p$ are closed affine subspaces : projection is a linear map and Dykstra algorithm reduces to classical alternating projections method (MAP)
- **Dykstra algorithm vs. MAP** : sequence of (pseudo) dual variables $z^{(k)}$ $(k = 1, 2, 3, \ldots)$ that track, in a cyclic fashion, the residuals from projecting onto $C_1, \ldots, C_p$
  - MAP will always converge to a feasible point in $C_1 \cap \ldots \cap C_p$
  - MAP will not necessarily converge to the solution of the (BAP) unless $C_1, \ldots, C_p$ are affine subspaces (in which case alternating projections and Dykstra algorithm coincide)

$\Rightarrow$ MAP does not converge to the solution of the BAP in the general case

Projection algorithm: Dykstra

- When convex sets C_i, $i = 1, \ldots, p$ are closed affine subspaces : projection is a linear map and Dykstra algorithm reduces to classical alternating projections method (MAP)
- **Dykstra algorithm vs. MAP** : sequence of (pseudo) dual variables $z^{(k)}$ $(k = 1, 2, 3, \ldots)$ that track, in a cyclic fashion, the residuals from projecting onto $C_1, \ldots, C_p$
  - MAP will always converge to a feasible point in $C_1 \cap \ldots \cap C_p$
  - MAP will not necessarily converge to the solution of the (BAP) unless $C_1, \ldots, C_p$ are affine subspaces (in which case alternating projections and Dykstra algorithm coincide)

  $\Rightarrow$ MAP does not converge to the solution of the BAP in the general case
- The Dykstra algorithm converges in general for any closed, convex sets $C_1, \ldots, C_p$ with nonempty intersection, i.e., $\bigcap_{i=1}^{p} C_i \neq \emptyset$

Projection algorithm: Dykstra (alternative)

The Dykstra algorithm solves (46) by generating two sequences: the sequence of **iterates** $\{x_i^{(k)}\}$ and the sequence of **increments** $\{z_i^{(k)}\}$ defined recursively as

Dykstra algorithm:

▷ `Initialize`

$$x_p^{(k=0)} = y \ (= x^0)$$

$$z_i^{(k=0)} = 0, \ i = 1, \ldots, p$$

`repeat` for $k = 1, 2, 3, \ldots$

$$x_0^{(k)} = x_p^{(k-1)}$$

`for` $i = 1, 2, \ldots, p$

$$x_i^{(k)} = \mathrm{P}_{C_i}\left(x_{i-1}^{(k)} + z_i^{(k-1)}\right)$$

$$z_i^{(k)} = \left(x_{i-1}^{(k)} - x_i^{(k)}\right) + z_i^{(k-1)}$$

Example : $p = 2$ thus $C = C_1 \cap C_2$

$\mathrm{k} = 1 : \; x_0^{(1)} = x_2^{(0)}$

i = 1:

$$x_1^{(1)} = \mathrm{P}_{C_1}\left(x_0^{(1)} + z_1^{(0)}\right)$$

$$z_1^{(1)} = \left(x_0^{(1)} - x_1^{(1)}\right) + z_1^{(0)}$$

i = 2:

$$x_2^{(1)} = \mathrm{P}_{C_2}\left(x_1^{(1)} + z_2^{(0)}\right)$$

$$z_2^{(1)} = \left(x_1^{(1)} - x_2^{(1)}\right) + z_2^{(0)}$$

$\mathrm{k} = 2 : \; x_0^{(2)} = x_2^{(1)}$

i = 1:

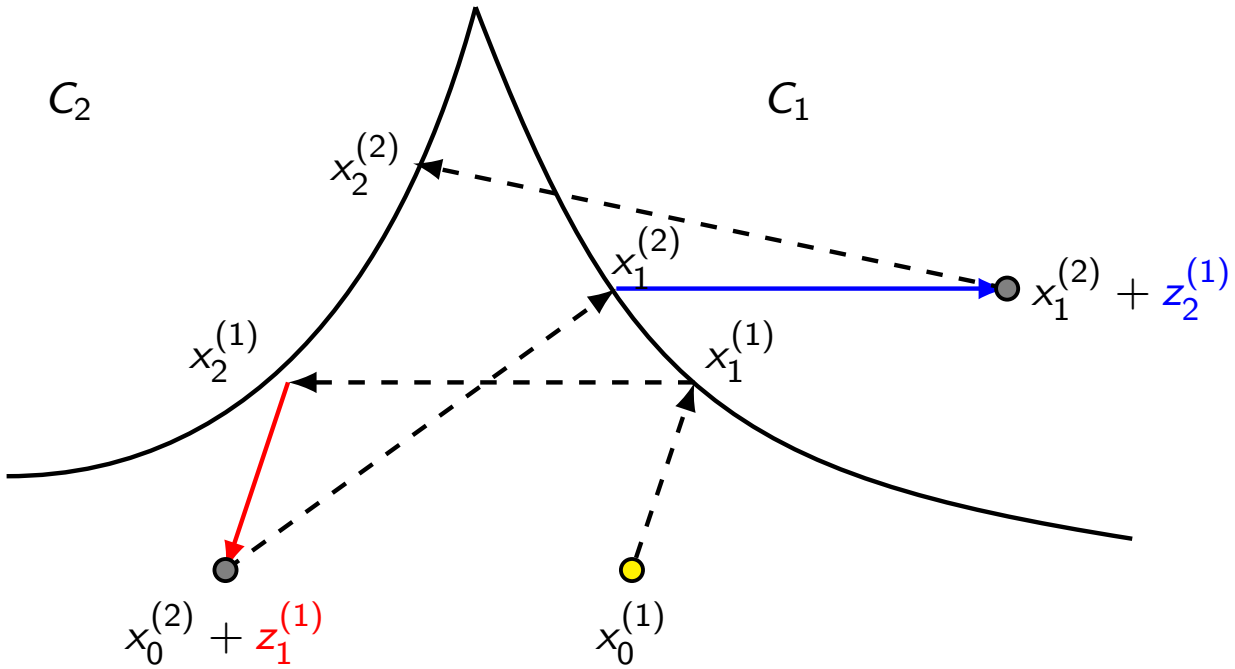
$$x_1^{(2)} = \mathrm{P}_{C_1}\left(x_0^{(2)} + z_1^{(1)}\right)$$

$$z_1^{(2)} = \left(x_0^{(2)} - x_1^{(2)}\right) + z_1^{(1)}$$

i = 2:

$$x_2^{(2)} = \mathrm{P}_{C_2}\left(x_1^{(2)} + z_2^{(1)}\right)$$

$$z_2^{(2)} = \left(x_1^{(2)} - x_2^{(2)}\right) + z_2^{(1)}$$



Projection algorithm: Dykstra (alternative)

Dykstra cyclic projections algorithm

- attractive method for approximating the intersection of closed convex sets
- converges to feasible point closest to (initial point) y by only requiring knowledge about the projections onto the individual convex sets C_i

Projection algorithm: Dykstra (alternative)

Dykstra cyclic projections algorithm

- attractive method for approximating the intersection of closed convex sets
- converges to feasible point closest to (initial point) y by only requiring knowledge about the projections onto the individual convex sets C_i

Particular cases

- Sets C_i are closed half-spaces, i.e., the intersection is a polyhedral subset of $\mathbb{R}^n$: (asymptotic) rate of convergence is linear
- The convergence rate for general C_i remains unknown

Projection algorithm: Hildreth

Hildreth algorithm (1957) : row action method for solving systems of linear inequalities

$$Ax \leq b, \ A \in \mathbb{R}^{m \times n}, \ x \in \mathbb{R}^n, b \in \mathbb{R}^m$$

Projection algorithm: Hildreth

Hildreth algorithm (1957) : row action method for solving systems of linear inequalities

$$Ax \leq b,\ A \in \mathbb{R}^{m \times n},\ x \in \mathbb{R}^n, b \in \mathbb{R}^m$$

Principle : orthogonal projection of the current iterate onto chosen half-space

while finding the closest point of the feasible set (polyhedron) to a given point y

$$x^* = \underset{x \in \mathbb{R}^n}{\operatorname{argmin}} \|y - x\|^2 \tag{47}$$

$$\text{subject to } Ax \leq b \tag{48}$$

Projection algorithm: Hildreth

Let

- a_i be i^{th} row of matrix $A \in \mathbb{R}^{m \times n}$
- b_i be i^{th} coordinate of vector $b \in \mathbb{R}^m$
- System of linear inequalities $\{x : \langle a_i \mid x \rangle \leq b_i\}, i = 1, \ldots, m$

Hildreth algorithm generates sequences $\{x^{(k)}\} \in \mathbb{R}^n$ and $\{z^{(k)}\} \in \mathbb{R}^m$

$\triangleright$ `Initialize`

`set` $x^{(0)}$ arbitrary initial approximation

`set` $z^{(0)} = 0$

`for` $k = 0, 1, 2, \ldots$

$i(k) = 1 + k \mod m$ // cyclical control sequence : constraints order

$$c^{(k)} = \min \left\{ z_{i(k)}^{(k)}, \frac{b_{i(k)} - \left\langle a_{i(k)} \mid x^{(k)} \right\rangle}{\left\| a_{i(k)} \right\|^2} \right\}$$

$x^{(k+1)} = x^{(k)} + c^{(k)} a_{i(k)}$

`for` $i = 1, 2, \ldots, m$

`if` $i == i(k)$ `then` $z_i^{(k+1)} = z_i^{(k)} - c^{(k)}$

`else` $z_i^{(k+1)} = z_i^{(k)}$

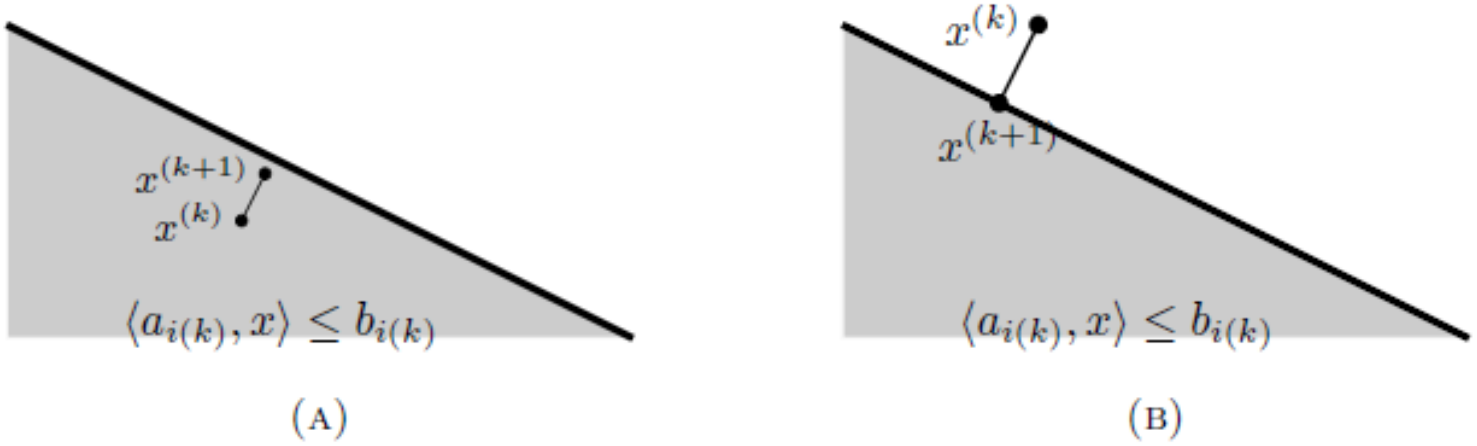
Projection algorithm: Hildreth

The sequence $i(k)$ indicates the order in which the constraints are used

- **If** constraint $\{x \in \mathbb{R}^n : \langle a_{i(k)} \mid x \rangle \leq b_{i(k)}\}$ is satisfied

  Then (A) move last approximate $x^{(k+1)}$ closer to the hyperplane $\{x \in \mathbb{R}^n : \langle a_{i(k)} \mid x \rangle = b_{i(k)}\}$

  Else (B) project last approximate $x^{(k+1)}$ onto the half-space $\{x \in \mathbb{R}^n : \langle a_{i(k)} \mid x \rangle \leq b_{i(k)}\}$
- The definition of $c^{(k)}$ guarantees no over projection (i.e. staying on half-space)
- Vector $z^{(k)}$ only updated at component $i(k) = i$ (all $z^{(k)}$ entries never negative)



Projection algorithm: Hildreth

Hildreth algorithm

- Particular instance of projection onto convex sets (POCS)

  Solution x^* is called the projection of $y = x^{(0)}$ onto $\mathcal{C} = \cap_{i=1}^{p} \mathcal{C}_i$, $\mathcal{C}_i = \{x \in \mathbb{R}^n \mid a_i^T x \leq b_i\}$, denoted $\mathrm{P}_C(y)$
- Iterates

  ◦) $x^{(k)}$ approximate the solution x^*

  ◦) $z^{(k)}$ approximate the Lagrange multipliers associated with x^*

  $\rightarrow$ referred to as primal-dual optimization method (where $z^{(k)}$ is the sequence of dual iterates)
- Convergence rate (i.e., the asymptotic behavior of sequence $x(k)$) produced by the Hildreth algorithm is linear

Projection algorithm: Hildreth

Hildreth algorithm

- Particular instance of projection onto convex sets (POCS)

  Solution x^* is called the projection of $y = x^{(0)}$ onto $\mathcal{C} = \cap_{i=1}^{p} \mathcal{C}_i$, $\mathcal{C}_i = \{x \in \mathbb{R}^n \mid a_i^T x \leq b_i\}$, denoted $\mathrm{P}_C(y)$
- Iterates

  ◦) $x^{(k)}$ approximate the solution x^*

  ◦) $z^{(k)}$ approximate the Lagrange multipliers associated with x^*

  $\rightarrow$ referred to as primal-dual optimization method (where $z^{(k)}$ is the sequence of dual iterates)
- Convergence rate (i.e., the asymptotic behavior of sequence $x(k)$) produced by the Hildreth algorithm is linear

Note: when projecting onto a polyhedron, e.g., when the closed convex sets $\mathcal{C}_i = \{x \in \mathbb{R}^n \mid a_i^T x \leq b_i\}$ are half spaces, Dykstra reduces to Hildreth algorithm

Gradient Methods: Limits

Advantages:

- Every iteration is computationally inexpensive (provided gradient can be efficiently computed)
- By definition (of first-order), the algorithm does not require second-order derivatives
- If contraction factor α not too small convergence rate $O(1/k)$ preserved compared to constant step size

Gradient Methods: Limits

Advantages:

- Every iteration is computationally inexpensive (provided gradient can be efficiently computed)
- By definition (of first-order), the algorithm does not require second-order derivatives
- If contraction factor α not too small convergence rate $O(1/k)$ preserved compared to constant step size

Disadvantages:

- Convergence rate strongly dependent on function properties (convex vs. non-convex, Lipschitz constant, etc.)
- Does not handle non-differentiable functions (converges to non-optimal point).

Gradient Methods: Addressing the limits

To address one or both shortcomings of the gradient method

1. Methods for non-differentiable and/or constrained problems

   ◦) Subgradient method - cf. Section 2.5

   ◦) Proximal gradient method - cf. Section 2.6

   ◦) Smoothing methods - cf. Remark end of Section 2.6 (Moreau envelope proximal smoothing)

   ◦) Cutting-plane methods - cf. Section 7a (partly)

Gradient Methods: Addressing the limits

To address one or both shortcomings of the gradient method

1. Methods for non-differentiable and/or constrained problems
   - ◦) Subgradient method - cf. Section 2.5
   - ◦) Proximal gradient method - cf. Section 2.6
   - ◦) Smoothing methods - cf. Remark end of Section 2.6 (Moreau envelope proximal smoothing)
   - ◦) Cutting-plane methods - cf. Section 7a (partly)

2. Methods with improved convergence
   - ◦) Conjugate gradient method
   - ◦) Accelerated (proximal) gradient method - cf. Section 2.6
   - ◦) Quasi-Newton methods - cf. Section 3.5

INFO-F524: Continuous Optimization
– Section 2 : Part 2 –

Bernard Fortz, Dimitri Papadimitriou

Bureau: **2N3 206**

Université Libre de Bruxelles

dimitrios.papadimitriou@ulb.be

Annee Académique: 2024-2025

References

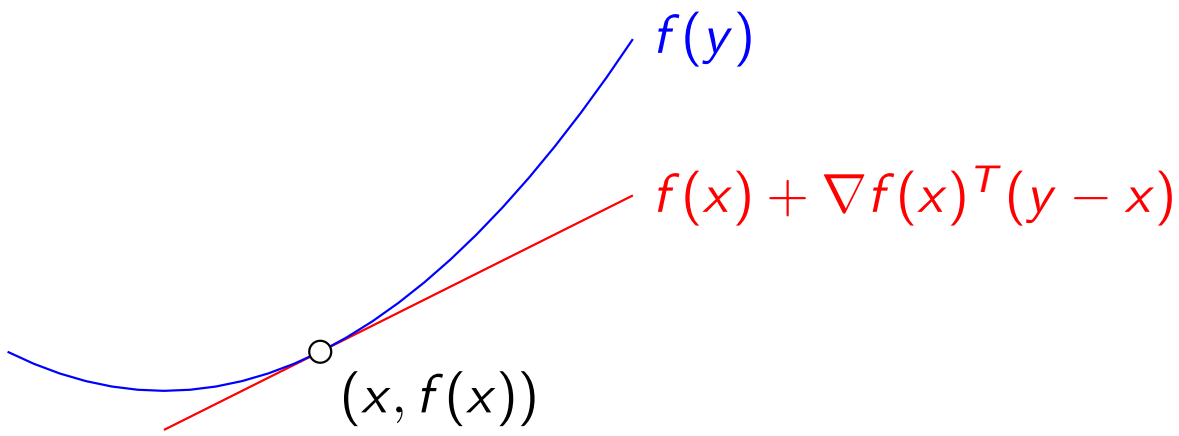
- R. Tyrrell Rockafellar and Roger J-B Wets, *Varational Analysis*, 1997, 2nd printing 2004, 3rd printing 2009, https://sites.math.washington.edu/~rtr/papers/rtr169-VarAnalysis-RockWets.pdf
- Boris S. Mordukhovich and Nguyen Mau Nam, *Convex Analysis and Beyond*, Volume I: Basic Theory, Springer International Publishing, 2022.
- A. Beck, *First-Order Methods in Optimization*, MOS-SIAM Series on Optimization, 2017.
- Jean-Pierre Aubin and Ivar Ekeland, *Applied Nonlinear Analysis*, Dover Books on Mathematics, 1984.

Subgradient method: motivation

Recall: if function f **convex** and differentiable, then

$$f(y) \geq f(x) + \nabla f(x)^T (y - x), \ \forall y \in \mathrm{dom}(f) \tag{1}$$

That is, the first-order approximation of f at x is a global underestimator, i.e., a global lower bound, of f at x



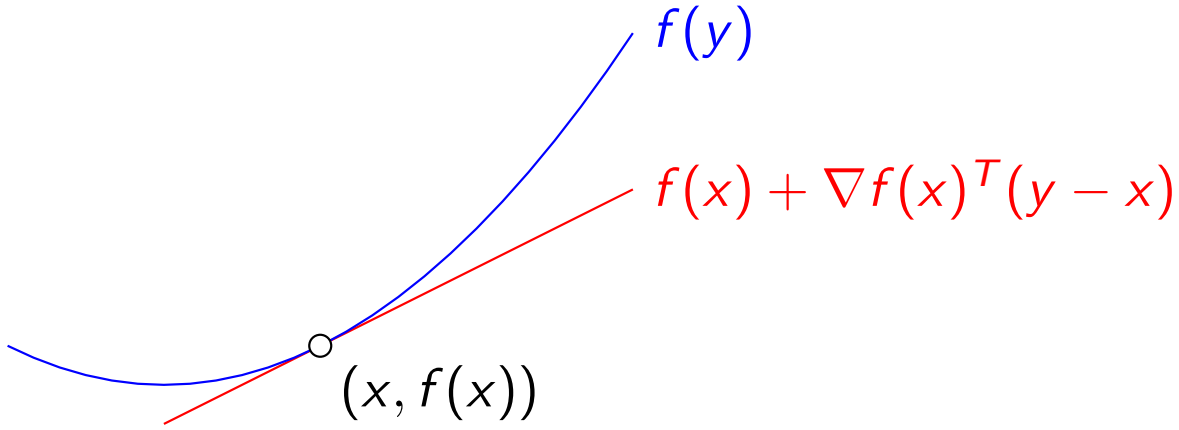
What if the function f is not differentiable?

Subgradient method: motivation

Recall: if function f **convex** and differentiable, then

$$f(y) \geq f(x) + \nabla f(x)^T (y - x), \ \forall y \in \mathrm{dom}(f) \tag{1}$$

That is, the first-order approximation of f at x is a global underestimator, i.e., a global lower bound, of f at x



What if the function f is not differentiable? This case can occur in, e.g.,

i) algorithms for nondifferentiable convex optimization

ii) convex analysis, e.g., optimality conditions, duality for nondifferentiable problems

Subgradient: definition

Subgradient

A subgradient of a convex function f at $x \in dom(f)$ is any $g \in \mathbb{R}^n$ such that

$$f(y) \geq f(x) + g^T(y - x), \ \forall y \in dom(f) \tag{2}$$

Thus, g is a subgradient of f at x **iff** $f(x) + g^T(y - x)$ is a global (affine) underestimator of f

Subgradient: definition

Subgradient

A subgradient of a convex function f at $x \in dom(f)$ is any $g \in \mathbb{R}^n$ such that

$$f(y) \geq f(x) + g^T(y - x), \ \forall y \in dom(f) \tag{2}$$

Thus, g is a subgradient of f at x **iff** $f(x) + g^T(y - x)$ is a global (affine) underestimator of f

Properties:

- The subgradient g always exists (in the relative interior of dom(f), denoted **relint dom**(f))
- If f is differentiable at x, then the subgradient of f at x is $g = \nabla f(x)$ uniquely; that is, $\nabla f(x)$ is a subgradient of f at x

For nonconvex f: generalized gradients (generalize subdifferentials)

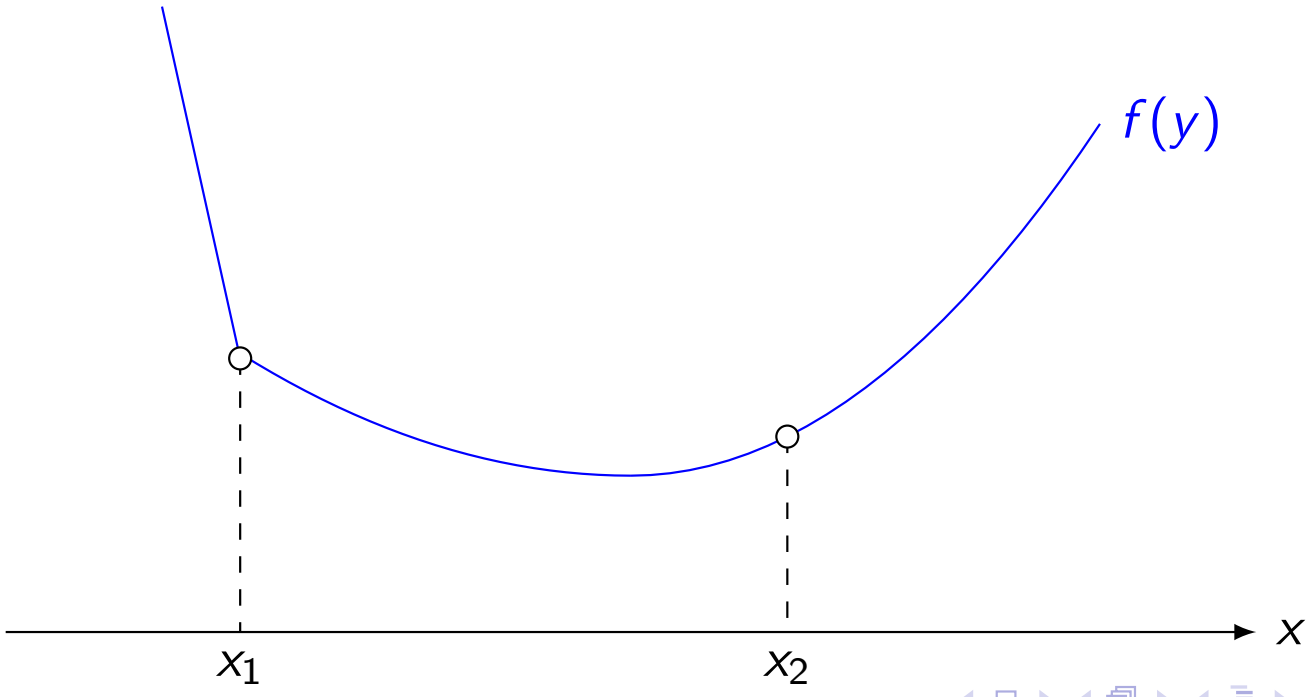
Subgradient: definition

Subgradient

A subgradient of a convex function f at $x \in \mathrm{dom}(f)$ is any $g \in \mathbb{R}^n$ such that

$$f(y) \geq f(x) + g^T(y - x), \ \forall y \in dom(f) \tag{3}$$

Thus, g is a subgradient of f at x **iff** $f(x) + g^T(y - x)$ is a global (affine) underestimator of f.



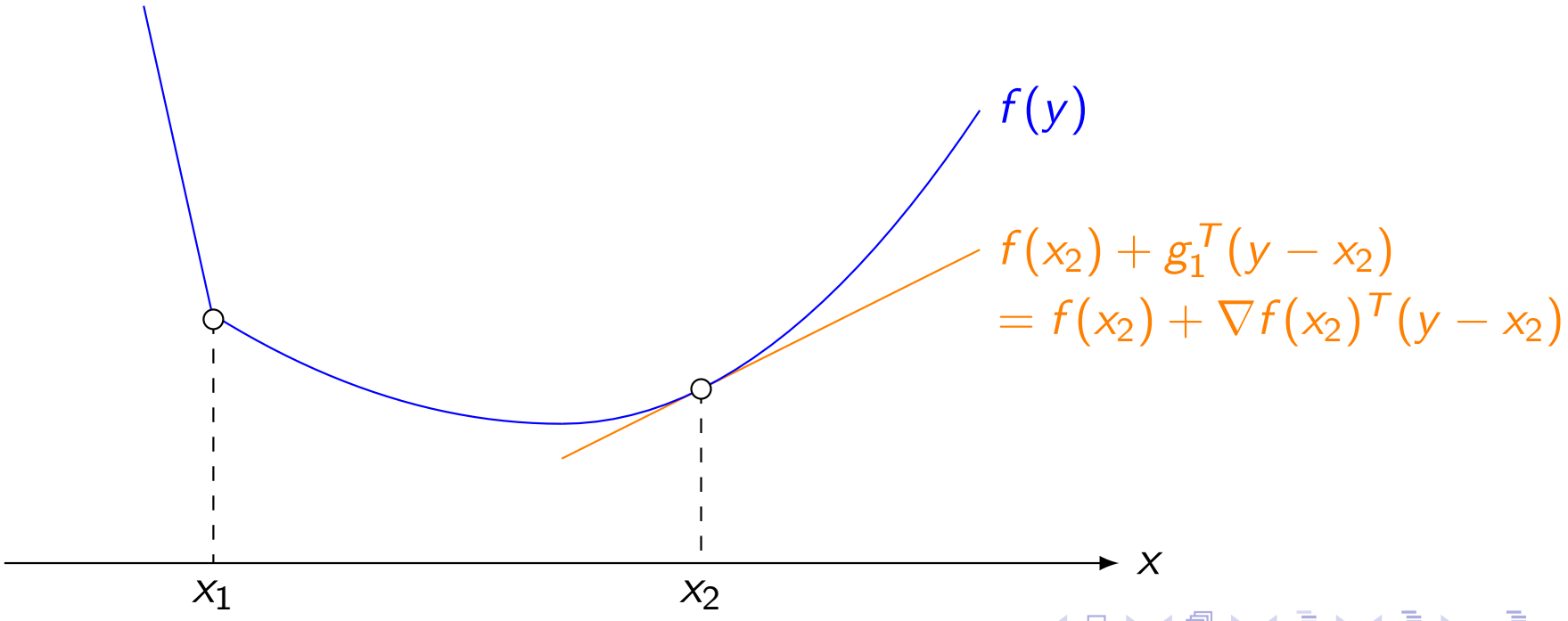
Subgradient: definition

Subgradient

A subgradient of a convex function f at $x \in \mathrm{dom}(f)$ is any $g \in \mathbb{R}^n$ such that

$$f(y) \geq f(x) + g^T(y - x), \ \forall y \in dom(f) \tag{4}$$

Thus, g is a subgradient of f at x **iff** $f(x) + g^T(y - x)$ is a global (affine) underestimator of f



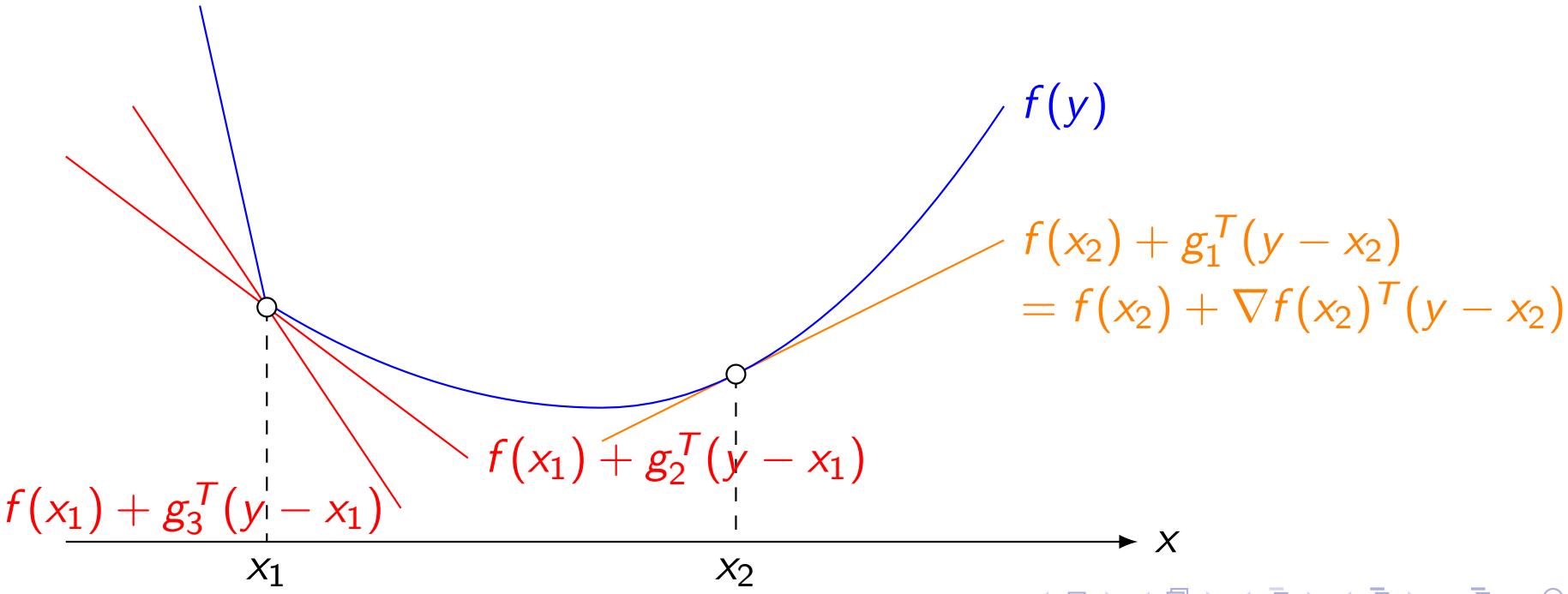
Subgradient: definition

Subgradient

A subgradient of a convex function f at $x \in \mathrm{dom}(f)$ is any $g \in \mathbb{R}^n$ such that

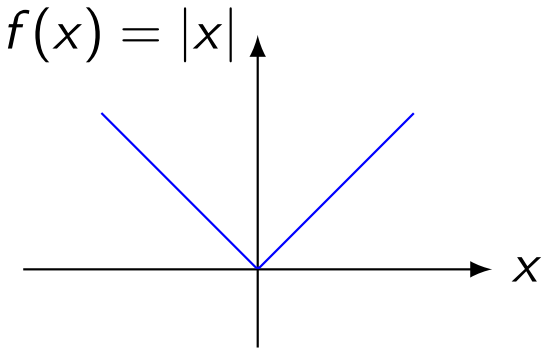
$$f(y) \geq f(x) + g^T(y - x), \ \forall y \in dom(f) \tag{5}$$

Thus, g is a subgradient of f at x **iff** $f(x) + g^T(y - x)$ is a global (affine) underestimator of f



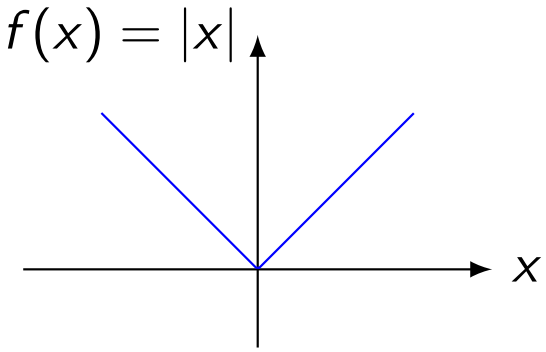
Subgradient: examples

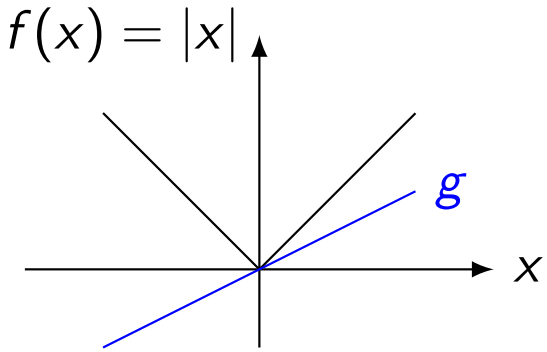
- $f : \mathbb{R} \to \mathbb{R} : x \to |x|$
  - For $x \neq 0$, f is differentiable and subgradient identical to gradient, $\text{sign}(x)$
  - For $x = 0$, any value in [-1, 1] is a valid subgradient
    because any line passing through $x = 0$ w/ slope in [-1, 1] lower bounds f



Subgradient: examples

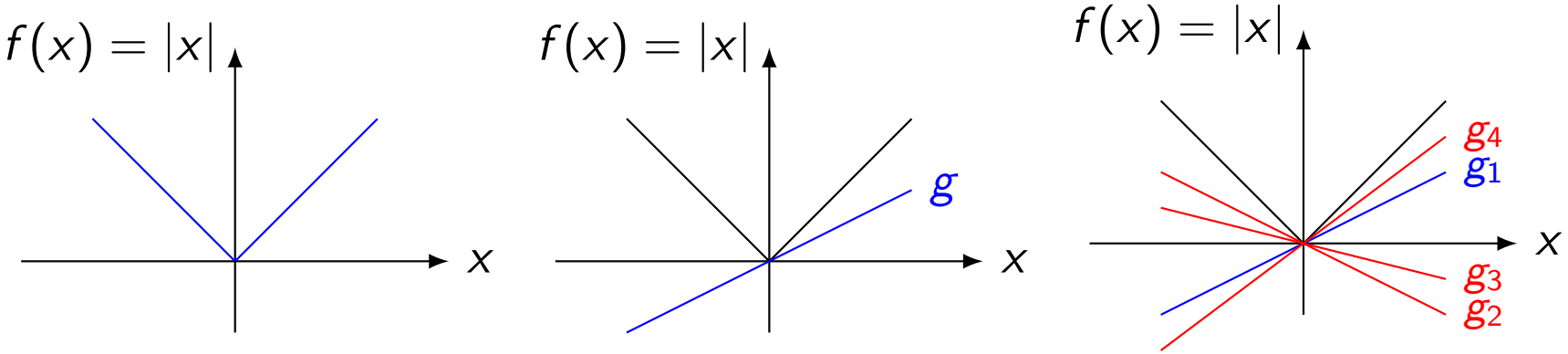
- $f : \mathbb{R} \to \mathbb{R} : x \to |x|$
  - For $x \neq 0$, f is differentiable and subgradient identical to gradient, $\text{sign}(x)$
  - For $x = 0$, any value in [-1, 1] is a valid subgradient
    because any line passing through $x = 0$ w/ slope in [-1, 1] lower bounds f





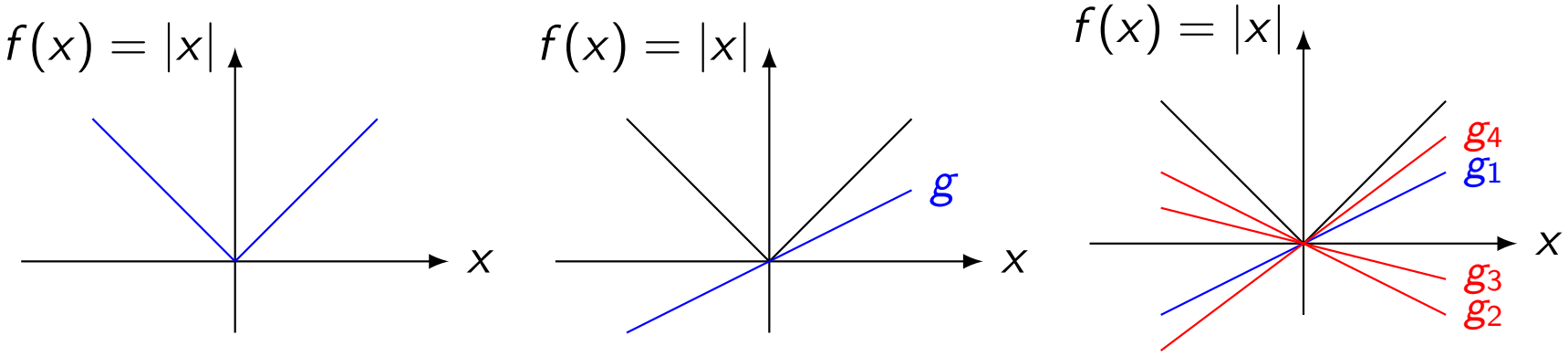
Subgradient: examples

- $f : \mathbb{R} \to \mathbb{R} : x \to |x|$
  - For $x \neq 0$, f is differentiable and subgradient identical to gradient, $\text{sign}(x)$
  - For $x = 0$, any value in [-1, 1] is a valid subgradient
    because any line passing through $x = 0$ w/ slope in [-1, 1] lower bounds f



Subgradient: examples

- $f : \mathbb{R} \to \mathbb{R} : x \to |x|$
  - For $x \neq 0$, f is differentiable and subgradient identical to gradient, $\text{sign}(x)$
  - For $x = 0$, any value in [-1, 1] is a valid subgradient
    because any line passing through $x = 0$ w/ slope in [-1, 1] lower bounds f



- ℓ_1-**norm**: $f : \mathbb{R}^n \to \mathbb{R} : x \to \|x\|_1$ –cf. infra
- ℓ_2-**norm** $f : \mathbb{R}^n \to \mathbb{R} : x \to \|x\|_2$ –cf. infra

Subgradient: Geometric interpretation

- From subdifferential inequality: g is a subgradient at x means

$$f(y) \geq f(x) + g^T(y - x) \tag{6}$$

Subgradient: Geometric interpretation

- From subdifferential inequality: g is a subgradient at x means

$$f(y) \geq f(x) + g^T(y - x) \tag{6}$$

- **Consequence**

$$f(y) \leq f(x) \Rightarrow g^T(y - x) \leq 0 \tag{7}$$

Subgradient: Geometric interpretation

- From subdifferential inequality: g is a subgradient at x means

$$f(y) \geq f(x) + g^T(y - x) \tag{6}$$

- **Consequence**

$$f(y) \leq f(x) \Rightarrow g^T(y - x) \leq 0 \tag{7}$$

  - **If** f is differentiable at x_0, **then** ∇x_0 is normal to the sublevel set through x_0: $\{x \mid f(x) \leq f(x_0)\}$
  - **If** f is nondifferentiable at x_0, **then** the nonzero subgradients at x_0 defines a supporting hyperplane to the sublevel set through x_0: $\{x \mid f(x) \leq f(x_0)\}$

Subgradient and Subdifferential

- A function f is called **subdifferentiable** at point x if there exists at least one subgradient at x
- The set of subgradients of f at point x is called the **subdifferential** of f at x, and is denoted $\partial f(x)$

Subgradient and Subdifferential

- A function f is called **subdifferentiable** at point x if there exists at least one subgradient at x
- The set of subgradients of f at point x is called the **subdifferential** of f at x, and is denoted $\partial f(x)$

Definition: subdifferential

The **subdifferential** of the function f at x (denoted $\partial \mathbf{f}(\mathbf{x})$) is defined as the set of all subgradients of f at x:

$$\begin{aligned} \partial f(x) &:= \{g \mid g^T(y-x) \leq f(y) - f(x), \forall y \in \mathrm{dom}(f)\} \\ &:= \bigcap_{y \in dom(f)} \{g \mid g^T(y-x) \leq f(y) - f(x)\} \end{aligned} \tag{8}$$

Subgradient and Subdifferential

- A function f is called **subdifferentiable** at point x if there exists at least one subgradient at x
- The set of subgradients of f at point x is called the **subdifferential** of f at x, and is denoted $\partial f(x)$

Definition: subdifferential

The **subdifferential** of the function f at x (denoted $\partial \mathbf{f}(\mathbf{x})$) is defined as the set of all subgradients of f at x:

$$\begin{aligned} \partial f(x) &:= \{g \mid g^T(y-x) \leq f(y) - f(x), \forall y \in \mathrm{dom}(f)\} \\ &:= \bigcap_{y \in dom(f)} \{g \mid g^T(y-x) \leq f(y) - f(x)\} \end{aligned} \tag{8}$$

Important: a function f is called subdifferentiable if it is subdifferentiable at **all** $x \in \mathrm{dom}(f)$ –notation: ∂f

Subdifferential: properties

Base properties (f convex):

- **If** $x \in \mathit{int}\,\mathrm{dom}(f)$, **then** the subdifferential $\partial f(x)$ is nonempty and bounded (follows from f being continuous at x)

Subdifferential: properties

Base properties (f convex):

- **If** $x \in \mathit{int}\,\mathrm{dom}(f)$, **then** the subdifferential $\partial f(x)$ is nonempty and bounded (follows from f being continuous at x)
- **If** f is differentiable at x, **then** $\partial f(x) = \{\nabla f(x)\}$, i.e., its gradient is its only subgradient

Subdifferential: properties

Base properties (f convex):

- **If** $x \in \mathit{int}\,\mathrm{dom}(f)$, **then** the subdifferential $\partial f(x)$ is nonempty and bounded (follows from f being continuous at x)
- **If** f is differentiable at x, **then** $\partial f(x) = \{\nabla f(x)\}$, i.e., its gradient is its only subgradient
- **If** $\partial f(x) = \{g\}$, **then** f is differentiable at x and $g = \nabla f(x)$

Subdifferential: properties

Base properties (f convex):

- **If** $x \in \mathit{int}\,\mathrm{dom}(f)$, **then** the subdifferential $\partial f(x)$ is nonempty and bounded (follows from f being continuous at x)
- **If** f is differentiable at x, **then** $\partial f(x) = \{\nabla f(x)\}$, i.e., its gradient is its only subgradient
- **If** $\partial f(x) = \{g\}$, **then** f is differentiable at x and $g = \nabla f(x)$

Note: $\partial f(x)$ is always a closed and convex set (even for nonconvex f)

- This property follows from the definition of $\partial f(x)$ defined as an intersection of an (infinite) set of half-spaces

Subgradient Calculus

Weak subgradient calculus: rules for finding **one subgradient**

- Many algorithms for nondifferentiable convex optimization require only one subgradient at each step, so weak calculus often suffices
- If $f(x)$ can be evaulated, then one can usually compute a subgradient $g \in \partial f(x)$

Subgradient Calculus

Weak subgradient calculus: rules for finding **one subgradient**

- Many algorithms for nondifferentiable convex optimization require only one subgradient at each step, so weak calculus often suffices
- If $f(x)$ can be evaulated, then one can usually compute a subgradient $g \in \partial f(x)$

Strong subgradient calculus: rules for finding $\partial f(x)$, i.e., **all subgradients**

- Some algorithms, optimality conditions, etc., need entire subdifferential
- Compute entire subdifferential can be quite complicated

Subgradient Calculus (1)

Let f is convex, and $x \in \textit{relint}\,\mathrm{dom}(f)$[1]

- **Scaling** ($\alpha > 0$): $\partial(\alpha f)(x) = \alpha \partial f(x)$

[1] Formally, the relative interior of a set X is defined as its interior within the affine hull of X

Subgradient Calculus (1)

Let f is convex, and $x \in \textit{relint}\,\mathrm{dom}(f)$[1]

- **Scaling** ($\alpha > 0$): $\partial(\alpha f)(x) = \alpha \partial f(x)$
- **Addition**: $\partial(f_1 + \ldots + f_m)(x) = \partial f_1(x) + \ldots + \partial f_m(x)$

[1] Formally, the relative interior of a set X is defined as its interior within the affine hull of X

Subgradient Calculus (1)

Let f is convex, and $x \in \textit{relint}\,\mathrm{dom}(f)$[1]

- **Scaling** ($\alpha > 0$): $\partial(\alpha f)(x) = \alpha \partial f(x)$
- **Addition**: $\partial(f_1 + \ldots + f_m)(x) = \partial f_1(x) + \ldots + \partial f_m(x)$
- **Nonnegative linear combination** ($\alpha_1, \alpha_2 \geq 0$):

$$\partial(\alpha_1 f_1 + \alpha_2 f_2)(x) = \alpha_1 \partial f_1(x) + \alpha_2 \partial f_2(x)$$

[1] Formally, the relative interior of a set X is defined as its interior within the affine hull of X

Subgradient Calculus (1)

Let f is convex, and $x \in \textit{relint}\,\mathrm{dom}(f)$[1]

- **Scaling** ($\alpha > 0$): $\partial(\alpha f)(x) = \alpha \partial f(x)$
- **Addition**: $\partial(f_1 + \ldots + f_m)(x) = \partial f_1(x) + \ldots + \partial f_m(x)$
- **Nonnegative linear combination** ($\alpha_1, \alpha_2 \geq 0$):

$$\partial(\alpha_1 f_1 + \alpha_2 f_2)(x) = \alpha_1 \partial f_1(x) + \alpha_2 \partial f_2(x)$$

- **Affine transformation of variables**: if $f(x) = h(Ax + b)$, then

$$\partial f(x) = A^\top \partial h(Ax + b)$$

[1] Formally, the relative interior of a set X is defined as its interior within the affine hull of X

Subgradient Calculus (2): Composition

Composition: suppose $f(x) = h(f_1(x), \ldots, f_m(x))$, with function h convex nondecreasing and f_i convex, for all $i = 1, \ldots, m$

- Find
$$q \in \partial h(f_1(x), \ldots, f_m(x)), g_i \in \partial f_i(x), \text{ for all } i = 1, \ldots, m$$
- Then, subgradient g of f at x
$$g = q_1 g_1 + \ldots + q_m g_m \in \partial f(x)$$
- <u>Proof</u>:

$$\begin{aligned} f(y) &= h(f_1(y), \ldots, f_m(y)) \\ &\geq h(f_1(x) + g_1^\top (y - x), \ldots, f_m(x) + g_m^\top (y - x)) \\ &\geq h(f_1(x), \ldots, f_m(x)) + q^\top (g_1^\top (y - x), \ldots, g_m^\top (y - x)) \\ &= f(x) + g^\top (y - x) \end{aligned}$$

Subgradient Calculus (2): Expectation

Expectation: If $f(x) = \mathbf{E}[f(x, \omega)]$, with ω random variable, f convex in x for every ω

To find a subgradient of f at $\hat{x}$

- for each ω, choose a function $\omega \to g(\omega)$ with $g(\omega) \in \partial_x f(\hat{x}, \omega)$
- then, subgradient $g = \mathbf{E}_\omega[g(\omega)] \in \partial f(\hat{x})$

Subgradient Calculus (3): Finite pointwise maximum

Pointwise maximum of finite set of subdifferentiable convex functions $f_1, \ldots, f_m$

$$f(x) = \max\{f_1(x), \ldots, f_m(x)\} = \max_{i=1,\ldots,m} f_i(x) \tag{9}$$

- Set of active functions at x: $I(x) := \{i \mid f_i(x) = f(x)\}$
- **Weak result**: to find a subgradient of f at x
  - Choose any of the functions that achieves the maximum at point x, i.e.,

$$f_\kappa(x) = f(x), \kappa \in I(x)$$

  - Choose any subgradient of that function at point x, i.e., $g \in \partial f_\kappa(x)$.
  - Then $g \in \partial f(x)$. This property follows from

$$f(y) \geq f_\kappa(y) \geq f_\kappa(x) + g^\top(y - x) = f(x) + g^\top(y - x)$$

Subgradient Calculus (3): Finite pointwise maximum

Pointwise maximum of finite set of subdifferentiable convex functions $f_1, \ldots, f_m$

$$f(x) = \max\{f_1(x), \ldots, f_m(x)\} = \max_{i=1,\ldots,m} f_i(x) \tag{9}$$

- Set of active functions at x: $I(x) := \{i \mid f_i(x) = f(x)\}$
- **Weak result**: to find a subgradient of f at x
  - Choose any of the functions that achieves the maximum at point x, i.e.,

$$f_\kappa(x) = f(x), \kappa \in I(x)$$

  - Choose any subgradient of that function at point x, i.e., $g \in \partial f_\kappa(x)$.
  - Then $g \in \partial f(x)$. This property follows from

$$f(y) \geq f_\kappa(y) \geq f_\kappa(x) + g^\top(y - x) = f(x) + g^\top(y - x)$$

- **Strong result**: the subdifferential of the maximum of functions is the convex hull (conv) of the union of subdifferentials of the active functions at x

$$\partial f(x) = \operatorname{conv} \bigcup \{\partial f_i(x) \mid f_i(x) = f(x)\} = \operatorname{conv} \bigcup_{i \in I(x)} \partial f_i(x)$$

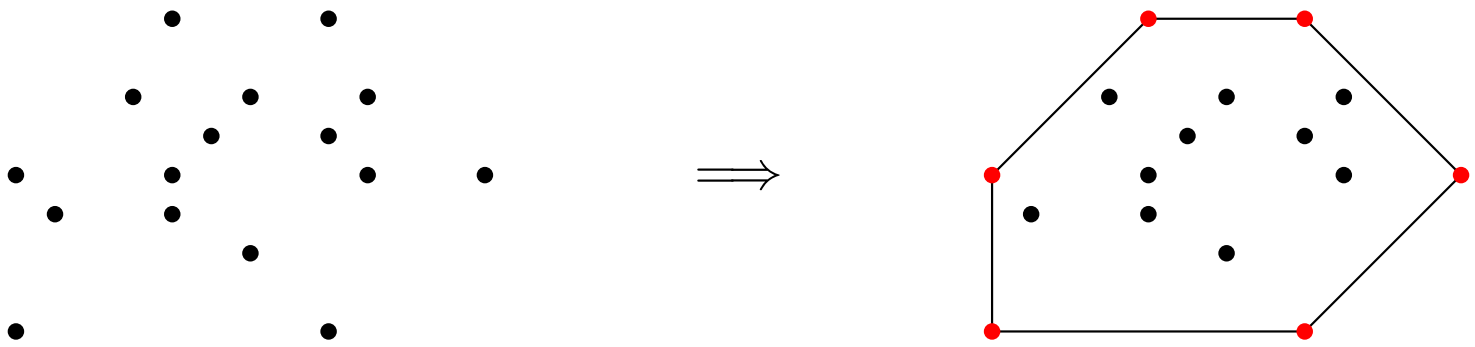
Subgradient Calculus (3): Finite pointwise maximum

Def: Convex hull

Let X be **any set** $X \subseteq \mathbb{R}^n$

- $\text{conv}(X)$: smallest **convex set** that contains X
- In finite dim., $\text{conv}(X)$: set of all convex combinations of finite number of points of X

$$conv(X) = \{x \in X \mid x = \sum_{i=1}^{m} \alpha_i x_i, \sum_{i=1}^{m} \alpha_i = 1, \alpha_i \geq 0\}$$



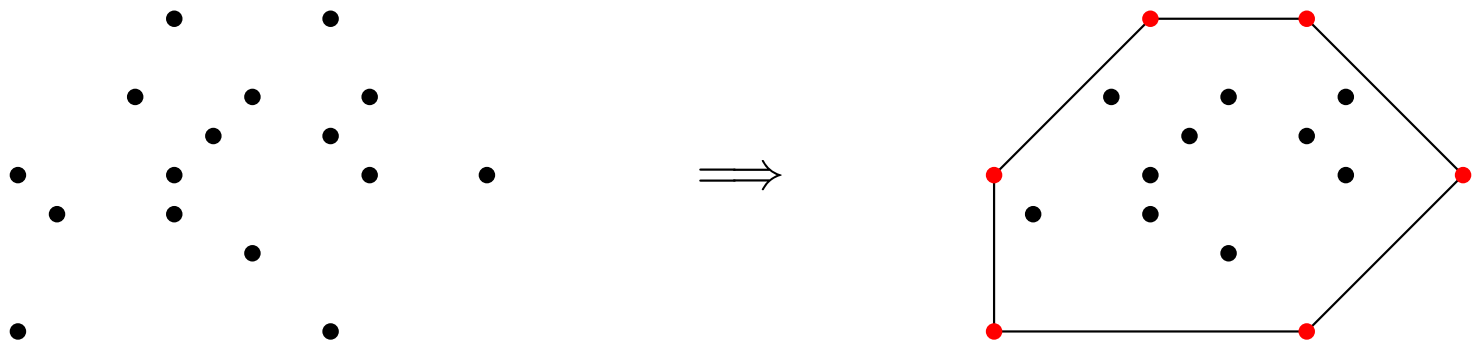
Subgradient Calculus (3): Finite pointwise maximum

Def: Convex hull

Let X be **any set** $X \subseteq \mathbb{R}^n$

- $\text{conv}(X)$: smallest **convex set** that contains X
- In finite dim., $\text{conv}(X)$: set of all convex combinations of finite number of points of X

$$conv(X) = \{x \in X \mid x = \sum_{i=1}^{m} \alpha_i x_i, \sum_{i=1}^{m} \alpha_i = 1, \alpha_i \geq 0\}$$

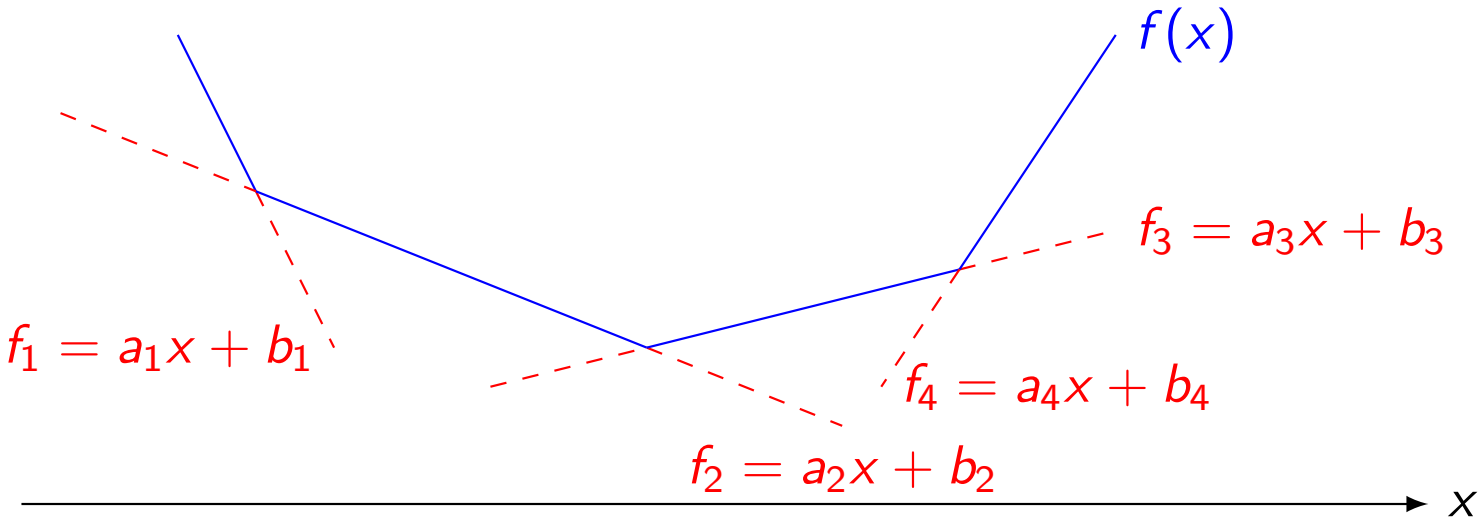


Algorithm: Jarvis march: $O(nh)$ time complexity (2D), ($n = |X|$, and $h = |\text{conv}(X)|$), Graham scan: $O(nlog(n))$, combination Chan's algorithm: $O(nlog(h))$

Example: Piecewise linear function

Intuition: application of finite pointwise maximum

$$\begin{aligned} f(x) &= \max\{a_1 x + b_1, a_2 x + b_2, a_3 x + b_3, a_4 x + b_4\} \\ &= \max_{i=1,\dots,4} f_i(x) \text{ with } f_i(x) = a_i x + b_i \end{aligned} \tag{10}$$



Since f_i differentiable: subdifferential at x is the **polyhedron** defined by

$$\begin{aligned} \partial f(x) &= \operatorname{conv} \bigcup_{i \in I(x)} \partial f_i(x) \text{ with } I(x) = \{i \mid a_i^\top x + b_i = f(x)\} \\ &= \operatorname{conv}\{\nabla f_i(x) \mid i \in I(x)\} = \operatorname{conv}\{a_i \mid i \in I(x)\} \end{aligned}$$

Example: ℓ_2-norm

ℓ_2-**norm** $f : \mathbb{R}^n \to \mathbb{R} : x \to \|x\|_2$

- For $x \neq 0$, f differentiable and subgradient (also gradient) given by $g = \dfrac{x}{\|x\|_2}$
- For $x = 0$
  - For g to be a subgradient of f we must have

$$f(y) = \|y\|_2 \geq f(x) + g^\mathsf{T}(y - x) = g^\mathsf{T} y, \ \forall y$$

  - For $\|y\|_2 \geq g^\mathsf{T} y$ to hold, the only condition for g is $\|g\|_2 \leq 1$

$\Rightarrow$ The subgradient g is any vector whose ℓ_2-norm is at most 1

Example: ℓ_2-norm

ℓ_2-**norm** $f : \mathbb{R}^n \to \mathbb{R} : x \to \|x\|_2$

- For $x \neq 0$, f differentiable and subgradient (also gradient) given by $g = \dfrac{x}{\|x\|_2}$
- For $x = 0$
  - For g to be a subgradient of f we must have

$$f(y) = \|y\|_2 \geq f(x) + g^\mathsf{T}(y - x) = g^\mathsf{T} y, \ \forall y$$

  - For $\|y\|_2 \geq g^\mathsf{T} y$ to hold, the only condition for g is $\|g\|_2 \leq 1$

$\Rightarrow$ The subgradient g is any vector whose ℓ_2-norm is at most 1

- Thus, the subdifferential is

$$\partial f(x) = \begin{cases} \left\{ \dfrac{x}{\|x\|_2} \right\} & x \neq 0 \\ \{g \mid \|g\|_2 \leq 1\} & x = 0 \end{cases}$$

Example: ℓ_1-norm

ℓ_1-**norm** $f : \mathbb{R}^n \to \mathbb{R} : x \to \|x\|_1 = \sum_{i=1}^{n} |x_i|$, non-differentiable convex function

- **Weak result**: f expressed as max. of 2^n linear functions

$$\|x\|_1 = \max\{s^\top x \mid s_i \in \{-1, 1\}, i = 1, \ldots, n\} = \max_{s \in \{-1,1\}^n} s^\top x$$

  ◦) Identify active function $s^\top x$: find $s \in \{-1, +1\}^n$ s.t. $s^\top x = \|x\|_1$
  - If $x_i \neq 0$: choose $s_i = 1$ if $x_i > 0$ and $s_i = -1$ if $x_i < 0$
  - If $x_i = 0$: more than one function is active, both $s_i = 1, s_i = -1$ acceptable

  ◦) Thus, subgradient g_i

$$g_i = \begin{cases} +1 & x_i > 0 \\ -1 \text{ or } +1 & x_i = 0 \\ -1 & x_i < 0 \end{cases}$$

Example: ℓ_1-norm

ℓ_1-**norm** $f : \mathbb{R}^n \to \mathbb{R} : x \to \|x\|_1 = \sum_{i=1}^{n} |x_i|$, non-differentiable convex function

- **Weak result**: f expressed as max. of 2^n linear functions

$$\|x\|_1 = \max\{s^\top x \mid s_i \in \{-1, 1\}, i = 1, \ldots, n\} = \max_{s \in \{-1,1\}^n} s^\top x$$

  ◦) Identify active function $s^\top x$: find $s \in \{-1, +1\}^n$ s.t. $s^\top x = \|x\|_1$
  - If $x_i \neq 0$: choose $s_i = 1$ if $x_i > 0$ and $s_i = -1$ if $x_i < 0$
  - If $x_i = 0$: more than one function is active, both $s_i = 1, s_i = -1$ acceptable

  ◦) Thus, subgradient g_i

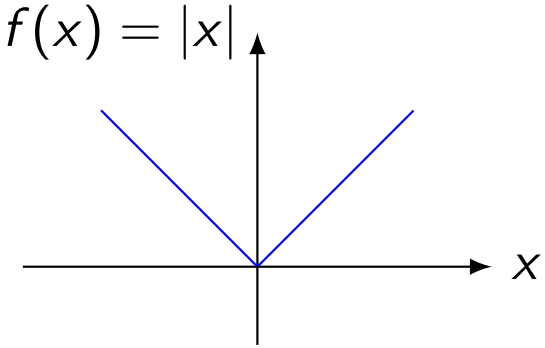
$$g_i = \begin{cases} +1 & x_i > 0 \\ -1 \text{ or } +1 & x_i = 0 \\ -1 & x_i < 0 \end{cases}$$

- **Strong result**: convex hull of all subgradients generated as product of intervals

$$\partial f(x) = \{g \mid \|g\|_\infty \leq 1, g^\top x = \|x\|_1\}$$

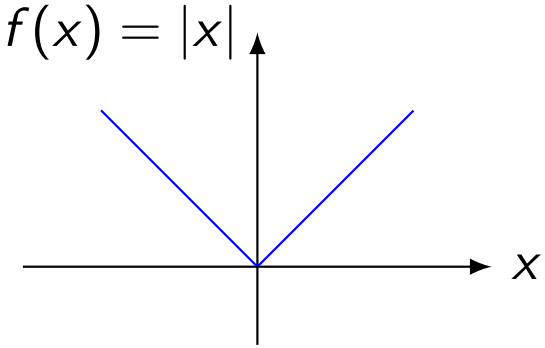
Example: ℓ_1-norm - Intuition

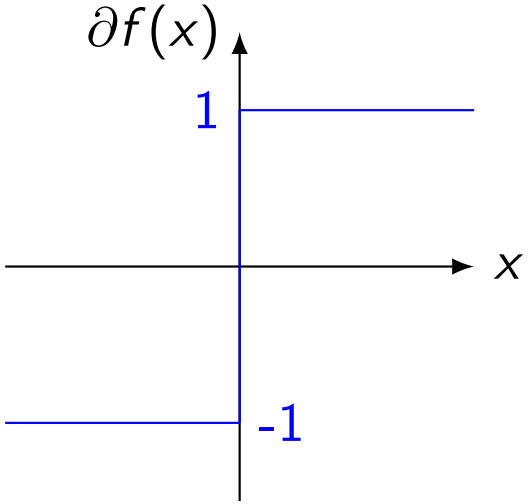
1-dim: $f := \mathbb{R} \to \mathbb{R} : x \to |x|$



Example: ℓ_1-norm - Intuition

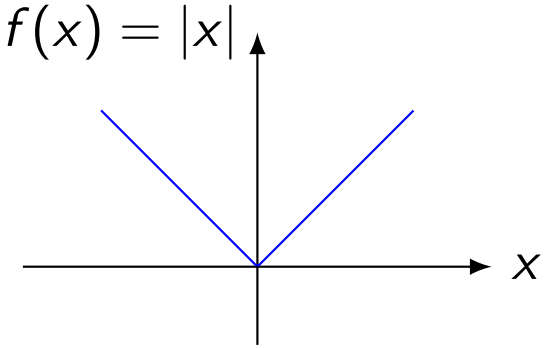
1-dim: $f := \mathbb{R} \to \mathbb{R} : x \to |x|$

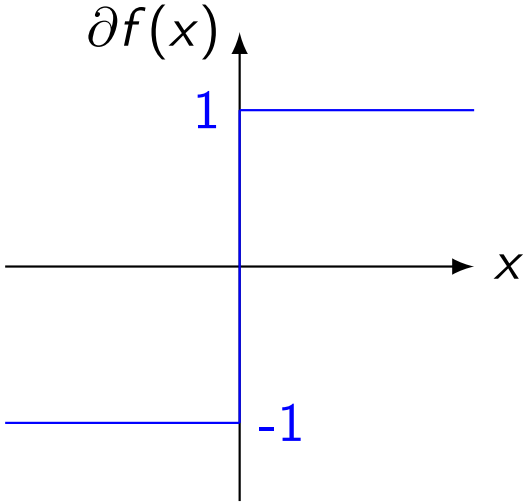




Example: ℓ_1-norm - Intuition

1-dim: $f := \mathbb{R} \to \mathbb{R} : x \to |x|$

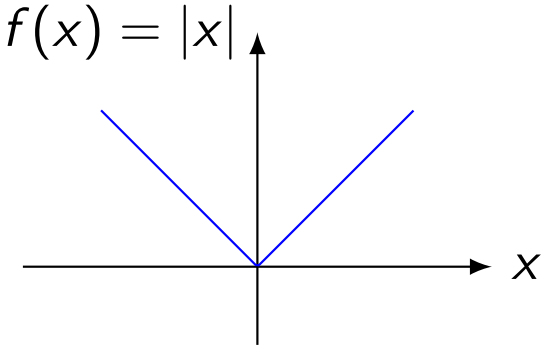


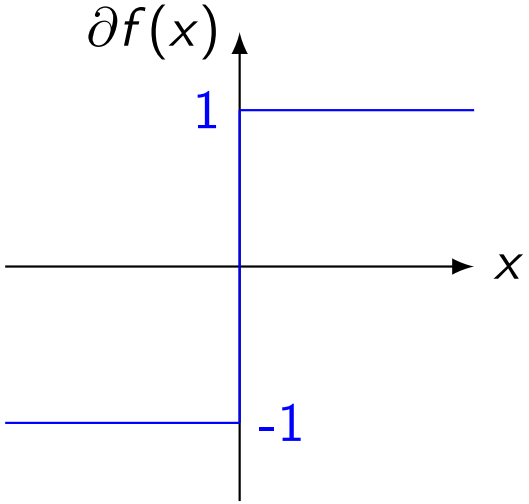


2-dim: $f := \mathbb{R}^n \to \mathbb{R} : x \to \|x\|_1 = \max_{s \in \{-1,1\}^n} s^\top x$

Example: ℓ_1-norm - Intuition

1-dim: $f := \mathbb{R} \to \mathbb{R} : x \to |x|$





2-dim: $f := \mathbb{R}^n \to \mathbb{R} : x \to \|x\|_1 = \max_{s \in \{-1,1\}^n} s^\top x$

Product of 2 intervals $\mathcal{I}_1 \times \mathcal{I}_2$, where $\mathcal{I}_i : \begin{cases} \{+1\} & x_i > 0 \\ [-1, +1] & x_i = 0 \\ \{-1\} & x_i < 0 \end{cases}$

Subgradient: Optimality conditions

1. **Unconstrained**
   - Recall: for a differentiable function f, **optimality condition** relates a minimizer of f with gradient at that point. For any f (convex or not),

$$f(x^\star) = \inf_x f(x) \Longleftrightarrow 0 = \nabla f(x^\star)$$

Subgradient: Optimality conditions

1. Unconstrained

- Recall: for a differentiable function f, **optimality condition** relates a minimizer of f with gradient at that point. For any f (convex or not),

$$f(x^\star) = \inf_x f(x) \Longleftrightarrow 0 = \nabla f(x^\star)$$

- Generalization to nondifferentiable **convex** function f: **subgradient optimality condition**

$$f(x^\star) = \inf_x f(x) \Longleftrightarrow 0 \in \partial f(x^\star) \tag{11}$$

  ◦) $x^\star$ minimizer of f **if and only if** 0 is a subgradient of f at $x^\star$

  ◦) By def. of subgradient: $g = 0$ implies that for all y

$$f(y) \geq f(x^\star) + 0^T(y - x^\star)\ \forall y \Longleftrightarrow 0 \in \partial f(x^\star) \tag{12}$$

Subgradient: Optimality conditions

1. **Unconstrained**
   - Recall: for a differentiable function f, **optimality condition** relates a minimizer of f with gradient at that point. For any f (convex or not),

$$f(x^\star) = \inf_x f(x) \Longleftrightarrow 0 = \nabla f(x^\star)$$

   - Generalization to nondifferentiable **convex** function f: **subgradient optimality condition**

$$f(x^\star) = \inf_x f(x) \Longleftrightarrow 0 \in \partial f(x^\star) \tag{11}$$

   ◦) $x^\star$ minimizer of f **if and only if** 0 is a subgradient of f at $x^\star$

   ◦) By def. of subgradient: $g = 0$ implies that for all y

$$f(y) \geq f(x^\star) + 0^T(y - x^\star)\ \forall y \Longleftrightarrow 0 \in \partial f(x^\star) \tag{12}$$

2. **Constrained** – see Section 4.

Subgradient: Optimality conditions - Example 1

Piecewise linear function

$$f(x) = \max_{i=1,\ldots,m} (a_i^\top x + b_i)$$

- Set of active functions at x is $I(x) = \{i \mid a_i^\top x + b_i = f(x)\}$
- Subdifferential at x: polyhedron defined by $\partial f(x) = \text{conv}\{a_i \mid i \in I(x)\}$

Subgradient: Optimality conditions - Example 1

Minimization of **Piecewise linear function**

$$\min_{x \in \mathbb{R}^n} f(x) = \max_{i=1,\ldots,m} (a_i^\top x + b_i)$$

Subgradient: Optimality conditions - Example 1

Minimization of **Piecewise linear function**

$$\min_{x \in \mathbb{R}^n} f(x) = \max_{i=1,\ldots,m} (a_i^\top x + b_i)$$

To find a subgradient g of f: given x: find index j for which

$$a_j^\top x + b_j = \max_{i=1,\ldots,m} (a_i^\top x + b_i)$$

Then, take as subgradient $g = a_j$

- Set Lipschitz constant $L = \max_{i=1,\ldots,m} \|a_i\|_2$
- Since f satisfies the Lipschitz condition

$$|f(u) - f(v)| \leq L \|u - v\|_2 , \ \forall u, v$$

The norm of subgradients is bounded: for any $g \in \partial f(x)$, $\|g\|_2 \leq L$

Subgradient: Optimality conditions - Example 2

The **Soft-thresholding** problem (particular case of the LASSO problem)

$$\min_{\beta \in \mathbb{R}^n} \frac{1}{2} \|y - \beta\|_2^2 + \lambda \|\beta\|_1 \tag{13}$$

where $\lambda > 0$, can be solved by using subgradient optimality condition

$$\begin{aligned} 0 \in \partial\Big(\frac{1}{2} \|y - \beta\|_2^2 + \lambda \|\beta\|_1\Big) &\Longleftrightarrow 0 \in -(y - \beta) + \lambda \partial \|\beta\|_1 \\ &\Longleftrightarrow (y - \beta) = \lambda u, u \in \partial \|\beta\|_1 \end{aligned}$$

Subgradient: Optimality conditions - Example 2

From the ℓ_1-norm, we derive

$$u_i = \begin{cases} 1 & \beta_i > 0 \\ [-1, 1] & \beta_i = 0 \\ -1 & \beta_i < 0 \end{cases}$$

The subgradient optimality conditions can thus be written

$$\begin{cases} y_i - \beta_i = \lambda \, \text{sign}(\beta_i) & \beta_i \neq 0 \\ |y_i - \beta_i| \leq \lambda & \beta_i = 0 \end{cases}$$

The solution is $\beta_i = [T_\lambda(y)]_i$, where T_λ = soft-thresholding operator

$$T_\lambda(y) = [|y| - \lambda]_+ \, \text{sign}(y) = \begin{cases} y - \lambda & y > \lambda \\ 0 & -\lambda \leq y \leq \lambda \\ y + \lambda & y < -\lambda \end{cases} \tag{14}$$

Note: $[|y| - \lambda]_+ = \max(|y| - \lambda, 0)$

Subgradient: Optimality conditions - Example 2

Thus, check that optimality conditions satisfied when $\beta_i = [T_\lambda(y)]_i$:

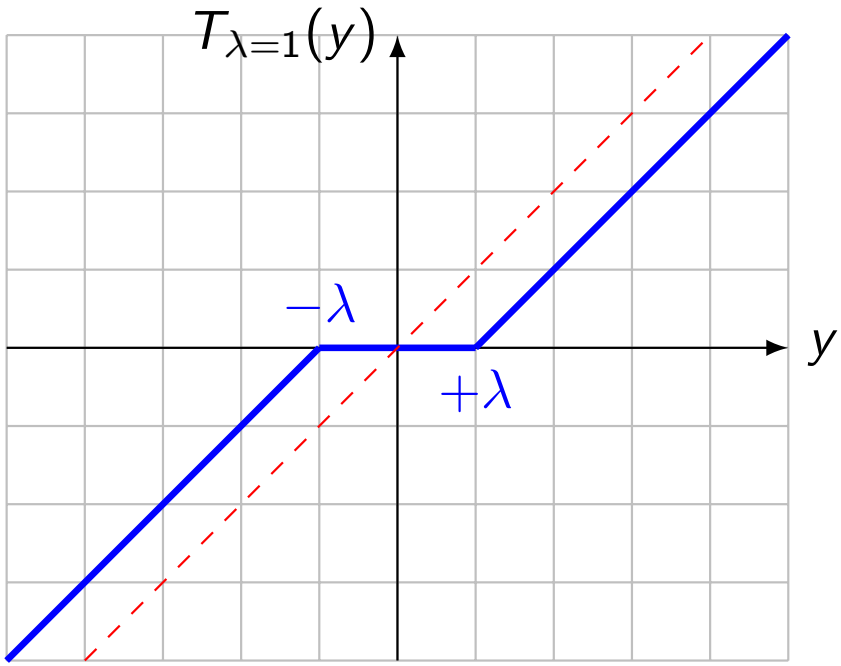
- When $y_i > \lambda$: $\beta_i = y_i - \lambda > 0$; thus $y_i - \beta_i = \lambda$
- When $|y_i| \leq \lambda$: $\beta_i = 0$ and $|y_i - \beta_i| = |y_i| \leq \lambda$
- When $y_i < -\lambda$: $\beta_i = y_i + \lambda > 0$; thus $y_i - \beta_i = -\lambda$

Subgradient: Optimality conditions - Example 2

Thus, check that optimality conditions satisfied when $\beta_i = [T_\lambda(y)]_i$:

- When $y_i > \lambda$: $\beta_i = y_i - \lambda > 0$; thus $y_i - \beta_i = \lambda$
- When $|y_i| \leq \lambda$: $\beta_i = 0$ and $|y_i - \beta_i| = |y_i| \leq \lambda$
- When $y_i < -\lambda$: $\beta_i = y_i + \lambda > 0$; thus $y_i - \beta_i = -\lambda$

Soft-threshold operator T_λ

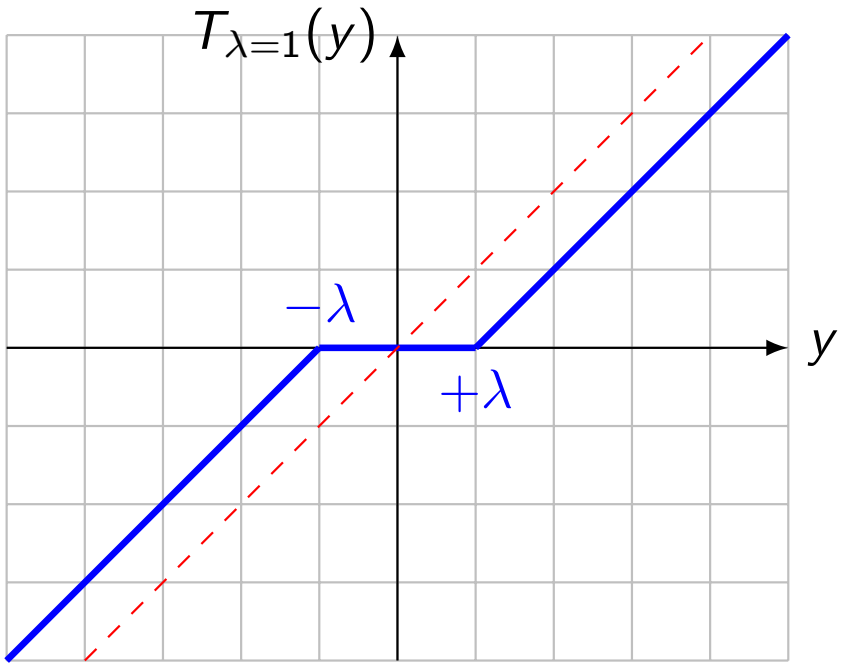


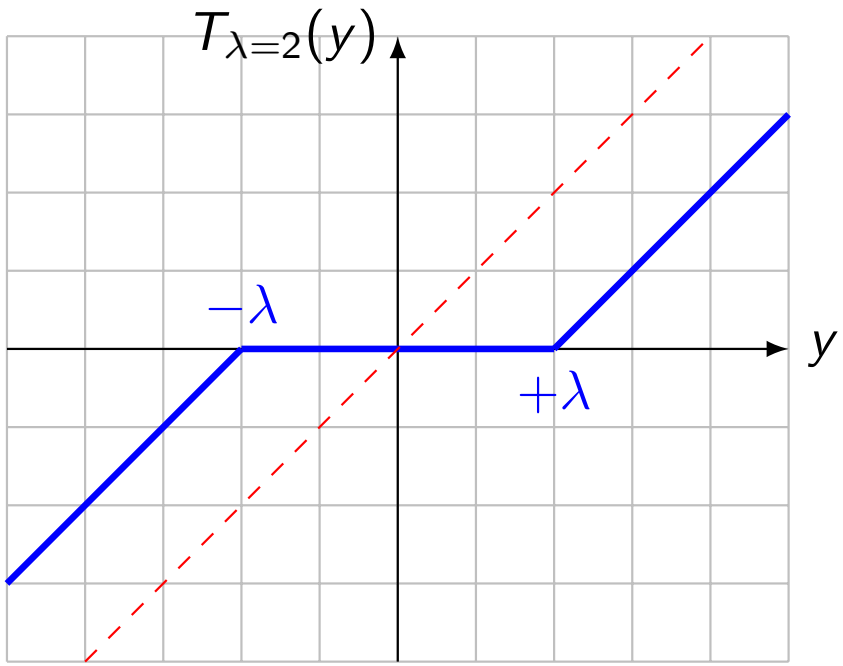
Subgradient: Optimality conditions - Example 2

Thus, check that optimality conditions satisfied when $\beta_i = [T_\lambda(y)]_i$:

- When $y_i > \lambda$: $\beta_i = y_i - \lambda > 0$; thus $y_i - \beta_i = \lambda$
- When $|y_i| \leq \lambda$: $\beta_i = 0$ and $|y_i - \beta_i| = |y_i| \leq \lambda$
- When $y_i < -\lambda$: $\beta_i = y_i + \lambda > 0$; thus $y_i - \beta_i = -\lambda$

Soft-threshold operator T_λ





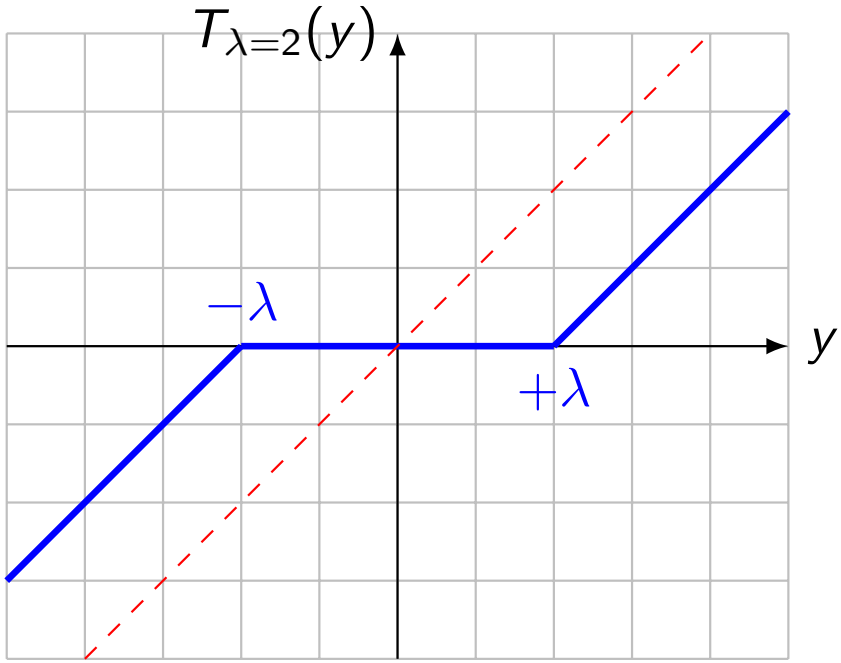
Subgradient: Optimality conditions - Example 2

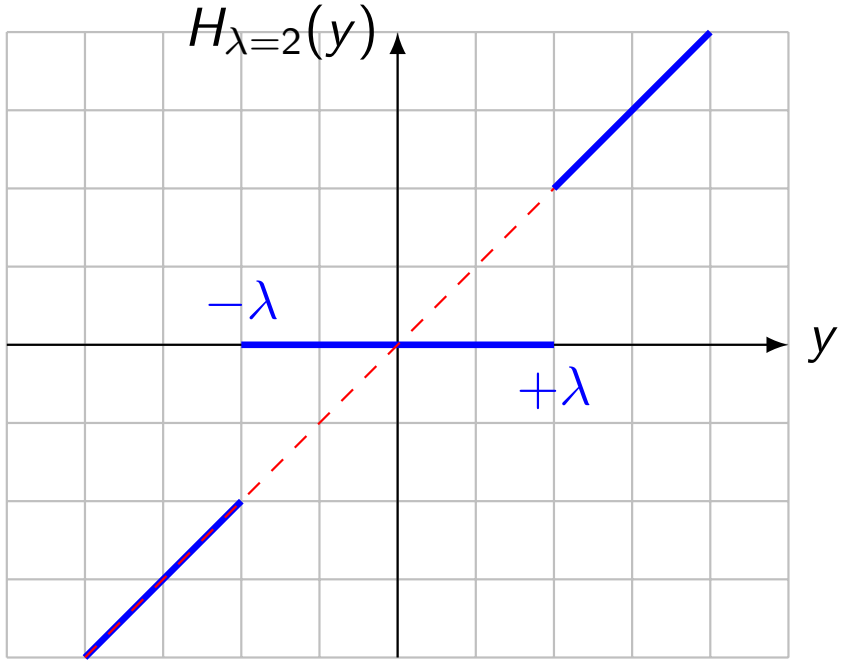
Soft-threshold operator T_λ

$$T_\lambda(y) = \begin{cases} y - \lambda & y > \lambda \\ 0 & -\lambda \le y \le \lambda \\ y + \lambda & y < -\lambda \end{cases}$$

Hard-threshold operator H_λ

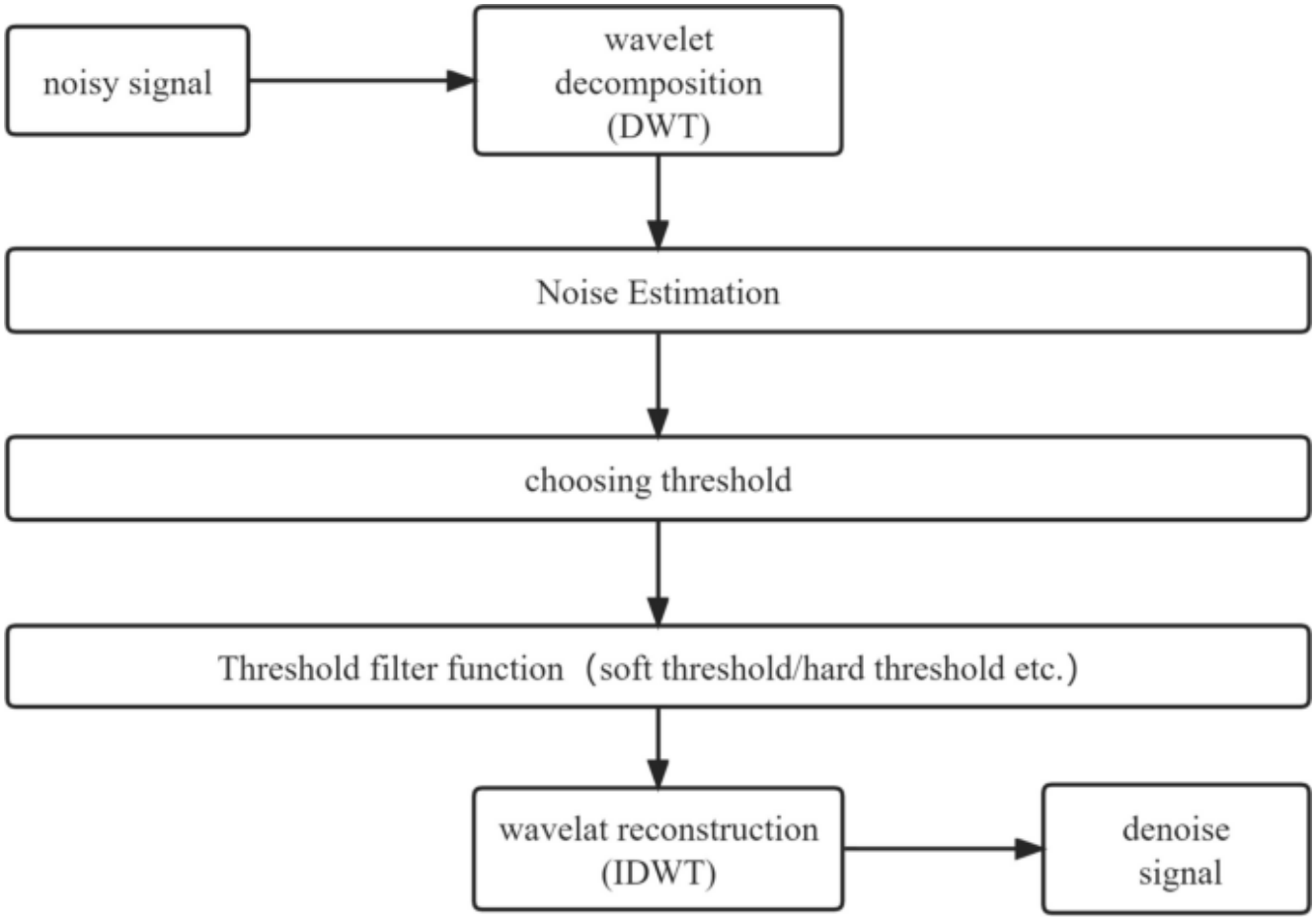
$$H_\lambda(y) = \begin{cases} y & |y| > \lambda \\ 0 & -\lambda \le y \le \lambda \end{cases}$$





Application: soft-thresholding

Method: **surface electromyography (sEMG)** wavelet denoising

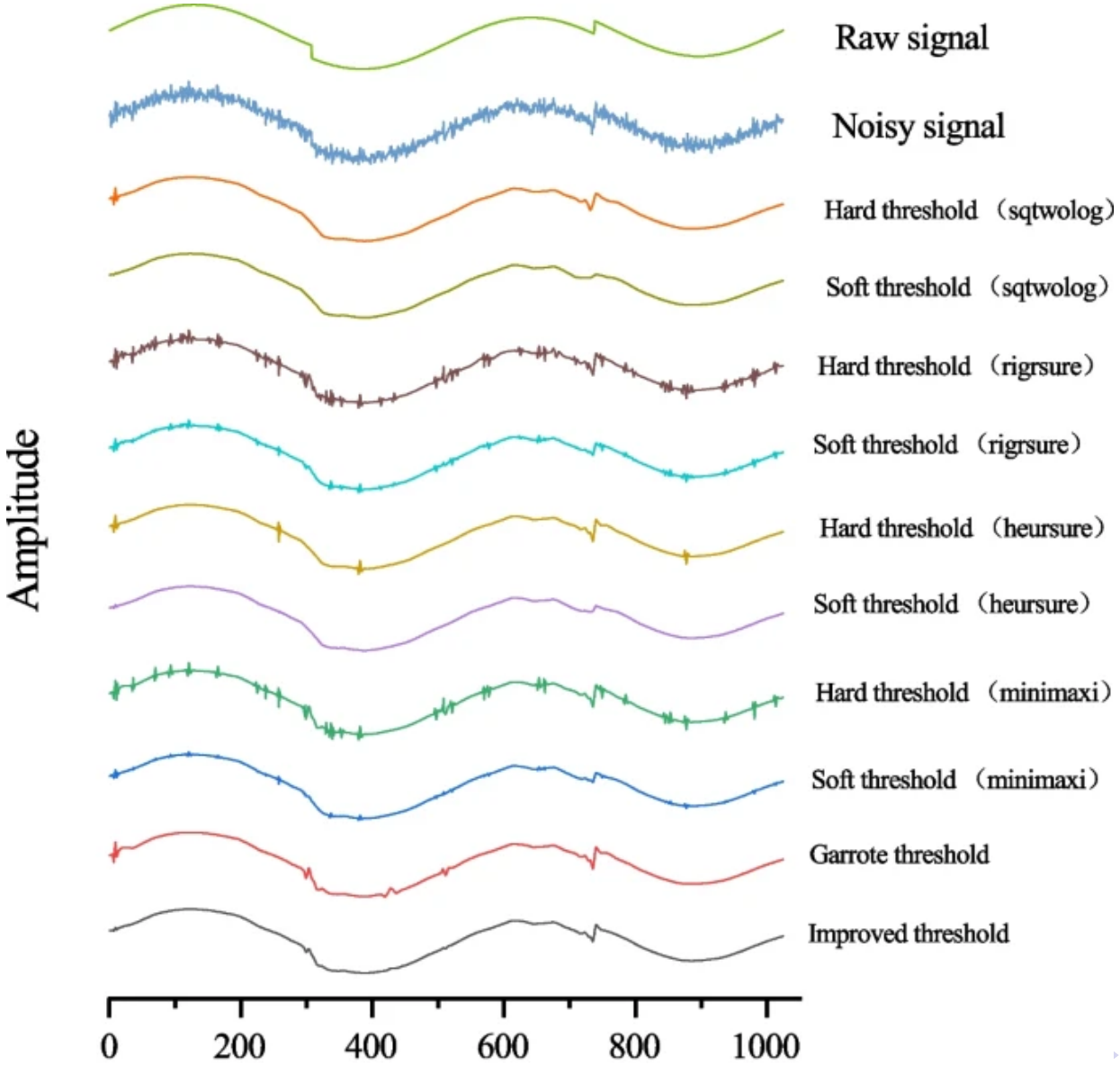


sEMG: weak physiological electrical signal that accompanies muscle contraction[2]

[2] Source: EURASIP Journal on Advances in Signal Processing volume 2023, Article number: 108 (2023)

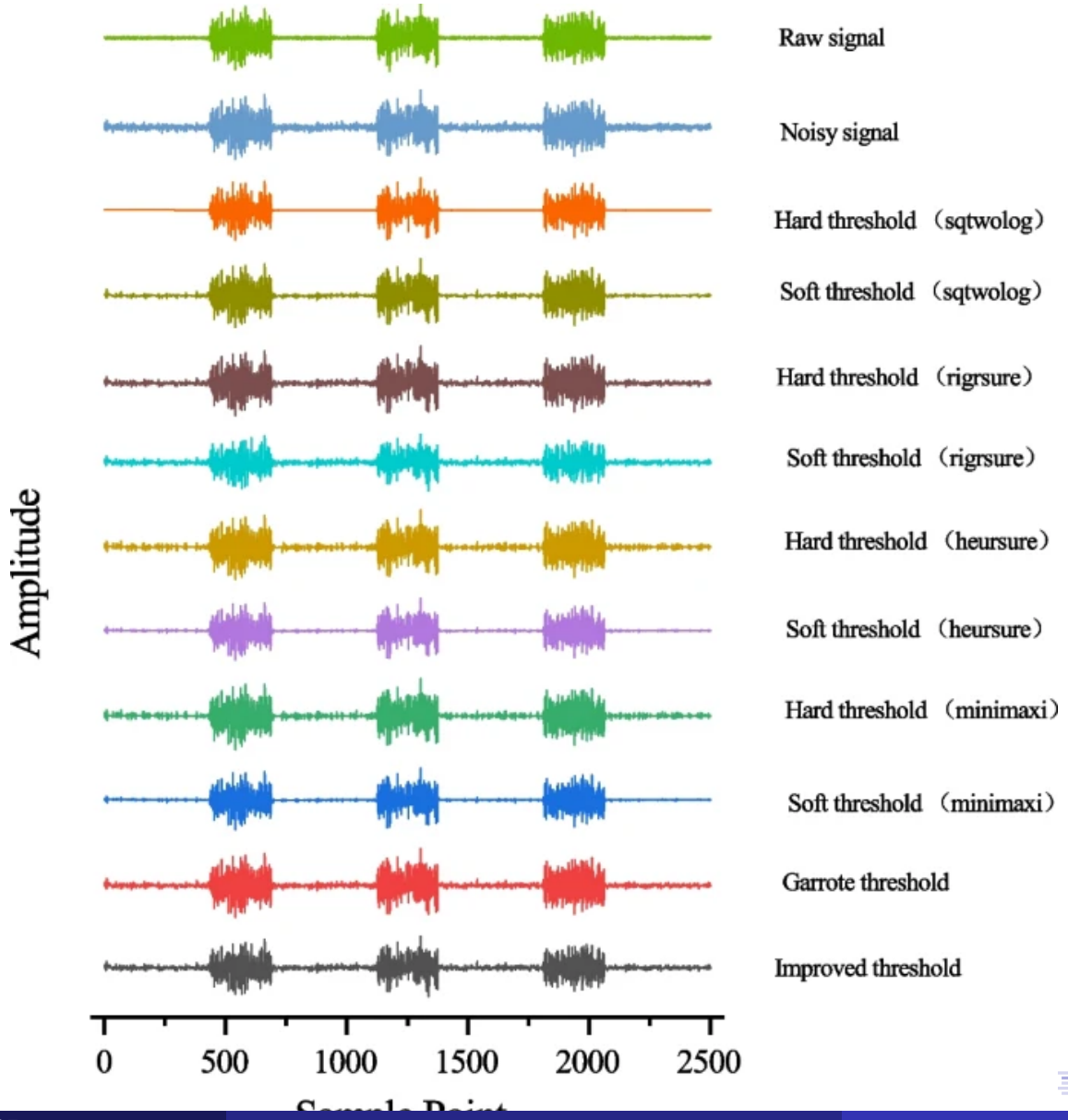
Application: soft-thresholding

Denoising effect



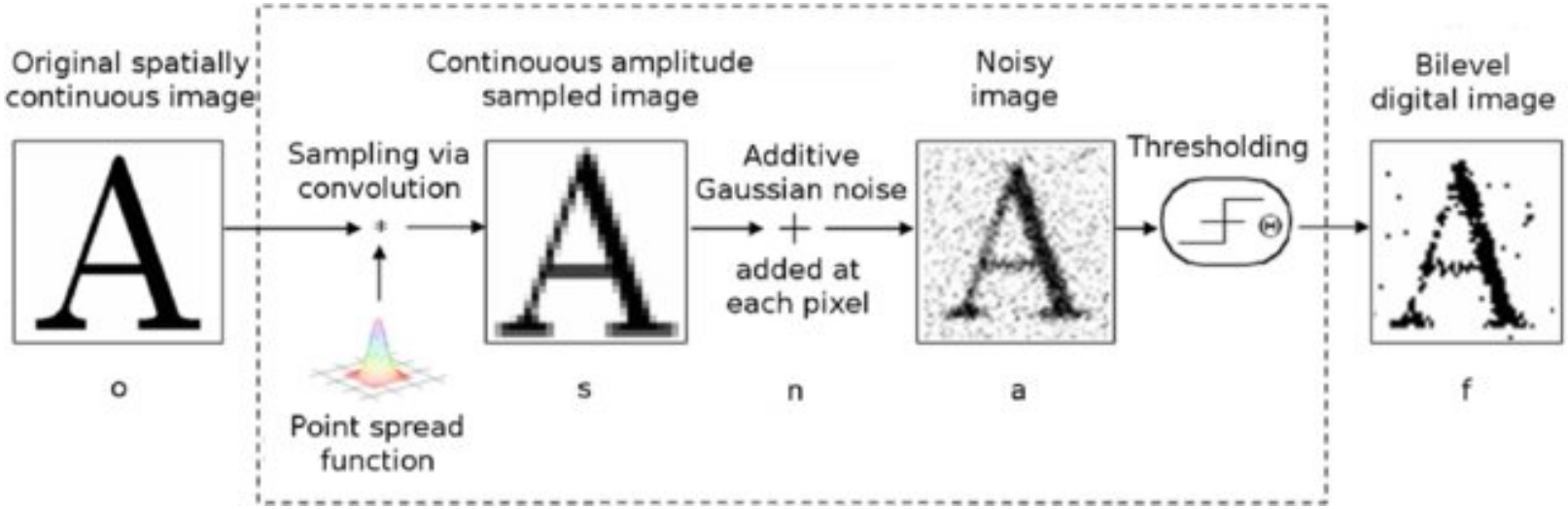
Application: soft-thresholding

Comparison between noisy signal and signal after improved threshold filtering



Application: soft-thresholding

Scanner model to determine the value of the pixel [i,j] centered on each sensor element[3]



Scanner model : multistage process whose steps include convolution with a point spread function (PSF), sampling, adding noise, and thresholding

[3] Source: Sparsity-based edge noise removal from bilevel graphical document images, International Journal on Document Analysis and Recognition (IJDAR), August 2014

Directional derivative and subgradients

Def: Directional derivative

Directional derivative of f at x in direction $d = \Delta x$

$$Df(x; \Delta x) := \lim_{t \to 0} \frac{f(x + t\Delta x) - f(x)}{t} \tag{15}$$

Directional derivative and subgradients

Def: Directional derivative

Directional derivative of f at x in direction $d = \Delta x$

$$Df(x; \Delta x) := \lim_{t \to 0} \frac{f(x + t\Delta x) - f(x)}{t} \tag{15}$$

Implications:

- If f convex and finite near $x \in \mathit{int}\,\mathrm{dom}(f)$, then $Df(x; \Delta x)$ exists for all Δx
- f is differentiable at x **if and only if** for some $g = \nabla f(x)$ and all Δx

$$Df(x; \Delta x) = g^T \Delta x = \nabla f(x)^T \Delta x$$

$\rightarrow$ $Df(x; \Delta x)$ is a linear function of Δx

Directional derivative and subgradients

Def: Directional derivative

Directional derivative of f at x in direction $d = \Delta x$

$$Df(x; \Delta x) := \lim_{t \to 0} \frac{f(x + t\Delta x) - f(x)}{t} \tag{15}$$

Implications:

- If f convex and finite near $x \in \mathit{int}\,\mathrm{dom}(f)$, then $Df(x; \Delta x)$ exists for all Δx
- f is differentiable at x **if and only if** for some $g = \nabla f(x)$ and all Δx

$$Df(x; \Delta x) = g^T \Delta x = \nabla f(x)^T \Delta x$$

$\rightarrow$ $Df(x; \Delta x)$ is a linear function of Δx

Generalization

For f convex and $x \in \mathit{int}\,\mathrm{dom}(f)$, $Df(x; \Delta x) = \sup_{g \in \partial f(x)} g^T \Delta x$

Subgradient direction

Δx is a descent direction for f at x if $Df(x; \Delta x) < 0$

- **For differentiable (convex) function** f: $\Delta x = -\nabla f(x)$ is always a descent direction except when it is zero

  $\Rightarrow$ Negative gradient of a differentiable function f is always a descent direction (if $\nabla f \neq 0$)

Subgradient direction

Δx is a descent direction for f at x if $Df(x; \Delta x) < 0$

- **For differentiable (convex) function** f: $\Delta x = -\nabla f(x)$ is always a descent direction except when it is zero

  $\Rightarrow$ Negative gradient of a differentiable function f is always a descent direction (if $\nabla f \neq 0$)

- **For nondifferentiable (convex) function** f: $\Delta x = -g$, with $g \in \partial f(x)$, need not be descent direction

  $\Rightarrow$ Negative subgradient is not always a descent direction

Subgradient direction: exemple

Example: $f(x_1, x_2) = |x_1| + 2\,|x_2|$

- Subgradient $g = (1, 2) \in \partial f(1, 0)$

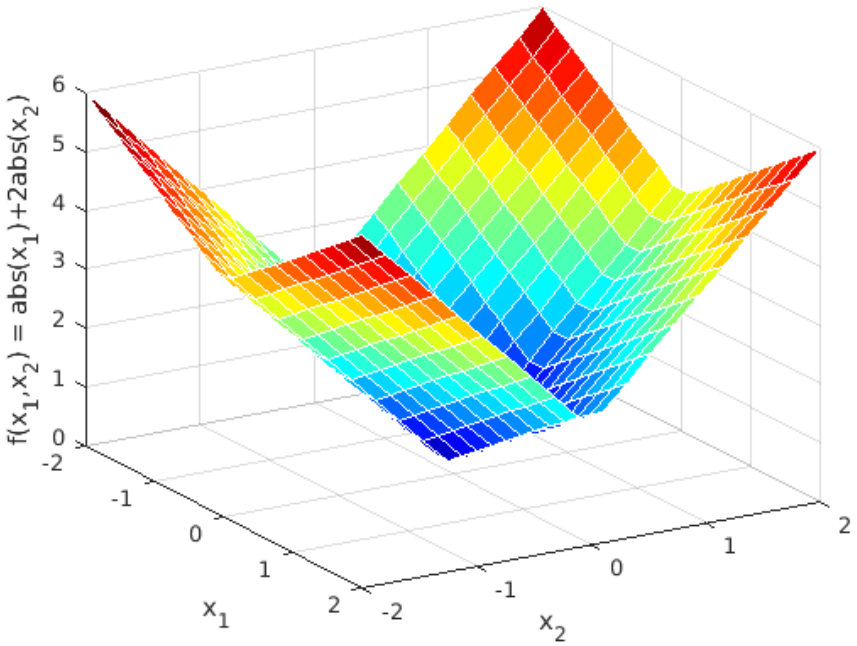
$$\begin{aligned} f(x) + g^T(y - x) &= 1 + 1(y_1 - 1) + 2(y_2 - 0) \\ &= y_1 + 2y_2 \leq f(y) \end{aligned}$$

Subgradient direction: exemple

Example: $f(x_1, x_2) = |x_1| + 2\,|x_2|$

- Subgradient $g = (1, 2) \in \partial f(1, 0)$

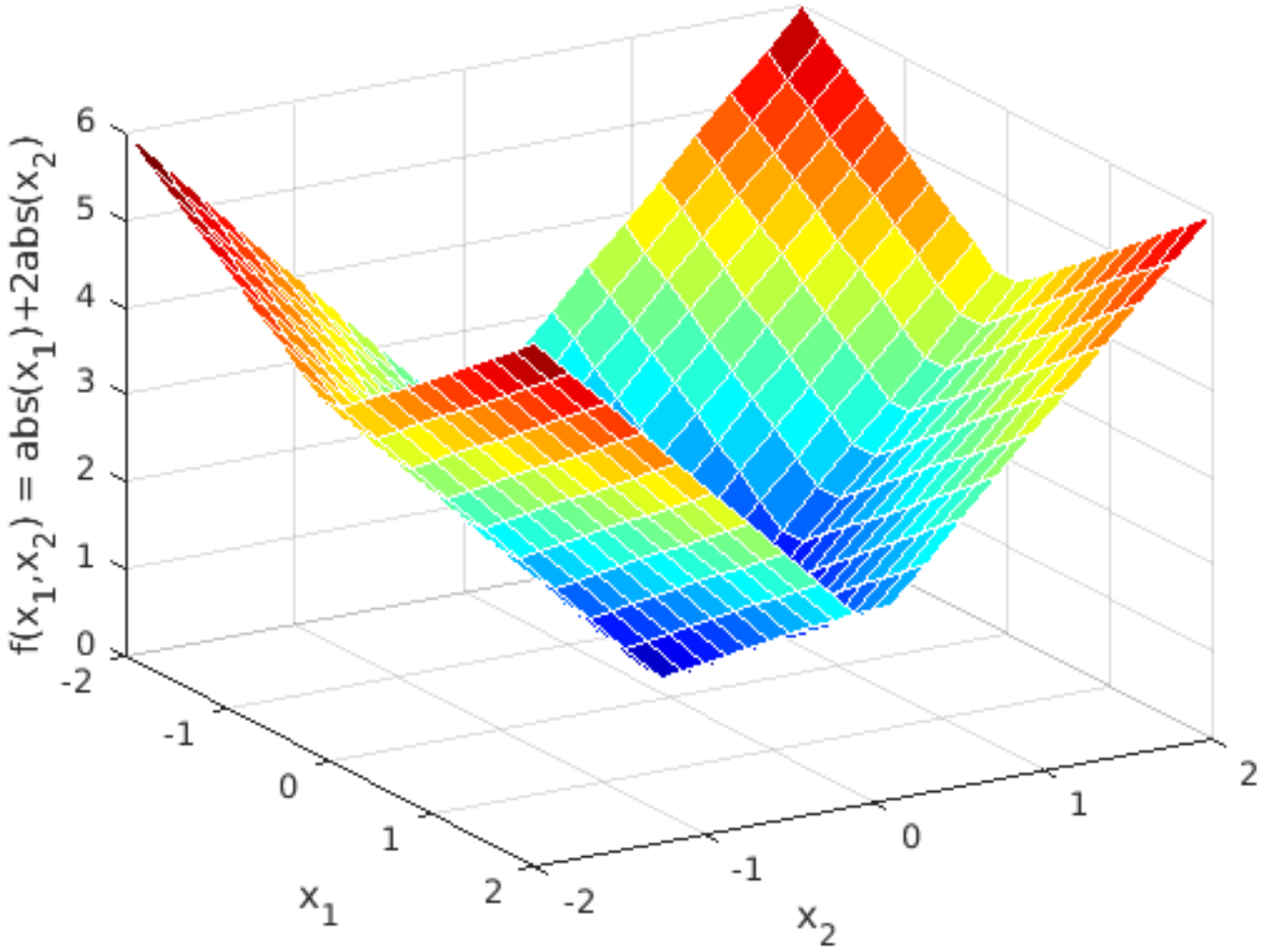
$$\begin{aligned} f(x) + g^T(y - x) &= 1 + 1(y_1 - 1) + 2(y_2 - 0) \\ &= y_1 + 2y_2 \leq f(y) \end{aligned}$$



Subgradient direction: exemple

Example: $f(x_1, x_2) = |x_1| + 2\,|x_2|$

- However $\Delta x = -g = (-1, -2)$ is not a descent direction at $(x_1, x_2) = (1, 0)$



Descent direction: Theorem

Theorem

Assume function f is convex and finite near $x \in \mathbb{R}^n$.

Then, either

i) $0 \in \partial f(x)$ in which case x minimizes f

ii) or there is a descent direction for f at x

Thus, x is optimal (it minimizes f) **if and only if** no descent direction for f at x

Descent direction: Theorem

Theorem

Assume function f is convex and finite near $x \in \mathbb{R}^n$.

Then, either

i) $0 \in \partial f(x)$ in which case x minimizes f

ii) or there is a descent direction for f at x

Thus, x is optimal (it minimizes f) **if and only if** no descent direction for f at x

Proof: Define

$$\Delta x = -\operatorname*{argmin}_{z \in \partial f(x)} \|z\|^2$$

- If $\Delta x = 0$, then $0 \in \partial f(x)$, so the point x is optimal
- Otherwise $Df(x; \Delta x) = -\operatorname*{argmin}_{z \in \partial f(x)} \|z\|^2$, so Δx is a descent direction

Subgradient method: algorithm

Subgradient algorithm: `input`: choose initial point $x_0 \in dom(f)$

`init`: select step size/length computation strategy

`either` Constant step size: $t_k = t$ positive constant, independent of k,

`or` Constant step length $t_k = \dfrac{\gamma}{\|g_k\|_2}$ with $\gamma = \|x_{k+1} - x_k\|_2 > 0$

`or` Diminishing step size ($\lim_{k\to\infty} t_k = 0$), e.g., $t_k = \dfrac{\tau}{\sqrt{k}}, \tau > 0$

`or` Diminishing step length ($\lim_{k\to\infty} t_k = 0$), e.g., $t_k = \dfrac{\gamma_k}{\|g_k\|_2}$, $\gamma_k > 0$

`repeat`:

1. Determine a **step**/search direction (vector starting at x_k) $\Delta x_k := -g_k$, i.e., subgradient of f at x_k ($g_k \in \partial f(x_k)$)
2. Compute step size/legnth t_k following selected strategy
3. Update: $x_{k+1} = x_k + t_k \Delta x_k = x_k - t_k g_k$

`until`: stopping criterion is satisfied

Subgradient method is not a Descent method

- The (basic) subgradient method: rather simple algorithm to minimize nondifferentiable convex function f

Subgradient method is not a Descent method

- The (basic) subgradient method: rather simple algorithm to minimize nondifferentiable convex function f
- However, **not a descent method**

Subgradient method is not a Descent method

- The (basic) subgradient method: rather simple algorithm to minimize nondifferentiable convex function f
- However, **not a descent method**

...Why ?

Subgradient method is not a Descent method

- The (basic) subgradient method: rather simple algorithm to minimize nondifferentiable convex function f
- However, **not a descent method**

...Why ?

- It can happen that $-g_k$ is not a descent direction for f at x_k, i.e., $Df(x_k; -g_k) > 0$. In such cases, $f(x_{k+1}) > f(x_k)$
- Even when $-g_k$ is a descent direction at x_k, step size/length can be such $f(x_{k+1}) > f(x_k)$

$\Rightarrow$ Iteration of subgradient algorithm can increase value of objective function

Subgradient method is not a Descent method

To mitigate this behaviour: several methods

Most common

- Keep track of best point found so far, i.e., point with smallest objective function value
- At each iteration, set
$$\tilde{f}_k = \min\{\tilde{f}_{k-1}, f(x_k)\}$$
- Then, best value of objective function found in k iterations
$$\tilde{f}_k = \min\{f(x_1), f(x_2), \ldots, f(x_k)\}$$
- <u>Note</u>: since $\tilde{f}_k$ is decreasing, the limit can be $-\infty$

Subgradient method: convergence properties

In subgradient method: **key quantity** is the Euclidean distance to the optimal set (not the function value which often increases)

Subgradient method: convergence properties

In subgradient method: **key quantity** is the Euclidean distance to the optimal set (not the function value which often increases)

Assume that

- A minimizer $x^\star$ in optimal set $X^\star$ of f exists and satisfies

$$\|x_0 - x^\star\|_2 \leq r \text{ (known)}$$

Thus, r : upper bound on $dist(x_0, X^\star)$ of initial point x_0 to optimal set $X^\star$

Subgradient method: convergence properties

In subgradient method: **key quantity** is the Euclidean distance to the optimal set (not the function value which often increases)

Assume that

- A minimizer $x^\star$ in optimal set $X^\star$ of f exists and satisfies

$$\|x_0 - x^\star\|_2 \leq r \text{ (known)}$$

  Thus, r : upper bound on $dist(x_0, X^\star)$ of initial point x_0 to optimal set $X^\star$

- Norm of subgradients is bounded

$$\exists\, L > 0 \text{ such that } \|g_k\|_2 \leq L, \forall k$$

  Assumption satisfied when f **Lipschitz continuous**

$$|f(u) - f(v)| \leq L\,\|u - v\|_2\,,\ \forall\, u, v \in \mathrm{dom}(f)$$

Subgradient method: convergence properties

With these assumptions

- For **constant step size** and **length** rules: subgradient algorithm guaranteed to converge to $\varepsilon-$suboptimal value

$$\lim_{k\to\infty} \tilde{f}_k - f^\star = \overline{f} - f^\star < \varepsilon$$

where $f^\star$ denotes the optimal value, i.e., $f^\star = \inf_x f(x)$. The number ε is a function of the step parameterization and decreases with it

○) For constant step size : $\overline{f} - f^\star \leq L^2 t/2$

○) For constant step length : $\overline{f} - f^\star \leq L\gamma/2$

$\Rightarrow$ Subgradient method finds $\varepsilon-$suboptimal point in finite number of steps

- For **diminishing step size** and **length** rules: subgradient algorithm guaranteed to converge to optimal value

$$\lim_{k\to\infty} \tilde{f}_k = f^\star, f^\star = \inf_x f(x)$$

Subgradient method: convergence properties

- The (lower bound on) convergence rate can be estimated by finding sequence of step sizes/lengths $\{t_1, t_2, \ldots, t_k\}$ that minimizes suboptimality bound:

$$t_i = \frac{r}{L\sqrt{k}}, \; i = 1, \ldots, k$$

- This choice of step size yields suboptimality bound $\tilde{f}_k - f^\star \leq \dfrac{rL}{\sqrt{k}}$

  That is, for any choice of step sizes, suboptimality bound must be at least as large as $\tilde{f}_k - f^\star \leq \dfrac{rL}{\sqrt{k}}$

- **If** this suboptimality bound used as stopping criterion

  Then number of steps to achieve a guaranteed accuracy of ε at least $(rL/\varepsilon)^2$ (no matter step sizes used)

- Thus, subgradient method very slow: reduction by factor 10^3 requires **at least** 10^6 iterations

Subgradient method: acceleration

Subgradient method

- in general, not a descent method
- main consequence: rather slowly converging

Subgradient method: acceleration

Subgradient method

- in general, not a descent method
- main consequence: rather slowly converging

Techniques to accelerate convergence of subgradient methods

- Convergence can be improved by keeping memory of past steps: **heavy ball** method

$$x_{k+1} = x_k - t_k g_k + \beta_k (x_k - x_{k-1})$$

  where t_k and β_k are positive
  - Such algorithms are **statefull** (basic subgradient method is stateless): second term referred to as **memory term**
  - Often use known ($f^\star = 0$) or estimated value $\hat{f}^\star$ of $f^\star = \inf_x f(x)$ to determine step sizes, e.g., Polyak step size
  - Thus, find applicability when $f^\star$ **can be estimated**
- Other ideas: localization methods, conjugate directions, etc.

Subgradient method: acceleration

- **Polyak step sizes**: when optimal value $f^\star$ known, take

$$t_k = \frac{f(x_{k-1}) - f^\star}{\|g_{k-1}\|_2^2}, \;\; k = 1, 2, 3, \ldots$$

- **Motivation**: from first step of subgradient proof

$$\|x_k - x^\star\|_2^2 \leq \|x_{k-1} - x^\star\|_2^2 - 2t_k\big(f(x_{k-1}) - f(x^\star)\big) + t_k^2 \|g_{k-1}\|_2^2 \qquad (16)$$

Polyak step size minimizes the RHS of (16)

Subgradient method: acceleration

- **Polyak step sizes**: when optimal value $f^\star$ known, take

$$t_k = \frac{f(x_{k-1}) - f^\star}{\|g_{k-1}\|_2^2}, \; k = 1, 2, 3, \ldots$$

- **Motivation**: from first step of subgradient proof

$$\|x_k - x^\star\|_2^2 \leq \|x_{k-1} - x^\star\|_2^2 - 2t_k\big(f(x_{k-1}) - f(x^\star)\big) + t_k^2 \|g_{k-1}\|_2^2 \qquad (16)$$

  Polyak step size minimizes the RHS of (16)

- With Polyak step sizes, subgradient method converges ($f_k \to f^\star$) at rate $O(\varepsilon^{-2})$

Subgradient method: acceleration

- **Polyak step sizes**: when optimal value $f^\star$ known, take

$$t_k = \frac{f(x_{k-1}) - f^\star}{\|g_{k-1}\|_2^2}, \quad k = 1, 2, 3, \ldots$$

- **Motivation**: from first step of subgradient proof

$$\|x_k - x^\star\|_2^2 \leq \|x_{k-1} - x^\star\|_2^2 - 2t_k\big(f(x_{k-1}) - f(x^\star)\big) + t_k^2 \|g_{k-1}\|_2^2 \tag{16}$$

  Polyak step size minimizes the RHS of (16)

- With Polyak step sizes, subgradient method converges ($f_k \to f^\star$) at rate $O(\varepsilon^{-2})$

Slow convergence rate

- Convergence rate $O(\varepsilon^{-2})$ over general convex, Lipschitz functions is **slow**
- Subgradient method cannot do better than $O(\varepsilon^{-2})$ rate (unless beyond non-smooth first-order methods)

INFO-F524: Continuous Optimization
– Section 2 : Part 3 –

Bernard Fortz, Dimitri Papadimitriou

Bureau: **2N3 206**

Université Libre de Bruxelles

dimitrios.papadimitriou@ulb.be

Annee Académique: 2024-2025

References

- R. Tyrrell Rockafellar and Roger J-B Wets, *Varational Analysis*, 1997, 2nd printing 2004, 3rd printing 2009, https://sites.math.washington.edu/~rtr/papers/rtr169-VarAnalysis-RockWets.pdf
- Boris S. Mordukhovich and Nguyen Mau Nam, *Convex Analysis and Beyond*, Volume I: Basic Theory, Springer International Publishing, 2022.
- A. Beck, *First-Order Methods in Optimization*, MOS-SIAM Series on Optimization, 2017.
- H. H. Bauschke and P. L. Combettes: *Convex Analysis and Monotone Operator Theory in Hilbert Spaces* (2nd Edition). Springer, New York, 2017.
- Jean-Pierre Aubin and Ivar Ekeland, *Applied Nonlinear Analysis*, Dover Books on Mathematics, 1984.

Definition

Let

- $\mathcal{X}$ be a vector space with norm $\|\cdot\|_{\mathcal{X}}$
- $f : \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$ is a proper closed convex function [1]

Proximal operator/mapping

The proximal operator $\text{prox}_{\lambda f} : \mathcal{X} \to \mathbb{R}$ associated to the scaled function λf, where $\lambda > 0$, a.k.a., the proximal operator of f with parameter λ, is defined as

$$\text{prox}_{\lambda f}(x) = \underset{u \in \mathcal{X}}{\text{argmin}} \left(f(u) + \frac{1}{2\lambda} \|u - x\|_{\mathcal{X}}^2 \right) \tag{1}$$

[1] Def: a proper convex function is an extended real-valued convex function with a non-empty effective domain that never takes on the value $-\infty$

Definition

Let

- $\mathcal{X}$ be a vector space with norm $\|\cdot\|_{\mathcal{X}}$
- $f : \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$ is a proper closed convex function [1]

Proximal operator/mapping

The proximal operator $\text{prox}_{\lambda f} : \mathcal{X} \to \mathbb{R}$ associated to the scaled function λf, where $\lambda > 0$, a.k.a., the proximal operator of f with parameter λ, is defined as

$$\text{prox}_{\lambda f}(x) = \underset{u \in \mathcal{X}}{\text{argmin}} \left(f(u) + \frac{1}{2\lambda} \|u - x\|_{\mathcal{X}}^2 \right) \tag{1}$$

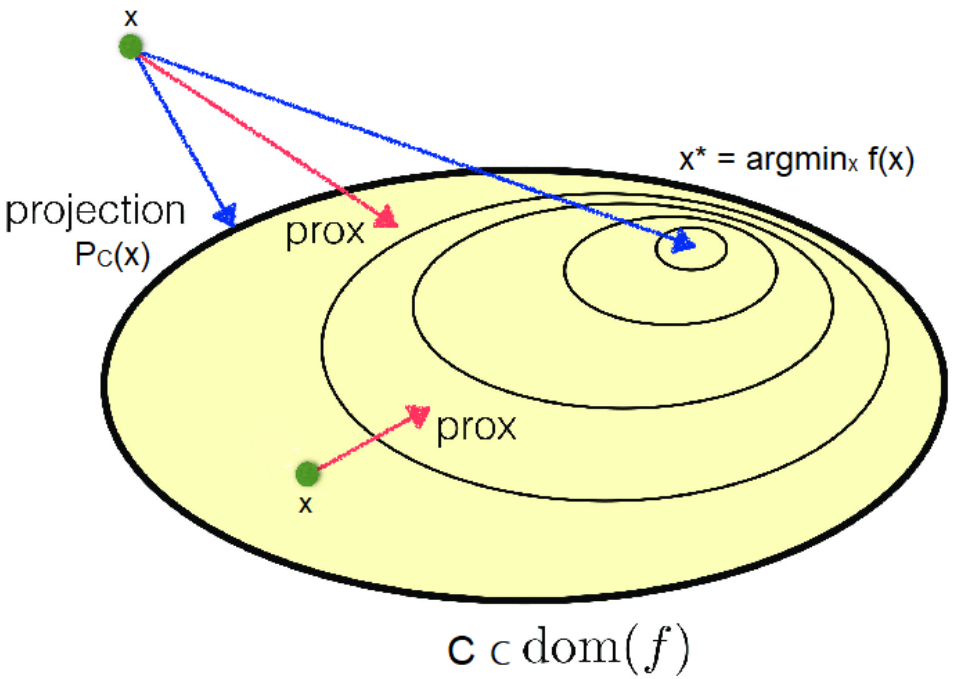
- The usual case is $\mathcal{X} = \mathbb{R}^n$ equipped with the ℓ_2-norm $\|\cdot\|_2$
- For a closed convex function f, $\text{prox}_f(x)$ exists and is unique for all x.

  The function to be minimized is strictly convex and not everywhere infinite (unique minimizer for any $v \in \mathbb{R}^n$); thus, $\text{prox}_{\lambda f}$ well-defined

[1] Def: a proper convex function is an extended real-valued convex function with a non-empty effective domain that never takes on the value $-\infty$

Interpretation

Interpretation (of proximal operator): $\text{prox}_f(x)$ yields a point that compromises between minimizing the function f and being near to x



Relation to proximal point: $\text{prox}_f(x)$ is sometimes called a proximal point of x with respect to f

Interpretation (of $\text{prox}_{\lambda f}(x)$): parameter λ interpreted as a relative weight or trade-off parameter between minimizing the function f and being near to x

First prox Theorem

First prox theorem

Let $f : \mathbb{R}^n \to]-\infty, \infty]$ be a closed and convex function

Then, $\text{prox}_f(x)$ is always singleton for any $x \in \mathbb{R}^n$

First prox Theorem

First prox theorem

Let $f : \mathbb{R}^n \to]-\infty, \infty]$ be a closed and convex function

Then, $\text{prox}_f(x)$ is always singleton for any $x \in \mathbb{R}^n$

Proof:

- For any $x \in \mathbb{R}^n$

$$prox_f(x) = \underset{u \in \mathbb{R}^n}{\text{argmin}}\ \tilde{f}(u, x), \text{ with } \tilde{f}(u, x) := f(u) + \frac{1}{2}\|u - x\|^2$$

- The function $\tilde{f}(u, x)$ is closed and (strongly) convex: sum of i) closed and (strongly) convex function $\frac{1}{2}\|u - x\|^2$ and ii) closed and convex function $f(u)$
- Therefore, there exists a unique minimizer to the problem

First prox Theorem

First prox theorem

Let $f : \mathbb{R}^n \to]-\infty, \infty]$ be a closed and convex function

Then, $\text{prox}_f(x)$ is always singleton for any $x \in \mathbb{R}^n$

Proof:

- For any $x \in \mathbb{R}^n$

$$prox_f(x) = \underset{u \in \mathbb{R}^n}{\text{argmin}}\ \tilde{f}(u, x), \text{ with } \tilde{f}(u, x) := f(u) + \frac{1}{2}\|u - x\|^2$$

- The function $\tilde{f}(u, x)$ is closed and (strongly) convex: sum of i) closed and (strongly) convex function $\frac{1}{2}\|u - x\|^2$ and ii) closed and convex function $f(u)$
- Therefore, there exists a unique minimizer to the problem

Implication: $\text{prox}_f(x)$ is a singleton $\Rightarrow$ $\text{prox}_f(x)$ exists and is unique

Examples

- Proximal operator of $f(x) = c$, constant $c \in \mathbb{R}$

$$\text{prox}_f(x) = \underset{u \in \mathbb{R}^n}{\text{argmin}} \left\{ c + \frac{1}{2} \left|\left|u - x\right|\right|^2 \right\} = x$$

thus, referred to as **identity mapping**

Examples

- Proximal operator of $f(x) = c$, constant $c \in \mathbb{R}$

$$\text{prox}_f(x) = \underset{u \in \mathbb{R}^n}{\text{argmin}} \left\{ c + \frac{1}{2} ||u - x||^2 \right\} = x$$

  thus, referred to as **identity mapping**

- Indicator function of a (non-empty) set $\mathcal{C}$

$$\iota_{\mathcal{C}}(x) = \begin{cases} 0 & \text{if } x \in \mathcal{C} \\ +\infty & \text{if } x \notin \mathcal{C} \end{cases} \tag{2}$$

  Note: indicator function of a (closed) convex set $\mathcal{C}$ is (closed) convex function

Examples

Proximal operator of indicator function $\iota_{\mathcal{C}}$ is **Euclidean projection on** $\mathcal{C}$

$$\begin{aligned} \mathrm{prox}_{\iota_{\mathcal{C}}}(x) &= \underset{u}{\mathrm{argmin}} \left\{ \iota_{\mathcal{C}}(u) + \frac{1}{2} \|u - x\|^2 \right\} \\ &= \underset{u \in \mathcal{C}}{\mathrm{argmin}} \frac{1}{2} \|u - x\|^2 = P_{\mathcal{C}}(x) \end{aligned} \tag{3}$$

Examples

Proximal operator of $f(x) = \lambda |x|$ is

$$\mathrm{prox}_f(x) = [|x| - \lambda]_+ \,\mathrm{sign}(x) = \begin{cases} x - \lambda & \text{if } x > \lambda \\ 0 & \text{if } |x| \geq \lambda \ (-\lambda \leq x \leq \lambda) \\ x + \lambda & \text{if } x < -\lambda \end{cases}$$

<u>Proof</u>: $\mathrm{prox}_f(x)$ minimizer of

$$g(u) = \begin{cases} g_1(u) := \lambda u + \frac{1}{2}(u - x)^2 & u > 0, \\ g_2(u) := -\lambda u + \frac{1}{2}(u - x)^2 & u \leq 0. \end{cases}$$

Examples

Proximal operator of $f(x) = \lambda\,|x|$ is

$$\mathrm{prox}_f(x) = [|x| - \lambda]_+ \,\mathrm{sign}(x) = \begin{cases} x - \lambda & \text{if } x > \lambda \\ 0 & \text{if } |x| \geq \lambda \ (-\lambda \leq x \leq \lambda) \\ x + \lambda & \text{if } x < -\lambda \end{cases}$$

<u>Proof</u>: $\mathrm{prox}_f(x)$ minimizer of

$$g(u) = \begin{cases} g_1(u) := \lambda u + \frac{1}{2}(u - x)^2 & u > 0, \\ g_2(u) := -\lambda u + \frac{1}{2}(u - x)^2 & u \leq 0. \end{cases}$$

- If minimizer attained at $u > 0$, then $0 = g_1'(u) = \lambda + u - x \;\Rightarrow\; u = x - \lambda$
  Therefore, **if** $x > \lambda$, **then** $\mathrm{prox}_{\mathbf{f}}(\mathbf{x}) = \mathbf{x} - \lambda$

Examples

Proximal operator of $f(x) = \lambda\,|x|$ is

$$\text{prox}_f(x) = [|x| - \lambda]_+ \,\text{sign}(x) = \begin{cases} x - \lambda & \text{if } x > \lambda \\ 0 & \text{if } |x| \geq \lambda \ (-\lambda \leq x \leq \lambda) \\ x + \lambda & \text{if } x < -\lambda \end{cases}$$

<u>Proof</u>: $\text{prox}_f(x)$ minimizer of

$$g(u) = \begin{cases} g_1(u) := \lambda u + \frac{1}{2}(u - x)^2 & u > 0, \\ g_2(u) := -\lambda u + \frac{1}{2}(u - x)^2 & u \leq 0. \end{cases}$$

- If minimizer attained at $u > 0$, then $0 = g_1'(u) = \lambda + u - x \;\Rightarrow\; u = x - \lambda$
  Therefore, **if** $x > \lambda$, **then** $\text{prox}_\mathbf{f}(\mathbf{x}) = \mathbf{x} - \lambda$
- If minimizer attained at $u < 0$, then $0 = g_2'(u) = -\lambda + u - x \;\Rightarrow\; u = x + \lambda$
  Therefore, **if** $x < -\lambda$, **then** $\text{prox}_\mathbf{f}(\mathbf{x}) = \mathbf{x} + \lambda$

Examples

Proximal operator of $f(x) = \lambda |x|$ is

$$\text{prox}_f(x) = [|x| - \lambda]_+ \,\text{sign}(x) = \begin{cases} x - \lambda & \text{if } x > \lambda \\ 0 & \text{if } |x| \geq \lambda \ (-\lambda \leq x \leq \lambda) \\ x + \lambda & \text{if } x < -\lambda \end{cases}$$

<u>Proof</u>: $\text{prox}_f(x)$ minimizer of

$$g(u) = \begin{cases} g_1(u) := \lambda u + \frac{1}{2}(u - x)^2 & u > 0, \\ g_2(u) := -\lambda u + \frac{1}{2}(u - x)^2 & u \leq 0. \end{cases}$$

- If minimizer attained at $u > 0$, then $0 = g_1'(u) = \lambda + u - x \;\Rightarrow\; u = x - \lambda$

  Therefore, **if** $x > \lambda$, **then** $\text{prox}_\mathbf{f}(\mathbf{x}) = \mathbf{x} - \lambda$
- If minimizer attained at $u < 0$, then $0 = g_2'(u) = -\lambda + u - x \;\Rightarrow\; u = x + \lambda$

  Therefore, **if** $x < -\lambda$, **then** $\text{prox}_\mathbf{f}(\mathbf{x}) = \mathbf{x} + \lambda$
- If minimizer of convex function exists and not attained at any point of differentiability, then it must be attained at a point of non-differentiability

  Thus, **if** $|x| \leq \lambda$, **then** $\text{prox}_f(x)$ only point of non-differentiability of g, namely, 0.

Examples

The proximal operator of affine function $f(x) = a^\top x + b$ is

$$\mathrm{prox}_f(x) = x - a$$

Examples

The proximal operator of affine function $f(x) = a^\top x + b$ is

$$\mathrm{prox}_f(x) = x - a$$

Proof:

$$\begin{aligned}
\mathrm{prox}_f(x) &= \underset{u}{\mathrm{argmin}} \left\{ a^\top u + b + \frac{1}{2}\|u - x\|^2 \right\} \\
&= \underset{u}{\mathrm{argmin}} \left\{ a^\top u + b - \frac{1}{2}\|a\|^2 + \frac{1}{2}\|u - x\|^2 + \frac{1}{2}\|a\|^2 \right\} \\
&= \underset{u}{\mathrm{argmin}} \left\{ a^\top x + b - \frac{1}{2}\|a\|^2 + \frac{1}{2}\|u - x\|^2 - a^T(u - x) + \frac{1}{2}\|a\|^2 \right\} \\
&= \underset{u}{\mathrm{argmin}} \left\{ a^\top x + b - \frac{1}{2}\|a\|^2 + \frac{1}{2}\|u - (x - a)\|^2 \right\} \\
&= \underset{u}{\mathrm{argmin}} \left\{ c(\mathrm{onstant}) + \frac{1}{2}\|u - (x - a)\|^2 \right\} \\
&= x - a
\end{aligned}$$

Proximal operator library: G. Chierchia, E. Chouzenoux, P. L. Combettes, and J.-C. Pesquet. "The Proximity Operator Repository." https://proximity-operator.net/

Base Calculus Rules

- **Separable sum**:

  If f separable across two variables, i.e., $f(x, y) = g(x) + h(y)$, then

  $$\text{prox}_f(x, y) = (\text{prox}_g(x), \text{prox}_h(y))$$

  Thus, evaluating proximal operator of separable function reduces to evaluating proximal operators for each separable part (independently)

Base Calculus Rules

- **Separable sum**:

  If f separable across two variables, i.e., $f(x, y) = g(x) + h(y)$, then

  $$\text{prox}_f(x, y) = (\text{prox}_g(x), \text{prox}_h(y))$$

  Thus, evaluating proximal operator of separable function reduces to evaluating proximal operators for each separable part (independently)

- **Generalization**: prox of separable functions

  For f closed convex and separable

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

  with f_i proper closed and convex univariate functions

  $$\text{prox}_f(x_1, \ldots, x_n) = \Big(\text{prox}_{f_i}(x_i)\Big)_{i=1}^{n}$$

Base Calculus Rules

- **Right scalar multiplication** ($\lambda > 0$): if $f(x) = \lambda g(x/\lambda)$, then $\mathrm{prox}_f(x) = \lambda \, \mathrm{prox}_{\lambda^{-1} g}(x/\lambda)$

Base Calculus Rules

- **Right scalar multiplication** ($\lambda > 0$): if $f(x) = \lambda g(x/\lambda)$, then $\mathrm{prox}_f(x) = \lambda\, \mathrm{prox}_{\lambda^{-1}g}(x/\lambda)$
- **Scaling** and **vector translation** of argument (pre-composition): if $f(x) = g(\lambda x + a), \lambda \neq 0$ then $\mathrm{prox}_f(x) = \dfrac{1}{\lambda}\big(\mathrm{prox}_{\lambda^2 g}(\lambda x + a) - a\big)$

Base Calculus Rules

- **Right scalar multiplication** ($\lambda > 0$): if $f(x) = \lambda g(x/\lambda)$, then $\mathrm{prox}_f(x) = \lambda\, \mathrm{prox}_{\lambda^{-1}g}(x/\lambda)$
- **Scaling** and **vector translation** of argument (pre-composition): if $f(x) = g(\lambda x + a), \lambda \neq 0$ then $\mathrm{prox}_f(x) = \dfrac{1}{\lambda}\big(\mathrm{prox}_{\lambda^2 g}(\lambda x + a) - a\big)$
- **Linear addition** (post-composition): if $f(x) = g(x) + a^\top x + b$, then $\mathrm{prox}_{\lambda f}(x) = \mathrm{prox}_{\lambda g}(x - \lambda a)$

Base Calculus Rules

- **Right scalar multiplication** ($\lambda > 0$): if $f(x) = \lambda g(x/\lambda)$, then $\mathrm{prox}_f(x) = \lambda\, \mathrm{prox}_{\lambda^{-1}g}(x/\lambda)$
- **Scaling** and **vector translation** of argument (pre-composition): if $f(x) = g(\lambda x + a), \lambda \neq 0$ then $\mathrm{prox}_f(x) = \dfrac{1}{\lambda}\big(\mathrm{prox}_{\lambda^2 g}(\lambda x + a) - a\big)$
- **Linear addition** (post-composition): if $f(x) = g(x) + a^\top x + b$, then $\mathrm{prox}_{\lambda f}(x) = \mathrm{prox}_{\lambda g}(x - \lambda a)$
- **Regularization**: if $f(x) = g(x) + \dfrac{\mu}{2}\|x - a\|_2^2$, then $\mathrm{prox}_{\lambda f}(x) = \mathrm{prox}_{\nu g}\left((\dfrac{\nu}{\lambda}x + \mu\nu a)\right)$ where $\nu = \dfrac{1}{\lambda^{-1} + \mu}$

  Particular case: $a = 0$

$$\mathrm{prox}_{\lambda f}(x) = \mathrm{prox}_{\nu g}\left((\frac{\nu}{\lambda}x)\right),\ \nu = \frac{1}{\lambda^{-1} + \mu}$$

Examples

Proximal operator of $f\colon \mathbb{R}^n \to \mathbb{R}\colon x \to \lambda \|x\|_1$ is soft-thresholding (shrinkage) operation

$$\mathrm{prox}_f(x) = \left(T_\lambda(x_j)\right)_{j=1}^n$$

- Indeed, let reformulate f as a sum of n functions:

$$g(x) = \sum_{i=1}^n \varphi(x_i)$$

- Since **i)** for $\varphi(t) = \lambda |t|$ the proximal operator is

$$\mathrm{prox}_\varphi(t) = [|t| - \lambda]_+ \,\mathrm{sign}(x) = \begin{cases} t - \lambda, & t \geq \lambda \\ 0, & -\lambda < t < \lambda \\ t + \lambda, & t \leq -\lambda \end{cases}$$

which corresponds to the soft-thresholding operator $T_\lambda(t)$

Examples

and **ii)** $\text{prox}_f(x_1, \ldots, x_n) = \Big(\text{prox}_{f_i}(x_i)\Big)_{i=1}^n$, we obtain

$$\text{prox}_f(x_1, \ldots, x_n) = \Big(T_\lambda(x_i)\Big)_{i=1}^n$$

- Thus, soft-thresholding operation is proximal operator of ℓ_1-norm

Examples

and **ii)** $\mathrm{prox}_f(x_1, \ldots, x_n) = \Big(\mathrm{prox}_{f_i}(x_i)\Big)_{i=1}^n$, we obtain

$$\mathrm{prox}_f(x_1, \ldots, x_n) = \Big(T_\lambda(x_i)\Big)_{i=1}^n$$

- Thus, soft-thresholding operation is proximal operator of ℓ_1-norm

Proximal operator of $f\colon \mathbb{R}^n \to \mathbb{R}\colon x \to \lambda \|x\|_2$ is

$$\mathrm{prox}_f(x) = \begin{cases} [\|x\|_2 - \lambda]_+ \dfrac{x}{\|x\|_2} & x \neq 0 \\ 0 & x = 0 \end{cases}$$

$$= \begin{cases} (1 - \dfrac{\lambda}{\|x\|_2})x & \|x\|_2 \geq \lambda \\ 0 & \|x\|_2 < \lambda \end{cases}$$

Second prox Theorem

Second prox theorem

Let $f : \mathbb{R}^n \to]-\infty, \infty]$ be a closed and convex function

Then, for any $x, u \in \mathbb{R}^n$, following statements are equivalent:

(i) $u = \text{prox}_f(x)$

(ii) $x - u \in \partial f(u)$

(iii) $f(y) \geq f(u) + \langle x - u \mid y - u \rangle$ for any $y \in \mathbb{R}^n$

Second prox Theorem

Second prox theorem

Let $f : \mathbb{R}^n \to]-\infty, \infty]$ be a closed and convex function

Then, for any $x, u \in \mathbb{R}^n$, following statements are equivalent:

(i) $u = \text{prox}_f(x)$

(ii) $x - u \in \partial f(u)$

(iii) $f(y) \geq f(u) + \langle x - u \mid y - u \rangle$ for any $y \in \mathbb{R}^n$

Proof:

- By definition, $u = \text{prox}_f(x)$ **if and only if** u minimizer of

$$\min_v \left\{ f(v) + \frac{1}{2} \|v - x\|^2 \right\} \quad (*)$$

- By first-order optimality condition and the sum rule of subdifferential calculus, (*) equivalent to

$$0 \in \partial f(u) + u - x \;\; \text{thus} \;\; x - u \in \partial f(u)$$

- By definition of subgradient, (ii) equivalent to (iii)

Consequences of second prox theorem

For a proper, closed and convex function f, $x^\star$ is a minimizer of f if and only if $x^\star = \text{prox}_f(x^\star)$

Fixed point property

The point $x^\star$ minimizes f (proper, closed, convex) if and only if $x^\star = prox_f(x^\star)$

Consequences of second prox theorem

For a proper, closed and convex function f, $x^\star$ is a minimizer of f if and only if $x^\star = \text{prox}_f(x^\star)$

Fixed point property

The point $x^\star$ minimizes f (proper, closed, convex) if and only if $x^\star = prox_f(x^\star)$

Proof:

- x is a minimizer of f if and only if $0 \in \partial f(x)$
  if and only if $x - x \in \partial f(x)$
- By second prox theorem (equivalence between (i) and (ii)), $x = \text{prox}_f(x)$

Consequences of second prox theorem

For a proper, closed and convex function f, $x^\star$ is a minimizer of f if and only if $x^\star = \text{prox}_f(x^\star)$

Fixed point property

The point $x^\star$ minimizes f (proper, closed, convex) if and only if $x^\star = prox_f(x^\star)$

Proof:

- x is a minimizer of f if and only if $0 \in \partial f(x)$
  if and only if $x - x \in \partial f(x)$
- By second prox theorem (equivalence between (i) and (ii)), $x = \text{prox}_f(x)$

Implication: f can be minimized by finding/searching for fixed point(s) of its proximal operator

Consequences of second prox theorem

When $f = \iota_{\mathcal{C}}$, with $\mathcal{C}$ nonempty closed and convex set, equivalence between (i) and (iii) in second prox theorem corresponds to **second projection theorem**:

Consequences of second prox theorem

When $f = \iota_{\mathcal{C}}$, with $\mathcal{C}$ nonempty closed and convex set, equivalence between (i) and (iii) in second prox theorem corresponds to **second projection theorem**:

Second projection theorem

Let $\mathcal{C} \subseteq \mathbb{R}^n$ be a nonempty closed and convex set, and let $u \in \mathcal{C}$

Then, $u = \mathrm{P}_{\mathcal{C}}(x)$ **if and only if** $\langle x - u \mid y - u \rangle \geq 0$, for any $y \in \mathcal{C}$

Consequences of second prox theorem

Firm nonexpansivity of prox operator

Let f be a proper closed and convex function. Then, for any $x, y \in \mathbb{R}^n$

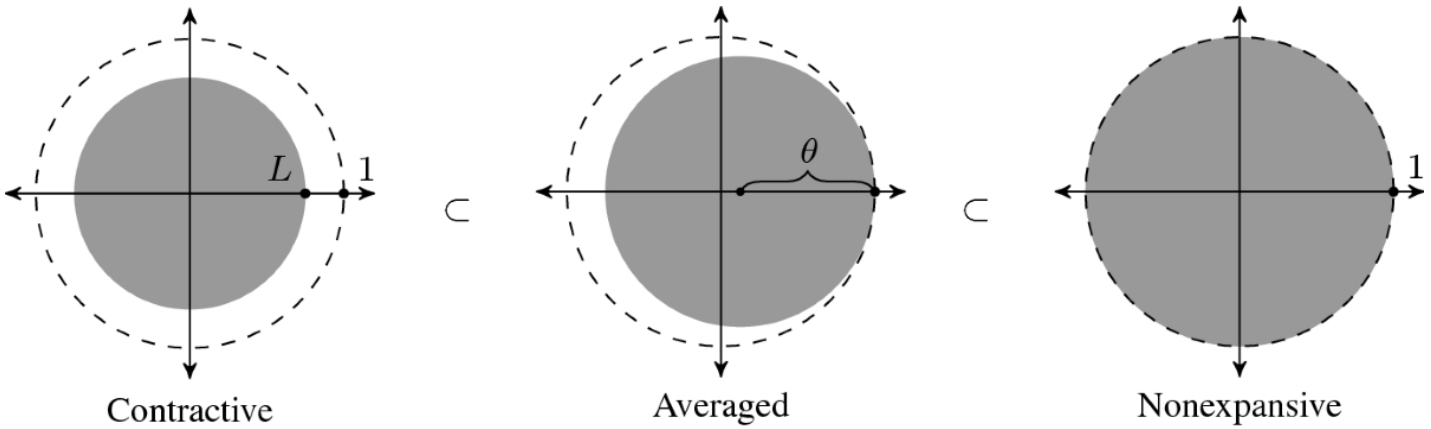
(a) **Firm nonexpansivity**:

$$\|\text{prox}_f(x) - \text{prox}_f(y)\|^2 \leq \langle x - y \mid \text{prox}_f(x) - \text{prox}_f(y)\rangle$$

(b) Nonexpansivity (1-Lipschitz, i.e., Lipschitz constant $L = 1$))[a]:

$$\|\text{prox}_f(x) - \text{prox}_f(y)\| \leq \|x - y\|$$

[a] mapping a pair of points by nonexpansive operator does not increase their distance



Consequences of second prox theorem

Property: Proximal operators are firmly nonexpansive
Proof:

- Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ be closed convex proper (CCP)
- Assume that
$$u_1 = \text{prox}_f(x_1) \text{ and } u_2 = \text{prox}_f(x_2)$$
- Equivalently (by (i)-(ii) of 2nd Prox Theorem)
$$x_1 - u_1 \in \partial f(u_1) \text{ and } x_2 - u_2 \in \partial f(u_2)$$
- Property
  - Subgradient : $f(y) \geq f(x) + g^T(y - x)$, $f(y) \leq f(x) \Rightarrow g^T(y - x) \leq 0$
  - ∂f operator is **monotone**: $u \in \partial f(x), v \in \partial f(y)$ yields $(u - v)^T(x - y) \geq 0$

  implies
$$\langle x_1 - u_1 - (x_2 - u_2) \mid u_1 - u_2 \rangle \geq 0$$
$$\Rightarrow \langle x_1 - x_2 \mid u_1 - u_2 \rangle \geq \|u_1 - u_2\|_2^2$$
- This shows that prox_f is firmly nonexpansive.

Example

Example : subdifferential operator ∂ of a lower semicontinuous convex function f

Example

Example : subdifferential operator ∂ of a lower semicontinuous convex function f

- Finding x that minimizes f : $0 \in \partial f(x)$
  $\iff x \in x + \partial f(x) \iff x \in (\mathrm{Id} + \partial f)(x)$
  $\iff x = (\mathrm{Id} + \partial f)^{-1}(x)$
- Since ∂f is convex, Minty's theorem guarantees that for any x the set defined by $(\mathrm{Id} + \partial f)^{-1}(x)$ is a singleton
- Thus, $(\mathrm{Id} + \partial f)^{-1}$ can be viewed as a function $\mathcal{H} \to \mathcal{H}$ called the **resolvent** of operator T $(= \partial f)$

$\Rightarrow$ Find x such that $0 \in \partial f(x)$ is equivalent to $x = (\mathrm{Id} + \partial f)^{-1}(x)$ that can be solved with fixed point iteration (FPI):

$$x_{k+1} = (\mathrm{Id} + \partial f)^{-1}(x_k) \tag{4}$$

Note: x is a fixed point of opertor T if $x = Tx \to$ the fixed-point iteration (FPI) is $x_{k+1} = Tx_k$

Example

The prox operator is intimately related to the subdifferential operator ∂

$$(\forall x \in \mathcal{H})\,(\forall u \in \mathcal{H}) \quad u \in \partial f(x) \Longleftrightarrow x = \mathrm{prox}_f(x+u) \tag{5}$$

The resolvent $(\mathrm{Id} + \lambda \partial f)^{-1}$ of the subdifferential operator ∂f is the proximal operator associated to the function λf

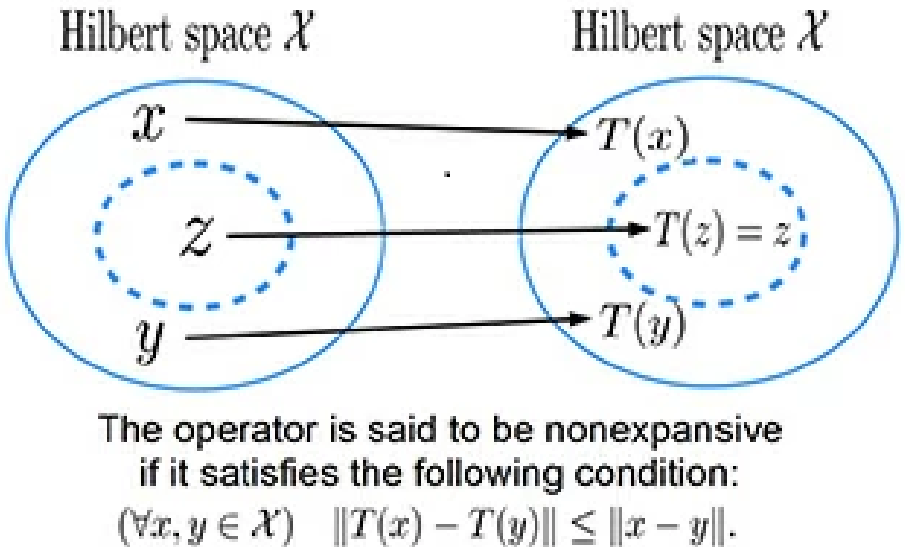
$$(\mathrm{Id} + \lambda \partial f)^{-1} = \mathrm{prox}_{\lambda f}$$

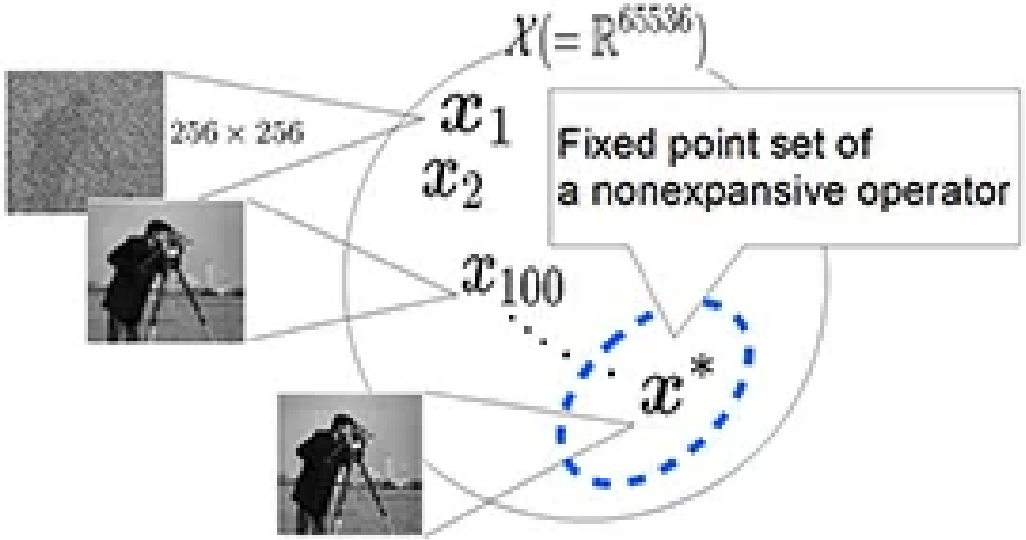
Proof:

$$\begin{aligned}
y = (\mathrm{Id} + \lambda \partial f)^{-1}(x) &\Longleftrightarrow (\mathrm{Id} + \lambda \partial f)(y) \ni x \\
&\Longleftrightarrow 0 \in \lambda \partial f(y) + (y - x) \\
&\Longleftrightarrow 0 \in \lambda \partial_y \left(f(y) + \frac{1}{2\lambda} \|y - x\|_2^2 \right) \\
&\Longleftrightarrow y = \underset{y}{\mathrm{argmin}} \left\{ f(y) + \frac{1}{2\lambda} \|y - x\|_2^2 \right\} \\
&\Longleftrightarrow y = \mathrm{prox}_{\lambda f}(x)
\end{aligned}$$

Example

Nonexpansive mapping and fixed point set: if z does not move by operator T, then z is a fixed point of T





Signal processing based on the fixed-point set of a computable nonexpansive operator

Consequences of second prox theorem

Implication: following firm nonexpansivity Property (a), successive iteration of the form

$$x_{k+1} = \text{prox}_{\lambda_k f}(x_k)$$

converge (weakly) to a fixed point of proximal operator; thus, to a minimizer of f (if it exists)

Proximal Point Method

- Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ be a proper closed convex function
- Consider minimization problem

$$\min_x f(x) \tag{6}$$

- General principle of **proximal point method** : iteratively apply proximal operator $\text{prox}_{\lambda f}$, $\lambda > 0$ to initial point x_0

$$x_{k+1} = \text{prox}_{\lambda_k f}(x_k) \tag{7}$$

Proximal Point Method: Properties

- **If** f has a minimum, **then** x_k converges to the set of minimizers of f and $f(x_k)$ converges to its optimal value
- Variation on proximal minimization algorithm : parameter λ values changes in each iteration k
  - ◦) Replace constant value λ with λ_k in the iteration
  - ◦) Convergence is guaranteed provided $\lambda_k > 0$ and $\sum_{k=1} \lambda_k = \infty$
- When $\text{prox}_f(x)$ not available (analytically), minimizations required in evaluation of proximal operator to be carried out with error, provided errors in minimizations satisfy certain conditions (such as summability)

Proximal Gradient Method

Consider unconstrained problem with composite objective function

$$\min_x \; f(x) = g(x) + h(x) \tag{8}$$

where

ia) $g : \mathbb{R}^n \to \mathbb{R}$ is convex, differentiable

ib) g admits a Lipschitz continuous gradient, i.e., ∇g is Lipschitz continuous with constant $L > 0$ over $int(\mathrm{dom}(g))$

$$\|\nabla g(x) - \nabla g(y)\| \leq L \|x - y\| \;\; \forall x, y$$

ii) $h : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is closed, convex, possibly nondifferentiable, assumed with inexpensive prox-operator (prox_h)

Proximal Gradient Method

Important consequences from these assumptions on the function g:

ia) Affine lower bound from the convexity of function g:

$$g(y) \geq g(x) + \nabla g(x)^\intercal (y - x), \ \forall x, y \tag{9}$$

Linear approximation always underestimates f

ib) Upper quadratic approximation of g (quadratic upper bound) from Lipschitz property of ∇g:

$$g(y) \leq g(x) + \nabla g(x)^\intercal (y - x) + \frac{L}{2} \|y - x\|^2, \ \forall x, y \tag{10}$$

Note: for convex function g, equivalence

Proximal Gradient Method

Idea

1) Make use of quadratic approximation to g

$$\hat{g}_t(u) = g(x) + \nabla g(x)^T (u - x) + \frac{1}{2t} \|u - x\|_2^2$$

2) Leave function h alone

Proximal Gradient Method

Idea

1) Make use of quadratic approximation to g

$$\hat{g}_t(u) = g(x) + \nabla g(x)^T (u - x) + \frac{1}{2t} \|u - x\|_2^2$$

2) Leave function h alone

Hence x^* minimizes $h(u)$ **plus** quadratic local approximation of $g(u)$ around x

$$\begin{aligned}
x^* &= \underset{u}{\operatorname{argmin}} \left\{ h(u) + \hat{g}_t(u) \right\} && (11)\\
&= \underset{u}{\operatorname{argmin}} \left\{ h(u) + g(x) + \nabla g(x)^T (u - x) + \frac{1}{2t} \|u - x\|_2^2 \right\} && (12)\\
&= \underset{u}{\operatorname{argmin}} \left\{ h(u) + g(x) + \nabla g(x)^T (u - x) + \frac{1}{2t} \|u - x\|_2^2 + \frac{t^2}{2} \|\nabla g(x)\|_2^2 \right\} \\
&= \underset{u}{\operatorname{argmin}} \left\{ h(u) + \frac{1}{2t} \|u - (x - t\nabla g(x))\|_2^2 \right\} && (13)\\
&\rightarrow \operatorname{prox}_{th} \left(x - t\nabla g(x) \right) && (14)
\end{aligned}$$

Proximal Gradient Method: Algorithm

Proximal Gradient algorithm:

`input`: Choose initial point $x_0 \in int(\mathrm{dom}(f))$

`repeat`:

1. Choose step size $t_k > 0$: fixed step size $t_k = 1/L$
   or inexact (computed by backtracking) line search on g.
2. Update: $x_{k+1} = \mathrm{prox}_{t_k h}\left(x_k - t_k \nabla g(x_k)\right)$

`until`: stopping criterion is satisfied

Proximal Gradient Method: Properties

Properties

- Proximal mapping $prox_{th}(.)$ does NOT depend on g at all, only on h

  Important: $prox_{th}(.)$ has closed-form for many important functions h -see http://proximity-operator.net/

Proximal Gradient Method: Properties

Properties

- Proximal mapping $prox_{th}(.)$ does NOT depend on g at all, only on h

  Important: $prox_{th}(.)$ has closed-form for many important functions h -see http://proximity-operator.net/

- Smooth part g can be complicated but only need to compute its gradients

Proximal Gradient Method: Properties

Properties

- Proximal mapping $prox_{th}(.)$ does NOT depend on g at all, only on h

  Important: $prox_{th}(.)$ has closed-form for many important functions h -see http://proximity-operator.net/

- Smooth part g can be complicated but only need to compute its gradients

- If L not known, step sizes t_k can be found by backtracking line search (t_k value chosen in each step)

Proximal Gradient Method: Properties

Properties

- Proximal mapping $prox_{th}(.)$ does NOT depend on g at all, only on h

  Important: $prox_{th}(.)$ has closed-form for many important functions h -see http://proximity-operator.net/

- Smooth part g can be complicated but only need to compute its gradients

- If L not known, step sizes t_k can be found by backtracking line search (t_k value chosen in each step)

- **Convergence**: convergence rate matches gradient descent rate (see below)

Proximal Gradient Method: Convergence properties

Consider unconstrained problem with composite objective function

$$\min_x f(x) = g(x) + h(x)$$

with

- g convex, differentiable, $\mathrm{dom}(g) = \mathbb{R}^n$
  ∇g is Lipschitz continuous with constant $L > 0$,
- h is convex, and prox_{th} can be evaluated
- Optimal value $f^\star$ is finite and attained at $x^\star$ (not necessarily unique)

Proximal Gradient Method: Convergence properties

Consider unconstrained problem with composite objective function

$$\min_x f(x) = g(x) + h(x)$$

with

- g convex, differentiable, $\mathrm{dom}(g) = \mathbb{R}^n$
  ∇g is Lipschitz continuous with constant $L > 0$,
- h is convex, and prox_{th} can be evaluated
- Optimal value $f^\star$ is finite and attained at $x^\star$ (not necessarily unique)

If fixed step size $t_k = t \in]0, 1/L]$
-or- backtracking line search with $t_k \geq t_{\min} \triangleq \min\{t_0, \alpha/L\}$ is used

Then proximal gradient algorithm convergence rate $O(1/k)$ or $O(1/\varepsilon)$

- $(f(x(k)) - f^\star)$ decreases at least as fast as $1/k$
- $(f(x(k)) - f^\star) \leq \varepsilon$ after $O(1/\varepsilon)$ iterations

Proximal Gradient Method: particular cases

Consider unconstrained problem with composite objective function

$$\min_x \; f(x) = g(x) + h(x) \tag{15}$$

$$\text{PGM update}: \; x_{k+1} = \text{prox}_{t_k h}\left(x_k - t_k \nabla g(x_k)\right)$$

Proximal Gradient Method: particular cases

Consider unconstrained problem with composite objective function

$$\min_x \; f(x) = g(x) + h(x) \tag{15}$$

$$\text{PGM update}: \; x_{k+1} = \text{prox}_{t_k h}\left(x_k - t_k \nabla g(x_k)\right)$$

- **Gradient method**: if $h(x) = 0$, then $\min_x \; f(x) = g(x)$ with $x \in \mathbb{R}^n$

$$x_{k+1} = x_k - t_k \nabla g(x_k)$$

Proximal Gradient Method: particular cases

Consider unconstrained problem with composite objective function

$$\min_x \; f(x) = g(x) + h(x) \tag{15}$$

$$\text{PGM update}: \; x_{k+1} = \text{prox}_{t_k h}\left(x_k - t_k \nabla g(x_k)\right)$$

- **Gradient method**: if $h(x) = 0$, then $\min_x \; f(x) = g(x)$ with $x \in \mathbb{R}^n$

$$x_{k+1} = x_k - t_k \nabla g(x_k)$$

- **Projected gradient method**: if $h(x) = \iota_{\mathcal{C}}(x)$, then $\min_x \; f(x) = g(x)$ s.t. $x \in \mathcal{C}$

$$x_{k+1} = P_C\left(x_k - t_k \nabla g_k(x_k)\right)$$

Proximal Gradient Method: ISTA

- **Proximal point method**: if $g(x) = 0$, then $\min_x f(x) = h(x)$

$$x_{k+1} = \text{prox}_{\lambda_k h}(x_k)$$

Proximal Gradient Method: ISTA

- **Proximal point method**: if $g(x) = 0$, then $\min_x f(x) = h(x)$

$$x_{k+1} = \text{prox}_{\lambda_k h}(x_k)$$

- **Iterative Soft-Thresholding Algorithm (ISTA)**:

$$\min_x f(x) = g(x) + h(x)$$

with

○) $g : \mathbb{R}^n \to]-\infty, +\infty]$ closed, differentiable over $int(\text{dom}(g))$

∇g Lipschitz continuous over $int(\text{dom}(g))$ (L-smooth)

○) $h : \mathbb{R}^n \to \mathbb{R} : x \to \lambda \|x\|_1$

$$\min_x g(x) + \lambda \|x\|_1$$

Proximal Gradient Method: ISTA

ISTA algorithm:

`init`: choose initial point x_0

`repeat`:

1. Compute step size $t_k > 0$:

   fixed $t_k = 1/L, L > 0$ or $t_k = 1/L_k, L_k \in]L/2, \infty[, L > 0$,

   or determined by **customized** backtracking line search on g.

2. Update: $x_{k+1} = T_{\lambda t_k}\big(x_k - t_k \nabla g(x_k)\big)$

`until`: stopping criterion is satisfied

Proximal Gradient Method: ISTA

- First-order approximation + proximal term (10)

$$
\begin{aligned}
x_{k+1} &= \underset{x\in\mathbb{R}^n}{\operatorname{argmin}}\ \lambda \|x\|_1 + \nabla f(x_k)^\top (x - x_k) + \frac{1}{2t}\|x - x_k\|_2^2 \\
&= \underset{x\in\mathbb{R}^n}{\operatorname{argmin}}\ \lambda \|x\|_1 + \frac{1}{2}\|x - (x_k - t\nabla f(x_k))\|_2^2 \\
&= \operatorname{shrink}(x_k - t\nabla f(x_k); \lambda t)
\end{aligned}
\tag{16}
$$

- Gradient step : bring elements with nonzero components
- Shrinkage step : eliminate some of them by soft-threshold shrinkage operation $\operatorname{shrink}(y = x_k - t\nabla f(x_k); \lambda t)$, where

$$
\begin{aligned}
\operatorname{shrink}(y; \lambda t) &= \underset{x\in\mathbb{R}^n}{\operatorname{argmin}}\ \lambda t\, |x|_1 + \frac{1}{2}\|x - y\|_2^2 \\
&= \operatorname{sign}(y)\max(|y| - \lambda t, 0) \\
&= \begin{cases} y - \lambda t \operatorname{sign}(y) & |y| \geq \lambda t \\ 0 & |y| < \lambda t \end{cases}
\end{aligned}
\tag{17}
$$

FISTA (Fast Iterative Shrinkage-Thresholding Algorithm)

Consider the model:

$$\min_{x \in \mathbb{R}^n} f(x) = g(x) + h(x)$$

where

i) $g : \mathbb{R}^n \to \mathbb{R}$ is convex, differentiable

∇g is Lipschitz continuous with constant $L > 0$ over $int(\mathrm{dom}(g))$

$$\|\nabla g(x) - \nabla g(y)\| \leq L \|x - y\| \ \ \forall x, y$$

ii) $h : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is closed, convex, possibly nondifferentiable

FISTA (Fast Iterative Shrinkage-Thresholding Algorithm)

Consider the model:

$$\min_{x \in \mathbb{R}^n} f(x) = g(x) + h(x)$$

where

i) $g : \mathbb{R}^n \to \mathbb{R}$ is convex, differentiable

∇g is Lipschitz continuous with constant $L > 0$ over $int(\mathrm{dom}(g))$

$$\|\nabla g(x) - \nabla g(y)\| \leq L \|x - y\| \ \ \forall x, y$$

ii) $h : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is closed, convex, possibly nondifferentiable

Main goals

- **Generalization**: any function h that is closed and convex
- **Acceleration**: to achieve optimal $O(1/k^2)$ convergence rate –compared to the rate $O(1/k)$ of PGM

FISTA (Fast Iterative Shrinkage-Thresholding Algorithm)

How ? instead of

$$x_{k+1} = \text{prox}_{t_k h}\left(x_k - t_k \nabla g(x_k)\right)$$

consider

$$x_{k+1} = \text{prox}_{t_k h}\left(y_k - t_k \nabla g(y_k)\right) \tag{18}$$

where y_k : linear combination of x_{k+1} and x_k

FISTA (Fast Iterative Shrinkage-Thresholding Algorithm)

How ? instead of

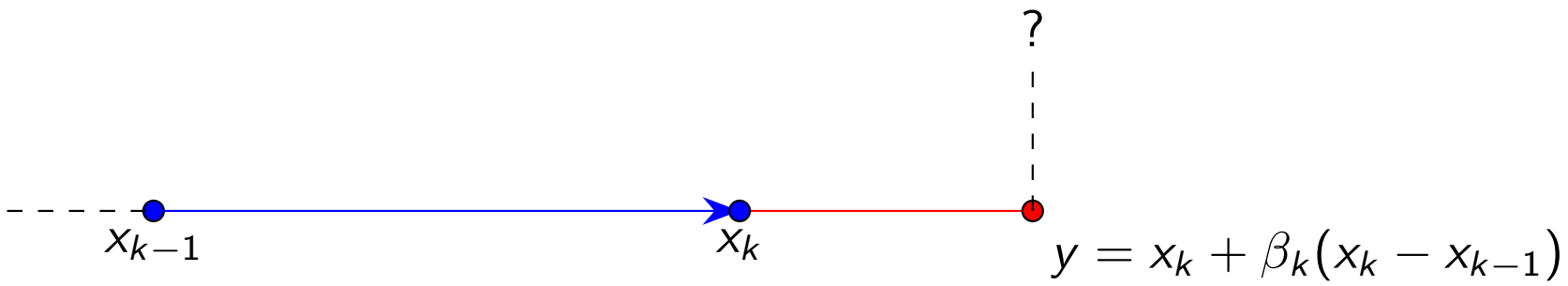
$$x_{k+1} = \text{prox}_{t_k h}\left(x_k - t_k \nabla g(x_k)\right)$$

consider

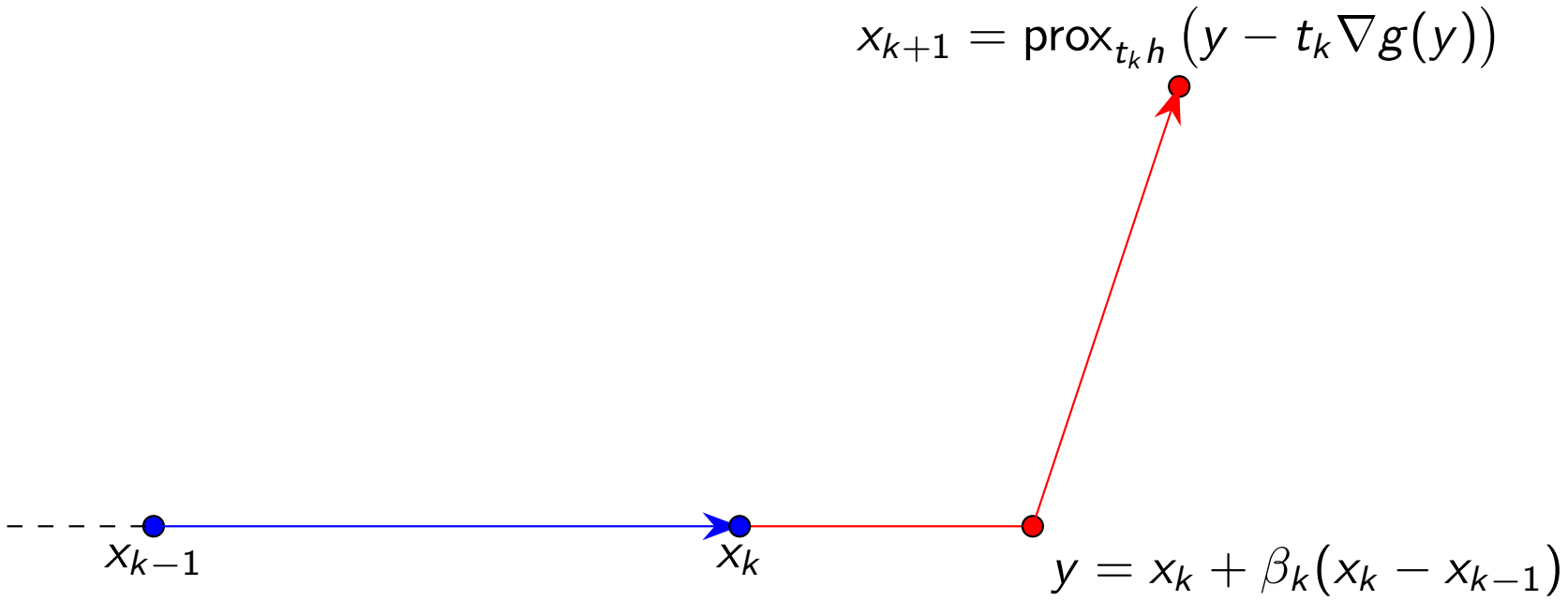
$$x_{k+1} = \text{prox}_{t_k h}\left(y_k - t_k \nabla g(y_k)\right) \tag{19}$$

where y_k : linear combination of x_k and x_{k-1} of the form

$$y_k = x_k + \beta_k \left(x_k - x_{k-1}\right) \tag{20}$$



FISTA (Fast Iterative Shrinkage-Thresholding Algorithm)

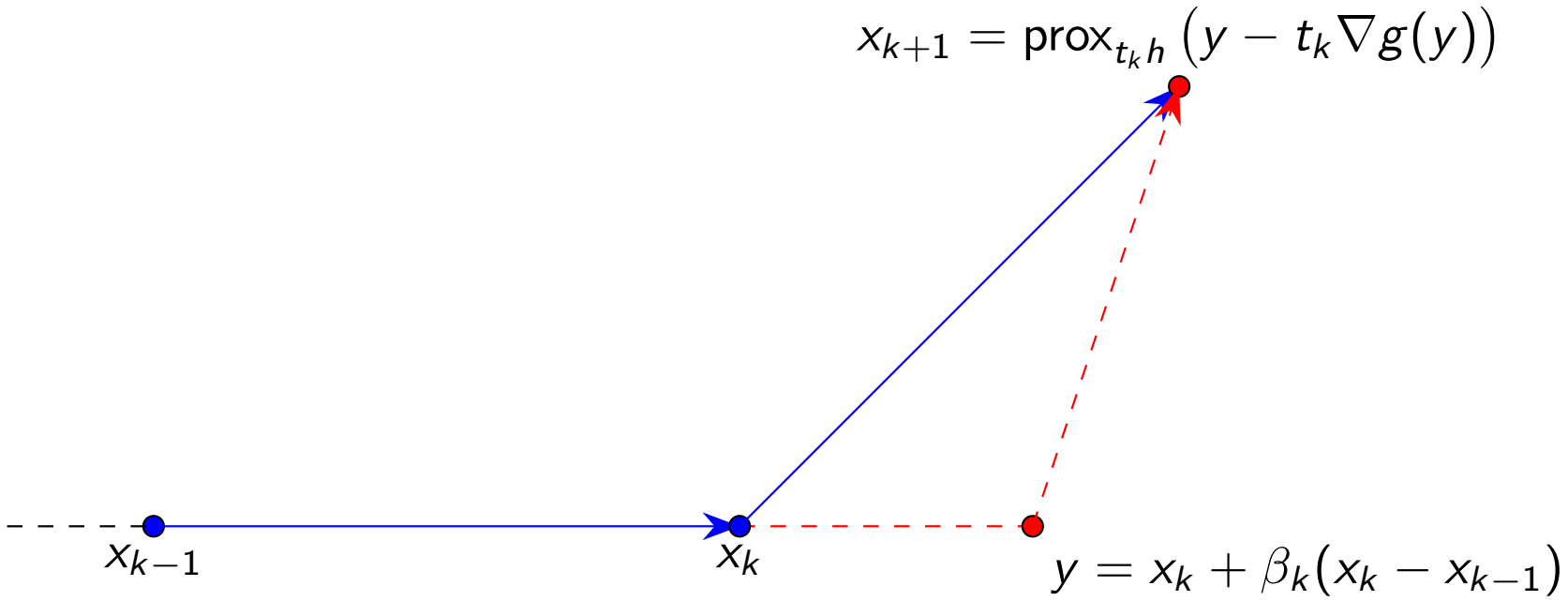


- After initial iteration, next iterations are proximal gradient steps at extrapolated points y

$$x_{k+1} = \text{prox}_{t_k h}\left(y_k - t_k \nabla g(y_k)\right)$$

- x_k is feasible (in $\text{dom}(h)$) but y may reside outside $\text{dom}(h)$

FISTA (Fast Iterative Shrinkage-Thresholding Algorithm)



- After initial iteration, next iterations are proximal gradient steps at extrapolated points y

$$x_{k+1} = \text{prox}_{t_k h}\left(y_k - t_k \nabla g(y_k)\right)$$

- x_k is feasible (in $\text{dom}(h)$) but y may reside outside $\text{dom}(h)$

FISTA (Fast Iterative Shrinkage-Thresholding Algorithm)

FISTA algorithm:

`input`: initial point $x_0 \in \mathbb{R}^n$

`init`: $y_0 \leftarrow x_0$, $\tilde{t}_0 \leftarrow 1$ // initial iteration is PG iteration

`repeat`:

1. Set step size $t_k > 0$:

   fixed $t_k = 1/L, L > 0$ // $L =$ Lipschitz constant of ∇g

   or $t_k = 1/L_k, L_k \in]\alpha L, L], \alpha < 1$

   or determined by **customized** backtracking line search on g.
2. Compute: $x_{k+1} = \text{prox}_{t_k h}\left(y_k - t_k \nabla g(y_k)\right)$
3. Compute **momentum factor** $\tilde{t}_{k+1} > 0$:

   $$\tilde{t}_{k+1} = \frac{1 + \sqrt{1 + 4\tilde{t}_k^2}}{2}$$
4. Update: $y_{k+1} = x_{k+1} + \dfrac{\tilde{t}_k - 1}{\tilde{t}_{k+1}}(x_{k+1} - x_k)$

`until`: stopping criterion is satisfied

FISTA: Convergence properties

Convergence properties:

- Minimize $f(x) = g(x) + h(x)$

  ◦) g is convex, differentiable, $\mathrm{dom}(f) = \mathbb{R}^n$ and ∇g is Lipschitz continuous with constant $L_g > 0$

  ◦) h is closed, convex and proximal function prox_h can be evaluated

FISTA: Convergence properties

Convergence properties:

- Minimize $f(x) = g(x) + h(x)$

  ○) g is convex, differentiable, $\mathrm{dom}(f) = \mathbb{R}^n$ and ∇g is Lipschitz continuous with constant $L_g > 0$

  ○) h is closed, convex and proximal function prox_h can be evaluated

- Let $\{x_k\}_{k \geq 0}$ sequence generated by FISTA with either constant stepsize $\alpha = 1$ rule (or backtracking procedure)

FISTA: Convergence properties

Convergence properties:

- Minimize $f(x) = g(x) + h(x)$

  ◦) g is convex, differentiable, $\mathrm{dom}(f) = \mathbb{R}^n$ and ∇g is Lipschitz continuous with constant $L_g > 0$

  ◦) h is closed, convex and proximal function prox_h can be evaluated

- Let $\{x_k\}_{k \geq 0}$ sequence generated by FISTA with either constant stepsize $\alpha = 1$ rule (or backtracking procedure)

  Then, for any $x^\star \in X^\star$ and $k \geq 1$

$$f(x_k) - f^\star \leq \frac{2\alpha L_g \left\| x_0 - x_k^\star \right\|^2}{(k+1)^2}$$

FISTA: Convergence properties

Convergence properties:

- Minimize $f(x) = g(x) + h(x)$

  ◦) g is convex, differentiable, $\mathrm{dom}(f) = \mathbb{R}^n$ and ∇g is Lipschitz continuous with constant $L_g > 0$

  ◦) h is closed, convex and proximal function prox_h can be evaluated

- Let $\{x_k\}_{k\geq 0}$ sequence generated by FISTA with either constant stepsize $\alpha = 1$ rule (or backtracking procedure)

  Then, for any $x^\star \in X^\star$ and $k \geq 1$

$$f(x_k) - f^\star \leq \frac{2\alpha L_g \left\| x_0 - x_k^\star \right\|^2}{(k+1)^2}$$

- Thus, FISTA achieves optimal rate $O(1/k^2)$ for first-order methods, i.e., rate of $O(1/\sqrt{\varepsilon})$ –compared to $O(1/k)$ of ISTA

FISTA: Convergence properties

Convergence properties:

- Convergence of the generated sequence $(x_k)_{k\in\mathbb{N}}$ requires to consider the following rule to update the momentum factor $\tilde{t}_k$

  Let $\delta > 2$

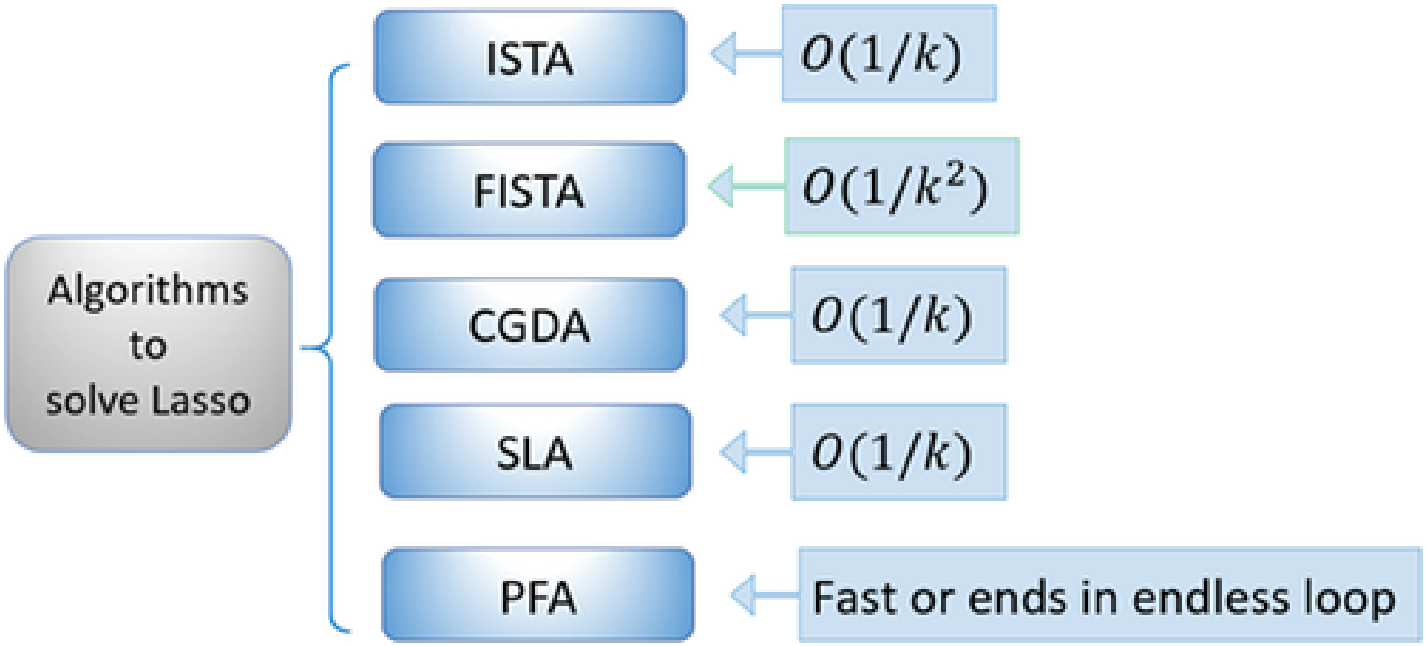
$$\tilde{t}_k = \frac{k+\delta}{\delta} \tag{21}$$

$$\frac{\tilde{t}_k - 1}{\tilde{t}_{k+1}} = \frac{k+\delta}{k+1+\delta} + \frac{-\delta}{k+1+\delta} = \frac{k}{k+1+\delta} \tag{22}$$

- This setting preserves the $O(1/k^2)$ convergence rate of the objective function.
- Note: the parameter $\frac{\tilde{t}_k - 1}{\tilde{t}_{k+1}}$ is referred to as the inertial parameter that controls the momentum term $x_k - x_{k-1}$

FISTA (Fast Iterative Shrinkage-Thresholding Algorithm)

Algorithms to optimize the objective function in Lasso [2]



Others: coordinate gradient descent algorithm (CGDA), smooth L1 algorithm (SLA), and path following algorithm (PFA)

[2] Source: Y.Zhao, A Survey of Numerical Algorithms that can Solve the Lasso Problems, 2023

FISTA: LASSO

LASSO = Least absolute shrinkage and selection operator

- A.k.a. ℓ_1-regularized least squares
- **Regression analysis method** : realizes both variable selection and regularization

$$\min_{x \in \mathbb{R}^n} f(x) = g(x) + \lambda_1 \|x\|_1 ,$$

$$\text{with } g(x) = \frac{1}{2} \|Ax - b\|_2^2 , \text{ where } A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m.$$

FISTA: LASSO

LASSO = Least absolute shrinkage and selection operator

- A.k.a. ℓ_1-regularized least squares
- **Regression analysis method** : realizes both variable selection and regularization

$$\min_{x \in \mathbb{R}^n} f(x) = g(x) + \lambda_1 \|x\|_1 ,$$

$$\text{with } g(x) = \frac{1}{2} \|Ax - b\|_2^2 , \text{ where } A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m.$$

  Parameter $\lambda > 0$ controls trade-off between model **sparsity** (induced by ℓ_1 norm: discontinuous gradient at zero) and goodness of fit

- Minimization of LASSO objective function : ℓ_1-regularization term computationally challenging due to its non-differentiability

FISTA: LASSO

LASSO = Least absolute shrinkage and selection operator

- A.k.a. ℓ_1-regularized least squares
- **Regression analysis method** : realizes both variable selection and regularization

$$\min_{x \in \mathbb{R}^n} f(x) = g(x) + \lambda_1 \|x\|_1 ,$$

$$\text{with } g(x) = \frac{1}{2} \|Ax - b\|_2^2 , \text{ where } A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m.$$

  Parameter $\lambda > 0$ controls trade-off between model **sparsity** (induced by ℓ_1 norm: discontinuous gradient at zero) and goodness of fit

- Minimization of LASSO objective function : ℓ_1-regularization term computationally challenging due to its non-differentiability

–Note: Lagrangian form of $\min_{x \in \mathbb{R}^n} \left\{ \frac{1}{2} \|Ax - b\|_2^2 \right\}$ subject to $\|x\|_1 \leq c$.

FISTA: LASSO

- Let vector $x = (1, \varepsilon) \in \mathbb{R}^2$ where $\varepsilon > 0$ is small
- The ℓ_1 and ℓ_2 norms of x given by

$$||x||_1 = 1 + \varepsilon, \quad ||x||_2^2 = 1 + \varepsilon^2$$

- Regularization: reduce magnitude of one element of x by $\delta \leq \varepsilon$.

  If $x_1 = 1$ reduced by δ, then

$$||x - (\delta, 0)||_1 = (1 - \delta) + \varepsilon, \quad ||x - (\delta, 0)||_2^2 = 1 - 2\delta + \delta^2 + \varepsilon^2$$

  If $x_2 = \varepsilon$ reduced by δ, then

$$||x - (0, \delta)||_1 = 1 + (\varepsilon - \delta), \quad ||x - (0, \delta)||_2^2 = 1 - 2\varepsilon\delta + \delta^2 + \varepsilon^2$$

  ○) Regularization by ℓ_2-**norm**: larger term x_1 results in a much greater reduction in norm compared to smaller term $x_2 \approx 0$

  ○) Regularization by ℓ_1-**norm**: reduction is the same, always equal to δ, regardless of quantity being penalized

FISTA: LASSO

Since $g(x) = \frac{1}{2}\|Ax - b\|_2^2$, where $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m$

- **ISTA** iteration:

$$x_{k+1} = T_{\lambda t_k}\big(x_k - t_k \nabla g(x_k)\big) \tag{23}$$

$$= T_{\lambda t_k}\big(x_k - t_k A^\top (Ax_k - b)\big) \tag{24}$$

FISTA: LASSO

Since $g(x) = \frac{1}{2}\|Ax - b\|_2^2$, where $A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m$

- **ISTA** iteration:

$$x_{k+1} = T_{\lambda t_k}\big(x_k - t_k \nabla g(x_k)\big) \tag{23}$$

$$= T_{\lambda t_k}\big(x_k - t_k A^\top (Ax_k - b)\big) \tag{24}$$

- **FISTA** iteration:

$$\begin{cases} x_{k+1} = T_{\lambda t_k}\big(y_k - t_k A^\top (Ay_k - b)\big) \\ \tilde{t}_{k+1} = \dfrac{1 + \sqrt{1 + 4\tilde{t}_k^2}}{2} \\ y_{k+1} = x_{k+1} + \dfrac{\tilde{t}_k - 1}{\tilde{t}_{k+1}}(x_{k+1} - x_k) \end{cases} \tag{25}$$

FISTA: Elastic-Net

Elastic-net[3]: regularized regression method that linearly combines the ℓ_1-norm penalty of Least Absolute Shrinkage and Selection Operator (LASSO) and ℓ_2-norm penalty of Ridge a.k.a. Tikhonov regularization

$$\min_{x\in\mathbb{R}^n} \frac{1}{2}\|Ax-b\|_2^2 + \frac{\lambda_2}{2}\|x\|_2^2 + \lambda_1\|x\|_1, \text{ where } A\in\mathbb{R}^{m\times n}, b\in\mathbb{R}^m.$$

[3] H. Zou, T. Hastie, Regularization and Variable Selection via the Elastic Net, Journal of the Royal Statistical Society, Series B, 67(2):301–320, 2005

FISTA: Elastic-Net

Elastic-net[3]: regularized regression method that linearly combines the ℓ_1-norm penalty of Least Absolute Shrinkage and Selection Operator (LASSO) and ℓ_2-norm penalty of Ridge a.k.a. Tikhonov regularization

$$\min_{x\in\mathbb{R}^n} \frac{1}{2}\|Ax-b\|_2^2 + \frac{\lambda_2}{2}\|x\|_2^2 + \lambda_1\|x\|_1, \text{ where } A\in\mathbb{R}^{m\times n}, b\in\mathbb{R}^m.$$

Iterations:

- For ISTA: $x_{k+1} = \text{prox}_{t_k h}\Big(x_k - t_k\big(A^\top(Ax_k - b) + \lambda_2 x_k\big)\Big)$,
  with $h(x) = \lambda_1\|x\|_1$
- For FISTA: $x_{k+1} = \text{prox}_{t_k h}\Big(y_k - t_k\big(A^\top(Ay_k - b) + \lambda_2 y_k\big)\Big)$,
  with $h(x) = \lambda_1\|x\|_1$

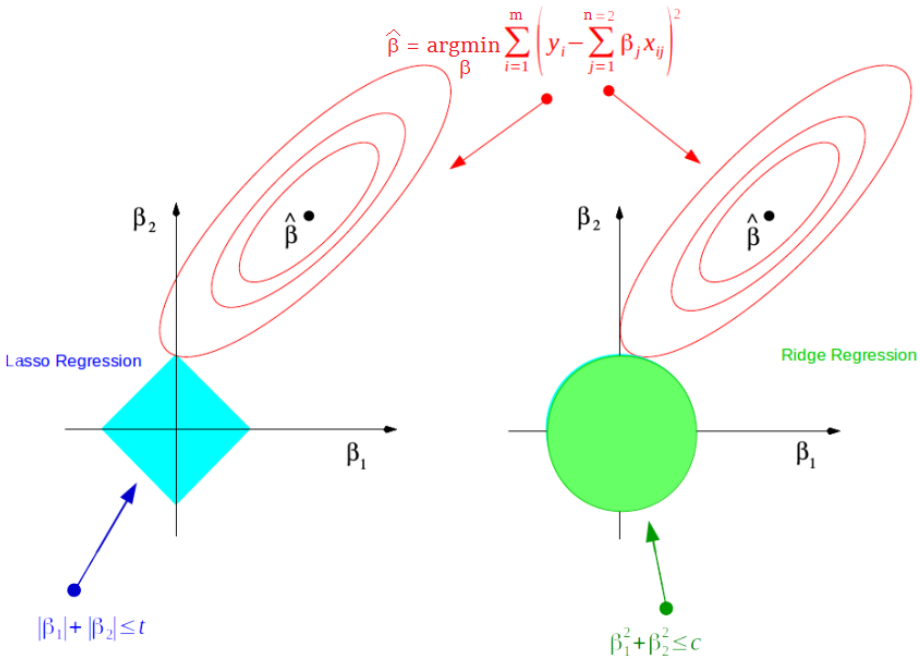
[3] H. Zou, T. Hastie, Regularization and Variable Selection via the Elastic Net, Journal of the Royal Statistical Society, Series B, 67(2):301–320, 2005

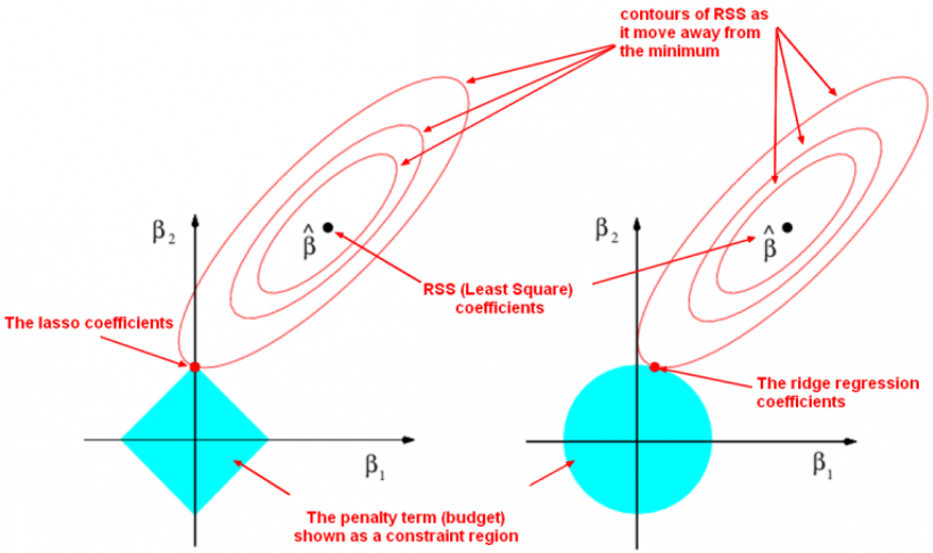
FISTA: Elastic-Net

Remember :

$$\underset{x\in\mathbb{R}^n}{\text{argmin}}\ \frac{1}{2}\left\|Ax-b\right\|^2 \Leftrightarrow \underset{\beta\in\mathbb{R}^n}{\text{argmin}}\ \left\|\mathbf{y}-\mathbf{X}\beta\right\|^2$$

$$\underset{x\in\mathbb{R}^n}{\text{argmin}}\ \sum_{i=1}^{m}\left(\left(\sum_{j=1}^{n} a_{ij}x_j\right)-b_i\right)^2 \Leftrightarrow \underset{\beta\in\mathbb{R}^n}{\text{argmin}}\ \sum_{i=1}^{m}\left(y_i-\left(\sum_{j=1}^{n} x_{ij}\beta_j\right)\right)^2$$





FISTA: Elastic-Net

- **Practice**: empirical evidence with regression algorithm (e.g., LARS) shows that the Elastic-Net does not perform satisfactorily unless very close to either ridge regression or LASS0. Thus called, **naïve** Elastic-Net

FISTA: Elastic-Net

- **Practice**: empirical evidence with regression algorithm (e.g., LARS) shows that the Elastic-Net does not perform satisfactorily unless very close to either ridge regression or LASS0. Thus called, **naïve** Elastic-Net

- **Data augmentation**

$$(A, b) \to (A^*, b^*) \text{ with } A^* = \frac{1}{\sqrt{1+\lambda_2}} \begin{bmatrix} A \\ \sqrt{2}I \end{bmatrix} \text{ and } b^* = \begin{bmatrix} y \\ 0 \end{bmatrix}$$

FISTA: Elastic-Net

- **Practice**: empirical evidence with regression algorithm (e.g., LARS) shows that the Elastic-Net does not perform satisfactorily unless very close to either ridge regression or LASS0. Thus called, **naïve** Elastic-Net

- **Data augmentation**

$$(A, b) \to (A^*, b^*) \text{ with } A^* = \frac{1}{\sqrt{1+\lambda_2}} \begin{bmatrix} A \\ \sqrt{2}I \end{bmatrix} \text{ and } b^* = \begin{bmatrix} y \\ 0 \end{bmatrix}$$

- Given data (A, b), penalty parameter (λ_1, λ_2), and augmented data (A^*, b^*), the naïve Elastic-Net solves a LASSO-type problem

$$\hat{x}^* = \underset{x}{\operatorname{argmin}} \ \frac{1}{2} \|A^* x - b^*\|_2^2 + \frac{\lambda_1}{\sqrt{1+\lambda_2}} \|x\|_1 \tag{26}$$

FISTA: Elastic-Net

- The corrected Elastic-Net estimates are defined by

$$\hat{x}_C = \sqrt{1+\lambda_2}\,\hat{x}^*$$

- Since naive elastic net estimates are given by

$$\hat{x}_N = \frac{1}{\sqrt{1+\lambda_2}}\,\hat{x}^*$$

we obtain

$$\hat{x}_C = (1+\lambda_2)\,\hat{x}_N$$

FISTA: Elastic-Net

- The corrected Elastic-Net estimates are defined by

$$\hat{x}_C = \sqrt{1+\lambda_2}\hat{x}^*$$

- Since naive elastic net estimates are given by

$$\hat{x}_N = \frac{1}{\sqrt{1+\lambda_2}}\hat{x}^*$$

  we obtain

$$\hat{x}_C = (1+\lambda_2)\,\hat{x}_N$$

- Thus, Elastic-Net coefficients are rescaled naïve Elastic-Net coefficients

FISTA: Elastic-Net

- The corrected Elastic-Net estimates are defined by

$$\hat{x}_C = \sqrt{1+\lambda_2}\hat{x}^*$$

- Since naive elastic net estimates are given by

$$\hat{x}_N = \frac{1}{\sqrt{1+\lambda_2}}\hat{x}^*$$

  we obtain

$$\hat{x}_C = (1+\lambda_2)\,\hat{x}_N$$

- Thus, Elastic-Net coefficients are rescaled naïve Elastic-Net coefficients

Scaling transformation preserves variable selection property and overcomes double shrinkage.

INFO-F524: Continuous Optimization
–Section 3: Second-order Methods–

Bernard Fortz, Dimitri Papadimitriou

Bureau: **2N3 206**

Université Libre de Bruxelles

dimitrios.papadimitriou@ulb.be

Annee Académique: 2024-2025

Context

For solving the minimization problem (P)

$$\min_{x \in \mathbb{R}^n} f(x), \tag{1}$$

Study choice of search directions used in basic updating scheme

$$x_{k+1} = x_k + t_k d_k \tag{2}$$

Assume B_k is a symmetric $n \times n$ matrix. All of search directions considered can be classified as Newton-like since they are all of the form

$$d_k = -B_k \nabla f(x_k)$$

- If $B_k = \mu_k I$, $\forall k$: scaled descent direction with scale factors μ_k
- If $B_k = (\nabla^2 f(x_k))^{-1}$: second-order Newton method
- If $B_k \approx (\nabla^2 f(x_k))^{-1}$: quasi-Newton method

Newton Method for Solving equations: finding roots

- Newton's method: iterative scheme to solve nonlinear equations

$$f(x) = 0 \tag{3}$$

where $f : \mathbb{R}^n \to \mathbb{R}^n$ is vector valued $f(x) = (f_1(x) \ldots, f_n(x))$ with $f_i(x) = f_i(x_1, \ldots, x_n)$, $\forall i = 1, \ldots, n$ and continuously differentiable

Newton Method for Solving equations: finding roots

- Newton's method: iterative scheme to solve nonlinear equations

$$f(x) = 0 \tag{3}$$

where $f : \mathbb{R}^n \to \mathbb{R}^n$ is vector valued $f(x) = (f_1(x) \ldots, f_n(x))$ with $f_i(x) = f_i(x_1, \ldots, x_n)$, $\forall i = 1, \ldots, n$ and continuously differentiable

- For **minimization** problem (P): compute stationary points, i.e., points at which $f'(x) = 0$, where f' denotes the $n \times n$ Jacobian matrix at x

$$(f'(x))_{ij} = \frac{\partial f_i}{\partial x_j}(x), \;\; i, j = 1, \ldots, n \tag{4}$$

Matrix notation

$$f'(x) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \nabla f_1(x)^T \\ \vdots \\ \nabla f_n(x)^T \end{bmatrix}$$

Newton Method for Solving equations: finding roots

Newton method for finding roots of

$$f(x) = 0 \tag{5}$$

- Let $x_0 \in \mathbb{R}^n$ approximate solution to (5)
- **Goal**: improve on approximate solution x_0

  If $\tilde{x}$ is a solution to (5), then

$$0 = f(\tilde{x}) = f(x_0) + f_0'(x_0)(\tilde{x} - x_0) + o\,\|x - x_0\|$$

  Thus, **if** x_0 close to $\tilde{x}$, **then** solution closer to linearized system

$$0 = f(x_0) + f'(x_0)(x - x_0) \tag{6}$$

Newton Method for Solving equations: finding roots

Newton method for finding roots of $f(x) = 0$

- Assume $f'(x_k)^{-1}$ exists (more generally, Jacobian matrix invertible)
- Iterative scheme: **Newton iteration**

  Set $x_0 = x_k$ and $x = x_{k+1}$ in (6)

$$x_{k+1} := x_k - [f'(x_k)]^{-1} f(x_k) \tag{7}$$

Newton Method for Solving equations: finding roots

Newton method for finding roots of $f(x) = 0$

- Assume $f'(x_k)^{-1}$ exists (more generally, Jacobian matrix invertible)
- Iterative scheme: **Newton iteration**

  Set $x_0 = x_k$ and $x = x_{k+1}$ in (6)

$$x_{k+1} := x_k - [f'(x_k)]^{-1} f(x_k) \tag{7}$$

- Associated direction d_k: **Newton direction**

  Since $x_{k+1} = x_k + (t_k) d_k$

$$d_k := -[f'(x_k)]^{-1} f(x_k) \tag{8}$$

Newton Method for Minimization

Newton method for minimization: solve optimization problem (P)

$$\min_{x \in \mathbb{R}^n} f(x) \tag{9}$$

Assumptions

- Function $f : \mathbb{R}^n \to \mathbb{R}$ twice differentiable
- B_k approximation to inverse of Hessian matrix $\nabla^2 f(x_k)$: $\boxed{B_k \approx (\nabla^2 f(x_k))^{-1}}$

Newton Method for Minimization

Newton method for minimization: solve optimization problem (P)

$$\min_{x \in \mathbb{R}^n} f(x) \tag{9}$$

Assumptions

- Function $f : \mathbb{R}^n \to \mathbb{R}$ twice differentiable
- B_k approximation to inverse of Hessian matrix $\nabla^2 f(x_k)$: $\boxed{B_k \approx (\nabla^2 f(x_k))^{-1}}$

Newton-like iteration:

$$x_{k+1} = x_k + t_k d_k \tag{10}$$

Newton Method for Minimization

- Minimizing quadratic approximation of (twice continuously differentiable) function f sequentially around current iterate x_k

Newton Method for Minimization

- Minimizing quadratic approximation of (twice continuously differentiable) function f sequentially around current iterate x_k
- Taylor expansion to second order and minimize resulting quadratic function

$$f(x) \approx f(x_k) + \nabla f(x_k)^T (x - x_k) + \frac{1}{2}(x - x_k)^T \nabla^2 f(x_k)(x - x_k)$$

Newton Method for Minimization

- Minimizing quadratic approximation of (twice continuously differentiable) function f sequentially around current iterate x_k
- Taylor expansion to second order and minimize resulting quadratic function

$$f(x) \approx f(x_k) + \nabla f(x_k)^T (x - x_k) + \frac{1}{2}(x - x_k)^T \nabla^2 f(x_k)(x - x_k)$$

- First order necessary condition $\nabla f(x^\star) = 0$

  Thus,

$$\begin{aligned} \nabla f(x) = 0 \;\Leftrightarrow \nabla f(x_k) + \nabla^2 f(x_k) x - \nabla^2 f(x_k) x_k = 0 \\ x = x_k - (\nabla^2 f(x_k))^{-1} \nabla f(x_k) \end{aligned}$$

Newton Method for Minimization

- Minimizing quadratic approximation of (twice continuously differentiable) function f sequentially around current iterate x_k
- Taylor expansion to second order and minimize resulting quadratic function

$$f(x) \approx f(x_k) + \nabla f(x_k)^T (x - x_k) + \frac{1}{2}(x - x_k)^T \nabla^2 f(x_k)(x - x_k)$$

- First order necessary condition $\nabla f(x^\star) = 0$
  Thus,

$$\begin{aligned} \nabla f(x) = 0 \;\Leftrightarrow \nabla f(x_k) + \nabla^2 f(x_k) x - \nabla^2 f(x_k) x_k = 0 \\ x = x_k - (\nabla^2 f(x_k))^{-1} \nabla f(x_k) \end{aligned}$$

- Consequently, Newton's method iteration

$$x_{k+1} = x_k + t_k d_k = x_k - (\nabla^2 f(x_k))^{-1} \nabla f(x_k)$$

Newton Method for Minimization

Newton method for minimization: solve optimization problem (P)

$$\min_{x \in \mathbb{R}^n} f(x) \tag{11}$$

Assumptions

- Function f twice differentiable
- B_k approximation to inverse of Hessian matrix $\nabla^2 f(x_k)$: $\boxed{B_k \approx (\nabla^2 f(x_k))^{-1}}$

Newton Method for Minimization

Newton method for minimization: solve optimization problem (P)

$$\min_{x \in \mathbb{R}^n} f(x) \tag{11}$$

Assumptions

- Function f twice differentiable
- B_k approximation to inverse of Hessian matrix $\nabla^2 f(x_k)$: $\boxed{B_k \approx (\nabla^2 f(x_k))^{-1}}$

Newton-like iteration:

$$\begin{aligned} x_{k+1} &= x_k + d_k \\ &= x_k - B_k \nabla f(x_k) \end{aligned} \tag{12}$$

PURE Newton method: step size $t = t_k = 1 \ \forall k$

Newton Method for Minimization

Invariance under linear transformation

- Consider a linear inversible transformation $y = Ax$ with nonsingular $A \in \mathbb{R}^{n \times n}$
- Let $g(y) = f(A^{-1}y) = f(x)$
- Newton steps on $g(y)$

$$
\begin{aligned}
y_{k+1} &= y_k - \left(\nabla^2 g(y_k)\right)^{-1} \nabla g(y_k) \\
&= Ax_k - \left(A^{-T} \nabla^2 f(x_k) A^{-1}\right)^{-1} A^{-T} \nabla f(x_k) \\
&= Ax_k - A\left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k) \\
&= A\left(x_k - \left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k)\right) \\
&= Ax_{k+1}
\end{aligned}
$$

$\Rightarrow$ Progress independent of problem scaling

- Property not verified by gradient descent method

Newton Method for Minimization

Damped Newton method

- **Step size**: backtracking line search with $0 < c \leq 1/2$ and $0 < \alpha < 1$
  - ◦) At each iteration, backtracking starts with $t = 1$
  - ◦) Contract $t = \alpha\, t$ while

$$f\left(x_k - t\left(\nabla^2 f(x_k)\right)^{-1}\nabla f(x_k)\right) > f(x) - c\, t\, \nabla f(x_k)^T\left(\left(\nabla^2 f(x_k)\right)^{-1}\nabla f(x_k)\right)$$

- **Newton update**

$$x_{k+1} = x_k - t_k\left(\nabla^2 f(x_k)\right)^{-1}\nabla f(x_k) \tag{13}$$

Newton Method for Minimization

Damped Newton method

- **Step size**: backtracking line search with $0 < c \leq 1/2$ and $0 < \alpha < 1$
  - ◦) At each iteration, backtracking starts with $t = 1$
  - ◦) Contract $t = \alpha\, t$ while

$$f\left(x_k - t\left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k)\right) > f(x) - c\, t\, \nabla f(x_k)^T \left(\left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k)\right)$$

- **Newton update**

$$x_{k+1} = x_k - t_k \left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k) \tag{13}$$

- **Stopping criterion**: based on gradient norm, i.e., $\|\nabla f\| \leq \varepsilon$ or Newton decrement

$$f(x_{k+1}) - \hat{f}(x^\star) = \frac{1}{2}\, \nabla f(x_k)^T\, \nabla^2 f(x_k)^{-1}\, \nabla f(x_k)$$

Newton Method for Minimization

Local rate of convergence: follows from properties of Hessian $\nabla^2 f(x^\star)$ around stationary point

- For all x in neighborhood of solution point $x^\star$ –such that $\nabla^2 f(x^\star)$ is positive definite– the Hessian $\nabla^2 f(x)$ is also positive definite

Newton Method for Minimization

Local rate of convergence: follows from properties of Hessian $\nabla^2 f(x^\star)$ around stationary point

- For all x in neighborhood of solution point $x^\star$ –such that $\nabla^2 f(x^\star)$ is positive definite– the Hessian $\nabla^2 f(x)$ is also positive definite
- Then, Newton method well defined in this region and converges quadratically (provided that t_k are eventually always 1)

Newton Method for Minimization

Theorem

Suppose

- f is twice differentiable
- Hessian $\nabla^2 f(x)$ Lipschitz continuous in neighborhood of a solution $x^\star$ at which **second-order sufficient conditions** satisfied

Consider $x_{k+1} = x_k + d_k$, where d_k is given by (12)

Newton Method for Minimization

Theorem

Suppose

- f is twice differentiable
- Hessian $\nabla^2 f(x)$ Lipschitz continuous in neighborhood of a solution $x^\star$ at which **second-order sufficient conditions** satisfied

Consider $x_{k+1} = x_k + d_k$, where d_k is given by (12)

Then

(i) if starting point x_0 sufficiently close to $x^\star$, then sequence of iterates converges to $x^\star$

(ii) rate of convergence of sequence $\{x_k\}$ is quadratic

(iii) sequence of gradient norms $\|\nabla f(x_k)\|$ converges quadratically to zero

Newton Method for Minimization

Theorem

Suppose

- f is twice differentiable
- Hessian $\nabla^2 f(x)$ Lipschitz continuous in neighborhood of a solution $x^\star$ at which **second-order sufficient conditions** satisfied

Consider $x_{k+1} = x_k + d_k$, where d_k is given by (12)

Then

(i) if starting point x_0 sufficiently close to $x^\star$, then sequence of iterates converges to $x^\star$

(ii) rate of convergence of sequence $\{x_k\}$ is quadratic

(iii) sequence of gradient norms $\|\nabla f(x_k)\|$ converges quadratically to zero

Note: Newton iteration not always a descent direction since $\nabla^2 f(x)$ not necessarily positive definite.

Gauss-Newton (GN) Method

- **Gauss–Newton method** : solve nonlinear least squares (NLLS)/nonlinear regression problems

Gauss-Newton (GN) Method

- **Gauss–Newton method** : solve nonlinear least squares (NLLS)/nonlinear regression problems
- Let $f : \mathbb{R}^n \to \mathbb{R}^m$ be a differentiable function

$$x \in \mathbb{R}^n \to f(x) = (f_1(x), f_2(x), \ldots, f_m(x))$$

Gauss-Newton (GN) Method

- **Gauss–Newton method** : solve nonlinear least squares (NLLS)/nonlinear regression problems
- Let $f : \mathbb{R}^n \to \mathbb{R}^m$ be a differentiable function

$$x \in \mathbb{R}^n \to f(x) = (f_1(x), f_2(x), \ldots, f_m(x))$$

- **Nonlinear least squares (NLLS)** problem:

$$\min_{x \in \mathbb{R}^n} F(x) = \|f(x)\|_2^2 = \sum_{i=1}^{m} f_i(x)^2 \tag{14}$$

Gauss-Newton (GN) Method

Let $f'(x) = J_f(x)$ denote the $m \times n$ Jacobian matrix of f at x:

$$f'(x) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \nabla f_1(x)^T \\ \vdots \\ \nabla f_m(x)^T \end{bmatrix}$$

Gauss-Newton (GN) Method

Let $f'(x) = J_f(x)$ denote the $m \times n$ Jacobian matrix of f at x:

$$f'(x) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \nabla f_1(x)^T \\ \vdots \\ \nabla f_m(x)^T \end{bmatrix}$$

Linearization of f around $\tilde{x}$

$$f(x) \approx f(\tilde{x}) + f'(\tilde{x})(x - \tilde{x}) \tag{15}$$

Gauss-Newton (GN) Method

- **Gradient** of NLLS objective function $F(x) = \|f(x)\|_2^2$ is

$$\nabla F(x) = 2f'(x)^T f(x) \tag{16}$$

- **Hessian** of NLLS objective function $F(x) = \|f(x)\|_2^2$ can be approximated by

$$\nabla^2 F(x) \approx 2f'(x)^T f'(x) \tag{17}$$

Each element of $\nabla^2 F(x)$ given by

$$H_{jk} \approx 2 \sum_{i=1}^{m} f'_{ij} f'_{jk}.$$

<u>Note</u>: justified when both f'_{ij} and f'_{ik} small

Gauss-Newton (GN) Method : Algorithm

- Start at some initial point $x_0 \in \mathrm{dom}(f)$
- Repeat for $k = 1, 2$, etc.
  - Linearize f around x_k

$$f(x) \approx f(x_k) + f'(x_k)(x - x_k)$$

  - Substitute affine approximation for f in $\|f(x)\|_2^2$ to obtain (linear LS):

$$\min_{x \in \mathbb{R}^n} \|f(x_k) + f'(x_k)(x - x_k)\|_2^2$$

If $f'(x_k)$ has full column rank, then set x_{k+1} as solution of this linear LS

$$\begin{aligned} x_{k+1} &= x_k - (f'(x_k)^T f'(x_k))^{-1} f'(x_k)^T f(x_k) \\ &= x_k - f'(x_k)^\dagger f(x_k) \end{aligned}$$

where $f'(x)^\dagger = J_f(\mathbf{x})^\dagger$: Moore-Penrose (or pseudo-) inverse of $J_f(\mathbf{x})$[1]

- End

[1] The Moore-Penrose or pseudo-inverse $A^\dagger$ of the matrix A is $A^\dagger = (A^T A)^{-1} A^T$

Gauss-Newton (GN) Method

- Gauss–Newton step $v_k = x_{k+1} - x_k$: solution of **linear least squares** (LLS)

$$\min_v \|f(x_k) + f'(x_k)v\|_2^2$$

- Thus, to improve convergence:
  add line search step to find t_k
  and then update

$$x_{k+1} = x_k + t_k v_k$$

Comparison: Newton vs. Gauss-Newton

Consider NLLS problem

$$\min_{x \in \mathbb{R}^n} F(x) = \|f(x)\|_2^2 = \sum_{i=1}^{m} f_i(x)^2$$

Newton step at x_k :

$$\begin{aligned} v_k = x_{k+1} - x_k &= -\left(\nabla^2 F(x_k)\right)^{-1} \nabla F(x_k) \\ &= -\left(f'(x_k)^T f'(x_k) + \sum_{i=1}^{m} f_i(x_k) \nabla^2 f_i(x_k)\right)^{-1} f'(x_k)^T f(x_k) \end{aligned} \tag{18}$$

- Computing Newton step v_k requires second-order derivatives of f
- Not always a descent direction: $\nabla^2 F$ is not necessarily positive definite
- Fast convergence near local minimum $x^\star$

Comparison: Newton vs. Gauss-Newton

Gauss-Newton step at x_k : let $H_k = 2f'(x_k)^T f'(x_k)$ then

$$\begin{aligned} v_k = x_{k+1} - x_k &= -(f'(x_k)^T f'(x_k))^{-1} f'(x_k)^T f(x_k) \\ &= -H_k^{-1} \nabla F(x_k) \end{aligned} \tag{19}$$

- Computing GN step v_k does NOT require second-order derivatives of f
- Defines descent direction when $H_k = 2f'(x_k)^T f'(x_k) \succ 0$

  Note: being the multiplication of a matrix by its transpose, this term is always **positive semi-definite**, i.e., $H_k = 2f'(x_k)^T f'(x_k) \succeq 0$
- Achieves convergence of Newton method near local minimum $x^\star$ if

$$\sum_{i=1}^{m} f_i(x^\star) \nabla^2 f_i(x^\star) \ll 1$$

  Indeed, when this condition is verified: (18) $\approx$ (19).

GN: Example of Nonlinear Least Squares

Nonlinear Least Squares (NLLS) problem:

$$\min_{x \in \mathbb{R}^n} F(x) = \|f(x)\|_2^2 = \sum_{i=1}^{m} f_i(x)^2 \tag{14}$$

where $f : \mathbb{R}^n \to \mathbb{R}^m : x \to (f_1(x), \ldots, f_m(x))$ differentiable (and typically nonlinear)

Note: the NLLS can be viewed as

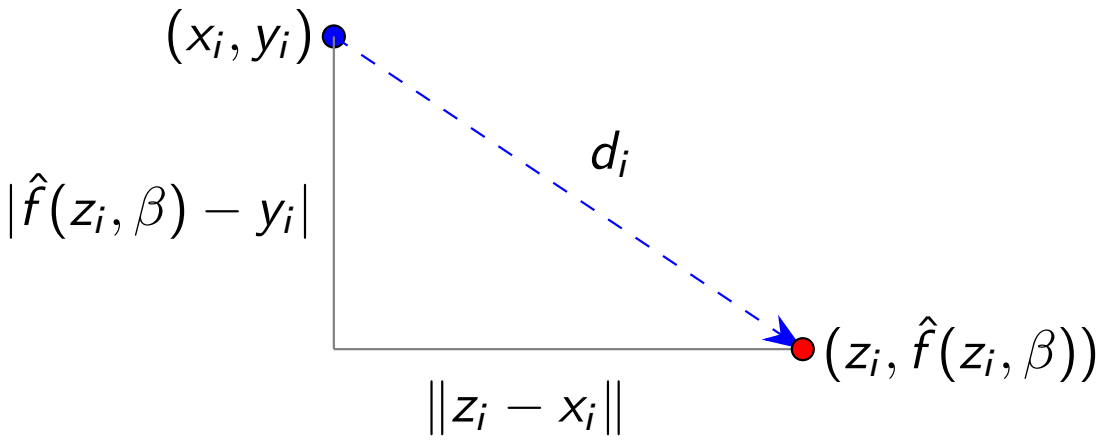
$$\min_{x \in \mathbb{R}^n} h(f(x))$$

where $f : \mathbb{R}^n \to \mathbb{R}^m$ is differentiable and $h : \mathbb{R}^m \to \mathbb{R}$ is convex

GN: Example of Nonlinear Least Squares

Nonlinear Least Squares: parameterized model fitting

- **Data**: n data points $\{(x_i, y_i)\}_{i=1}^n$
- **Variables**: model $\hat{f}(z, \beta)$ depends on p parameters $\beta = (\beta_1, \beta_2, \ldots, \beta_p)$ and n points $\{z_i\}_{i=1}^n$



- **Objective function**: squared distance d_i^2 of data point (x_i, y_i) to graph of the model function $\hat{f}$ at z_i, i.e., at $(z_i, \hat{f}(z_i, \beta))$

$$d_i^2 = \left(\hat{f}(z_i, \beta) - y_i\right)^2 + \|z_i - x_i\|_2^2$$

GN: Example of Nonlinear Least Squares

- **Data**: n data points $\{(x_i, y_i)\}_{i=1}^n$
- **Variables**: model $\hat{f}(z, \beta)$, p parameters $\beta = (\beta_1, \beta_2, \ldots, \beta_p)$ and $\{z_i\}_{i=1}^n$
- **Objective function**: squared distance d_i^2 of data point (x_i, y_i) to graph of the function $\hat{f}$ at z_i, i.e., at $(z_i, \hat{f}(z_i, \beta))$

$$d_i^2 = \left(\hat{f}(z_i, \beta) - y_i\right)^2 + \|z_i - x_i\|_2^2$$

- **Goal**: minimize the mean squared distance

  $\equiv$ minimize the **sum** of squared distance d_i^2 over z_i (i.e., squared distance of (x_i, y_i) to the graph) and over β (parameters vector)

$$\min_{z,\beta} \sum_{i=1}^n d_i^2 = \sum_{i=1}^n \left(\hat{f}(z_i, \beta) - y_i\right)^2 + \|z_i - x_i\|_2^2 \tag{20}$$

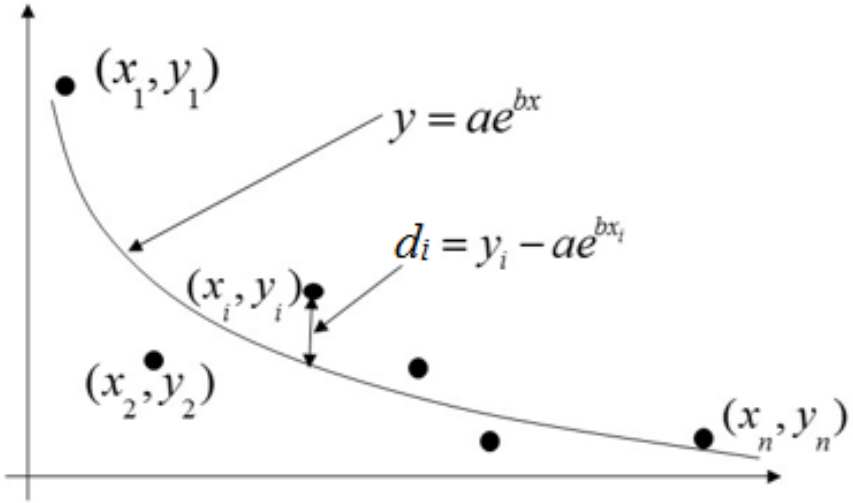
GN: Nonlinear LS $\equiv$ (Regularized) Nonlinear Regression

Nonlinear Regression : in (20), $z_i = x_i \ \forall i$.

Thus, minimize the **sum** of squared distance d_i^2 over parameters vector β (only)

$$\boxed{\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{n} d_i^2 = \sum_{i=1}^{n} \left(\hat{f}(x_i, \beta) - y_i\right)^2} \tag{21}$$

Exemple: exponential model fitting



GN: Nonlinear LS $\equiv$ (Regularized) Nonlinear Regression

Regularized Nonlinear Regression :

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{n} d_i^2 = \sum_{i=1}^{n} \left(\hat{f}(x_i, \beta) - y_i\right)^2$$

$$\rightarrow \hat{\beta} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{n} \left(\hat{f}(x_i, \beta) - y_i\right)^2 + \alpha \left\|\beta\right\|_2^2 + (1-\alpha) \left\|\beta\right\|_1, \ \alpha = \frac{\lambda_2}{\lambda_1 + \lambda_2}$$

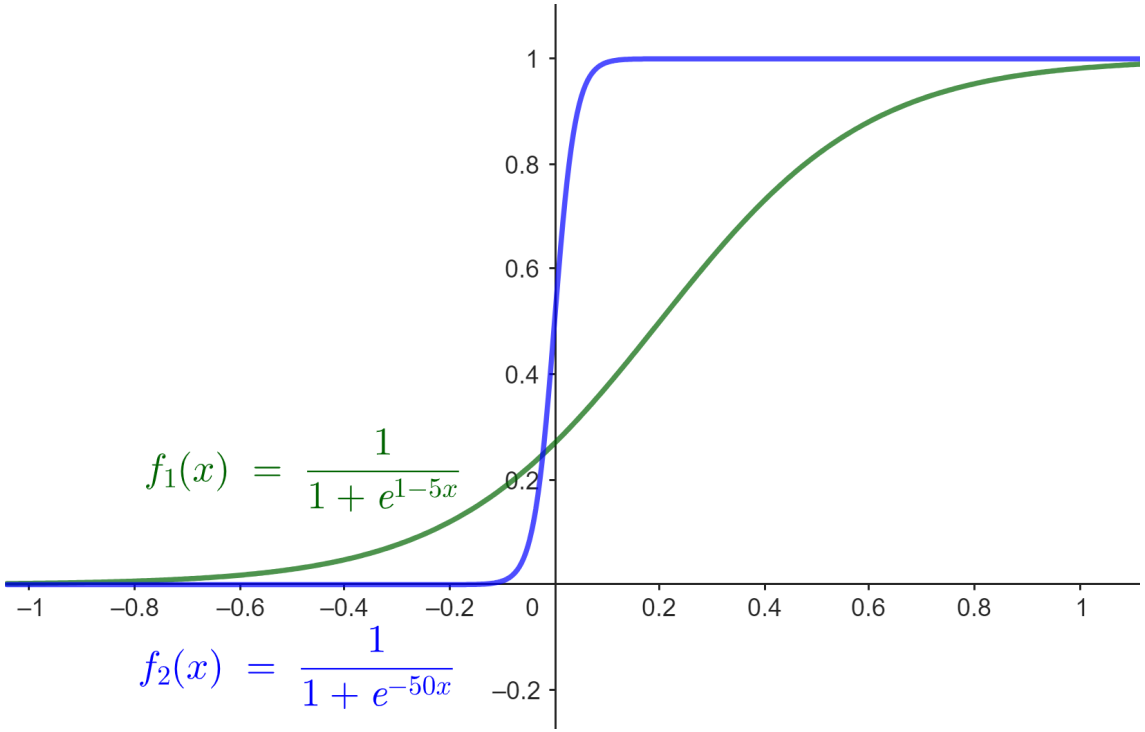
Rôle of the (elastic net) regularization term : $\alpha \left\|\beta\right\|_2^2 + (1-\alpha) \left\|\beta\right\|_1$

GN: Nonlinear LS $\equiv$ (Regularized) Nonlinear Regression

Examples: Logistic model (S-curve) $\rightarrow$ Logistic regression

$$\hat{f}(x, \beta) := \sigma(x; \beta) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}} \tag{22}$$

where β_0 is the intercept and β_1 the rate parameter



When $\beta \gg 1$ performs as a (probabilistic) linear classifier

GN: Nonlinear LS $\equiv$ (Regularized) Nonlinear Regression

The logistic regression leads to a linear classifier

$$P(y = 1 \,|\, x, \beta) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

$$P(y = 0 \,|\, x, \beta) = 1 - P(y = 1 \,|\, x, \beta) = \frac{e^{-(\beta_0 + \beta_1 x)}}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

- Probability $P(y = 1 \,|\, x, \beta) > \frac{1}{2} \;\Rightarrow y = 1$ (classification); $y = 0$ (otherwise)
- Classification boundary is at:

$$P(y = 1 \,|\, x, \beta) = \frac{1}{2}$$

$$\rightarrow \quad \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}} = \frac{1}{2}$$

$\Rightarrow$ Classification boundary $\beta_0 + \beta_1 x = 0$ is an (hyper)plane

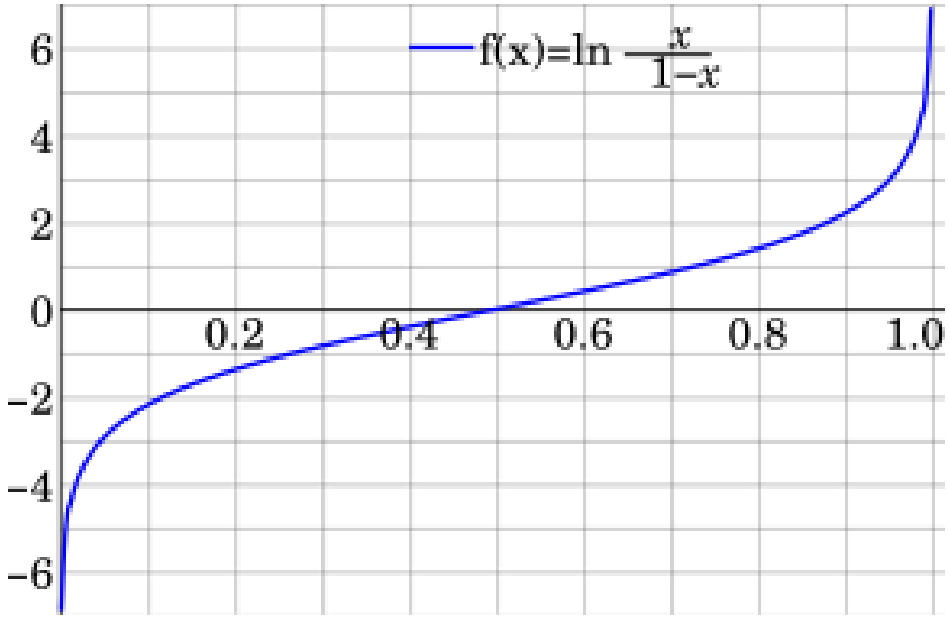
GN: Nonlinear LS $\equiv$ (Regularized) Nonlinear Regression

Logit function (inverse of logistic function):

$$logit(P) = \sigma^{-1}(P) := \log \frac{P}{1-P}$$

$$\text{Since } P(y=1 \,|\, x, \beta) = \frac{1}{1+e^{-(\beta_0+\beta_1 x)}} \Rightarrow \frac{P(y=1)}{1-P(y=1)} = exp(\beta_0 + \beta_1 x)$$

$$\rightarrow logit(P) := \log \frac{P(y=1)}{1-P(y=1)} = \beta_0 + \beta_1 x$$



Thus, **statistical** logit function maps probability values $[0,1] \rightarrow \mathbb{R} \cup \{-\infty, +\infty\}$

GN: Nonlinear LS $\equiv$ (Regularized) Nonlinear Regression

Examples

- Power model : $\hat{f}(x, \beta) \sim x^{\beta}$
- Saturating model : $\hat{f}(x, \beta) = \dfrac{poly(x, \beta)}{1 + poly(x, \beta)}$

  $poly(x, \beta)$: polynomial of order $m < n$ (usually $\ll$ when n large)

Levenberg–Marquardt

Levenberg–Marquardt (LM) algorithm a.k.a damped least-squares (DLS) method

Levenberg–Marquardt

Levenberg–Marquardt (LM) algorithm a.k.a damped least-squares (DLS) method

- **Main application**: solve nonlinear least squares problems (least squares curve / model fitting)

Levenberg–Marquardt

Levenberg–Marquardt (LM) algorithm a.k.a damped least-squares (DLS) method

- **Main application**: solve nonlinear least squares problems (least squares curve / model fitting)
- **LM algorithm**: interpolates between Gauss–Newton (GN) algorithm and gradient descent (GD) method

Levenberg–Marquardt

Levenberg–Marquardt (LM) algorithm a.k.a damped least-squares (DLS) method

- **Main application**: solve nonlinear least squares problems (least squares curve / model fitting)
- **LM algorithm**: interpolates between Gauss–Newton (GN) algorithm and gradient descent (GD) method
- **Main properties**: more robust than GN algorithm (even when starting very far off the minimum)

Levenberg–Marquardt

LM algorithm addresses two difficulties of GN method

- How to update x_k when columns of $f'(x_k)$ are **linearly dependent**
- What to do when the Gauss–Newton update **does not reduce** $\|f(x)\|_2^2$

Levenberg–Marquardt

Working principle

- Starting from the LS problem

$$\min_{x \in \mathbb{R}^n} F(x) = \|f(x)\|_2^2$$

- Substitution of affine approximation for f around x_k

$$f(x) \approx f(x_k) + f'(x_k)(x - x_k)$$

  in the LS problem yields

$$\min_{x \in \mathbb{R}^n} \|f(x_k) + f'(x_k)(x - x_k)\|_2^2$$

- LM method adds regularization term on x

Levenberg–Marquardt

Working principle

- Starting from the LS problem

$$\min_{x \in \mathbb{R}^n} F(x) = \|f(x)\|_2^2$$

- Substitution of affine approximation for f around x_k

$$f(x) \approx f(x_k) + f'(x_k)(x - x_k)$$

in the LS problem yields

$$\min_{x \in \mathbb{R}^n} \|f(x_k) + f'(x_k)(x - x_k)\|_2^2$$

- LM method adds regularization term on x

$\Rightarrow$ LM algorithm computes x_{k+1} by solving regularized LS problem

$$\boxed{\min_{x \in \mathbb{R}^n} \|f(x_k) + f'(x_k)(x - x_k)\|_2^2 + \lambda_k \|x - x_k\|_2^2} \tag{23}$$

Levenberg–Marquardt

Role of the **regularization term**

- Forces x to be close to x_k when the local approximation is accurate

Levenberg–Marquardt

Role of the **regularization term**

- Forces x to be close to x_k when the local approximation is accurate
- With regularization parameter $\lambda_k > 0$: minimization problem always unique solution (no rank condition on $f'(x_k)$)

Levenberg–Marquardt

Role of the **regularization term**

- Forces x to be close to x_k when the local approximation is accurate
- With regularization parameter $\lambda_k > 0$: minimization problem always unique solution (no rank condition on $f'(x_k)$)
- Can also be interpreted as a proximal point update with convexified cost function

Levenberg–Marquardt

Role of the **regularization term**

- Forces x to be close to x_k when the local approximation is accurate
- With regularization parameter $\lambda_k > 0$: minimization problem always unique solution (no rank condition on $f'(x_k)$)
- Can also be interpreted as a proximal point update with convexified cost function

Note: LM method can be derived directly from Newton method (no linearization). Let $H_k = \nabla^2 f(x_k)$, LM update can be formulated as

$$x_{k+1} = x_k - (H_k + \lambda_k I)^{-1} \nabla f(x_k)$$

Levenberg–Marquardt

Levenberg–Marquardt:

$$\min_{x \in \mathbb{R}^n} \|f(x_k) + f'(x_k)(x - x_k)\|_2^2 + \lambda_k \|x - x_k\|_2^2 \tag{24}$$

Levenberg–Marquardt

Levenberg–Marquardt:

$$\min_{x \in \mathbb{R}^n} \|f(x_k) + f'(x_k)(x - x_k)\|_2^2 + \lambda_k \|x - x_k\|_2^2 \tag{24}$$

LM iteration (update)

$$x_{k+1} = x_k - (f'(x_k)^T f'(x_k) + \lambda_k I)^{-1} f'(x_k)^T f(x_k) \tag{25}$$

Levenberg–Marquardt

Levenberg–Marquardt:

$$\min_{x \in \mathbb{R}^n} \|f(x_k) + f'(x_k)(x - x_k)\|_2^2 + \lambda_k \|x - x_k\|_2^2 \tag{24}$$

LM iteration (update)

$$x_{k+1} = x_k - (f'(x_k)^T f'(x_k) + \lambda_k I)^{-1} f'(x_k)^T f(x_k) \tag{25}$$

LM step $v_k = x_{k+1} - x_k$

$$\begin{aligned} v_k &= -(f'(x_k)^T f'(x_k) + \lambda_k I)^{-1} f'(x_k)^T f(x_k) \\ &= -\frac{1}{2}(f'(x_k)^T f'(x_k) + \lambda_k I)^{-1} \nabla F(x_k) \end{aligned} \tag{26}$$

Levenberg–Marquardt: Particular cases

- For $\lambda_k = 0$: LM reduces to Gauss–Newton step (if defined)

$$\begin{aligned} v_k &= -(f'(x_k)^T f'(x_k) + \lambda_k I)^{-1} f'(x_k)^T f(x_k) \\ &= -\frac{1}{2}(f'(x_k)^T f'(x_k))^{-1} \nabla F(x_k) \end{aligned}$$

- For $\lambda_k \gg 1$: term $\lambda_k I$ takes over in denominator

$$\begin{aligned} v_k &= -(f'(x_k)^T f'(x_k) + \lambda_k I)^{-1} f'(x_k)^T f(x_k) \\ &\approx -\frac{1}{2}(\lambda_k I)^{-1} \nabla F(x_k) \end{aligned}$$

Levenberg–Marquardt: Particular cases

- For $\lambda_k = 0$: LM reduces to Gauss–Newton step (if defined)

$$\begin{aligned} v_k &= -(f'(x_k)^T f'(x_k) + \lambda_k I)^{-1} f'(x_k)^T f(x_k) \\ &= -\frac{1}{2}(f'(x_k)^T f'(x_k))^{-1} \nabla F(x_k) \end{aligned}$$

- For $\lambda_k \gg 1$: term $\lambda_k I$ takes over in denominator

$$\begin{aligned} v_k &= -(f'(x_k)^T f'(x_k) + \lambda_k I)^{-1} f'(x_k)^T f(x_k) \\ &\approx -\frac{1}{2}(\lambda_k I)^{-1} \nabla F(x_k) \end{aligned}$$

If regularization parameter (damping factor) $\lambda_k \gg f'(x_k)^T f'(x_k)$
Then inverting $f'(x_k)^T f'(x_k) + \lambda_K I$ not necessary –update well-approximated by (small) gradient step

Levenberg–Marquardt: Particular cases

When $\lambda_k \gg f'(x_k)^T f'(x_k)$, LM step

$$v_k \approx -\frac{1}{2}(\lambda_k I)^{-1} \nabla F(x_k)$$

$\rightarrow$ (approximated) Hessian matrix not used

Levenberg–Marquardt: Particular cases

When $\lambda_k \gg f'(x_k)^T f'(x_k)$, LM step

$$v_k \approx -\frac{1}{2}(\lambda_k I)^{-1} \nabla F(x_k)$$

$\rightarrow$ (approximated) Hessian matrix not used

- Scale each gradient component according to the curvature
- Then larger movement along direction of smaller gradient
- **How ?**

Levenberg–Marquardt: Particular cases

When $\lambda_k \gg f'(x_k)^T f'(x_k)$, LM step

$$v_k \approx -\frac{1}{2}(\lambda_k I)^{-1} \nabla F(x_k)$$

$\rightarrow$ (approximated) Hessian matrix not used

- Scale each gradient component according to the curvature
- Then larger movement along direction of smaller gradient
- **How ?**

  Identity matrix I replaced by diagonal elements of matrix $f'(x_k)^T f'(x_k)$

$$v_k = -\left(f'(x_k)^T f'(x_k) + \lambda_k \mathrm{diag}(f'(x_k)^T f'(x_k))\right)^{-1} f'(x_k)^T f(x_k)$$

$\Rightarrow$ Technique to avoid slow convergence in direction of small gradient

Levenberg–Marquardt: damping factor

Regularization parameter: several strategies can be adopted

- At iteration k, compute the solution v_k of

$$\min \|f(x_k) + f'(x_k)v\|_2^2 + \lambda_k \|v\|_2^2$$

Levenberg–Marquardt: damping factor

Regularization parameter: several strategies can be adopted

- At iteration k, compute the solution v_k of

$$\min \|f(x_k) + f'(x_k)v\|_2^2 + \lambda_k \|v\|_2^2$$

- Update rule for regularization parameter λ
  - **if** $\|f(x_k + v_k)\|_2^2 \leq \|f(x_k)\|_2^2$
  - **then** set $x_{k+1} = x_k + v_k$ and decrease λ_k
  - **otherwise** set $x_{k+1} = x_k$ and increase λ_k

Levenberg–Marquardt: algorithm

Levenberg-Marquardt algorithm:

`input:`

choose initial point $x_0 \in \mathrm{dom}(f)$

λ_0 and constants γ_1, γ_2 such that $0 < \gamma_1 < 1 < \gamma_2$

`repeat:`

1. Evaluate $f(x_k)$ and Jacobian $J_f(x_k) = f'(x_k)$
2. Compute (estimator) of solution for regularized least squares problem

$$\hat{x} = x_k - (J_f(x_k)^T J_f(x_k) + \lambda_k I)^{-1} J_f(x_k)^T f(x_k)$$

3. Update:

`if` $\|f(\hat{x})\|_2^2 \leq \|f(x_k)\|_2^2$

`then set` $x_{k+1} = \hat{x}$ and $\lambda_{k+1} = \gamma_1 \lambda_k$ (decrease)

`otherwise set` $x_{k+1} = x_k$ and $\lambda_{k+1} = \gamma_2 \lambda_k$ (increase)

`until:` $\nabla F(x_k) = 2 J_f(x_k)^T f(x_k) \leq \varepsilon$

Levenberg–Marquardt: algorithm

- LM method: **heuristic**, works well in practice for moderately sized models

Levenberg–Marquardt: algorithm

- LM method: **heuristic**, works well in practice for moderately sized models
- Main disadvantage: matrix inversion as part of the update

$$\hat{x} = x_k - (J_f(x_k)^T J_f(x_k) + \lambda_k I)^{-1} J_f(x_k)^T f(x_k)$$

Levenberg–Marquardt: algorithm

- LM method: **heuristic**, works well in practice for moderately sized models
- Main disadvantage: matrix inversion as part of the update

$$\hat{x} = x_k - (J_f(x_k)^T J_f(x_k) + \lambda_k I)^{-1} J_f(x_k)^T f(x_k)$$

- Matrix inversion usually implemented using pseudo-inverse methods such as SVD: cost of update may become prohibitive after model size increases to a few thousand parameters

Levenberg–Marquardt: algorithm

- LM method: **heuristic**, works well in practice for moderately sized models
- Main disadvantage: matrix inversion as part of the update

$$\hat{x} = x_k - (J_f(x_k)^T J_f(x_k) + \lambda_k I)^{-1} J_f(x_k)^T f(x_k)$$

- Matrix inversion usually implemented using pseudo-inverse methods such as SVD: cost of update may become prohibitive after model size increases to a few thousand parameters
- For moderately sized models (few hundred parameters): faster and more robust than GD method while avoiding limitations of GN

INFO-F524: Continuous Optimization
–Section 3: Quasi-Newton methods–

Bernard Fortz, Dimitri Papadimitriou

Bureau: **2N3 206**

Université Libre de Bruxelles

dimitrios.papadimitriou@ulb.be

Annee Académique: 2024-2025

Motivation

When minimizing a twice continuously differentiable function f that is convex, **Newton method** provides

- advantages: fast convergence, robustness, affine invariance
- disadvantages: requires second derivatives and solution of linear equation which can be too expensive for large scale applications

Motivation

When minimizing a twice continuously differentiable function f that is convex, **Newton method** provides

- advantages: fast convergence, robustness, affine invariance
- disadvantages: requires second derivatives and solution of linear equation which can be too expensive for large scale applications

Quasi-Newton method

Method which aims at preserving advantages while mitigating computational cost

Quasi-Newton methods: secant equation

Assume positive definite matrix H_k: approximation of Hessian matrix at x_k

$$\boxed{\mathrm{H}_k \approx \nabla^2 f(x_k)} \tag{1}$$

H_k chosen to

- avoid computation of second-order derivatives
- simplify computation of search direction

Quasi-Newton methods: secant equation

Assume positive definite matrix H_k: approximation of Hessian matrix at x_k

$$\boxed{\mathrm{H}_k \approx \nabla^2 f(x_k)} \tag{1}$$

H_k chosen to

- avoid computation of second-order derivatives
- simplify computation of search direction

Quasi-Newton method **iteration**

$$x_{k+1} = x_k + t_k d_k \tag{2}$$

$$= x_k - t_k H_k^{-1} \nabla f(x_k) \tag{3}$$

Quasi-Newton **direction**

$$d_k = -H_k^{-1} \nabla f(x_k) \tag{4}$$

Algorithm

Quasi-Newton algorithm:

`input`: choose initial point $x_0 \in dom(f)$

`repeat`:

1. Determine **step**/search direction (starting at x_k) $d_k := -H_k^{-1}\nabla f(x_k)$
2. Compute **step size** $t_k \in \mathbb{R}^+$: fixed $t_k = t$

   or via inexact (e.g. backtracking) line search
3. Update: $x_{k+1} = x_k + t_k d_k = x_k - t_k H_k^{-1}\nabla f(x_k)$
4. Compute: H_{k+1}

`until`: stopping criterion is satisfied

Algorithm

Quasi-Newton (QN) algorithm:

`input`: choose initial point $x_0 \in \mathrm{dom}(f)$

`repeat`:

1. Determine **step**/search direction (starting at x_k) $d_k := -H_k^{-1}\nabla f(x_k)$
2. Compute **step size** $t_k \in \mathbb{R}^+$: fixed $t_k = t$

   or via inexact (e.g. backtracking) line search
3. Update: $x_{k+1} = x_k + t_k d_k = x_k - t_k H_k^{-1}\nabla f(x_k)$
4. Compute: H_{k+1}

`until`: stopping criterion is satisfied

QN Algorithm

- **Technique 1**: approximate the Hessian of f at x_k, i.e., $\nabla^2 f(x_k)$ by H_k and then compute its inverse H_k^{-1}

QN Algorithm

- **Technique 1**: approximate the Hessian of f at x_k, i.e., $\nabla^2 f(x_k)$ by H_k and then compute its inverse H_k^{-1}
- **Technique 2**: directly approximate the inverse of the Hessian of f to avoid any matrix inversion $B_k \approx (\nabla^2 f(x_k))^{-1}$

QN Algorithm

- **Technique 1**: approximate the Hessian of f at x_k, i.e., $\nabla^2 f(x_k)$ by H_k and then compute its inverse H_k^{-1}
- **Technique 2**: directly approximate the inverse of the Hessian of f to avoid any matrix inversion $B_k \approx (\nabla^2 f(x_k))^{-1}$

  Principle: low-rank update on a matrix corresponds to a low rank update on its inverse

QN Algorithm

- **Technique 1**: approximate the Hessian of f at x_k, i.e., $\nabla^2 f(x_k)$ by H_k and then compute its inverse H_k^{-1}
- **Technique 2**: directly approximate the inverse of the Hessian of f to avoid any matrix inversion $B_k \approx (\nabla^2 f(x_k))^{-1}$

  Principle: low-rank update on a matrix corresponds to a low rank update on its inverse

  Inverse of the Hessian can be approximated (instead of Hessian itself) by using the Sherman–Morrison formula

$$\left(A + uv^T\right)^{-1} = A^{-1} - \frac{A^1 uv^T A^{-1}}{1 + v^T A^{-1} u}$$

  Given an invertible square matrix A and outer product uv^{T} of vectors u and v, computes an updated matrix inverse of rank-1 perturbation of $\left(A + uv^T\right)$

QN Algorithm

H_k: approximation of $\nabla^2 f(x_k)$; thus, with displacement s_k

$$\nabla f(x_k + s_k) \approx \nabla f(x_k) + H_k s_k$$

QN Algorithm

H_k: approximation of $\nabla^2 f(x_k)$; thus, with displacement s_k

$$\nabla f(x_k + s_k) \approx \nabla f(x_k) + H_k s_k$$

Once $x_{k+1} = x_k + s_k$ is computed, **how to obtain** H_{k+1} ?

Idea: since H_k already contains some information, make some suitable update

- Requirement for H_{k+1}

$$\nabla f(x_{k+1}) = \nabla f(x_k) + H_{k+1} s_k$$

- or equivalently

$$H_{k+1} s_k = \nabla f(x_{k+1}) - \nabla f(x_k)$$

QN Algorithm

Secant equation imposes

$$y_k = H_{k+1} s_k \iff B_{k+1} y_k = s_k \tag{5}$$

$$\begin{cases} y_k & := \nabla f(x_{k+1}) - \nabla f(x_k) \\ s_k & := x_{k+1} - x_k = t_k d_k \quad : \text{ QN step} \end{cases} \tag{6}$$

Interpretation: by considering a quadratic approximation of f around x_{k+1}

$$\hat{f}(x) \approx f(x_{k+1}) + \nabla f(x_{k+1})^T (x - x_{k+1}) + \frac{1}{2}(x - x_{k+1})^T H_{k+1} (x - x_{k+1})$$

By construction $\nabla \hat{f}(x_{k+1}) = \nabla f(x_{k+1})$.

QN Algorithm

Secant equation imposes

$$y_k = H_{k+1} s_k \iff B_{k+1} y_k = s_k \tag{5}$$

$$\begin{cases} y_k & := \nabla f(x_{k+1}) - \nabla f(x_k) \\ s_k & := x_{k+1} - x_k = t_k d_k \quad : \text{ QN step} \end{cases} \tag{6}$$

Interpretation: by considering a quadratic approximation of f around x_{k+1}

$$\hat{f}(x) \approx f(x_{k+1}) + \nabla f(x_{k+1})^T (x - x_{k+1}) + \frac{1}{2}(x - x_{k+1})^T H_{k+1}(x - x_{k+1})$$

The secant condition (5) implies also

$$\begin{aligned} \nabla_x \hat{f}(x_k) &= \nabla f(x_{k+1}) + H_{k+1}(x_k - x_{k+1}) \\ &= \nabla f(x_{k+1}) - H_{k+1} s_k \\ &= \nabla f(x_{k+1}) - \nabla f(x_{k+1}) + \nabla f(x_k) \\ &= \nabla f(x_k) \end{aligned}$$

QN Algorithm

- Given displacement s_k and change of gradients y_k, the **secant equation** requires that (symmetric positive definite matrix) H_{k+1} maps s_k into y_k

$$\boxed{y_k = H_{k+1} s_k}$$

QN Algorithm

- Given displacement s_k and change of gradients y_k, the **secant equation** requires that (symmetric positive definite matrix) H_{k+1} maps s_k into y_k

$$\boxed{y_k = H_{k+1} s_k}$$

- **Condition**: s_k and y_k satisfy the curvature condition

$$s_k^T y_k > 0 \tag{7}$$

If curvature condition is verified, H_k update preserves its positive definiteness

QN Algorithm

- Given displacement s_k and change of gradients y_k, the **secant equation** requires that (symmetric positive definite matrix) H_{k+1} maps s_k into y_k

$$\boxed{y_k = H_{k+1} s_k}$$

- **Condition**: s_k and y_k satisfy the curvature condition

$$s_k^T y_k > 0 \tag{7}$$

  If curvature condition is verified, H_k update preserves its positive definiteness

- This condition ensures that $d_k = -H_k^{-1} \nabla f(x_k)$ is a descent direction

QN Algorithm

- Given displacement s_k and change of gradients y_k, the **secant equation** requires that (symmetric positive definite matrix) H_{k+1} maps s_k into y_k

$$\boxed{y_k = H_{k+1} s_k}$$

- **Condition**: s_k and y_k satisfy the curvature condition

$$s_k^T y_k > 0 \tag{7}$$

  If curvature condition is verified, H_k update preserves its positive definiteness

- This condition ensures that $d_k = -H_k^{-1} \nabla f(x_k)$ is a descent direction

Quasi-Newton methods differ based on update rules: $H_k \to H_{k+1}$ $(B_k \to B_{k+1})$

Algorithm: SR1

Symmetric Rank 1 (SR1)

- QN method to update the second-order derivative (Hessian) based on the derivatives (gradients) calculated at two points
- Generalization to the secant method for a multidimensional problem
- SR1 update maintains symmetry of H_k (B_k) matrix but does not guarantee that update be positive definite

Algorithm: SR1

Symmetric Rank 1 (SR1)

- QN method to update the second-order derivative (Hessian) based on the derivatives (gradients) calculated at two points
- Generalization to the secant method for a multidimensional problem
- SR1 update maintains symmetry of H_k (B_k) matrix but does not guarantee that update be positive definite

Algorithm: SR1

SR1 update

$$H_{k+1} = H_k + auu^T$$

where

$$\left.\begin{array}{l} a = -1 \text{ or } +1 \\ \textbf{and } u \end{array}\right\} \text{ chosen to satisfy secant equation } y = H_{k+1}s$$

Algorithm: SR1

SR1 update

$$H_{k+1} = H_k + auu^T$$

where

$$\left.\begin{array}{l} a = -1 \text{ or } +1 \\ \textbf{and } u \end{array}\right\} \text{ chosen to satisfy secant equation } y = H_{k+1}s$$

Hence, by substituting the SR1 update into secant equation

$$\begin{aligned} y &= (H_k + auu^T)s \\ &= H_k s + auu^T s \\ &= H_k s + (au^T s)u \quad (*) \end{aligned}$$

Algorithm: SR1

- By substituting SR1 update into secant equation

$$y = H_k s + (au^T s)u \quad (*)$$

Thus,

$$y - H_k s = (au^T s)u$$

Algorithm: SR1

- By substituting SR1 update into secant equation

$$y = H_k s + (au^T s)u \quad (*)$$

Thus,

$$y - H_k s = (au^T s)u$$

- Since $(au^T s)$ is scalar, u must be a multiple of $y - H_k s$
That is, $u = b(y - H_k s)$ for some scalar b

Algorithm: SR1

- By substituting SR1 update into secant equation

$$y = H_k s + (au^T s)u \quad (*)$$

Thus,

$$y - H_k s = (au^T s)u$$

- Since $(au^T s)$ is scalar, u must be a multiple of $y - H_k s$
  That is, $u = b(y - H_k s)$ for some scalar b
- By substituting $u = b(y - H_k s)$ into $(*)$

$$(y - H_k s) = ab^2 \left[s^T (y - H_k s)\right] (y - H_k s)$$

which is satisfied **iff** parameters a and b

$$\begin{cases} a &= \text{sign}[s^T (y - H_k s)] \\ b &= \pm \left|s^T (y - H_k s)\right|^{-1/2} \end{cases}$$

Algorithm: SR1

Therefore, the only SR1 updating formula that satisfies the secant equation is

$$H_{k+1} = H_k + \frac{(y_k - H_k s_k)(y_k - H_k s_k)^T}{(y_k - H_k s_k)^T s_k} \tag{8}$$

Algorithm: SR1

Therefore, the only SR1 updating formula that satisfies the secant equation is

$$H_{k+1} = H_k + \frac{(y_k - H_k s_k)(y_k - H_k s_k)^T}{(y_k - H_k s_k)^T s_k} \tag{8}$$

By applying the Sherman–Morrison formula, updating formula for the inverse Hessian approximation B_k

$$B_{k+1} = B_k + \frac{(s_k - H_k y_k)(s_k - H_k y_k)^T}{(s_k - H_k y_k)^T y_k} \tag{9}$$

Algorithm: SR1

SR1 update formula:

$$H_{k+1} = H_k + \frac{(y_k - H_k s_k)(y_k - H_k s_k)^T}{(y_k - H_k s_k)^T s_k} \tag{8}$$

Shortcoming of SR1 update: does not preserve positive definiteness of H_k

- Even if H_k is positive definite, H_{k+1} not necessarily

Algorithm: SR1

SR1 update formula:

$$H_{k+1} = H_k + \frac{(y_k - H_k s_k)(y_k - H_k s_k)^T}{(y_k - H_k s_k)^T s_k}$$

Algorithm: SR1

SR1 update formula:

$$H_{k+1} = H_k + \frac{(y_k - H_k s_k)(y_k - H_k s_k)^T}{(y_k - H_k s_k)^T s_k}$$

Denominator in (8): three cases can occur

- If $(y_k - H_k s_k)^T s_k \neq 0$: unique SR1 updating formula satisfying the secant equation, given by (8)

Algorithm: SR1

SR1 update formula:

$$H_{k+1} = H_k + \frac{(y_k - H_k s_k)(y_k - H_k s_k)^T}{(y_k - H_k s_k)^T s_k}$$

Denominator in (8): three cases can occur

- If $(y_k - H_k s_k)^T s_k \neq 0$: unique SR1 updating formula satisfying the secant equation, given by (8)
- If $y_k = H_k s_k$: only rank-one update formula satisfying the secant equation is $H_{k+1} = H_k$

Algorithm: SR1

SR1 update formula:

$$H_{k+1} = H_k + \frac{(y_k - H_k s_k)(y_k - H_k s_k)^T}{(y_k - H_k s_k)^T s_k}$$

Denominator in (8): three cases can occur

- If $(y_k - H_k s_k)^T s_k \neq 0$: unique SR1 updating formula satisfying the secant equation, given by (8)
- If $y_k = H_k s_k$: only rank-one update formula satisfying the secant equation is $H_{k+1} = H_k$
- If $y_k \neq H_k s_k$ and $(y_k - H_k s_k)^T s_k = 0$: no rank-one update formula satisfying the secant equation

  Numerical instabilities and even breakdown of SR1 method can occur

Algorithm: SR1

- **Consequence**: **rank-one** update does not provide enough freedom to develop a matrix with all the desired characteristics, and that **rank-two** correction required

Algorithm: SR1

- **Consequence**: **rank-one** update does not provide enough freedom to develop a matrix with all the desired characteristics, and that **rank-two** correction required

- Note: SR1 may perform well for constrained problem ($y_k^T s_k > 0$ condition not verified) or by skipping the update if denominator is small

Algorithm: BFGS

The Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm

- Determines the descent direction by preconditioning the gradient with curvature information.

Algorithm: BFGS

The Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm

- Determines the descent direction by preconditioning the gradient with curvature information.
- Technique: gradually improving an approximation to the Hessian matrix of the objective function

Algorithm: BFGS

The Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm

- Determines the descent direction by preconditioning the gradient with curvature information.
- Technique: gradually improving an approximation to the Hessian matrix of the objective function
- Approximation of Hessian matrix: obtained only from gradient evaluations (or approximate gradient evaluations) via a generalized secant method

Algorithm: BFGS

- If the curvature condition $s_k^T y_k > 0$ is verified then the BFGS update preserves positive definiteness of H_k
- This property ensures ensures that $d_k = -H_k^{-1} \nabla f(x_k)$ is a descent direction
- **Key property** of BFGS updates
  - **If** H_k is positive-definite
    and step size t_k satisfies (strong/weak) Wolfe conditions
  - **Then** H_{k+1} is also positive-definite

Algorithm: BFGS

BFGS update: rank two update of the form

$$H_{k+1} = H_k + auu^T + bvv^T$$

Algorithm: BFGS

BFGS update: rank two update of the form

$$H_{k+1} = H_k + auu^T + bvv^T$$

By substitution in secant equation $y = H_{k+1}s$

$$\begin{aligned} y &= (H_k + auu^T + bvv^T)s \\ y - H_k s &= (au^T s)u + (bv^T s)v \end{aligned}$$

Algorithm: BFGS

Using

$$y - H_k s = (au^T s)u + (bv^T s)v \ (*)$$

By setting $u = y$, $v = Hs$, we obtain

$$\begin{cases} a &= \dfrac{1}{y^T s} \\ b &= -\dfrac{1}{s^T H_k s} \end{cases}$$

Algorithm: BFGS

Using

$$y - H_k s = (au^T s)u + (bv^T s)v \ (*)$$

By setting $u = y$, $v = Hs$, we obtain

$$\begin{cases} a &= \dfrac{1}{y^T s} \\ b &= -\dfrac{1}{s^T H_k s} \end{cases}$$

Finally, substitute the expressions of a, b in $(*)$ to obtain the approximation H_{k+1} of the Hessian matrix $\nabla^2 f(x_k)$: **Broyden- Fletcher-Goldfarb-Shanno** update:

$$\boxed{H_{k+1} = H_k - \frac{H_k s_k s_k^T H_k}{s_k^T H_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}} \tag{10}$$

Algorithm: BFGS

- BFGS algorithm requires initial approximation H_0 to be computed at initialization
- **No generic procedure or formula** to find H_0 that works well in all cases
- **Techniques**

  ◦ Use specific information about the problem, for instance by setting H_0 to (inverse of an) approximate Hessian calculated by finite differences at x_0

  ◦ Otherwise, set H_0 to be the identity matrix $H_0 = I$, or a multiple of the identity matrix $H_0 = \gamma I$, where γ chosen to reflect scaling of variables

Algorithm: BFGS

BFGS algorithm:

`input`: choose initial point $x_0 \in \mathrm{dom}(f)$

`init`: compute Hessian approximation H_0

`repeat`:

1. Compute search direction (vector starting at x_k) $d_k := -H_k^{-1}\nabla f(x_k)$
2. Set $x_{k+1} = x_k + t_k d_k$

   t_k computed from a line search procedure

   t_k satisfies Wolfe conditions to ensure stable updates
3. Set $s_k = x_{k+1} - x_k$ and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$
4. Compute H_{k+1} using (10)

`until`: stopping criterion is satisfied ($|\nabla f(x_k)\| \leq \varepsilon$)

Algorithm: BFGS

- To avoid any matrix inversion: approximate the inverse of the Hessian instead of the Hessian matrix itself

Algorithm: BFGS

- To avoid any matrix inversion: approximate the inverse of the Hessian instead of the Hessian matrix itself
- First step of BFGS algorithm carried out using inverse of matrix H_k, i.e.

$$B_k = H_k^{-1}$$

Algorithm: BFGS

- To avoid any matrix inversion: approximate the inverse of the Hessian instead of the Hessian matrix itself
- First step of BFGS algorithm carried out using inverse of matrix H_k, i.e.

$$B_k = H_k^{-1}$$

- Expression of B_k (inverse update) obtained by applying Sherman–Morrison formula to Step 4 of BFGS algorithm

$$B_{k+1} = \left(I - \frac{s_k y_k^T}{y_k^T s_k}\right) B_k \left(I - \frac{y_k s_k^T}{y_k^T s_k}\right) + \frac{s_k s_k^T}{y_k^T s_k} \tag{11}$$

Algorithm: BFGS

BFGS algorithm:

`input`: choose initial point $x_0 \in \mathrm{dom}(f)$

`init`: compute Hessian approximation $B_0 = H_0^{-1}$

`repeat`:

1. Compute search direction (vector starting at x_k) $d_k := -B_k \nabla f(x_k)$
2. Set $x_{k+1} = x_k + t_k d_k$

   t_k computed from a line search procedure

   t_k satisfies Wolfe conditions to ensure stable updates
3. Set $s_k = x_{k+1} - x_k$ and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$
4. Compute B_{k+1} using (11)

`until`: stopping criterion is satisfied ($\|\nabla f(x_k)\| \leq \varepsilon$)

Algorithm: BFGS properties

Property: positive definiteness

- BFGS update preserves **positive definiteness**:
  - H_k positive definite $\Rightarrow$ H_{k+1} positive definite

Algorithm: BFGS properties

Property: positive definiteness

- BFGS update preserves **positive definiteness**:
  - H_k positive definite $\Rightarrow$ H_{k+1} positive definite
  - Equivalently: B_k positive definite $\Rightarrow$ B_{k+1} positive definite

Algorithm: BFGS properties

Property: **self-correcting property** of BFGS update formula (10) and (11)

- **If**

  i) H_k incorrectly estimates curvature in objective function

  and ii) this bad estimate slows down the iteration
- **Then** the Hessian approximation tends to correct itself within a few BFGS iterations

Algorithm: BFGS properties

Property: **self-correcting property** of BFGS update formula (10) and (11)

- **If**

  i) H_k incorrectly estimates curvature in objective function

  and ii) this bad estimate slows down the iteration
- **Then** the Hessian approximation tends to correct itself within a few BFGS iterations
- Hold only when adequate line search is performed

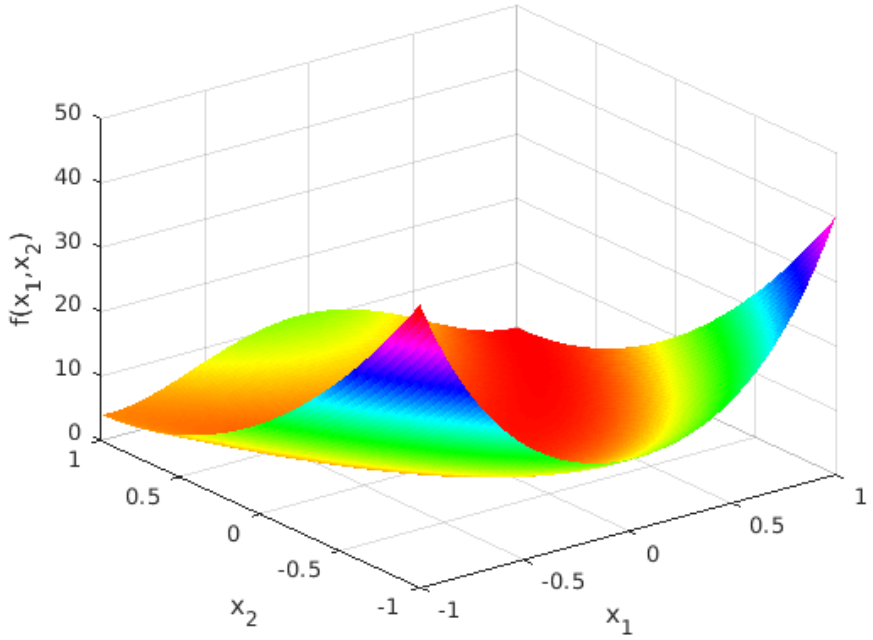
  ◦ Wolfe line search conditions ensure that gradients are sampled at points that allow second-order model to capture appropriate curvature information

Algorithm: BFGS Example

Rosenbrock function

$$\min_{x_1,x_2} f(x_1,x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \tag{12}$$

optimal solution $x^\star = (x_1, x_2) = (1, 1)$, optimal value: $f(x^\star) = 0$.



$\Rightarrow$ Dependence on starting point x_0 (valley vs. top)

Algorithm: BFGS Example

```
example_minFunc_ex1(10,20,25,50,100)
x1 = 0.0000, x2 = 0.0000 (starting point)
x1 = 1.0000, x2 = 1.0000 (optimal solution)
----------------------------------------
Result after 10 evaluations of limited-memory solvers on 2D rosenbrock:
x1 = 0.2345, x2 = 0.0399 (minFunc with steepest descent)
x1 = 0.5335, x2 = 0.2670 (minFunc with limited-memory BFGS - default)
----------------------------------------
Result after 20 evaluations of limited-memory solvers on 2D rosenbrock:
x1 = 0.3160, x2 = 0.0999 (minFunc with steepest descent)
x1 = 0.9458, x2 = 0.8915 (minFunc with limited-memory BFGS - default)
----------------------------------------
Result after 25 evaluations of limited-memory solvers on 2D rosenbrock:
x1 = 0.3654, x2 = 0.1230 (minFunc with steepest descent)
x1 = 1.0000, x2 = 1.0000 (minFunc with limited-memory BFGS - default)
----------------------------------------
Result after 50 evaluations of limited-memory solvers on 2D rosenbrock:
x1 = 0.4659, x2 = 0.2125 (minFunc with steepest descent)
x1 = 1.0000, x2 = 1.0000 (minFunc with limited-memory BFGS - default)
----------------------------------------
Result after 100 evaluations of limited-memory solvers on 2D rosenbrock:
x1 = 0.5845, x2 = 0.3375 (minFunc with steepest descent)
x1 = 1.0000, x2 = 1.0000 (minFunc with limited-memory BFGS - default)
----------------------------------------
```

Algorithm: BFGS Example

```
example_minFunc_ex1(10,20,25,50,100)
x1 = -3.0000, x2 = 5.0000 (starting point)
x1 = 1.0000, x2 = 1.0000 (optimal solution)
----------------------------------------
Result after 10 evaluations of limited-memory solvers on 2D rosenbrock:
x1 = 0.2345, x2 = 0.0399 (minFunc with steepest descent)
x1 = 0.5335, x2 = 0.2670 (minFunc with limited-memory BFGS - default)
----------------------------------------
Result after 20 evaluations of limited-memory solvers on 2D rosenbrock:
x1 = 0.3160, x2 = 0.0999 (minFunc with steepest descent)
x1 = 0.9458, x2 = 0.8915 (minFunc with limited-memory BFGS - default)
----------------------------------------
Result after 25 evaluations of limited-memory solvers on 2D rosenbrock:
x1 = 0.3654, x2 = 0.1230 (minFunc with steepest descent)
x1 = 1.0000, x2 = 1.0000 (minFunc with limited-memory BFGS - default)
----------------------------------------
Result after 50 evaluations of limited-memory solvers on 2D rosenbrock:
x1 = 0.4659, x2 = 0.2125 (minFunc with steepest descent)
x1 = 1.0000, x2 = 1.0000 (minFunc with limited-memory BFGS - default)
----------------------------------------
Result after 100 evaluations of limited-memory solvers on 2D rosenbrock:
x1 = 0.5845, x2 = 0.3375 (minFunc with steepest descent)
x1 = 1.0000, x2 = 1.0000 (minFunc with limited-memory BFGS - default)
----------------------------------------
```

Algorithm: BFGS Example

L-BFGS result

```
----------------------------------------
Result after 25 evaluations on 2D rosenbrock:
x1 = 0.3654, x2 = 0.1230 (minFunc with steepest descent)
x1 = 1.0000, x2 = 1.0000 (minFunc with --BFGS - default)
----------------------------------------
```

To be compared with GD + backtracking

Result (8): $x_0 = [-3, 5], \varepsilon = 10^{-5}, t0 = 0.9, c = 0.5, \text{alpha} = 0.99$

```
iter = 1 t = 0.000132 norm grad = 469.847756 fval = 34.298387
...
iter = 245 t = 0.001008 norm grad = 0.000012 fval = 0.000000
iter = 246 t = 0.144493 norm grad = 0.000143 fval = 0.000000
iter = 247 t = 0.000998 norm grad = 0.000012 fval = 0.000000
iter = 248 t = 0.900000 norm grad = 0.000015 fval = 0.000000
iter = 249 t = 0.001296 norm grad = 0.000008 fval = 0.000000
ans = [0.999991724727687,0.999983407525798]
```

Algorithm: Computational Complexity

Computational Complexity

- Updates of BFGS curvature matrix do not require matrix inversion

Algorithm: Computational Complexity

Computational Complexity

- Updates of BFGS curvature matrix do not require matrix inversion
- **Implication**: each iteration can be performed at a computational cost of $O(n^2)$ arithmetic operations (plus cost of objective function and gradient evaluations)

Algorithm: Computational Complexity

Computational Complexity

- Updates of BFGS curvature matrix do not require matrix inversion
- **Implication**: each iteration can be performed at a computational cost of $O(n^2)$ arithmetic operations (plus cost of objective function and gradient evaluations)

Comparison

- Compared to Newton's method: no $O(n^3)$ operations such as solving linear systems or matrix–matrix operations

Algorithm: DFP

Fundamental idea of Quasi-Newton updating rule

- Instead of recomputing approximate Hessians (or inverse Hessians) from scratch at every iteration
- Apply modification that combines i) most recently observed information about objective function with ii) existing knowledge embedded in current Hessian approximation

Algorithm: DFP

Fundamental idea of Quasi-Newton updating rule

- Instead of recomputing approximate Hessians (or inverse Hessians) from scratch at every iteration
- Apply modification that combines i) most recently observed information about objective function with ii) existing knowledge embedded in current Hessian approximation

BFGS method:

- Perform rank-two update for H_k
- Then apply the Shermann-Morrison-Woodbury formula to find how to apply a rank-two update to the inverse B_k

Algorithm: DFP

Fundamental idea of Quasi-Newton updating rule

- Instead of recomputing approximate Hessians (or inverse Hessians) from scratch at every iteration
- Apply modification that combines i) most recently observed information about objective function with ii) existing knowledge embedded in current Hessian approximation

BFGS method:

- Perform rank-two update for H_k
- Then apply the Shermann-Morrison-Woodbury formula to find how to apply a rank-two update to the inverse B_k

DFP method: instead consider applying rank-two update to B_k first

$$B_{k+1} = B_k + auu^T + bvv^T$$

Algorithm: DFP

Davidson–Fletcher–Powell (DFP) formula:

- Finds solution to (multidimensional) secant equation that

  i) is closest to current estimate

  and **ii**) satisfies curvature condition

Algorithm: DFP

Davidson–Fletcher–Powell (DFP) formula:

- Finds solution to (multidimensional) secant equation that

  i) is closest to current estimate

  and **ii**) satisfies curvature condition

- Such update maintains i) symmetry and ii) positive definiteness of (approximate) Hessian matrix

Algorithm: DFP

Davidson–Fletcher–Powell (DFP) formula:

- Finds solution to (multidimensional) secant equation that

  i) is closest to current estimate

  and **ii**) satisfies curvature condition

- Such update maintains i) symmetry and ii) positive definiteness of (approximate) Hessian matrix

Note: first quasi-Newton method to generalize secant equation-based method to a multidimensional problem

Algorithm: DFP

Rank-two update applied to B_k

$$B_{k+1} = B_k + auu^T + bvv^T$$

Algorithm: DFP

Rank-two update applied to B_k

$$B_{k+1} = B_k + auu^T + bvv^T$$

According to secant condition $s = B_{k+1}y$, we have

$$B_k y_k + auu^T y_k + bvv^T y_k = s_k$$

Algorithm: DFP

Rank-two update applied to B_k

$$B_{k+1} = B_k + auu^T + bvv^T$$

According to secant condition $s = B_{k+1}y$, we have

$$B_k y_k + auu^T y_k + bvv^T y_k = s_k$$

An obvious choice for u and v is

$$\begin{cases} u = s_k \rightarrow au^T y_k = 1 \text{ with } a = \dfrac{1}{u^T y_k} \\ v = B_k y_k \rightarrow bv^T y_k = -1 \text{ with } b = \dfrac{-1}{v^T y_k} \end{cases}$$

Then, we get

$$B_{k+1} = B_k - \frac{B_k y_k y_k^T B_k}{y_k^T B_k y_k} + \frac{s_k s_k^T}{y_k^T s_k} \tag{13}$$

Algorithm: DFP

DFP update:

$$B_{k+1} = B_k - \frac{B_k y_k y_k^T B_k}{y_k^T B_k y_k} + \frac{s_k s_k^T}{y_k^T s_k} \tag{14}$$

Algorithm: DFP

DFP update:

$$B_{k+1} = B_k - \frac{B_k y_k y_k^T B_k}{y_k^T B_k y_k} + \frac{s_k s_k^T}{y_k^T s_k} \tag{14}$$

The equations (14) mirror equations (10) and (11): one can obtain the one from the other by interchanging $s \leftrightarrow y$, $B \leftrightarrow H$

Algorithm: DFP

Properties of David-Fletcher-Powell (DFP) updates:

- **Computational complexity**: of order $O(n^2)$
  - Complexity of BFGS is the same as DFP

Algorithm: DFP

Properties of David-Fletcher-Powell (DFP) updates:

- **Computational complexity**: of order $O(n^2)$
  - Complexity of BFGS is the same as DFP
- Preserves **positive definiteness** of B_k

Algorithm: DFP

Properties of David-Fletcher-Powell (DFP) updates:

- **Computational complexity**: of order $O(n^2)$
  - Complexity of BFGS is the same as DFP
- Preserves **positive definiteness** of B_k
- DFP **more sensitive to step size**
  - BFGS convergence speed/rate is (in general) much less sensitive to approximate step size compared to DFP
  - Explains preferential adoption of BFGS over DFP

Algorithm: Convergence QN methods

- **Fact**: Hessian approximations evolve by means of updating formulas

  $\Rightarrow$ Analysis of QN methods much more complex than that of steepest descent and Newton's method

Algorithm: Convergence QN methods

- **Fact**: Hessian approximations evolve by means of updating formulas

  $\Rightarrow$ Analysis of QN methods much more complex than that of steepest descent and Newton's method

- **QN methods**: BFGS, DFP, SR1, etc.

  ◦ Known empirically to be robust

  ◦ Not possible to establish global convergence results for general nonlinear objective functions

Consequence

No formal proof that iterates of QN methods converge to a stationary point from any starting point and any (suitable) initial Hessian approximation

Algorithm: Convergence QN methods

Superlinear convergence results verified under reasonable assumptions (objective function, iterates, etc.)

Algorithm: Convergence QN methods

Superlinear convergence results verified under reasonable assumptions (objective function, iterates, etc.)

Assumptions

- **Search direction**

$$d_k = -H_k^{-1} \nabla f(x_k)$$

  where H_k (symmetric and positive definite matrix) updated at every iteration by QN updating formula

- **Step length/size** t_k computed by inexact line search algorithm satisfying (strong or weak) Wolfe conditions
  - Line search algorithm always trials $t = 1$ first
  - Accepts $t = 1$ if it satisfies the Wolfe conditions
  - Crucial implementation detail to obtain a fast rate of convergence

Algorithm: Convergence QN methods

Theorem

Consider

- $f : \mathbb{R}^n \to \mathbb{R}$ twice continuously differentiable
- Iterate $x_{k+1} = x_k + t_k d_k$, where d_k is a QN descent direction and t_k satisfies Wolfe conditions with $c_1 \leq 1/2$

If

- $\{x_k\}_{k \in \mathbb{N}}$ converges to $x^\star$ such that $\nabla f(x^\star) = 0$ and $\nabla^2 f(x^\star) \succ 0$
- QN search direction d_k satisfies

$$\lim_{k \to +\infty} \frac{\left\| \left(H_k - \nabla^2 f(x^\star) \right) d_k \right\|}{\| d_k \|} = 0 \tag{15}$$

Then

i) Step length $t_k = 1$ admissible for all $k, k > k_0$

ii) if $t_k = 1$ for all $k > k_0$, sequence $\{x_k\}_{k \in \mathbb{N} : k > k_0}$ converges to $x^\star$ superlinearly

Algorithm: Convergence QN methods

Intepretation:

- **If** search direction of QN method approximates Newton direction well enough, i.e., satisfies (15)
- **Then** implications

  ◦ Unit step length ($t = 1$) satisfies Wolfe cond. as iterates converge to $x^\star$

  ◦ **Superlinear** convergence rate can be attained even if the sequence of quasi-Newton matrices H_k does not converge to $\nabla^2 f(x^\star)$

  Note: Newton method converges more rapidly, i.e., quadratically

Algorithm: Convergence QN methods

Intepretation:

- **If** search direction of QN method approximates Newton direction well enough, i.e., satisfies (15)
- **Then** implications

  ◦ Unit step length ($t = 1$) satisfies Wolfe cond. as iterates converge to $x^\star$

  ◦ **Superlinear** convergence rate can be attained even if the sequence of quasi-Newton matrices H_k does not converge to $\nabla^2 f(x^\star)$

  Note: Newton method converges more rapidly, i.e., quadratically

- **Condition** : sequence H_k become increasingly accurate approximations to $\nabla^2 f(x^\star)$ along the search directions d_k

Algorithm: Convergence QN methods

Intepretation:

- **If** search direction of QN method approximates Newton direction well enough, i.e., satisfies (15)
- **Then** implications

  ◦ Unit step length ($t = 1$) satisfies Wolfe cond. as iterates converge to $x^\star$

  ◦ **Superlinear** convergence rate can be attained even if the sequence of quasi-Newton matrices H_k does not converge to $\nabla^2 f(x^\star)$

  Note: Newton method converges more rapidly, i.e., quadratically

- **Condition** : sequence H_k become increasingly accurate approximations to $\nabla^2 f(x^\star)$ along the search directions d_k

- Note well: condition (15) is both necessary and sufficient for superlinear convergence of QN methods

Algorithm: Convergence QN methods

Theorem

If

- $f : \mathbb{R}^n \to \mathbb{R}$ twice continuously differentiable
- Iterate $x_{k+1} = x_k + d_k$, where d_k is a QN descent direction given by $d_k = -H_k^{-1} \nabla f(x_k)$ and $t_k = 1$ (uniformly)
- Sequence $\{x_k\}$ converges to $x^\star$ such that $\nabla f(x^\star) = 0$ and $\nabla^2 f(x^\star) \succ 0$

Algorithm: Convergence QN methods

Theorem

If

- $f : \mathbb{R}^n \to \mathbb{R}$ twice continuously differentiable
- Iterate $x_{k+1} = x_k + d_k$, where d_k is a QN descent direction given by $d_k = -H_k^{-1} \nabla f(x_k)$ and $t_k = 1$ (uniformly)
- Sequence $\{x_k\}$ converges to $x^\star$ such that $\nabla f(x^\star) = 0$ and $\nabla^2 f(x^\star) \succ 0$

Then

Sequence $\{x_k\}$ converges superlinearly **if and only if** (15) holds

Algorithm: Convergence QN methods

Convergence analysis of BFGS : characterization (15) of superlinear convergence and applies to general nonlinear—not just convex—objective functions

Theorem

If

- $f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable
- Iterates generated by BFGS algorithm converge to a minimizer $x^\star$ at which Hessian matrix $\nabla^2 f$ is Lipschitz continuous with constant $M > 0$

$$\left\|\nabla^2 f(x) - \nabla^2 f(x^\star)\right\| \leq M \left\|x - x^\star\right\|, \ \forall x \text{ near } x^\star$$

- Sequence $\|x_k - x^\star\|$ converges rapidly enough to 0 such that

$$\sum_{k=1}^{\infty} \|x_k - x^\star\| < \infty$$

Then, sequence $\{x_k\}$ converges to $x^\star$ at a superlinear rate

Limited-memory BFGS (L-BFGS)

BFGS still uses an $n \times n$ dense matrix $\Rightarrow$ Problem for storage of dense Hessian matrices in large scale problems $n \gg 1$

Limited-memory BFGS (L-BFGS)

BFGS still uses an $n \times n$ dense matrix $\Rightarrow$ Problem for storage of dense Hessian matrices in large scale problems $n \gg 1$

Limited-memory quasi-Newton methods:

- useful for solving large problems whose Hessian matrices cannot be computed at a reasonable cost or are not sparse
- maintain simple and compact approximations of Hessian matrices: instead of storing fully dense $n \times n$ approximations, store only a few vectors of length n that represent the approximations implicitly

Limited-memory BFGS (L-BFGS)

- Despite modest storage requirements, limited-memory QN methods often yield an acceptable (albeit linear) rate of convergence
- Among various limited-memory QN methods that have been proposed: L-BFGS most commonly considered.

Main idea

- L-BFGS relies on the BFGS updating formula.
- Use curvature information from only the most recent iterations to construct the Hessian approximation.
- Discard curvature information from earlier iterations

  less likely to be relevant to the actual properties of the Hessian matrix at the current iteration discarded in the interest of saving memory space

Limited-memory BFGS (L-BFGS)

- Let t_k be step size at iteration k, each step of the BFGS method has the form

$$x_{k+1} = x_k - t_k B_k \nabla f(x_k)$$

Limited-memory BFGS (L-BFGS)

- Let t_k be step size at iteration k, each step of the BFGS method has the form

$$x_{k+1} = x_k - t_k B_k \nabla f(x_k)$$

- **Notation**: approximation B_k to the inverse of the Hessian matrix H_k^{-1}

$$B_{k+1} = V_k^T B_k V_k + \rho_k s_k s_k^T \tag{16}$$

$$\text{where } \rho_k = \frac{1}{y_k^T s_k}, \ V_k = I - \rho_k y_k s_k^T \tag{17}$$

$$\text{and } s_k = x_{k+1} - x_k, \ y_k = \nabla f(x_{k+1}) - \nabla f(x_k) \tag{18}$$

Limited-memory BFGS (L-BFGS)

- Let t_k be step size at iteration k, each step of the BFGS method has the form

$$x_{k+1} = x_k - t_k B_k \nabla f(x_k)$$

- **Notation**: approximation B_k to the inverse of the Hessian matrix H_k^{-1}

$$B_{k+1} = V_k^T B_k V_k + \rho_k s_k s_k^T \tag{16}$$

$$\text{where } \rho_k = \frac{1}{y_k^T s_k}, \;\; V_k = I - \rho_k y_k s_k^T \tag{17}$$

$$\text{and } s_k = x_{k+1} - x_k, \;\; y_k = \nabla f(x_{k+1}) - \nabla f(x_k) \tag{18}$$

- **Problem**: Since approximation B_k to the inverse of the Hessian matrix will generally be dense, the cost of storing and manipulating B_k matrix is prohibitive when the number of variables is large

Limited-memory BFGS (L-BFGS)

Procedure circumvent this problem:

0) Select some initial approximation B_k^0 (in contrast to the standard BFGS iteration, initial approximation allowed to vary from iteration to iteration)

   A method for choosing B_k^0 that has proved effective in practice is to set

$$B_k^0 = \gamma_k I = \frac{s_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}} I$$

   The scaling factor γ_k attempts to estimate the size of the true Hessian matrix along the most recent search direction

   This choice helps ensuring that the search direction d_k is well scaled, and as a result the step length $t_k = 1$ is accepted in most iterations

Limited-memory BFGS (L-BFGS)

1) Store instead a modified version of B_k implicitly, by storing a m vector pairs $\{s_i, y_i\}, i = k-m, \ldots, k-1$.

   Assume some initial Hessian approximation B_k^0 and find by repeated application of (16) that L-BFGS approximation B_k satisfies

$$\begin{aligned} B_k = & \left(V_{k-1}^T \ldots V_{k-m}^T\right) B_k^0 \left(V_{k-m} \ldots V_{k-1}\right) \\ & + \rho_{k-m} \left(V_{k-1}^T \ldots V_{k-m+1}^T\right) s_{k-m} s_{k-m}^T \left(V_{k-m+1} \ldots V_{k-1}\right) \\ & + \rho_{k-m+1} \left(V_{k-1}^T \ldots V_{k-m+2}^T\right) s_{k-m+1} s_{k-m+1}^T \left(V_{k-m+2} \ldots V_{k-1}\right) \\ & \ldots \\ & + \rho_{k-1} s_{k-1} s_{k-1}^T \end{aligned} \tag{19}$$

2) Next, product $B_k \nabla f(x_k)$ obtained by recursively performing a sequence of inner products and vector summations involving $\nabla f(x_k)$ and pairs $\{s_i, y_i\}, i = 1, \ldots, m$

Limited-memory BFGS (L-BFGS)

2) From B_k expression (19), recursive procedure to compute product $B_k \nabla f(x_k)$

L-BFGS Two-Loop recursion algorithm:

1. $q \leftarrow \nabla f(x_k)$
2. `for` $i = k-1, k-2, \ldots, k-m$

   $t_i \leftarrow \rho_i s_i^T q$

   $q \leftarrow q - t_i y_i$

   `end for`
3. $r \leftarrow B_k^0 q$
4. `for` $i = k-m, k-m+1, \ldots, k-1$

   $b \leftarrow \rho_i y_i^T r$

   $r \leftarrow r + s_i(t_i - b)$

   `end for`

5 `stop` with result $B_k \nabla f(x_k) = r$

Limited-memory BFGS (L-BFGS)

Remarks concerning **Step 2**

- Without considering the multiplication $B_k^0 q$, the two-loop recursion scheme requires $4mn$ multiplications
- If B_k^0 is diagonal, then n additional multiplications are needed
- This recursion has also the advantage that the multiplication by the initial matrix B_k^0 is isolated from the rest of the computations, allowing this matrix to be chosen freely and to vary between iterations

Limited-memory BFGS (L-BFGS)

3) After the new iterate is computed, the oldest vector pair in the set of pairs $\{s_i, y_i\}$ is replaced by the new pair $\{s_k, y_k\}$ obtained from the current step by means of (18)

Limited-memory BFGS (L-BFGS)

3) After the new iterate is computed, the oldest vector pair in the set of pairs $\{s_i, y_i\}$ is replaced by the new pair $\{s_k, y_k\}$ obtained from the current step by means of (18)

4) In this way, the set of vector pairs includes curvature information from the m most recent iterations. In practice, relatively small values of m (between 3 and 20) often produce satisfactory results

Limited-memory BFGS (L-BFGS)

Formal specification of L-BFGS algorithm

L-BFGS algorithm:

`input`:

Choose initial point $x_0 \in \mathrm{dom}(f)$

Select integer $m > 1$ (usually in range [3,20])

`repeat`:

0. Compute the approximation B_k^0 to the inverse of the Hessian matrix
1. Compute the search direction $d_k := -B_k \nabla f(x_k)$ using the Two-loop recursion algorithm
2. Compute $x_{k+1} = x_k + t_k d_k$
   where t_k is computed to satisfy the Wolfe (decrease and curvature) conditions
3. `If` $k > m$
   `Then` Discard the vector pair $\{s_{k-m}, y_{k-m}\}$ from storage;
4. Compute $s_k = x_{k+1} - x_k$ and $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$

`until`: stopping criterion is satisfied ($|\nabla f(x_k)\| \leq \varepsilon$)

Limited-memory BFGS (L-BFGS)

In practice

- Strategy of keeping the m most recent correction pairs $\{s_i, y_i\}$ works well in practice
- Optimal choice of m is nevertheless problem dependent
- Since the cost of each iteration increases with the amount of storage, the best CPU time is often obtained for small values of m
- Moreover, L-BFGS algorithm tends to be less robust when m is small

Limited-memory BFGS (L-BFGS)

Remarks

- During its first $m-1$ iterations, L-BFGS algorithm is equivalent to BFGS **if**

  i) initial matrix B_0 is the same in both methods

  ii) L-BFGS chooses $B_k^0 = B_0$ at each iteration

Limited-memory BFGS (L-BFGS)

Remarks

- During its first $m-1$ iterations, L-BFGS algorithm is equivalent to BFGS **if**

  i) initial matrix B_0 is the same in both methods

  ii) L-BFGS chooses $B_k^0 = B_0$ at each iteration

- **Main weakness** of L-BFGS method

  ◦) Converges slowly on **ill-conditioned problems**, i.e., problems where the Hessian matrix contains a wide distribution of eigenvalues

  ◦) Spread of eigenvalues measured by the **condition number** κ of $\nabla^2 f(x^\star)$

$$\kappa := \frac{\lambda_{\max}(\nabla^2 f(x^\star))}{\lambda_{\min}(\nabla^2 f(x^\star))}$$

INFO-F524: Continuous Optimization
–Section 4: Constrained Optimization –

Bernard Fortz, Dimitri Papadimitriou

Bureau: **2N3 206**

Université Libre de Bruxelles

dimitrios.papadimitriou@ulb.be

Annee Académique: 2023-2024

Lagrange function

Consider the problem with both equality and inequality constraints of the form

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\text{minimize}} \quad & f(x) = f(x_1,\dots,x_n) \\ \text{subject to} \quad & g_j(x) \le 0 && j = 1,\dots,m \\ & h_i(x) = 0 && i = 1,\dots,\ell, \end{aligned} \tag{1}$$

Assumption

The functions f, $h_i, i = 1,\dots,\ell$ and $g_j, j = 1,\dots,\ell$ are continuously differentiable (at least within an open set containing a local minimum)

Lagrange function: $L : \mathbb{R}^{n+\ell+m} \to \mathbb{R}$ associated to (1)

$$L(x,\lambda,\mu) = f(x) + \sum_{i=1}^{\ell} \lambda_i h_i(x) + \sum_{j=1}^{m} \mu_j g_j(x) \tag{2}$$

The vectors $\lambda \in \mathbb{R}^\ell$ and $\mu \in \mathbb{R}^m$ are called the **dual variables** or multiplier vectors associated with the problem (1).

Lagrange Dual function

Lagrange dual function : $\varphi\colon \mathbb{R}^\ell \times \mathbb{R}^m \to \mathbb{R}$ is defined at the **infimum** of the Lagrange function (2) over x: for $\lambda \in \mathbb{R}^\ell, \mu \in \mathbb{R}^m$

$$\begin{aligned}\varphi(\lambda,\mu) &= \inf_{x\in\mathcal{D}} L(x,\lambda,\mu) \\ &= \inf_{x\in\mathcal{D}} \left\{ f(x) + \sum_{j=1}^{m} \mu_j g_j(x) + \sum_{i=1}^{\ell} \lambda_i h_i(x) \right\}\end{aligned} \tag{3}$$

- The Lagrange dual function is the pointwise infimum of a family of affine functions of (λ,μ): for x $(\in \mathcal{D})$ fixed, $L(x,\lambda,\mu)$ is affine in (λ,μ).
- The Lagrange dual function is **concave**, even when the problem (1) is not convex.

Lagrange Dual function

The Lagrange dual function is **concave**, even when (2) is not convex

Proof:

- Def: function $f : \mathbb{R}^N \to \mathbb{R}$ is convex if $\forall (x, y) \in \mathrm{dom}(f) \times \mathrm{dom}(f)$ and $\forall a \in [0, 1]$

$$f(ax + (1-a)y) \leq af(x) + (1-a)f(y) \tag{4}$$

- A function f is concave if $\forall a \in [0, 1]$

$$f(ax + (1-a)y) \geq af(x) + (1-a)f(y). \tag{5}$$

- For $x \in \mathcal{D}$ fixed, $L(x, \lambda)$ is **affine** in $\lambda \to \forall a \in [0, 1]$

$$\begin{aligned} L(x, a\lambda_1 + (1-a)\lambda_2) &= L(x, a\lambda_1) + L(x, (1-a)\lambda_2) \\ &= aL(x, \lambda_1) + (1-a)L(x, \lambda_2) \end{aligned} \tag{6}$$

- Hence, since $L(\lambda) \geq \varphi(\lambda)$

$$\begin{aligned} L(x, a\lambda_1 + (1-a)\lambda_2) &= aL(x, \lambda_1) + (1-a)L(x, \lambda_2) \\ &\geq a\varphi(\lambda_1) + (1-a)\varphi(\lambda_2) \end{aligned} \tag{7}$$

Lagrange Dual function

Proof:

- Since this inequality holds for any $x \in \mathcal{D}$, we have

$$\begin{aligned} \inf_x L(x, a\lambda_1 + (1-a)\lambda_2) &= \varphi\big(a\lambda_1 + (1-a)\lambda_2\big) \\ &\geq a\varphi(x, \lambda_1) + (1-a)\varphi(x, \lambda_2) \end{aligned} \tag{8}$$

- Similar reasoning for φ: $\varphi(\mu)$
- Then, combination by introduction vector $\theta = (\lambda, \mu)$: $\varphi = \varphi(\theta)$

Lagrange Dual function: Example

$$\text{minimize } f(x_1, x_2, x_3, x_4) = (x_1 + 1)(x_2 - 12) + \frac{1}{2}(x_3 - 16)^2 + \frac{1}{2}(x_4 - 32)^2$$

$$\text{subject to } 2x_1 + x_2 - \frac{1}{5}x_3 - x_4 = 10$$

- Lagrange function:

$$L(x_1, x_2, x_3, x_4, \lambda) = f(x_1, x_2, x_3, x_4) + \lambda(2x_1 + x_2 - \frac{1}{5}x_3 - x_4 - 10)$$

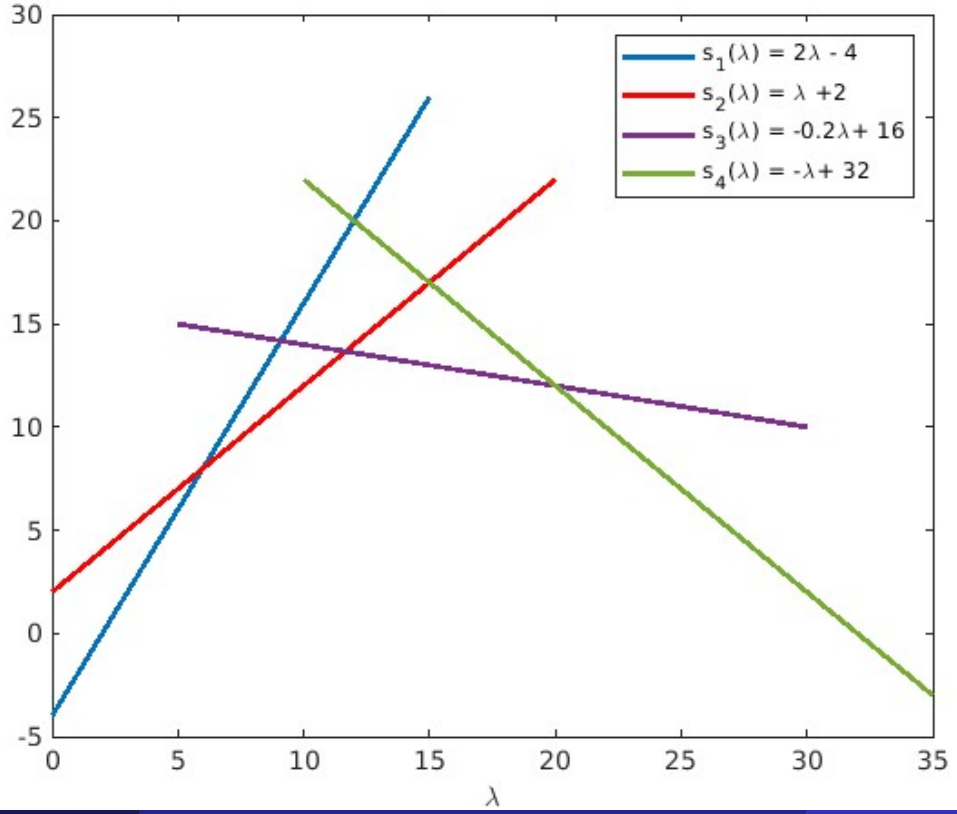
- Computing Lagrangian dual function $\inf_{x_i, i=1:4} L$ requires solving the system

$$\begin{cases} \partial L/\partial x_1 = x_2 - 12 + 2\lambda = 0 \\ \partial L/\partial x_2 = x_1 + 1 + 1\lambda = 0 \\ \partial L/\partial x_3 = x_3 + 16 - \frac{1}{5}\lambda = 0 \\ \partial L/\partial x_4 = x_4 + 32 - 1\lambda = 0 \end{cases} \xRightarrow{x=(1,8,0,0)} \begin{cases} s_1(\lambda) := 2\lambda - 4 \\ s_2(\lambda) := 1\lambda + 2 \\ s_3(\lambda) := -\frac{1}{5}\lambda + 16 \\ s_4(\lambda) := -1\lambda + 32 \end{cases}$$

Lagrange Dual function: Example

System

$$\begin{cases} \partial L/\partial x_1 = x_2 - 4 + 2\lambda = 0 \\ \partial L/\partial x_2 = x_1 + 1 + 1\lambda = 0 \\ \partial L/\partial x_3 = x_3 + 16 - \frac{1}{5}\lambda = 0 \\ \partial L/\partial x_4 = x_4 + 32 - 1\lambda = 0 \end{cases} \xRightarrow{x=(1,8,0,0)} \begin{cases} s_1(\lambda) := 2\lambda - 4 \\ s_2(\lambda) := 1\lambda + 2 \\ s_3(\lambda) := -\frac{1}{5}\lambda + 16 \\ s_4(\lambda) := -1\lambda + 32 \end{cases}$$



Duality gap

Lower-bound property

- The Lagrange dual function $\varphi(\lambda, \mu)$ yields lower bounds on the optimal value $p^\star$ of (1): for any λ (unrestricted) and any $\mu \geq 0$, we have

$$\varphi(\lambda, \mu) \leq p^\star \tag{9}$$

- For each pair $(\lambda, \mu) \in \mathbb{R}^\ell \times \mathbb{R}^m$ with $\mu \geq 0$, the Lagrange dual function $\varphi(\lambda, \mu)$ gives a lower (upper) bound on the optimal value $p^\star$ of the minimization (maximization) problem (1)

Duality gap

Lagrange dual problem

- To determine the best lower bound that can be obtained from the Lagrange dual function (3) solve the concave **Lagrange dual problem**

$$\begin{array}{ll} \text{maximize} & \varphi(\lambda, \mu) \\ \text{subject to} & \mu \geq 0 \end{array} \tag{10}$$

- Optimal value of Lagrange dual problem, denoted $d^\star$: by definition, the best lower bound on $p^\star$ that can be obtained from the Lagrange dual function

- **Duality gap**: for minimization (maximization) problem, the optimal duality gap of the original problem is defined as the difference

$$\text{minimization} : \; p^\star - d^\star$$
$$\text{maximization} : \; d^\star - p^\star$$

Equality-constrained Newton Method

Two key properties of Newton's method

1. **First-order/Linearized optimality condition**
   - When at point x, seek a direction d such that $\nabla f(x+d) = 0$.
   - Using first order Taylor expansion of f at $x+d$

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

     we get

     $$\nabla f(x+d) \approx \nabla f(x) + \nabla^2 f(x) d = 0$$

   - Solving this linearized system for d, we get Newton update

     $$d = -(\nabla^2 f(x))^{-1} \nabla f(x)$$

Equality-constrained Newton Method

Two key properties of Newton's method

1. **First-order/Linearized optimality condition**
   - When at point x, seek a direction d such that $\nabla f(x+d) = 0$.
   - Using first order Taylor expansion of f at $x+d$

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

     we get

     $$\nabla f(x+d) \approx \nabla f(x) + \nabla^2 f(x) d = 0$$

   - Solving this linearized system for d, we get Newton update

     $$d = -(\nabla^2 f(x))^{-1} \nabla f(x)$$

Remember that $x_{k+1} = x_k + t_k d_k$; thus, Newton step $\Delta x \equiv t_k d_k = x_{k+1} - x_k$ with $t_k = 1$ for the pure Newton method

Equality-constrained Newton Method

2. **Affine Invariance** of Newton's method:
   - Consider an affine transformation $x = Ay$ of the variable x in the original minimization problem, $\min_x f(x)$ where $A \in \mathbb{R}^{n \times n}$ is an invertible matrix
   - Let $\phi(y) = f(Ay)$: Newton update on ϕ is the same as that on f
   - Thus, Newton method independent of linear scaling

Equality-constrained Newton Method

2. **Affine Invariance** of Newton's method:
   - Consider an affine transformation $x = Ay$ of the variable x in the original minimization problem, $\min_x f(x)$ where $A \in \mathbb{R}^{n \times n}$ is an invertible matrix
   - Let $\phi(y) = f(Ay)$: Newton update on ϕ is the same as that on f
   - Thus, Newton method independent of linear scaling

Consequence: Newton's method not affected by problem's conditioning (due to affine invariance)

Equality-constrained Newton Method

Assume f is convex and twice continuously differentiable; hence, dom f is an open set matrix $A \in \mathbb{R}^{m \times n}$ of $rank(A) = m$ optimal value is finite and attained

$$\min_x \; f(x) \;\; \text{s.t.} \;\; Ax = b$$

- For any **convex optimization** problem with differentiable objective and constraint functions, any points that satisfy the Lagrange conditions are primal and dual optimal, and have zero duality gap
- The point $x^\star$ (and $\lambda^\star$) is optimal **if and only if** the optimality conditions are verified by $x^\star \in \operatorname{dom} f$:

$$\begin{cases} \nabla f(x^\star) + A^T \lambda^\star = 0 \\ Ax^\star - b = 0 \end{cases} \tag{11}$$

Equality-constrained Newton Method

- In **equality constrained Newton method**, take Newton steps restricted to a region satisfied by constraints

$$h(x) := Ax = b$$

- Newton step Δx obtained by minimizing second-order approximation of f at x

$$\begin{aligned}\Delta x &= \underset{A(x+\tilde{\Delta}x)=b}{\operatorname{argmin}} \quad f(x+\tilde{\Delta}x) \\ &= \underset{A(x+\tilde{\Delta}x)=b}{\operatorname{argmin}} \quad f(x) + \nabla f(x)^T \tilde{\Delta}x + \frac{1}{2}\tilde{\Delta}x^T \nabla^2 f(x) \tilde{\Delta}x\end{aligned}$$

Equality-constrained Newton Method

- Starting from initial point x_0 that must be feasible to ensure that x_{k+1} is feasible as

$$\begin{aligned} Ax_{k+1} &= A(x_k + t_k \Delta x) \\ &= Ax_k + t_k A\Delta x = b \end{aligned}$$

- Δx and λ are solutions of the linearized approximation of the **optimality condition**

$$\begin{cases} \nabla f(x + \Delta x) + A^T \lambda \approx \nabla f(x) + \nabla^2 f(x)\Delta x + A^T \lambda = 0 \\ A(x + \Delta x) - b = 0 \end{cases} \tag{12}$$

- Hence, since $Ax = b$, the system

$$\begin{cases} \nabla f(x) + \nabla^2 f(x)\Delta x + A^T \lambda = 0 \\ A\Delta x = 0 \end{cases} \tag{13}$$

Equality-constrained Newton Method

- Newton step Δx of f at a feasible point x for the linear equality constrained problem is given by (first block of) the solution of

$$\underbrace{\begin{bmatrix} \nabla^2 f(x) & A^T \\ A & 0 \end{bmatrix}}_{KKTMatrix} \begin{bmatrix} \Delta x \\ \lambda \end{bmatrix} = \begin{bmatrix} -\nabla f(x) \\ 0 \end{bmatrix} \tag{14}$$

  Important: the 0 in the RHS of (14), comes from the Newton's update for solving $h(x) = 0 := Ax - b = 0$ which is $\Delta x = x_{k+1} - x_k = -A^{-1}h(x)$.

- Hence, KKT conditions (see infra) result in an an equality-constrained **KKT matrix**, which maintains the structure and sparseness of the Hessian

  Note: larger linear system in the Hessian

Equality-constrained Newton Method

Generalization : $h(x) = 0$ with $h(x) = (h_1(x), \ldots, h_\ell(x))$

- Defining $J_h(x)$ as the Jacobian of $h(x)$, the system becomes

$$\begin{cases} \nabla f(x) + \nabla^2 f(x)\Delta x + J_h^T(x)\lambda = 0 \\ J_h(x)\Delta x = -h(x) \end{cases} \tag{15}$$

  Note : second eq. comes from Newton update for solving $h(x) = 0$, which is $\Delta x = x_{k+1} - x_k = -J(x)^{-1}h(x)$

- Newton step Δx of f at a feasible point x for the equality constrained problem is given by (first block Δx of) the solution of

$$\underbrace{\begin{bmatrix} \nabla^2 f(x) & J_h^T(x) \\ J_h(x) & 0 \end{bmatrix}}_{KKTMatrix} \begin{bmatrix} \Delta x \\ \lambda \end{bmatrix} = \begin{bmatrix} -\nabla f(x) \\ -h(x) \end{bmatrix} \tag{16}$$

Equality-constrained Newton Method

Newton algorithm:

`input:`

initial point x_0 feasible, i.e., $h(x_0) = 0$

tolerance $\varepsilon > 0$

`repeat:`

1. Compute: Newton step Δx
2. Compute: Newton decrement $\delta \approx f(x) - f^\star = (\Delta x \nabla f^2(x) \Delta x)^{1/2}$
3. Compute step size $t_k > 0$ by backtracking line search
4. Update: $x_{k+1} = x_k + t_k \Delta x$

`until:` $\delta^2/2 < \varepsilon$

Assuming i) f is convex, $\nabla^2 f(x)$ has full-rank and ii) independent constraints (Jacobian has full row-rank), then KKT matrix is invertible. **Implication**

- Newtons step calculated by direct matrix inversion : $J_h(x)\Delta x = -h(x) \rightarrow \Delta x = J_h(x)^{-1} h(x)$

Equality-constrained Newton Method

Remark:

- The Newton's method for the reduced problem (by elimination of equality constraints):

$$\underset{z}{\text{minimize}}\ f(Fz + \tilde{x})$$

  starting at z_0, generates iterates z_k

- The Newton's method with equality constraints, when started at $x_0 = Fz_0 + \tilde{x}$, iterates are given by $x_k = Fz_k + \tilde{x}$

- Hence, the iterates in Newton's method for the equality constrained problem coincide with the iterates in Newton's method applied to the unconstrained reduced problem.

Equality-constrained Newton Method

- In case of general constraints, we cannot use projected methods like we did for gradient method
- Instead, we rely on Barrier methods which will be introduced in Section 6
- Barrier methods is a particular case of the more general interior point methods (IPM)

Nonconvex case

Consider the optimization problem with both equality and inequality constraints

$$\begin{array}{lll} \underset{x\in\mathbb{R}^n}{\text{minimize}} & f(x) = f(x_1,\ldots,x_n) & \\ \text{subject to} & g_j(x) \le 0 & j = 1,\ldots,m \\ & h_i(x) = 0 & i = 1,\ldots,\ell, \end{array} \tag{17}$$

Assumption

The functions f, $h_i, i = 1,\ldots,\ell$ and $g_j, j = 1,\ldots,\ell$ are continuously differentiable (at least within an open set containing a local minimum)

Nonconvex cases : simplest instances

- BoxQP : $x^T Qx$ with Q indefinite and (box) constraints $\ell \le x \le u$
- Nonlinear equality constraints : f convex, g_j convex for all j, and $\exists\, i, h_i$ nonlinear

Nonconvex case: linear equality constraints

- Consider the minimization problem

$$\begin{aligned} \text{maximize} \quad & f(x_1, x_2) = x_1 x_2 \\ \text{subject to} \quad & h(x_1, x_2) = x_1 + 4x_2 - 16 = 0 \end{aligned} \tag{18}$$

- The objective function f is bilinear and thus nonconvex since it includes both convex (e.g. x^2) and concave (e.g. $x_1(1 - x_2)$) functions.
- The constraint function h does not admit a stationary point (linear function), so the qualification condition (see Section 4.8) is satisfied.

Nonconvex case: linear equality constraints

- The **Lagrangian function** is given by

$$L(x_1, x_2, \lambda) = x_1 x_2 - \lambda(x_1 + 4x_2 - 16).$$

- Thus, finding a **stationary point** of the Lagrangian requires solving the following system

$$\frac{\partial L}{\partial x_1} = x_2 - \lambda = 0$$
$$\frac{\partial L}{\partial x_2} = x_1 - 4\lambda = 0$$
$$\frac{\partial L}{\partial \lambda} = -(x_1 + 4x_2 - 16) = 0$$

- We obtain the candidate solution $x_1 = 8,\ x_2 = 2,\ \lambda = 2.$

- **Remember** : first order condition does not enable distinguishing between local minimum and maximum.

Nonconvex case: nonlinear equality constraints

Consider the minimization problem

$$\begin{aligned} \text{maximize} \quad & f(x_1, x_2) = x_1^2 x_2 \\ \text{subject to} \quad & h(x_1, x_2) = 2x_1^2 + x_2^2 - 3 = 0 \end{aligned} \tag{19}$$

- The objective function f is nonconvex.
- The equality constraint is nonconvex.
- The constraint function h does admit a stationary point at (0,0); however, this point doesn't belong to the feasbility set.
- Thus, the rank-based constraint qualification a.k.a. LICQ condition (see Section 4.8) is satisfied.

Nonconvex case: nonlinear equality constraints

The Lagrangian function is given by

$$L(x_1, x_2, \lambda) = x_1^2 x_2 - \lambda(2x_1^2 + x_2^2 - 3)$$

Finding a stationary point of the Lagrangian requires solving the following system

$$\begin{aligned}
\frac{\partial L}{\partial x_1} &= 2x_1 x_2 - 4\lambda x_1 &&= 0 \\
\frac{\partial L}{\partial x_2} &= x_1^2 - 2\lambda x_2 &&= 0 \\
\frac{\partial L}{\partial \lambda} &= -(2x_1^2 + x_2^2 - 3) &&= 0
\end{aligned}$$

Nonconvex case: nonlinear equality constraints

From the first equation, one deduces that $x_1 = 0$ or $x_2 = 2\lambda$

- If $x_1 = 0$, then $x_2 = \pm\sqrt{3}$ and $\lambda = 0$.
- If $x_1 \neq 0$, $x_2 = 2\mu$, we obtain $x_1^2 = x_2^2$ and $3x_1^2 = 3$.

  Thus $x_1 = \pm 1$ and $x_2 = \pm 1$
- If $x_2 = 1$, $\lambda = 0.5$ and if $x_2 = -1$, $\lambda = -0.5$

The candidate solution points are:

$$(0, \sqrt{3}, 0),\ (0, -\sqrt{3}, 0),\ (1, 1, 0.5),$$
$$(-1, -1, -0.5),\ (1, -1, -0.5) \text{ and } (-1, 1, 0.5)$$

By comparing the objective functions, we see that the function is maximized in $(1, 1)$ and $(-1, 1)$.

Nonconvex case: nonlinear equality constraints

Consider the problem

$$\begin{aligned} \text{maximize} \quad & f(x_1, x_2, x_3) = x_1 x_2 x_3 \\ \text{subject to} \quad & h_1(x_1, x_2) = x_1^2 + x_2^2 = 1 \\ & h_2(x_1, x_2, x_3) = x_1 + x_3 = 1 \end{aligned} \tag{20}$$

Written in standard form

$$\begin{aligned} \text{maximize} \quad & f(x_1, x_2, x_3) = x_1 x_2 x_3 \\ \text{subject to} \quad & h_1(x_1, x_2) = x_1^2 + x_2^2 - 1 = 0 \\ & h_2(x_1, x_2, x_3) = x_1 + x_3 - 1 = 0 \end{aligned} \tag{21}$$

Nonconvex case: nonlinear equality constraints

First compute the Jacobian matrix of the constraint functions

$$J_h(x_1, x_2, x_3) = \begin{pmatrix} \frac{\partial h_1}{\partial x_1} & \frac{\partial h_1}{\partial x_2} & \frac{\partial h_1}{\partial x_3} \\ \frac{\partial h_2}{\partial x_1} & \frac{\partial h_2}{\partial x_2} & \frac{\partial h_2}{\partial x_3} \end{pmatrix} = \begin{pmatrix} 2x_1 & 2x_2 & 0 \\ 1 & 0 & 1 \end{pmatrix}$$

- The rank of the Jacobian $rank(J_h) < 2$ for any point with $x_1 = x_2 = 0$.
- Such point would violate the first constraint h_1.
- Consequently, all points in the feasible set satisfy the non-degeneracy CQ condition.

Next, the Lagrangian associated to this problem is:

$$L(x_1, x_2, x_3, \lambda_1, \lambda_2) = x_1 x_2 x_3 - \lambda_2(x_1^2 + x_2^2 - 1) - \lambda_1(x_1 + x_3 - 1)$$

Nonconvex case: nonlinear equality constraints

Lagrangian associated to this problem is:

$$L(x_1, x_2, x_3, \lambda_1, \lambda_2) = x_1 x_2 x_3 - \lambda_2(x_1^2 + x_2^2 - 1) - \lambda_1(x_1 + x_3 - 1)$$

By setting, its first-order partial derivatives equal to 0, we obtain the following system

$$\begin{array}{rclcl}
\dfrac{\partial L}{\partial x_1} & = & x_2 x_3 - 2\lambda_1 x_1 - \lambda_2 & = & 0 \\
\dfrac{\partial L}{\partial x_2} & = & x_1 x_3 - 2\lambda_1 x_2 & = & 0 \\
\dfrac{\partial L}{\partial x_3} & = & x_1 x_2 - \lambda_2 & = & 0 \\
\dfrac{\partial L}{\partial \lambda_1} & = & 1 - x_1 - x_3 & = & 0 \\
\dfrac{\partial L}{\partial \lambda_2} & = & 1 - x_1^2 - x_2^2 & = & 0
\end{array}$$

Nonconvex case: nonlinear equality constraints

- Solve the second and third equations for λ_1 and λ_2 in terms of x_1, x_2 and x_3 and plug these into the first equation to obtain

$$x_2^2 x_3 - x_1^2 x_3 - x_1 x_2^2 = 0$$

- Then, solve the fourth equation for x_2^2 in terms of x_1^2 and the last equation for x_3 in terms of x_1, to obtain

$$(1 - x_1^2)(1 - x_1) - x_1^2(1 - x_1) - x_1(1 - x_1^2) = 0$$

Thus, $x_1 = \frac{1}{6}(-1 \pm \sqrt{13})$

- We obtain the four candidates

$$\begin{cases} x_1 \simeq 0.4343, x_2 \simeq \pm 0.9008, x_3 \simeq 0.5657; \\ x_1 \simeq -0.7676, x_2 \simeq \pm 0.6409, x_3 \simeq 1.7676; \end{cases}$$

- Evaluate the objective functions at these four points: the maximum is attained at $-(0.7676, -0.6409, 1.7676)$.

Inequality and Bound constraints

1. **Linear program with inequality constraints**

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax \leq b \end{aligned} \tag{22}$$

- The Lagrange function is

$$\begin{aligned} L(x, \mu) &= c^T x + \mu^T (Ax - b) \\ &= -b^T \mu + (A^T \mu + c)^T x \end{aligned} \tag{23}$$

- The Lagrange dual function is defined by

$$\begin{aligned} \varphi(\mu) &= \inf_x L(x, \mu) \\ &= -b^T \mu + \inf_x (A^T \mu + c)^T x \end{aligned} \tag{24}$$

Inequality and Bound constraints

- The infimum of a linear function is $-\infty$, except when 0; thus, Lagrange dual function is

$$\varphi(\mu) = \begin{cases} -b^T\mu & A^T\mu + c = 0 \\ -\infty & \text{otherwise} \end{cases} \tag{25}$$

  The dual variable μ is dual feasible if $\mu \geq 0$ and $A^T\mu + c = 0$

- Lagrange dual problem (of the primal LP) : maximize $\varphi(\mu)$ over $\mu \geq 0$

  Reformulation by explicitly including dual feasibility conditions $\mu \geq 0$ as constraints to obtain the Lagrange dual problem (LP in standard form)

$$\begin{aligned} \text{maximize} \quad & -b^T\mu \\ \text{subject to} \quad & A^T\mu + c = 0 \\ & \mu \geq 0 \end{aligned} \tag{26}$$

Inequality and Bound constraints

2. **Quadratic program with inequality constraints**: quadratic programming problem where Q is real positive semi-definite matrix of size $n \times n$

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & \tfrac{1}{2} x^T Q x + c^T x \\ \text{subject to} \quad & Ax \leq b \end{aligned} \tag{27}$$

- The Lagrange dual function is given by

$$\varphi(\mu) = \inf_{x \in \mathbb{R}^n} \left\{ \frac{1}{2} x^T Q x + c^T x + \mu(Ax - b) \right\}$$

  By grouping the terms in x, the infimum is attained at

$$x = -Q^{-1}(c + A^T \mu)$$

- Substituting this expression in $\varphi(\mu)$ yields

$$\varphi(\mu) = -\frac{1}{2} \mu^T A Q^{-1} A^T \mu - \mu^T (b + A Q^{-1} c) - \frac{1}{2} c^T Q c$$

Inequality and Bound constraints

- The dual problem obtained by dropping the last constant term

$$\begin{aligned} \underset{\mu \in \mathbb{R}^n}{\text{maximize}} \quad & \tfrac{1}{2}\mu^T P \mu + q^T \mu \\ \text{subject to} \quad & \mu \geq 0 \end{aligned} \tag{28}$$

  where $P = AQ^{-1}A^T$ and $q = b + AQ^{-1}c$.

- If $\mu^\star$ is any dual optimal solution, then the optimal solution to the primal problem is given by

$$x = -Q^{-1}(c + A^T \mu^\star)$$

Quadratic program with inequality constraints

Hildreth quadratic programming procedure [1] for solving the dual problem (28)

- The direction vectors are selected to be equal to the basis vectors $e_i = [0\ 0\ \ldots\ 1\ \ldots\ 0\ 0]^T$
- Then, the vector μ of dual variables can be varied one component at a time
- At a given step, having obtained a vector $\mu \geq 0$, the objective function may then be regarded as a quadratic function in single component μ_i of the vector μ
- The component μ_i is then adjusted to minimize the objective function. If that requires $\mu_i < 0$, set $\mu_i = 0$. In either case, the objective function is decreased.
- Then, the algorithm takes the next component μ_{i+1}.

[1] D. G. Luenberger. Optimization by Vector Space Methods. John Wiley and Sons, New York, 1969.

Quadratic program with inequality constraints

- If one complete cycle through the components corresponds to one iteration taking the vector $\mu^{(k)}$ to $\mu^{(k+1)}$, the method can be expressed explicitly as

$$\mu^{(k+1)} = \max(0, \omega_i^{(k+1)}) \tag{29}$$

by involving two vectors of dual variable values $\mu^{(k)}$ and its update $\mu^{(k+1)}$

$$\omega_i^{(k+1)} = -\frac{1}{p_{ii}} \left[q_i + \sum_{j=1}^{i-1} p_{ij}\mu_j^{(k+1)} + \sum_{j=i+1}^{m} p_{ij}\mu_j^{(k)} \right] \tag{30}$$

where scalar p_{ij} : ij^{th} element in matrix $P = AQ^{-1}A^T$, and q_i : i^{th} element in vector $q = b + AQ^{-1}c$.

Quadratic program with inequality constraints

- If one complete cycle through the components corresponds to one iteration taking the vector $\mu^{(k)}$ to $\mu^{(k+1)}$, the method can be expressed explicitly as

$$\mu^{(k+1)} = \max(0, \omega_i^{(k+1)}) \tag{29}$$

by involving two vectors of dual variable values $\mu^{(k)}$ and its update $\mu^{(k+1)}$

$$\omega_i^{(k+1)} = -\frac{1}{p_{ii}} \left[q_i + \sum_{j=1}^{i-1} p_{ij}\mu_j^{(k+1)} + \sum_{j=i+1}^{m} p_{ij}\mu_j^{(k)} \right] \tag{30}$$

where scalar p_{ij} : ij^{th} element in matrix $P = AQ^{-1}A^T$, and q_i : i^{th} element in vector $q = b + AQ^{-1}c$.

- Because the converged $\mu^\star$ vector contains either zero or positive values of dual variables, we have

$$x = -Q^{-1}(c + A^T\mu^\star) \tag{31}$$

Quadratic program with inequality constraints

Hildreth QP algorithm : based on an element-by-element search $\rightarrow$ does not require any matrix inversion

- If i) active constraints (primal) are linearly independent -**and**- ii) their number is less than or equal to number of primal variables, then dual variables will converge

- If one or both of these conditions are violated

  Then dual variables will not converge to a set of fixed values

  The iteration will terminate when the iterative counter reaches its maximum value.

- No matrix inversion $\Rightarrow$ computation will continue without interruption : Hildreth QP algorithm outputs **near-optimal solution** with constraints if the situation of conflicting constraints arises.

  Note: one of the key strengths of this algorithm: its ability to automatically recover from an ill-conditioned constrained problem

Quadratic program with inequality constraints

- When both conditions are satisfied, the one-dimensional search technique in Hildreth QP procedure converges to a vector $\mu^\star$ that contains
  - ◦) zeros for inactive constraints
  - ◦) and (strictly) positive components corresponding to active constraints
- Positive component collected as a vector is called $\mu^\star_{act}$ with its value defined by

$$\mu^\star_{act} = -P^{-1}_{act}\, q_{act} = (A_{act} Q^{-1} A^T_{act})^{-1} (b_{act} + A_{act} Q^{-1} c) \tag{32}$$

  where A_{act} and b_{act} are the constraint matrix and coefficient vector with the deletion of the row elements that corresponding to the zero elements in $\mu^\star$
- The proof of convergence relies on existence of a set of bounded $\mu^\star_{act}$ that is determined by the existence of inverse of $(A_{act} Q^{-1} A^T_{act})$

Quadratic program with inequality constraints

Example: Consider the following minimization problem

$$\begin{aligned} \underset{x \in \mathbb{R}^2}{\text{minimize}} \quad & \frac{1}{2} x^T Q x + c^T x \\ \text{subject to} \quad & Ax \leq b \end{aligned} \tag{33}$$

where $Q = \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix}$, $c = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$, $A = \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 3 & 2 \end{bmatrix}$, and $b = \begin{bmatrix} 0 \\ 0 \\ 4 \end{bmatrix}$

- Optimal solution to the unconstrained problem is given by

$$\begin{bmatrix} x_1^{(0)} \\ x_2^{(0)} \end{bmatrix} = -Q^{-1} c = -\begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} -1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{34}$$

- Not to the constrained problem : inequality constraints are violated, with respect to the third constraint, i.e., $3 + 2 > 4$.

Quadratic program with inequality constraints

To find the optimal $\mu^\star$, form

$$
\begin{aligned}
P &= AQ^{-1}A^T \\
&= \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 3 & 2 \end{bmatrix} \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} -1 & 0 & 3 \\ 0 & -1 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & -5 \\ 1 & 2 & -7 \\ -5 & -7 & 29 \end{bmatrix} \\
q &= b + AQ^{-1}c \\
&= \begin{bmatrix} 0 \\ 0 \\ 4 \end{bmatrix} + \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 3 & 2 \end{bmatrix} \begin{bmatrix} -1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}
\end{aligned}
\tag{35}
$$

At iteration $k = 0$: start from initial conditions $\mu_1^{(0)} = 0, \mu_2^{(0)} = 0, \mu_3^{(0)} = 0$

Quadratic program with inequality constraints

- At $k = 1$, using (30) we get

$$\begin{aligned} \omega_1^{(1)} + 1 &= 0 \\ \mu_1^{(1)} + 2\omega_2^{(1)} + 1 &= 0 \\ -5\mu_1^{(1)} - 7\omega_2^{(1)} + 29\omega_3^{(1)} - 1 &= 0 \end{aligned} \tag{36}$$

  Solving the first equation gives $\mu_1^{(1)} = \max(0, \omega_1^{(1)}) = 0$, the second $\mu_2^{(1)} = \max(0, \omega_2^{(1)}) = 0$, and the third $\mu_3^{(1)} = \max(0, \omega_3^{(1)}) = 0.0345$.

- At $k = 1$, using (30) we get

$$\begin{aligned} \omega_1^{(2)} + \mu_2^{(1)} - 5\mu_3^{(1)} + 1 &= 0 \\ \mu_1^{(2)} + 2\omega_2^{(2)} - 7\mu_3^{(1)} + 1 &= 0 \\ -5\mu_1^{(2)} - 7\omega_2^{(2)} + 29\omega_3^{(2)} - 1 &= 0 \end{aligned} \tag{37}$$

  Solving the first equation gives $\mu_1^{(2)} = \max(0, \omega_1^{(2)}) = 0$, the second $\mu_2^{(2)} = \max(0, \omega_2^{(2)}) = 0$, and the third $\mu_3^{(2)} = \max(0, \omega_3^{(2)}) = 0.0345$.

Quadratic program with inequality constraints

- Since $\mu^{(1)} = \mu^{(2)}$, iterative procedure has converged and dual optimal solution is $\mu_1^\star = 0, \mu_2^\star = 0, \mu_3^\star = 0.0345$.
- Primal optimal solution is given by

$$\begin{aligned} x^\star = -Q^{-1}c - Q^{-1}A^T\mu^\star &= \begin{bmatrix} x_1^{(0)} \\ x_2^{(0)} \end{bmatrix} - Q^{-1}A^T\mu^\star \\ &= \begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 0.1724 \\ 0.2414 \end{bmatrix} = \begin{bmatrix} 0.8276 \\ 0.7586 \end{bmatrix} \end{aligned} \tag{38}$$

- From (38), constrained primal optimal solution consists of two parts

  ○) first part : identical to the primal optimal solution of the unconstrained primal problem

  ○) second part : correction term due to active constraint.

Quadratic program with inequality constraints

Example: consider the following minimization problem

$$\begin{aligned} \underset{x \in \mathbb{R}^2}{\text{minimize}} \quad & \frac{1}{2}\left[(x_1-2)^2+(x_2-2)^2\right] \\ \text{subject to} \quad & 0 \le x_1 \le 1,\ 0 \le x_2 \le 1 \end{aligned} \tag{39}$$

which can be written in matrix form as

$$\begin{aligned} \underset{x \in \mathbb{R}^2}{\text{minimize}} \quad & \frac{1}{2}x^T Q x + c^T x \\ \text{subject to} \quad & Ax \le b \end{aligned} \tag{40}$$

where $Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $c = \begin{bmatrix} -2 \\ -2 \end{bmatrix}$, $A = \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix}$, and $b = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$

Bound or Box constraints

- Consider the optimization problem

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & l_i \leq x_i \leq u_i \quad i = 1, \ldots, n, \end{aligned} \tag{41}$$

  Constraints called variable bounds (since they give lower and upper bounds for each variable x_i) or box constraints (since the feasible set is a box)
- In **standard form**

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & l_i - x_i \leq 0 \quad i = 1, \ldots, n \\ & x_i - u_i \leq 0 \quad i = 1, \ldots, n \end{aligned} \tag{42}$$

  Thus, $2n$ inequality constraints

$$\begin{aligned} & g_i(x) = l_i - x_i, \ i = 1, \ldots, n, \\ & \text{and} \\ & g_i(x) = x_{i-n} - u_{i-n}, \ i = n+1, \ldots, 2n \end{aligned} \tag{43}$$

Bound or Box constraints

Linear program with linear constraints ($A \in \mathbb{R}^{\ell \times n}$) and variable bounds or box constraints $l \leq x \leq u,\ l < u$

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & l \leq x \leq u \end{aligned} \tag{44}$$

Dual problem : Lagrange multiplier

- λ associated with the equality constraint
- μ_1 associated with the inequality constraint $x \leq u \rightarrow x - u \leq 0$
- μ_2 associated with the inequality constraint $l \leq x \rightarrow l - x \leq 0$

$$\begin{aligned} \underset{\lambda, \mu_1, \mu_2 \in \mathbb{R}^n}{\text{maximize}} \quad & -b^T \lambda - \mu_1^T u + \mu_2^T l \\ \text{subject to} \quad & A^T \lambda + \mu_1 - \mu_2 + c = 0 \\ & \mu_1 \geq 0,\ \mu_2 \geq 0 \end{aligned} \tag{45}$$

Bound or Box constraints

- Instead, reformulate the primal problem as

$$\begin{aligned} &\underset{x\in\mathbb{R}^n}{\text{minimize}}\ f(x) = \begin{cases} c^T x & l \le x \le u \\ \infty & \text{otherwise} \end{cases} \\ &\text{subject to } Ax = b \end{aligned} \tag{46}$$

- This problem is equivalent to the original primal problem (44): the explicit box constraints are implicit

- Defining $z_i^+ = \max\{z_i, 0\}, z_i^- = \max\{-z_i, 0\}$, the **Lagrange dual function** for problem (46) is

$$\begin{aligned} \varphi(\lambda) &= \inf_{l\le x\le u}\ c^T x + \lambda^T(Ax - b) \\ &= -b^T\lambda - u^T(A^T\lambda + c)^- + l^T(A^T\lambda + c)^+ \end{aligned} \tag{47}$$

Bound or Box constraints

- Thus, derivation of analytical formula for the Lagrange dual function φ, which is a **concave** piecewise-linear function
- **Dual problem**: different from the dual of the original problem (although dual problems are equivalent) since defined by the unconstrained problem

$$\underset{\lambda \in \mathbb{R}^n}{\text{maximize}} - b^T\lambda - u^T(A^T\lambda + c)^- + l^T(A^T\lambda + c)^+ \tag{48}$$

Bound or Box constraints

Box-constrained Quadratic Program (BoxQP)

- Consider quadratic program where $Q \in \mathbb{R}^{n \times n}$ is symmetric matrix

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & \tfrac{1}{2} x^T Q x + c^T x \\ \text{subject to} \quad & l \leq x \leq u \end{aligned} \tag{49}$$

- When the objective function is **nonconvex**, a locally optimal solution to the BoxQP may not be globally optimal
- Particular role: although **NP-Hard**, the nonconvex BoxQP problem is arguably the simplest nonconvex mathematical optimization problem

Bound or Box constraints

- **Global methods** : typically combine convex relaxations of the objective function with branch-and-bound to find globally optimal solutions
- **Local methods** tailored variants of proximal algorithm and semi-smooth Newton direction with exact linesearch

Bound or Box constraints

For the **convex** case (matrix of quadratic polynomial is PSD): two main strategies

- **Active Set algorithm**
  - Generates iterates on a face of the feasible box until either a minimizer of the objective function on that face or a point on the boundary of that face is reached
  - Since function values are strictly decreasing and iterates are feasible, finite convergence can be proved
  - Disadvantage (large-scale case): constraints added or removed one at a time, requiring so a number of iterations proportional to the problem size
  - Combined with gradient projection: many constraints added or removed per iteration

- Other strategy : handle inequality constraints using **interior point algorithms** (see Section 6), Hildreth QP (if Q is positive definite)

Optimality conditions

Consider the following optimization problem in the standard form:

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & f(x) = f(x_1, \ldots, x_n) \\ \text{subject to} \quad & g_j(x) \leq 0 \qquad\qquad j = 1, \ldots, m \\ & h_i(x) = 0 \qquad\qquad i = 1, \ldots, \ell, \end{aligned} \tag{50}$$

The domain $\mathcal{D} = \bigcap_{j=1}^{m} \operatorname{dom} g_j \cap \bigcap_{i=1}^{\ell} \operatorname{dom} h_i$ is nonempty. Here too, $p^\star$ denotes the optimal value of the problem (50) which is **not necessarily convex**.

Optimality conditions

Certificate of suboptimality

- If we can find a dual feasible solution (λ, μ), we establish a lower bound on the optimal value of the primal problem

$$\varphi(\lambda, \mu) \leq p^\star$$

- Thus, dual feasible point (λ, μ) provides a proof or certificate that $\varphi(\lambda, \mu) \leq p^\star$

- Dual feasible points allow to bound how suboptimal a given feasible point is, without knowing the exact value of $p^\star$: if x is primal feasible and (λ, μ) is dual feasible, then

$$f(x) - p^\star \leq f(x) - \varphi(\lambda, \mu)$$

Optimality conditions

Consequences

- x is ε-suboptimal for the primal problem, with $\varepsilon = f(x) - \varphi(\lambda, \mu)$
- (λ, μ) is ε-suboptimal for the dual problem

Duality gap

The duality gap associated with the primal feasible point x and dual feasible point (λ, μ) is defined as the gap between primal and dual objectives:

$$f(x) - \varphi(\lambda, \mu)$$

Optimality conditions

Consequences

- x is ε-suboptimal for the primal problem, with $\varepsilon = f(x) - \varphi(\lambda, \mu)$
- (λ, μ) is ε-suboptimal for the dual problem

Duality gap

The duality gap associated with the primal feasible point x and dual feasible point (λ, μ) is defined as the gap between primal and dual objectives:

$$f(x) - \varphi(\lambda, \mu)$$

A primal-dual feasible pair $(x, (\lambda, \mu))$ localizes the optimal value of the primal (and dual) problems into an interval:

$$p^\star \in [\varphi(\lambda, \mu), f(x)], d^\star \in [\varphi(\lambda, \mu), f(x)],$$

the width of which corresponds to the **duality gap**

Optimality conditions

- **If** duality gap of primal-dual feasible pair $(x, (\lambda, \mu))$ is zero, i.e. $f(x) = \varphi(\lambda, \mu)$
  Then, x is primal optimal and (λ, μ) is dual optimal

Consequently

- (λ, μ) provides a certificate that proves that x is optimal
- x provides a certificate that proves that (λ, μ) is dual optimal

Optimality conditions

Nonheuristic stopping criteria

- Suppose an algorithm produces a sequence of primal feasible points x_k and dual feasible (λ_k, μ_k) points, for $k = 1, 2, \ldots$, and $\varepsilon_{abs} > 0$ is a given required absolute accuracy
- Then **stopping criterion** given by

$$f(x(k)) - \varphi(\lambda_k, \mu_k) \leq \varepsilon_{abs}$$

  This condition guarantees that x_k is ε_{abs}-suboptimal when the algorithm terminates

  Note: (λ_k, μ_k) provides a certificate that proves it
- Obviously, strong duality must hold if this method is to work for arbitrarily small tolerances ε_{abs}

Complementary slackness

- Suppose strong duality holds: primal and dual optimal values are equal
- Let $x^\star$ be a primal optimal and $(\lambda^\star, \mu^\star)$ be a dual optimal point, we obtain

$$
\begin{aligned}
f(x^\star) &= \varphi(\lambda^\star, \mu^\star) \quad \text{by zero duality gap assumption} && (51)\\
&= \inf_x \left\{ f(x) + \sum_{i=1}^{\ell} \lambda_i^\star h_i(x) + \sum_{j=1}^{m} \mu_j^\star g_j(x) \right\} \quad \text{by definition} && (52)\\
&\leq f(x^\star) + \sum_{i=1}^{\ell} \lambda_i^\star h_i(x^\star) + \sum_{j=1}^{m} \mu_j^\star g_j(x^\star) && (53)\\
&\leq f(x^\star) && (54)
\end{aligned}
$$

- Since inequality (third line) is an equality, $x^\star$ minimizes $L(x, \lambda^\star, \mu^\star)$ over x
- Note: $L(x, \lambda^\star, \mu^\star)$ can have other minimizers; $x^\star$ is simply a minimizer

Complementary slackness

Implications

- The (sub)differential of $L(x, \lambda^\star, \mu^\star)$ must contain 0 at $x = x^\star$ —is exactly the stationarity condition
- Condition known as **complementary slackness**

$$\sum_{j=1}^{m} \mu_j^\star g_j(x^\star) = 0. \tag{55}$$

- Since each term in this sum is nonpositive (each term $\mu_j^\star g_j(x^\star) \leq 0$), we conclude that

$$\mu_j^\star g_j(x^\star) = 0, \; j = 1, \ldots, m \tag{56}$$

- <u>Note</u>: this condition holds for any primal optimal $x^\star$ and any dual optimal $(\lambda^\star, \mu^\star)$ point when strong duality holds

Complementary slackness

Complementary slackness condition

$$\boxed{\begin{cases} \mu_j^\star > 0 & \implies g_j(x^\star) = 0 \\ g_j(x^\star) < 0 & \implies \mu_j^\star = 0 \end{cases}} \tag{57}$$

Interpretation of this condition (applies only to inequality constraints)

- for the jth optimal Lagrange multiplier μ_j: $\mu_j = 0$ unless the jth constraint is active at the optimum
- for the jth inequality constraint: at $x^\star$, either $g_j(x^\star) = 0$ or the corresponding dual variable $\mu_j = 0$

KKT Conditions

Consider the minimization problem

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & f(x) = f(x_1, \ldots, x_n) \\ \text{subject to} \quad & g_j(x) \leq 0 \qquad\qquad j = 1, \ldots, m \\ & h_i(x) = 0 \qquad\qquad i = 1, \ldots, \ell, \end{aligned} \tag{58}$$

Functions $f, g_1, \ldots, g_m, h_1, \ldots, h_\ell$ differentiable but not necessarily convex

KKT Conditions

Lagrange function

$$L(x, \lambda, \mu) = f(x) + \sum_{i=1}^{\ell} \lambda_i h_i(x) + \sum_{j=1}^{m} \mu_j g_j(x) \tag{59}$$

First-order KKT conditions

First-order **KKT conditions** associated to problem (58)

1. Stationarity: $\nabla f(x) + \sum_{i=1}^{\ell} \lambda_i \nabla h_i(x) + \sum_{j=1}^{m} \mu_j \nabla g_j(x) = 0$
2. Primal feasibility: $h_i(x) = 0, i = 1, \ldots, \ell,\ g_j(x) \leq 0, j = 1, \ldots, m$
3. Dual feasibility: $\mu_j \geq 0, j = 1, \ldots, m$
4. Complementary slackness: $\sum_{j=1}^{m} \mu_j g_j(x) = 0$

KKT Conditions

In indexed notation: stationarity condition reads

$$\begin{cases} \dfrac{\partial L(x,\lambda,\mu)}{\partial x_1} & = 0 \\ \dots \\ \dfrac{\partial L(x,\lambda,\mu)}{\partial x_n} & = 0 \end{cases} \tag{60}$$

Thus

$$\frac{\partial L(x,\lambda,\mu)}{\partial x_i} = 0, \forall i = 1, \dots, n$$

Moreover, complementary slackness conditions often expressed as

$$\mu_j g_j(x) = 0, \; j = 1, \dots, m$$

Note: case $m = 0$, i.e., when there are **no inequality constraints**, the KKT conditions turn into the Lagrange conditions, and the KKT multipliers are called Lagrange multipliers

KKT Conditions

In case of maximization problem

$$
\begin{aligned}
L(x, \lambda, \mu) &= f(x) + \sum_{i=1}^{\ell} \lambda_i (0 - h_i(x)) + \sum_{j=1}^{m} \mu_j (0 - g_j(x)) \\
&= f(x) - \sum_{i=1}^{\ell} \lambda_i h_i(x) - \sum_{j=1}^{m} \mu_j g_j(x)
\end{aligned}
\tag{61}
$$

KKT conditions

1. Stationarity: $\nabla f(x) - \sum_{i=1}^{\ell} \lambda_i \nabla h_i(x) - \sum_{j=1}^{m} \mu_j \nabla g_j(x) = 0$
2. Primal feasibility: $h_i(x) = 0, i = 1, \ldots, \ell$, $g_j(x) \leq 0, j = 1, \ldots, m$
3. Dual feasibility: $\mu_j \geq 0, j = 1, \ldots, m$
4. Complementary slackness: $\sum_{j=1}^{m} \mu_j g_j(x) = 0$

KKT Conditions: necessity for nonconvex problems

- If $x^\star$ and $\lambda^\star$, $\mu^\star$ are any primal and dual points respectively to a nonconvex problem (with zero duality gap)
- Then $x^*, (\mu^\star, \lambda^\star)$ must satisfy the KKT conditions

Thus, for any solution $x^\star$ to the primal problem and any solution $(\mu^\star, \lambda^\star)$ to the dual problem with zero duality gap (strong duality holds), $x^\star, (\mu^\star, \lambda^\star)$ must satisfy the KKT conditions

1. Stationarity: $\nabla f(x^\star) + \sum_{i=1}^{\ell} \lambda_i^\star \nabla h_i(x^\star) + \sum_{j=1}^{m} \mu_j^\star \nabla g_j(x^\star) = 0$
2. Primal feasibility: $h_i(x^\star) = 0, i = 1, \ldots, \ell$, and $g_j(x^\star) \leq 0, j = 1, \ldots, m$
3. Dual feasibility: $\mu_j^\star \geq 0, j = 1, \ldots, m$
4. Complementary slackness: $\sum_{j=1}^{m} \mu_j^\star g_j(x^\star) = 0$

KKT Conditions: sufficiency for convex problems

Suppose $f, g_1, \ldots, g_m$ are convex, and $h_1, \ldots, h_\ell$ are affine

- **If** there exists a solution $\tilde{x}$ to the primal problem and a solution $(\tilde{\mu}, \tilde{\lambda})$ to the dual problem, that together satisfy the KKT conditions
- **Then** $\tilde{x}$ and $(\tilde{\mu}, \tilde{\lambda})$ are primal and dual optimal points, with zero duality gap.

KKT Conditions: sufficiency for convex problems

Suppose $f, g_1, \ldots, g_m$ are convex, and $h_1, \ldots, h_\ell$ are affine

- **If** there exists a solution $\tilde{x}$ to the primal problem and a solution $(\tilde{\mu}, \tilde{\lambda})$ to the dual problem, that together satisfy the KKT conditions
- **Then** $\tilde{x}$ and $(\tilde{\mu}, \tilde{\lambda})$ are primal and dual optimal points, with zero duality gap.

Consequence: for convex optimization problem with differentiable objective and constraint functions, points that satisfy the KKT conditions are primal and dual optimal **and** have zero duality gap

Slater condition

- Remember: if the optimal duality gap is zero, then strong duality holds

$$d^\star = p^\star \tag{62}$$

$\rightarrow$ best bound that can be obtained from the Lagrange dual function is tight

Slater condition

- Remember: if the optimal duality gap is zero, then strong duality holds

$$d^\star = p^\star \tag{62}$$

$\rightarrow$ best bound that can be obtained from the Lagrange dual function is tight

- If the primal problem is convex: $f, g_1, \ldots, g_m$ convex and linear equality constraints

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & g_j(x) \leq 0 \quad j = 1, \ldots, m \\ & Ax = b \end{array} \tag{63}$$

Then, we usually (but not always) have strong duality

Slater condition

Slater conditions

There exists an $x \in relint(\mathcal{D})$[a] such that $g_j(x) < 0$, $j = 1, \ldots, m$ and $Ax = b$

[a] The interior relative to the affine dimension of a set

Slater condition

Slater conditions

There exists an $x \in relint(\mathcal{D})^a$ such that $g_j(x) < 0$, $j = 1, \ldots, m$ and $Ax = b$

[a] The interior relative to the affine dimension of a set

- Note : such a point is sometimes called **strictly feasible** since the inequality constraints hold with strict inequalities $g_j(x) < 0, j = 1, \ldots, m$
- **Slater's theorem** : strong duality holds if the Slater's condition holds and the problem is convex

Slater conditions

Slater conditions **implications**

- Strong duality for convex optimization problems
- The dual optimal value attained when $d^\star > -\infty$
  there exists a dual feasible $(\lambda^\star, \mu^\star)$ with $\varphi(\lambda^\star, \mu^\star) = d^\star = p^\star$
- If a convex optimization problem with differentiable objective and constraint functions satisfies Slater's condition then **KKT conditions** provide necessary and sufficient conditions for optimality
- The optimal duality gap is zero and the dual optimum is attained, so $x^\star$ is optimal if and only if there are (λ, μ) that, together with $x^\star$, satisfy the KKT conditions

CQ conditions

Slater's constraint qualification (CQ)

- Guarantees strong duality (and that the dual optimum is attained) for a convex problem

More generally

- **Constraint Qualification** (CQ) : conditions over the constraints under which a local minimum of the optimization problem satisfy the KKT conditions

- **First-order** CQ : conditions over the constraints under which it can be claimed that

  if x is a feasible minimum point, then x is a stationary point of the Lagrangian function associated to each objective function it minimizes

CQ conditions

First-order CQ

- Properties of analytical description of a set which ensure that its structure around a given feasible point can be constructively captured by (first-order) approximations of the constraint functions defining the set
- Essential for deriving primal and **primal-dual characterizations** of solutions of optimization problems, for duality relations, and for convergence and rate of convergence of computational methods for solving optimization problems

CQ conditions

First-order CQ

- Properties of analytical description of a set which ensure that its structure around a given feasible point can be constructively captured by (first-order) approximations of the constraint functions defining the set
- Essential for deriving primal and **primal-dual characterizations** of solutions of optimization problems, for duality relations, and for convergence and rate of convergence of computational methods for solving optimization problems

Appropriate description of **geometry** of feasible set $D \in \mathbb{R}^n$ around a feasible point $\tilde{x} \in D$ that includes all feasible directions (as well as "almost-feasible" ones)

$$\mathcal{T}_D(\tilde{x}) = \Big\{ d \in \mathbb{R}^n \mid \exists \, \{t_k\} \subset \mathbb{R}_0^+, \; \{t_k\} \to 0, \\ \exists \, \{d_k\} \subset \mathbb{R}^n, \; \{d_k\} \to d, \text{ such that } \tilde{x} + t_k d_k \in D, \forall k \Big\} \tag{64}$$

CQ conditions

- Suppose further that D (i.e. the feasible set) is defined by a finite number of equality and inequality constraints

$$D = \left\{ x \in \mathbb{R}^n \mid h(x) = 0, g(x) \leq 0 \right\} \tag{65}$$

where $h\colon \mathbb{R}^n \to \mathbb{R}^\ell$ and $g\colon \mathbb{R}^n \to \mathbb{R}^m$ are functions assumed to be continuously differentiable in the region of interest

- In turn the optimization can be formulated as

$$\text{minimize } f(x) \text{ subject to } x \in D \tag{66}$$

CQ conditions

- CQ : conditions imposed on h, g and/or their derivatives at or around the point $\tilde{x}$, that guarantee that this description has an explicit algebraic representation in terms of the **derivatives of the constraint functions**
- Optimality conditions : whenever a point $\tilde{x}$ is a local solution of (66) where $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable at $\tilde{x}$, it holds that

$$\langle \nabla f(\tilde{x}) \mid x - \tilde{x} \rangle \geq 0 \ \forall x \in \tilde{x} + \mathcal{T}_D(\tilde{x}) \tag{67}$$

Def: Constraint qualification (CQ)

CQ provide a characterization of the set $\mathcal{T}_D(\tilde{x})$ defined by (64) and its dual

$$\mathcal{T}_D(\tilde{x}) = \Big\{ d \in \mathbb{R}^n \mid \exists \, \{t_k\} \subset \mathbb{R}_0^+, \ \{t_k\} \to 0, \\ \exists \, \{d_k\} \subset \mathbb{R}^n, \ \{d_k\} \to d, \text{ such that } \tilde{x} + t_k d_k \in D, \forall k \Big\}$$

to make the optimality conditions (67) tractable

CQ conditions

Active and inactive constraints

- Consider set D defined by finite number of equality and inequality constraints
- Let $\tilde{x} \in D$ be any feasible point
- If $g_j(\tilde{x}) < 0$ for some $j \in \{1, \ldots, m\}$, by continuity $g_j(x) < 0$ for all $x \in \mathbb{R}^n$ close to $\tilde{x}$, and such constraints (inactive at $\tilde{x}$) do not influence the geometry of the set D around the point $\tilde{x}$

Active constraints

The set of inequality constraints active at $\tilde{x}$ is defined by

$$\mathcal{A}(\tilde{x}) = \{j = 1, \ldots, m \mid g_j(\tilde{x}) = 0\}$$

All the equality constraints are, of course, active at any feasible point.

CQ conditions

- As candidate to represent directions tangent to D at the point $\tilde{x}$, consider the set of directions obtained by linearizing all the constraints active at $\tilde{x}$

$$\mathcal{L}_D(\tilde{x}) = \{d \in \mathbb{R}^n \mid \nabla h_i(\tilde{x})^T d = 0, i = 1, \ldots, \ell, \ \nabla g_j(\tilde{x})^T d \leq 0 \ \forall j \in \mathcal{A}(\tilde{x})\}$$

<u>Note</u>: $\mathcal{T}_D(\tilde{x}) \subset \mathcal{L}_D(\tilde{x})$ always

- Thus, **fundamental question** is whether it holds that

$$\mathcal{T}_D(\tilde{x}) = \mathcal{L}_D(\tilde{x}) \tag{68}$$

- When (68) holds, the dual of $\mathcal{L}_D(\tilde{x})$ is

$$(\mathcal{T}_D(\tilde{x}))^\circ = (\mathcal{L}_D(\tilde{x}))^\circ \tag{69}$$

$$= \left\{ z \in \mathbb{R}^n \mid z = \sum_{i=1}^{\ell} \lambda_i \nabla h_i(\tilde{x}) + \sum_{j=1}^{m} \mu_j \nabla g_j(\tilde{x}), \lambda \in \mathbb{R}^\ell, \mu_j \geq 0 \ \forall j \in \mathcal{A} \right. \tag{70}$$

CQ conditions

Consequently: characterization of solutions of

$$D = \left\{ x \in \mathbb{R}^n \mid h(x) = 0, g(x) \leq 0 \right\} \tag{65}$$

translates into

There exists $(\lambda, \mu) \in \mathbb{R}^\ell \times \mathbb{R}^m$ such that KKT optimality conditions verified at $\tilde{x}$

$$\nabla f(\tilde{x}) + \sum_{i=1}^{\ell} \lambda_i \nabla h_i(\tilde{x}) + \sum_{j=1}^{m} \mu_j \nabla g_j(\tilde{x}) = 0, \tag{71}$$

$$h(\tilde{x}) = 0, g(\tilde{x}) \leq 0, \tag{72}$$

$$\mu \geq 0, \tag{73}$$

$$\mu_j = 0, \ \forall j \notin \mathcal{A}(\tilde{x}) \tag{74}$$

where $L : \mathbb{R}^n \times \mathbb{R}^\ell \times \mathbb{R}^m \to \mathbb{R}, L(x, \lambda, \mu) = f(x) + \langle \lambda \mid h(x) \rangle + \langle \mu \mid g(x) \rangle$

CQ conditions

- **Abadie CQ** : condition (68) is sufficient for condition (69), referred to as **Guignard CQ**, to hold but not necessary

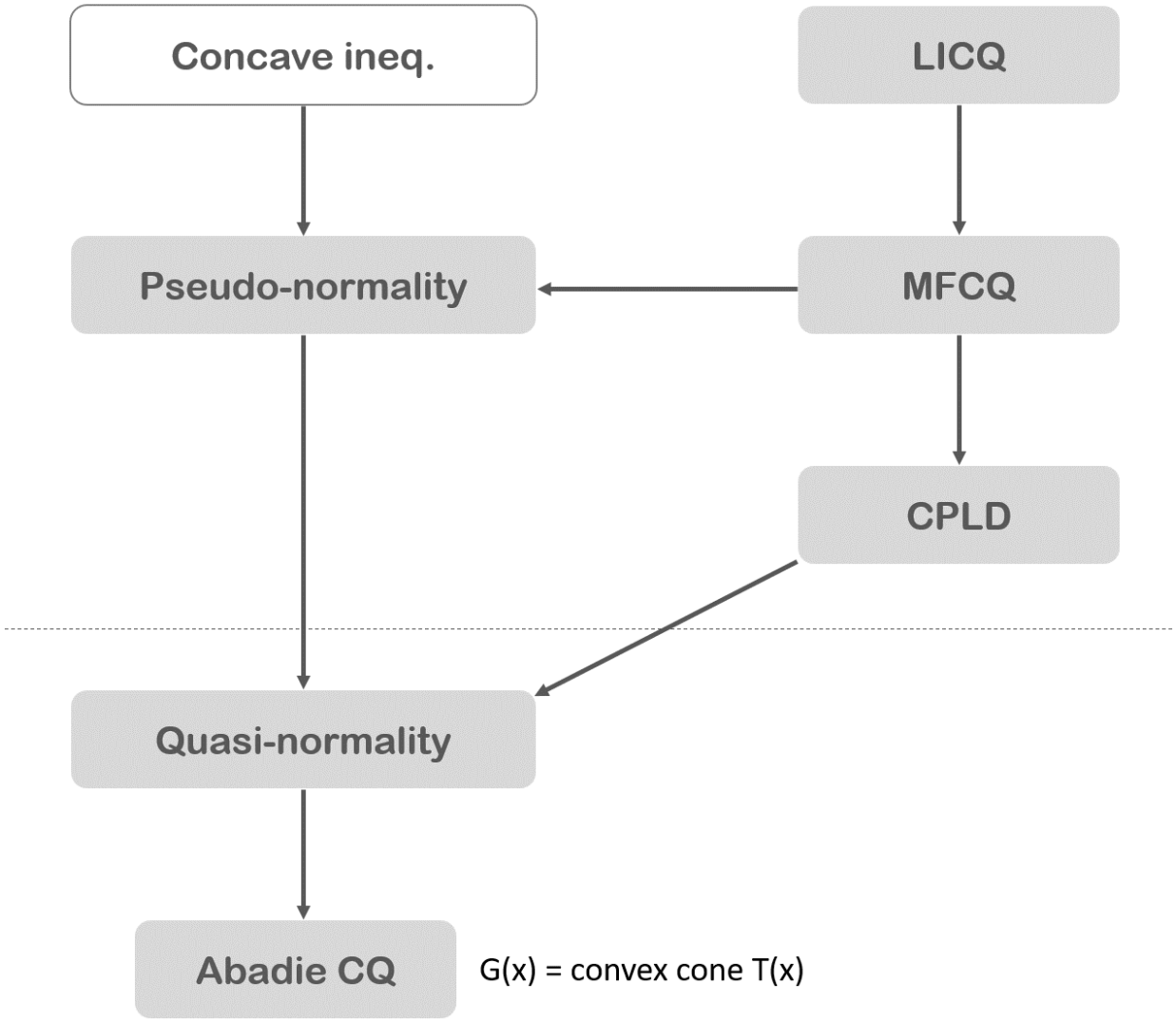
$$\mathcal{T}_D(\tilde{x}) = \mathcal{L}_D(\tilde{x}) \tag{68}$$

- **Guignard CQ** : weakest constraint qualification that ensures that KKT conditions are necessary optimality conditions for the associated optimization problem

$$(\mathcal{T}_D(\tilde{x}))^\circ = (\mathcal{L}_D(\tilde{x}))^\circ \tag{69}$$

- Important : neither Abadie CQ nor Guignard CQ are verifiable directly in general, since they require knowledge of $\mathcal{T}_D$ or its dual $(\mathcal{T}_D)^\circ$

CQ conditions



CQ conditions

LCQ : Linear CQ

- Assumes there exists a neighborhood U of $\tilde{x}$ such that $h_i, i = 1, \ldots, \ell$ and $g_j, \forall j \in \mathcal{A}(\tilde{x})$ are affine in U (thus, constraints are linear in a neighborhood U of $\tilde{x}$)
- In that case, $\mathcal{T}_D$ is the set of feasible directions at $\tilde{x}$

LICQ: Linear Independence CQ

- Imposes that the gradients of equality and **active** inequality constraints are linearly independent at $\tilde{x} \in D$: the set

$$\{\nabla h_i(\tilde{x}), i = 1, \ldots, \ell\} \cup \{\nabla g_j(\tilde{x}), j \in \mathcal{A}(\tilde{x})\}$$

  is linearly independent
- Ensures **existence**, **boundedness** and uniqueness of the KKT multipliers

CQ conditions

MFCQ : Mangasarian-Fromovitz CQ

- Assumes set $\{\nabla h_i(\tilde{x}), i = 1, \ldots, \ell\}$ is linearly independent and there exists a vector $d \in \mathbb{R}^n$ such that

$$\nabla h_i(\tilde{x})^T d = 0, \forall i \in 1, \ldots, \ell \text{ and } \nabla g_j(\tilde{x})^T d < 0, \ \forall j \in \mathcal{A}(\tilde{x})$$

- Equivalent dual form : $(\lambda, \mu) = 0$ unique solution of the linear system

$$\sum_{i=1}^{\ell} \lambda_i \nabla h_i(\tilde{x}) + \sum_{j \in \mathcal{A}(\tilde{x})} \mu_j \nabla g_j(\tilde{x}) = 0, \ \mu_j \geq 0 \ \forall j \in \mathcal{A}(\tilde{x})$$

- Ensures **existence** and **boundedness** -but not uniqueness- of KTT multipliers
- Stability : if MFCQ holds at $\tilde{x} \in D$ then there exists a neighborhood U of $\tilde{x}$ such that MFCQ holds at each $x \in D \cap U$

CQ conditions

QNCQ : Quasi-Normal CQ

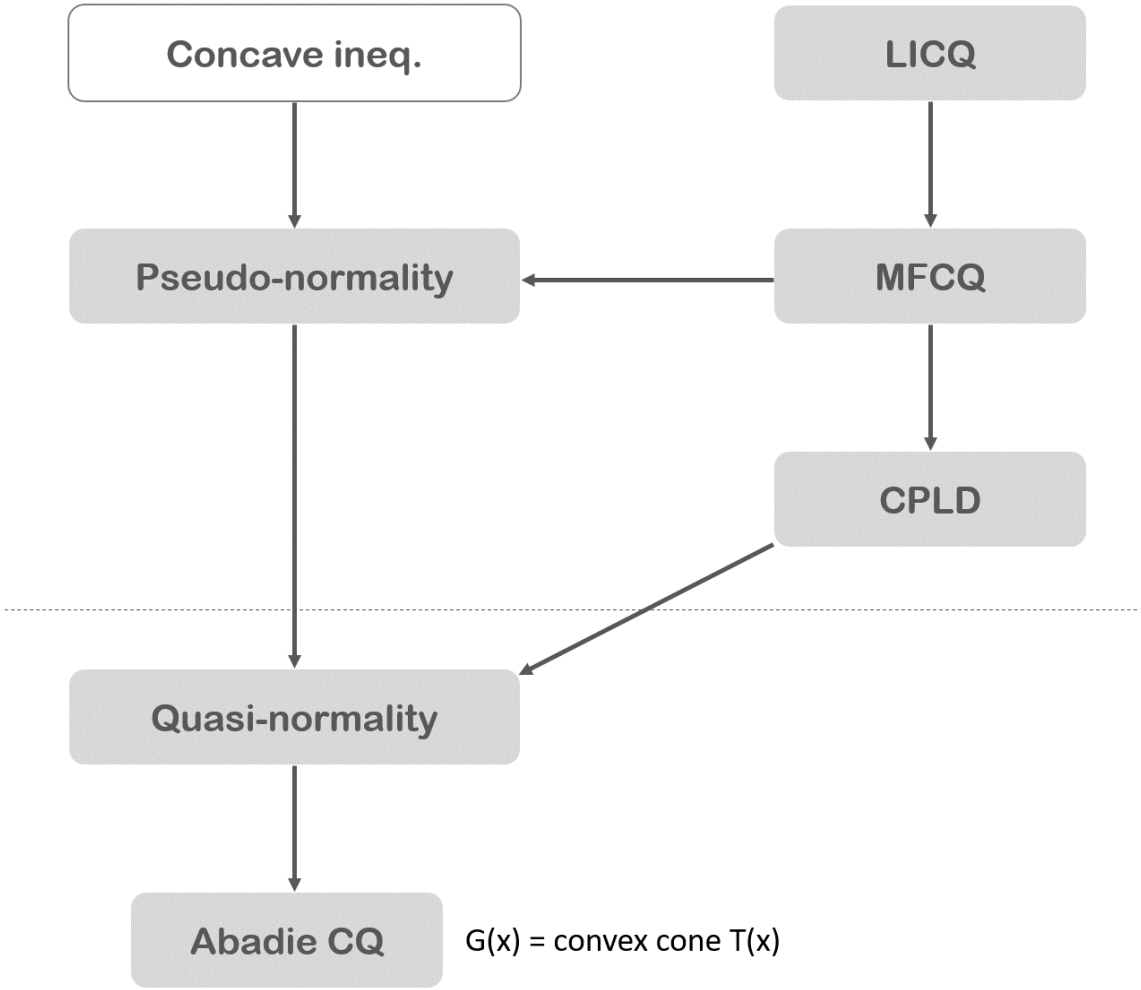
The point $\tilde{x} \in D$ is said to be quasinormal if there exist no nonzero $(\lambda, \mu) \in \mathbb{R}^{\ell} \times \mathbb{R}^{m}_{+}$ and no sequence $\{x_k\} \to \tilde{x}$ such that

$$\sum_{i=1}^{\ell} \lambda_i \nabla h_i(\tilde{x}) + \sum_{j=1}^{m} \mu_j \nabla g_j = 0$$

and for all k, $\lambda_i h_i(x_k) > 0, \forall i$ with $\lambda_i \neq 0$ and $\mu_j g_j(x_k) > 0, \forall j$ with $\mu_j \neq 0$

CQ conditions

Strict implication: LCQ $\Rightarrow$ LICQ $\Rightarrow$ MFCQ $\Rightarrow$ QNCQ



CQ conditions: remarks

Slater CQ

- **Assumptions**: each h_i is affine function, each g_j is convex function, and $\exists\, \hat{x} \in \mathbb{R}^n$ such that $h_i(\hat{x}) = 0$, $\forall i \in \{1, .., \ell\}$ and $g_j(\hat{x}) \leq 0$, $\forall j \in \{1, .., m\}$

- **If** g is convex differentiable and no equality constraints appear in the optimization problem

  Then Slater CQ is equivalent to MFCQ holding at every point $\tilde{x} \in D$

CQ conditions: remarks

Slater CQ

- **Assumptions**: each h_i is affine function, each g_j is convex function, and $\exists\, \hat{x} \in \mathbb{R}^n$ such that $h_i(\hat{x}) = 0$, $\forall i \in \{1, .., \ell\}$ and $g_j(\hat{x}) \leq 0$, $\forall j \in \{1, .., m\}$

- **If** g is convex differentiable and no equality constraints appear in the optimization problem

  Then Slater CQ is equivalent to MFCQ holding at every point $\tilde{x} \in D$

Note

- In the case the description of the set D does not contain inequality constraints both LICQ, MFCQ reduce to the classical regularity condition

$$\text{rank} J_h(\tilde{x}) = \ell$$

- Furthermore, the KKT conditions reduce in this case to the classical Lagrange optimality conditions