Refresher course on Numerical Matrix Analysis and Optimization

Franck Iutzeler & Jérôme Malick

2021

Contents

Chapter 1 Introduction **1**

Chapter 2 Matrix Analysis **5**
2.1 Matrices 5
2.2 Linear Systems 7
2.3 Spectral Decompositions 8
2.4 Matrix Norms 10

Chapter 3 Optimization **11**
3.1 Recalls on differentiation 11
3.2 What is optimization? 14
3.3 The gradient algorithm 17
3.4 To go further 19

Tutorial 1 Matrix Analysis **23**
1.1 Decompositions 23
1.2 Linear Systems Resolution with applications to Regression 23
1.3 PageRank 24

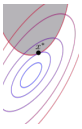
Tutorial 2 Optimization **27**
2.1 Using the definitions 27
2.2 Smoothness and Optimization 28

CHAPTER 1 INTRODUCTION

WHY do we have this preliminary course in numerical matrix analysis and optimization ?

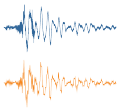
Matrix and optimization are at the **heart of computational mathematics**, with applications everywhere, e.g.

Machine Learning



Energy Management

Signal/Image Processing



	theory		**practice**
Mix between	*meaning of a problem; existence/uniqueness of solution; math properties*	and	*Computability, speed, and use of standard libraries to solve numerically these problems.*

This short course focuses on **matrix analysis and optimization in action** with exercises inspired from:

- Google PageRank, Image processing
- Machine learning applications (regression, classification)

This course is not a standard course on linear algebra or optimization

- not a math course
  basic knowledge is assumed *(take a look to your undergraduate courses)*
- not an algorithmic course
  basic programming skills are expected *(check-out the Python tutorial at* `https://github.com/iutzeler/refresher-course`*)*

This course is

- a review of basics of matrix analysis – from numerical perspective
- a short overview of numerical optimization
- includes quick recalls from matrix calculus and differential calculus

Contents

Matrix Analysis

1. Basics on matrices
    - Matrices and operations between matrices
    - Operations on matrices : transpose, trace, determinant
    - Special matrices (triangular, symmetric, orthogonal, invertible, SDP)
    - Decomposition : (P)LU, QR
2. Linear systems
    - Invertible systems, linear least-squares, linear least-norm
    - Easy systems for special matrices (triangular, orthogonal,...)
    - Solving systems : by factorization, by iterative methods, by optimization
    - Practical considerations (preconditioning, software,...)
3. Spectral decompostions
    - Eigenvalues : real, complex, spectral radius
    - Eigenvalue decomposition, geometric interpretation
    - Singular value decomposition : SVD, compact SVD, link with eigenvalues

\+ Note on matrix norms : standard norms, induced/operator norms, connection with spectral radius

Numerical optimization

1. Introduction : what is optimization ?
    - Optimization problems : definitions, examples, first properties
    - How to solve an optimization problem : exact/approximate solutions, difficult/impossible in general, "easy" for linear... and convex problems
2. Convexity and optimization
    - Convex sets and functions, examples
    - Convex optimization problems : global solutions, convex set of solutions
    - Recognizing convexity: definition, convexity-preserving operations, Hessian
3. The gradient algorithm
    - Unconstrained convex differentiable problems, optimality conditions
    - Convergence theory vs numerical experiments
    - Beyond gradient : acceleration, 2nd order, Newton

\+ Recalls on derivatives : gradient, Hessian, chain rule, examples.

Organization

Teachers

- Franck Iutzeler – Assistant Professor in Applied Maths
  http://www.iutzeler.org – *franck.iutzeler@univ-grenoble-alpes.fr*
- Gilles Bareilles – TA, PhD Candidate in Applied Maths
  http://gbareilles.fr – *gilles.bareilles@univ-grenoble-alpes.fr*

Course

This is mainly a blackboard course, with room for discussions and clarifications.

The main points of the course are recalled in Chapters 2 and 3.

Exercise sessions

They are present to give you the opportunity to manipulate the notions on simple examples. *no fancy maths and no computations - they will be done on machines*

You can find the exercises in Tutorials 1 and 2.

Practical sessions

In these, you will manipulate the objects seen in the course in Python.

The practical sessions are based on Jupyter notebooks available at `https://github.com/iutzeler/refresher-course`

Useful links & References

- Horn, R. & Johnson, C.: **Matrix analysis**.
- Boyd, S. & Vandenberghe, L.: **Convex optimization**.
- Rockafellar, R.T. & Wets, J.-B.: **Variational Analysis**.
- Bubeck, S.: **Convex Optimization: Algorithms and Complexity**, `https://arxiv.org/abs/1405.4980`.
- Hiriart-Urruty J.-B., Lemaréchal C.: **Fundamentals of convex analysis**.
- Stephen's Boyd website (check the courses, quizzes, and exercises) *http://web.stanford.edu/~boyd/*

CHAPTER 2 Matrix Analysis

Linear Algebra is at the core of most numerical methods. It plays an essential role in applied mathematics, especially in data science. In this chapter, we review the basic notions of matrix analysis from a numerical perspective.

2.1 Matrices

2.1.1 Matrix operations

A matrix is an m(lines/rows) $\times$ n(columns) array of real[1] numbers. The set of all $m \times n$ real matrices is denoted by $\mathbb{R}^{m\times n}$ (or sometimes $\mathcal{M}_{m\times n}(\mathbb{R})$). If $m = n$, the matrix is said to be square.

[1] in this course

$$A = (A_{ij})_{i=1,..,m;j=1,..,n}$$

Operations between matrices:

- Addition
- Product with a scalar
- Product between matrices

Special cases:

- Matrix/vector product
- Matrix powers

Operations on matrices:

- Transposition
- Trace
- Determinant

2.1.2 Special matrices

- Identity

$$I \in \mathbb{R}^{n\times n} : I_{ii} = 1 \text{ and } I_{ij} = 0 \text{ for } i \neq j$$

$AI = A$ for any A; $\det(I) = 1$; $\text{trace}(I) = n$

- Diagonal

$$D \in \mathbb{R}^{n\times n} : D_{ij} = 0 \text{ for } i \neq j$$

$\det(D) = \prod_i D_{ii}$

- Triangular

$$T \in \mathbb{R}^{n\times n} : T_{ij} = 0 \text{ for } i > j$$

$\det(T) = \prod_i T_{ii}$

- Symmetric

$$S \in \mathbb{R}^{n\times n} : S = S^\top$$

e.g. $A^\top A$ is symmetric.

- Orthogonal

$$Q \in \mathbb{R}^{n\times n} : Q^\top Q = I$$

the column of Q are unitary and orthogonal, e.g. permutation matrices.

- Invertible (a.k.a. nonsingular)

$$A \in \mathbb{R}^{n\times n} : \exists B \text{ s.t. } AB = BA = I$$

if B exists, it is unique and called A^{-1} the inverse of A.
A invertible $\Leftrightarrow \det(A) \neq 0$.
A, B invertible $\Rightarrow AB$ invertible with $(AB)^{-1} = B^{-1}A^{-1}$.

- positive (semi-)definite

$$A \in \mathbb{R}^{n\times n} : \text{symmetric } + x^\top Ax > 0 \ \ \forall x$$

semi-definite if the inequality is not strict. If A is positive definite, then it is invertible.

2.1.3 *Factorizations*

- LU

$$\text{For } A \text{ invertible, } A = PLU$$

with P a permutation, L lower-triangular, U (upper-)triangular.

- Cholesky

$$\text{For } A \text{ positive definite, } A = LL^\top$$

with L lower-triangular.

- QR

$$\text{For any } A, A = QR$$

with Q orthogonal, R (upper-)triangular.

2.2 Linear Systems

2.2.1 Linear equations

Let $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$, the problem of finding $x \in \mathbb{R}^n$ such that $Ax = b$ appears as a subproblem in virtually all branches of numerical mathematics (ODE, optimization, statistics, etc.).

This problem can be seen as finding the coefficients x of a linear combination of the columns of A giving b.

Vocabulary reminder:

- The *rank* of A is the number of linearly independent columns
- The set $Im(A) = \{Ax : x \in \mathbb{R}^n\} \subset \mathbb{R}^m$ is called the *image* of A
- The set $Ker(A) = \{x \in \mathbb{R}^n : Ax = 0\} \subset \mathbb{R}^n$ is called the *kernel* of A
- If $m = n$ and (rank$(A) = n$ or $Im(A) = \mathbb{R}^n$ or $Ker(A) = \{0\}$), A is *invertible* i.e. there is an inverse matrix A^{-1} such that $AA^{-1} = A^{-1}A = I$

Different situations can arise:

- There might might be *no* solutions (if $Im(A) \neq \mathbb{R}^m$) $\Rightarrow$ change the problem to least-squares $\min_x \|Ax - b\|_2$
- There might might be *multiple* solutions (if $Ker(A) \neq \{0\}$) $\Rightarrow$ change the problem to least norm $\min_x \|x\|$ s.t. $\|Ax - b\|_2$
- If A is invertible, $x = A^{-1}b$ is the unique solution

How to *compute* a solution to one of the above problems?

2.2.2 "Easy" cases

The problem of finding x such that $Ax = b$ is easy for certain matrices:

- If you *have the inverse* A^{-1}; however computing A^{-1} is roughly n times harder than solving $Ax = b$ so this should never be done numerically[2].
- Diagonal

[2] Use `solve(A,b)` and not `Ainv = inv(A); Ainv.dot(b)`.

- Triangular

- Orthogonal

If we are not in an easy case, we need to be a bit more inventive.

2.2.3 3 types of algorithms for solving linear systems

1. *using pre-computed factorizations*
   If $A = A_1 A_2$ eg. LU or QR, then

$$\begin{aligned} & Ax = b \\ \Leftrightarrow & A_1 \underbrace{A_2\, x}_{:=y} = b \\ \Leftrightarrow & \begin{cases} A_1\, y & = b \\ A_2\, x & = y \end{cases} \end{aligned}$$

2. *iterative methods* (fixed-point algorithm)
   Choose m, n such that $A = M - N$
   eg. for $A = D$(diagonal) $+L$ (lower part) $+U$ (upper part), Jacobi takes $M = D$ and $N = -L - U$, Gauss-Siedel takes $M = D + L$ and $N = -U$.
   Then

$$\begin{aligned} & Ax = b \\ \Leftrightarrow & (M - N)x = b \\ \Leftrightarrow & Mx = Nx + b \end{aligned}$$

   Initialize x_0 and iteratively set x_{k+1} as a solution of $Mx = Nx_k + b$ (M should be easy)

3. *optimization methods*

$$\begin{aligned} & Ax = b \\ \Leftrightarrow & Ax - b = 0 \end{aligned}$$

   Find x that minimizes $\|Ax - b\|$ (see in the second part).

So what to do in practice?

- In most cases, trust your solver !
- Preconditionning might help: solve $MAx = Mb$ instead with M "reducing the difference between columns/lines" (eg. standard scaler in ML)
- Otherwise, use optimization/iterative methods and use the structure of your matrix

2.3 Spectral Decompositions

2.3.1 Eigenvalues

[3] eigenvalues are complex in general even if A is real

For a *squared* matrix $A \in \mathbb{R}^{n \times n}$, a scalar $\lambda \in \mathbb{R}$ or $\mathbb{C}$ is an eigenvalue of A if[3]

$$\exists x \neq 0 \quad Ax = \lambda x$$

and a vector x satisfying the above relation is called an *eigenvector* (associated with λ). It is not unique, and globally means that in the direction of x, A is simply a dilatation. The set of all x such that $Ax = \lambda x$ (0 included) is called the *eigenspace* associated with λ.

There are between 1 and n *distinct* eigenvalues; an important quantity being the *spectral radius* defined as $\rho(A) = \max\{|\lambda| : \lambda \text{ is an eigenvalue of } A\}$.

- There are many applications (dynamical systems, graphs, image processing, etc.) and lot of theory surrounding eigenvalues
- $Ax = \lambda x \Leftrightarrow (A - \lambda I)x = 0 \Leftrightarrow x \in Ker(A - \lambda I)$

2.3.2 Eigenvalue decomposition for symmetric matrices

For a *symmetric square* matrix $A \in \mathbb{R}^{n \times n}$, one can say a bit more.

For each λ, such that $\exists x \neq 0 \quad Ax = \lambda x \Leftrightarrow x \in Ker(A - \lambda I)$ the associated eigenspace has dimension $m_\lambda = n - \text{rank}(A - \lambda I)$, called the (geometric) multiplicity of λ, and thus one can extract m_λ independent eigenvectors from it.

In the symmetric case, there are *exactly n* eigenvalues counting multiplicities[4] (that take between 1 and *n distinct* values) and A admits an *eigendecomposition*, that is

$$A = PDP^\top = \sum_{i=1}^{n} \lambda_i p_i p_i^\top$$

[4] If A is not symmetric, the sum of all (geometric) multiplicities may not be n and such a decomposition may not exist.

where $P \in \mathbb{R}^{n \times n}$ is orthogonal and $D \in \mathbb{R}^{n \times n}$ is diagonal. D contains the n eigenvalues of A and P contains n associated eigenvectors (orthonormalized, with vector 1 corresponding to eigenvalue 1 etc., without any particular order for the eigenvalues).

- The eigenvalues of a symmetric matrix are real

  Proof. $\lambda\langle x; x\rangle = \langle \lambda x; x\rangle = \langle Ax; x\rangle = \langle x; Ax\rangle = \langle x; \lambda x\rangle = \overline{\langle \lambda x; x\rangle} = \overline{\lambda}\langle x; x\rangle$ □

- If A is furthermore positive (semi-)definite, the eigenvalues are positive (non-negative)

  Proof. $0 < \langle Ax; x\rangle = \lambda \|x\|^2$ □

- in the non-symmetric case, the *singular value decomposition* is often used, it is linked to the eigendecomposition of the symmetric matrix $A^\top A$

2.3.3 Singular Value Decomposition (SVD)

A real matrix $A \in \mathbb{R}^{m \times n}$ admits a singular value decomposition

$$A = UDV^\top = \sum_{i=1}^{\min\{m,n\}} \sigma_i u_i v_i^\top$$

where $U \in \mathbb{R}^{m \times m}$ and $V \in \mathbb{R}^{n \times n}$ are orthogonal and $D \in \mathbb{R}^{m \times n}$ (same dimension as A) is "diagonal" with non-negative entries (σ_i).

- The (σ_i) are the *singular values* of A. Thy are related to the eigenvalues of $A^\top A$:

$$\begin{aligned} A^\top A &= VD^\top U^\top UDV^\top \\ &= VD^\top DV^\top \end{aligned}$$

  thus $\sigma_i(A)^2 = \lambda_i(A^\top A)$ (≥ 0 and real since $A^\top A$ is positive semi-definite)

- The rank of A is equal to the number r of non-zero singular values of A, i.e. $r = \text{rank}(A) \leq \min\{m, n\}$.

- In practice, we do not need information (vectors) corresponding to null singular values. It is common to use[5] the *reduced/compact* SVD

$$A = \sum_{i=1}^{\operatorname{rank}(A)} \sigma_i u_i v_i^\top = \tilde{U}\tilde{D}\tilde{V}^\top$$

where $\tilde{U} \in \mathbb{R}^{m\times r}$ and $\tilde{V} \in \mathbb{R}^{r\times n}$ have orthonormal columns and $\tilde{D} \in \mathbb{R}^{r\times r}$ is diagonal with positive entries (σ_i).

[5] In Python, this is done with the option `full_matrices=False`

2.4 Matrix Norms

- Norms in $\mathbb{R}^n$
  - $\|\alpha x\| = |\alpha|\|x\|$ (homogeneous)
  - $\|x + y\| \leq \|x\| + \|y\|$ (triangle inequality or sub-additivity)
  - $\|x\| \geq 0$ and $\|x\| = 0 \Leftrightarrow x = 0$ (definition)
- *Matrix Norms* have to verify the additional property[6]
  - $\|AB\| \leq \|A\|\|B\|$ (sub-multiplicativity for squared matrices)
- Typical matrix norms include:
  - the Frobenius norm

$$\|A\|_F = \sqrt{\sum_{i,j=1}^{m,n} |A_{i,j}|^2} = \sqrt{\operatorname{trace} A^\top A} = \sqrt{\sum_{i=1}^{\min\{m,n\}} \sigma_i^2(A)}$$

  - Induced/Operator norms

$$\|A\|_{\alpha,\beta} = \sup\left\{\frac{\|Ax\|_\beta}{\|x\|_\alpha} : x \neq 0\right\}$$

    where for $A \in \mathbb{R}^{m\times n}$, $\|\cdot\|_\alpha$ is a norm on $\mathbb{R}^n$, $\|\cdot\|_\beta$ is a norm on $\mathbb{R}^m$. For such a norm, we have $\|Ax\|_\beta \leq \|A\|_{\alpha,\beta}\|x\|_\alpha$. The most common case is $\|A\|_{2,2} = \sigma_{\max}(A)$.
  - Schatten norms

$$\|A\|_{*p} = \left(\sum_{i=1}^{\min\{m,n\}} \sigma_i^p(A)\right)^{\frac{1}{p}}$$

    They are vector norms on the singular values of A. $\|A\|_{*\infty} = \|A\|_{2,2}$ and $\|A\|_{*1}$ (also called nuclear norm) are the most frequently used.
- Gelfand's formula: For any matrix norm, $\rho(A) = \lim_{k\to\infty} \|A^k\|^{1/k}$

[6] One can also get norms on matrices by taking the norm of its entries viewed as a vector, but it may not be sub-mutliplicative eg. $\|x\|_\infty = \max_i |x_i|$.

CHAPTER 3 Optimization

Optimization is everywhere in science and industry. It is at the core of the most recent advances in Machine Learning

3.1 Recalls on differentiation

Differentiability plays a central role in optimization. This is somehow a special case of the notion of subgradient defined above but the treatment of differentiable functions will be rather different algorithmically. In order to promote even more this difference, we will adopt the following convention for the name of generic functions: (*i*) f if it is differentiable; (*ii*) g if it is not assumed differentiable; and (*iii*) f if the differentiability does not play a role in the result.

3.1.1 Derivative of a function from $\mathbb{R}$ to $\mathbb{R}$

In this basic case, the notion of differentiability is quite direct.

Definition 3.1. A function $f : \mathcal{V} \subset \mathbb{R} \to \mathbb{R}$ defined on a open subset[7] $\mathcal{V}$ of $\mathbb{R}$ is differentiable at $x \in \mathcal{V}$ if the derivative (ie. the limit)

$$f'(x) := \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

exists. This function f is differentiable on $\mathcal{V}$ if it is differentiable at every point of $\mathcal{V}$.

[7] At first read, you can take $\mathcal{V}$ as the full space to fix ideas

This definition is equivalent to the existence of a real number $f'(x)$ such that

$$f(x+h) = f(x) + f'(x)h + o(|h|).$$

Note that we now only consider an open subset of $\mathbb{R}$ over which the function is finite-valued. If f takes infinite values on any open set containing x, then it cannot be differentiable at that point.

In addition, if f is differentiable at x, it is necessarily continuous at x. The derivative f' is itself a function from $\mathcal{V} \to \mathbb{R}$ and may also be continuous (on $\mathcal{V}$), in which case, we say that f is continuously differentiable, often denoted $C^1(\mathcal{V})$ or simply C^1.

The derivative of the derivative is called the second-order derivative, noted f''. If it exists and is continuous, we say that f is C^2. Iterating, we can easily define higher order derivatives and differentiability classes up to C^∞.

3.1.2 Gradient of a function from $\mathbb{R}^n$ to $\mathbb{R}$

Let us now consider a function defined over an open subset $\mathcal{V}$ of $\mathbb{R}^n$

$$\begin{array}{rccc} f: & \mathcal{V} \subset \mathbb{R}^n & \longrightarrow & \mathbb{R} \\ & x = [x_1, .., x_n] & \longmapsto & f(x) \end{array} .$$

For every $x \in \mathcal{V}$, the i-th *partial function* is defined on $\mathcal{V}' \subset \mathbb{R}$ as

$$\begin{array}{rccc} \phi_{i,x}: & \mathcal{V}' & \longrightarrow & \mathbb{R} \\ & u & \longmapsto & f(x_1, .., x_{i-1}, u, x_{i+1}, .., x_n) \end{array} ,$$

and since this function falls into the case of the previous section, we can study its differentiability. If for all i, $\phi_{i,x}$ is differentiable at x_i, then, we will say that f is differentiable at x.

Definition 3.2. A function $f : \mathcal{V} \subset \mathbb{R}^n \to \mathbb{R}$ defined on a open subset $\mathcal{V}$ of $\mathbb{R}^n$ is differentiable at $x \in \mathcal{V}$ if for all $i = 1, .., n$, the derivative (ie. the limit)

$$\frac{\partial f}{\partial x_i}(x) := \lim_{h \to 0} \frac{\phi_{i,x}(x_i + h) - \phi_{i,x}(x_i)}{h}$$

exists. This function f is differentiable on $\mathcal{V}$ if it is differentiable at every point of $\mathcal{V}$. Further, if f is differentiable on $\mathcal{V}$, we define its *gradient* as the $\mathcal{V} \subset \mathbb{R}^n \to \mathbb{R}^n$ mapping

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(x) \\ \vdots \\ \frac{\partial f}{\partial x_n}(x) \end{bmatrix}.$$

Similar to what was obtained in the one-dimensional case, we have a *first-order* development of f at a point x at which f is differentiable:

$$f(x + h) = f(x) + \langle \nabla f(x), h \rangle + o(\|h\|).$$

Remark 3.3. This development can actually be used to compute the gradient itself. Indeed if one finds a vector v such that

$$f(x + h) = f(x) + \langle v, h \rangle + o(\|h\|),$$

then v is exactly the gradient of f at x: $\nabla f(x)$. ◂

3.1.3 Jacobian of a mapping $\mathbb{R}^m$ to $\mathbb{R}^n$

Now, let us consider the case of a mapping (ie. a multi-valued function) from $\mathbb{R}^m$ to $\mathbb{R}^n$

$$\begin{array}{rccc} c: & \mathcal{V} \subset \mathbb{R}^m & \longrightarrow & \mathbb{R}^n \\ & x = [x_1, .., x_m] & \longmapsto & c(x) = [c_1(x), .., c_n(x)] \end{array} .$$

A mapping is differentiable if and only if each of its *component functions* is differentiable as formalized in the following definition.

Definition 3.4. A mapping $c : \mathcal{V} \subset \mathbb{R}^m \to \mathbb{R}^n$ defined on a open subset $\mathcal{V}$ of $\mathbb{R}^m$ is differentiable at $x \in \mathcal{V}$ if for all $i = 1, .., n$, and all $j \in 1, .., m$, the derivative $\frac{\partial c_i}{\partial x_j}(x)$ exists. This mapping c is differentiable on $\mathcal{V}$ if it is differentiable at every point of $\mathcal{V}$.

Further, if c is differentiable on $\mathcal{V}$, we define its *Jacobian* as the $\mathcal{V} \subset \mathbb{R}^m \to \mathbb{R}^n \times \mathbb{R}^m$ mapping[8]

$$Jc(x) = \begin{bmatrix} \nabla c_1(x)^\top \\ \vdots \\ \nabla c_n(x)^\top \end{bmatrix} = \begin{bmatrix} \frac{\partial c_1}{\partial x_1}(x) & \dots & \frac{\partial c_1}{\partial x_m}(x) \\ \vdots & \ddots & \vdots \\ \frac{\partial c_n}{\partial x_1}(x) & \dots & \frac{\partial c_n}{\partial x_m}(x) \end{bmatrix}.$$

[8] The name comes from Carl Gustav Jacob Jacobi (1804-1851), a German mathematician.

While, we do not often differentiate mappings, we often differentiate compositions of a function and mapping. For this, the *chain rule* gives a efficient formula based on the respective gradients and Jacobian of the functions.

Lemma 3.5 (Chain rule)**.** *Take a function $f : \mathcal{V}' \subset \mathbb{R}^n \to \mathbb{R}$ and a mapping $c : \mathcal{V} \subset \mathbb{R}^m \to \mathbb{R}^n$. If c is differentiable at $x \in \mathcal{V}$ and f is differentiable at $c(x) \in \mathcal{V}'$, then $f \circ c$ is differentiable at x and its gradient can be obtained by*[9]

$$\nabla f \circ c(x) = Jc(x)^\top \nabla f(c(x)). \qquad \text{(Chain rule)}$$

[9] $f \circ c(x) = f(c(x))$

The first-order development of $f \circ c$ is thus

$$f \circ c(x + h) = f \circ c(x) + \langle Jc(x)^\top \nabla f(c(x)), h \rangle + o(\|h\|).$$

3.1.4 Second-order differentiability

The derivative of the gradient, that is the second-order derivative of the function, is often used in numerical optimization methods.

Definition 3.6. A function $f : \mathcal{V} \subset \mathbb{R}^n \to \mathbb{R}$ defined on a open subset $\mathcal{V}$ of $\mathbb{R}$ is twice differentiable at $x \in \mathcal{V}$ if its gradient is differentiable at $x \in \mathcal{V}$.
Further, if f is twice differentiable on $\mathcal{V}$, we define its *Hessian* as the $\mathcal{V} \subset \mathbb{R}^n \to \mathbb{R}^n \times \mathbb{R}^n$ mapping[10]

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

[10] also denoted by Hf, its name comes from Ludwig Otto Hesse (1811-1874), a German mathematician.

This definition comes with the following important property.

Lemma 3.7. *The Hessian of a function $f : \mathcal{V} \subset \mathbb{R}^n \to \mathbb{R}$ at $x \in \mathcal{V}$ is a symmetric matrix.*

Proof. This follows directly from Schwarz's theorem.[11] □

[11] Hermann Schwarz (1843-1921), German mathematician, was the first to propose a rigorous proof of the symmetry of second derivatives (also called the equality of mixed partials).

3.1.5 Fréchet derivatives [⋆]

The notion of Fréchet derivatives generalizes the notion of gradient and Jacobian seen above. A mapping $c : \mathcal{V} \subset \mathbb{R}^m \to \mathbb{R}^n$ defined on a open subset $\mathcal{V}$ of $\mathbb{R}^m$ is *Fréchet differentiable* at $x \in \mathcal{V}$ if there exists a linear operator

$$\begin{array}{rccc} \mathrm{D}\, c(x) : & \mathbb{R}^m & \longrightarrow & \mathbb{R}^n \\ & h & \longmapsto & \mathrm{D}\, c(x)[h] \end{array}$$

called the (Fréchet) *differential* of c at x,[12] such that

$$c(x+h) = c(x) + \mathrm{D}\,c(x)[h] + o(\|h\|)$$
$$\text{or, equivalently } \lim_{h\to 0} \frac{\|c(x+h) - c(x) - \mathrm{D}\,c(x)[h]\|}{\|h\|} = 0.$$

[12] from Maurice René Fréchet (1878-1973), a French mathematician.

Then, if f is a $\mathcal{V} \subset \mathbb{R}^n \to \mathbb{R}$ function, the gradient of f can be defined as the unique element of $\mathbb{R}^n$ that satisfies

$$\mathrm{D}\,f(x)[h] = \langle \nabla f(x), h\rangle \text{ for all } h \in \mathbb{R}^n$$

and thus, it can also be defined as

$$\nabla f(x) = \{v : f(u) = f(x) + \langle v, u-x\rangle + o(\|u-x\|) \text{ for all } u \in \mathbb{R}^n\}.$$

The same can be done for mappings and the Jacobian of c can be defined as the unique $\mathbb{R}^n \times \mathbb{R}^m$ operator $Jc(x)$ such that $\mathrm{D}\,c(x)[h] = Jc(x)h$.

Finally, the Chain rule for differentials is

$$\mathrm{D}(f\circ c)(x)[h] = \mathrm{D}\,f(c(x))[\mathrm{D}\,c(x)[h]] = \langle \nabla f(c(x)), Jc(x)h\rangle = \langle Jc(x)^\top \nabla f(c(x)), h\rangle.$$

3.2 What is optimization?

3.2.1 Optimization problems

Given $f : \mathbb{R}^n \to \mathbb{R}$, the *objective function*, and $C \subset \mathbb{R}^n$, the *constraint set*, the problem of minimizing a function f over a set C consists in finding $x^\star \in C$ such that $f(x^\star) \leq f(x)$ for all $x \in C$. We note

$$x^\star \in \operatorname{argmin}_C f \Leftrightarrow x^\star \text{ is a solution of } \inf_{x\in C} f(x)$$

for $x^\star$ a solution of the problem

$$\begin{cases} \min f(x) \\ x \in C \end{cases} \tag{$\mathcal{P}$}$$

[13] *infeasible* in the optimization language.

We directly note that if C is empty, the problem is impossible[13] and if C is open it may be impossible to find a solution. Hence, we will restrict our analysis to closed sets.

How to solve optimization problems?

First, what does solving means? Finding $f^\star$, $x^\star$, or an approximation of one/both? It is extremely rare to have explicit/exact solutions so most of the time numerical methods are used to approximate them.

In general, optimization is a (NP-)hard problem but there are situations that are more favorable than others eg. linear programs, convex problems.

3.2.2 Convex sets

Let us now introduce the definition of a convex set.

Definition 3.8. A subset C of $\mathbb{R}^n$ is convex if and only if for any $x, u \in C$, $(1-\alpha)x+\alpha u \in C$ for any $\alpha \in (0,1)$.

The crucial property here is that any (weighted) average of points of a convex set belongs stay in the set. Equivalently, the set C is convex if and only if for any $(x_1, .., x_N) \in C^N$,

$$\sum_{i=1}^{N} \alpha_i x_i \in C \text{ for any } (\alpha_1, .., \alpha_N) \in \mathbb{R}_+^N \text{ with } \sum_{i=1}^{N} \alpha_i = 1,$$

where $\sum_{i=1}^N \alpha_i x_i$ is called a *convex combination* of $(x_1, .., x_N)$.

Examples of convex sets:

- Affine spaces $\{x : \langle s, x\rangle = r\}$
- Balls $\{x : \|x - s\| \leq r\}$
- Half spaces $\{x : \langle s, x\rangle \leq r\}$ and open half spaces $\{x : \langle s, x\rangle < r\}$
- Simplices $\{x : \sum_{i=1}^n x_i = 1 \text{ and } x_i \geq 0 \text{ for all } i = 1, .., n\}$
- Intersections of convex sets $\cap_{i=1}^N C_i$

Examples of non-convex sets:

- Discrete sets (eg. $\{0\} \cup \{1\}$) or disjoint sets
- Spheres $\{x : \|x - s\| = r\}$
- Sets with "holes"

Theorem 3.9. *Let C be a closed nonempty convex set. Then, for any $y \in \mathbb{R}^n$, there is a unique projection* $\text{proj}_C(y)$*, solution of* $\min_{x\in C} \|x - y\|^2$.
Furthermore, $\text{proj}_C(y)$ *is the projection of y onto C if and only if*

$$\langle y - \text{proj}_C(y), z - \text{proj}_C(y)\rangle \leq 0 \text{ for all } z \in C.$$

3.2.3 Minimization over convex sets

First, if $C = \mathbb{R}^n$, Fermat's rule indicates that critical points (ie. points at which the gradient is null) are good candidates to be (local) minimizers.

Theorem 3.10 (Fermat's rule)**.** *If a differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ has a local minimum at x (ie. if there is a neighborhood $\mathcal{U}$ of x such that $f(x) \leq f(u)$ for all $u \in \mathcal{U}$) then $\nabla f(x) = 0$.*

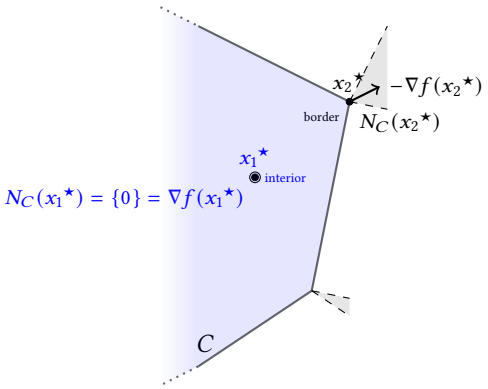
The *constrained* variant of Fermat's rule writes as follows.

Theorem 3.11 (Fermat's rule – constrained case)**.** *If a differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ has a local minimum at x then $0 \in \nabla f(x) + N_C(x)$ or,[14] equivalently,*

$$\langle y - x, \nabla f(x)\rangle \geq 0 \quad \text{for all } y \in C.$$

[14] *The* normal cone *of a convex set C at a point $x \in C$ is defined as the set $N_C(x) := \{u : \langle y - x, u\rangle \leq 0 \text{ for all } y \in C\}$.*

Note that if x belongs to the relative interior of C, then $N_C(x) = \{0\}$.



3.2.4 Convex functions

The notion of convexity is as important for functions as for sets. Notably, this is the notion that will enable us to go from the (sub)gradient inequalities and local minimizers above to *global* minimizers.

[15] This is the set $\mathrm{epi} f := \{(x,t) : f(x) \leq t\}$

A function is convex if and only if its *epigraph*[15] is convex. However, the following definition is much more direct.

Definition 3.12. A function $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ is convex if and only if for any $x, u \in \mathrm{dom}\, f$, $f((1-\alpha)x + \alpha u) \leq (1-\alpha)f(x) + \alpha f(u)$ for any $\alpha \in (0,1)$.

More generally convex functions verify *Jensen's inequality*. For any convex combination $\sum_{i=1}^N \alpha_i x_i$,

$$f\left(\sum_{i=1}^N \alpha_i x_i\right) \leq \sum_{i=1}^N \alpha_i f(x_i).$$

Checking the definition directly may be possible but it is often simpler to rely on convexity-preserving operations:

- all norms are convex;
- a sum of convex functions is convex;
- affine substitution of the argument (if f is convex, $x \mapsto f(Ax+b)$ is convex for any affine map $Ax+b$);
- the (pointwise) maximum of convex functions is convex.

The most striking point of convex functions is that local minimizers are actually global.

Theorem 3.13. *Consider a differentiable function $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ and a convex set C. Then, $x \in \mathrm{argmin}_C f$ if and only if $0 \in \nabla f(x) + N_C(x)$ which means that*

$$\langle y - x, \nabla f(x) \rangle \geq 0$$

for all $y \in C$.

In addition, for a differentiable f, convexity can be seen directly as a property on the gradient mapping.

[16] *typically here,* $\mathrm{dom}\, f = \mathbb{R}^n$.

Theorem 3.14. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a differentiable function with open domain.*[16] *Then the following are equivalent:*

i) f *is convex;*

ii) $f(u) \geq f(x) + \langle \nabla f(x), u - x \rangle$ *for all* $x, u \in \operatorname{dom} f$*;*

iii) $\langle \nabla f(x) - \nabla f(u), x - u \rangle \geq 0$ *for all* $x, u \in \operatorname{dom} f$*, ie.* ∇f *is monotone.*

Furthermore, if f *is twice differentiable on* $\operatorname{dom} f$*, any of the above is equivalent to*

iv) $\langle u, \nabla^2 f(x) u \rangle \geq 0$ *for all* $x, u \in \operatorname{dom} f$*, ie.* $\nabla^2 f$ *is positive semi-definite.*

3.3 The gradient algorithm

We are now given a (first-order) *oracle* $x \rightsquigarrow (f(x); \nabla f(x))$ and want to construct iteratively a sequence (x_k) such that $x_k \to x^\star$ (in some sense).

3.3.1 Smoothness

In addition to differentiability, smoothness is important in terms of optimization since it allows us to have a quadratic upper approximation of our function, obtained directly from the fundamental theorem of calculus. This is the crucial point for the use of gradient methods.

Definition 3.15. We say that a function $f : \mathbb{R}^n \to \mathbb{R}$ is L-smooth if it has a L-Lipschitz continuous gradient, ie. if

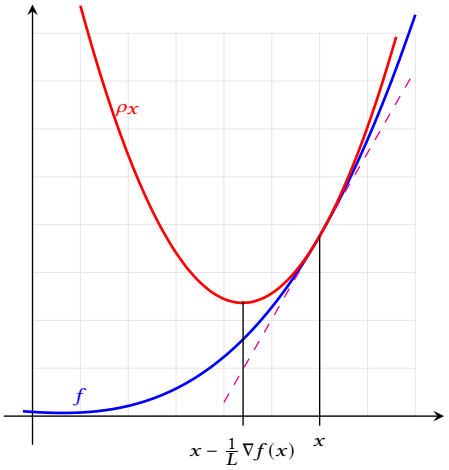
$$\|\nabla f(x) - \nabla f(u)\| \leq L\|x - u\| \text{ for all } x, u \in \mathbb{R}^n.$$

From this property, we can derive this highly important lemma.

Lemma 3.16. *Consider a function* $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ *with a L-Lipschitz continuous gradient, then for any* $x, u \in \mathbb{R}^n$*, one has*

$$|f(u) - f(x) - \langle \nabla f(x), u - x \rangle| \leq \frac{L}{2}\|x - u\|^2.$$

Thus, if we fix a point x, the function $\rho_x : u \mapsto f(x) + \langle \nabla f(x), u - x \rangle + \frac{L}{2}\|u - x\|^2$ is quadratic in its argument and majorizes f, that is to say $\rho_x(u) \geq f(u)$ for any u. Furthermore, the minimum of ρ_x is attained at $x^\star = x - \frac{1}{L}\nabla f(x)$.



Such a quadratic approximation can be leveraged using gradients steps, ie. taking

$$u = x - \gamma \nabla f(x)$$

for some $\gamma > 0$. Indeed, in that case, Lemma 3.16 gives us

$$f(u) \leq f(x) - \left(\frac{1}{\gamma} - \frac{L}{2}\right) \|x - u\|^2 = f(x) - \left(\gamma - \frac{L\gamma^2}{2}\right) \|\nabla f(x)\|^2.$$

Thus, taking a gradient step leads to a strict functional decrease ($f(u) < f(x)$) as soon as $\gamma < 2/L$. This is the core idea behind the *gradient descent* algorithm.[17] Take $x_0 \in \mathbb{R}^n$ and $\gamma > 0$, the gradient descent algorithm consists in iterating

$$x_{k+1} = x_k - \gamma \nabla f(x_k) \qquad \text{(Gradient descent)}$$

and leads to the following guarantees.

[17] introduced by Louis Augustin Cauchy (1789–1857), a French mathematician, in his "Compte Rendu à l'Académie des Sciences" of October 18, 1847.

Theorem 3.17. *Consider a function $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ with a L-Lipschitz continuous gradient and such that $\inf f > -\infty$. Assume that (Gradient descent) is run with $0 < \gamma < 2/L$, then $(f(x_k))$ converges and any limit point $\overline{x}$ of (x_k) satisfies $\nabla f(\overline{x}) = 0$.*

Thus, the functional values are decreasing and all limit points are critical points of f. However, we had no convergence guarantee and no rate. Convexity will help us get these rates.

3.3.2 Gradient algorithm for convex functions

When f is L-smooth and convex, we can guarantee convergence and a $O(1/k)$ rate.

Theorem 3.18. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex L-smooth function. Then, the iterates (x_k) generated by (Gradient descent) with $\gamma = 1/L$ satisfy:*

- *(convergence) $x_k \to x^\star$ for some minimizer $x^\star$ of f;*
- *(rate) $f(x_k) - f(x^\star) \leq \dfrac{2L\|x_0 - x^\star\|^2}{k}$ for any minimizer $x^\star$ of f.*

In the above theorem, any $\gamma \in (0, 1/L)$ actually works for the convergence and gets a similar complexity but $\gamma = 1/L$ is the optimal value in terms of rate.

3.3.3 Projected Gradient algorithm

Now let us consider the problem of minimizing a smooth convex function F over a nonempty closed convex set C. Thanks to the ability to project onto C, we can easily define a projected gradient method:

$$x_{k+1} = \mathrm{proj}_C\left(x_k - \gamma \nabla f(x_k)\right) \qquad \text{(Projected gradient descent)}$$

for some initialization $x_0 \in \mathbb{R}^n$ and stepsize $\gamma > 0$.

Theorem 3.19. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex L-smooth function. Then, the iterates (x_k) generated by (Gradient descent) with $\gamma = 1/L$ belong to C and satisfy:*

- *(convergence) $x_k \to x^\star$ for some minimizer $x^\star$ of f on C;*
- *(rate) $f(x_k) - f(x^\star) \leq \dfrac{3L\|x_0 - x^\star\|^2 + f(x_0) - f(x^\star)}{k+1}$ for any minimizer $x^\star$ of f on C.*

3.4 To go further

3.4.1 Beyond gradient descent

The gradient method, as the archetype of first-order methods, has several advantages but also many drawbacks (slowness, global parameters), to get over them:

- More advanced methods
  - Nesterov's accelerated gradient
  - Conjugate gradient for linear systems
- Second order methods
  - Newton's method
  - Quasi-Newton (BFGS, etc.)

3.4.2 A classification of optimization problems

- Convex problems
  - smooth unconstrained problems *(least-squares, logistic regression)*
  - smooth constrained *(non-negative least-squares)*
  - non-smooth *(SVM, Lasso)*
- Non-Convex problems
  - continuous problems *(neural networks, Nash equilibrium finding)*
  - discrete optimization *(bandits)*
- Other special cases
  - stochastic optimization *(signal processing, finite-sums)*
  - infinite dimension *(optimal control, calculus of variations)*

3.4.3 So in practice?

- There is an important work of *modeling*
- Rely on *optimization software* for medium-scale problems
- Use the *structure* of the problem

TUTORIALS

TUTORIAL 1 MATRIX ANALYSIS

1.1 DECOMPOSITIONS

Exercise 1.1 (Rank 1 matrices)**.**

a. Justify why a rank 1 matrix A can always be written $A = uv^\top$.

b. Express matrix $B = \begin{bmatrix} 1 & 2 & 5 \\ 2 & 4 & 10 \\ 0 & 0 & 0 \end{bmatrix}$ as the product of two vectors: $B = uv^\top$.

c. Compute eigenvalues and associated eigenvectors of matrix B.

d. Justify why B is rank 1.

Exercise 1.2 (Projection)**.** We call projection a real square matrix P such that $P = P^2$.

a. Show that if $\|P\| < 1$ for some operator norm, then $P = 0$.

b. Let P be a rank 1 matrix. Show that Px is the projection of x on the span of some vector. Justify why if P is symmetric, the projection is said orthogonal.

Exercise 1.3 (Singular Value Decomposition)**.** Let $A = \begin{bmatrix} 2 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}$

a. Compute the eigenvalues of A. Is the matrix invertible?

b. Let $B = AA^\top$. Check that B is symmetric. What can you say about its eigenvalues? Compute the eigenvalues of B.

c. Without further computation, give the eigenvalues of $C = A^\mathrm{T}A$.

d. Give conditions for C to be invertible. Notably, show that if A is full column rank, then C is also full column rank.

e. Show that C^{-1} is symmetric whenever it exists. What can you then say about its eigenvalues?

1.2 LINEAR SYSTEMS RESOLUTION WITH APPLICATIONS TO REGRESSION

In this example, that we will also study in the Labs, we use linear algebra to extract information from data; more precisely, we predict final notes of a group of student

from their profiles[18].

Profiles include features such as student grades, demographic, social and school related features and were collected by using school reports and questionnaires. We wish to predict the final grade by a *good* linear combination of the features.

Mathematically, from the *learning matrix* A of size $n \times d$, $n \geq d$, comprising of the features values of each training student in line, and the vector of the values of the target features b; we seek a *regression vector* that minimizes the squared error between Ax and b. This problem boils down to the following least square problem:

$$\min_x \|Ax - b\|_2^2. \tag{1.1}$$

Exercise 1.4. Using basic analysis[19], one can prove that any solution of Pb (1.1) verifies the following relation:

$$A^{\mathrm{T}}Ax = A^{\mathrm{T}}b.$$

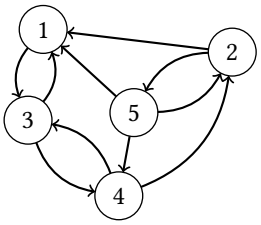
a. Assume that A has full rank, show that there is a unique solution to Pb (1.1).

b. Express this solution using the singular value decomposition of A.

In the Lab, we are going to learn a linear predictor using linear regression over a part of the data called the *learning set* and we will check our prediction by comparing the results for the rest of the data, the *testing set*.

1.3 PageRank

The problem of ranking webpages is of the utmost importance for search engines. To this end, a very popular approach is to represent webpages as a graph where the nodes are the pages themselves and the edges are the links between them (if page i contains a links pointing toward page j, there is a directed edge from node i to node j in the graph). Then, a page/node has a high score (it is ranked high) if there are many links pointing toward it, especially coming from highly ranked pages. This approach is at the core of the PageRank algorithm developed in 1996 by Larry Page and Sergey Brin, the founders of Google. This exercise illustrates the mathematics used in this process by working on a simple example.



$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Figure 1.1: a simple graph and its incidence matrix

More formally, consider the graph of Fig. 1.1. We could choose as a score the number of incoming links; node 1 is ranked first with 3, nodes 2, 3, 4 are second with 2, 5 is last with 1. The limits of this scoring is that 2, 3, 4 have the same score but are very different in nature as 3 is pointed by the most important page. To correct this

[18] The Student Performance data can be found at https://archive.ics.uci.edu/ml/datasets/Student+Performance; it include secondary education students of two Portuguese schools.

[19] this will be quickly covered in the optimization part.

phenomenon, the following scoring was introduced: the score x_i of page i is equal to the sum over the pages j pointing toward i of the scores (x_j) divided by their number of outgoing links n_j. Mathematically, the (implicit) score of page i is

$$x_i = \sum_{j \in \mathcal{P}_i} \frac{x_j}{n_j} \tag{1.2}$$

where $\mathcal{P}_i$ is the set of nodes pointing toward i (i itself is not in $\mathcal{P}_i$).

Exercise 1.5 (Some algebraic graph theory)**.** The graph of Fig. 1.1 can be represented by its incidence matrix A which verifies $A_{i,j} = 1$ if there is a link pointing towards i in page j, and $A_{i,j} = 0$ elsewhere.

a. Let $x \in \mathbb{R}^5$ be the vector of the pages scores. Write the score equation (1.2) as a linear equation $x = Rx$ where R is a matrix to define. Does a solution to this equation exist? Is it unique?

b. Show that matrix R is column stochastic, that is, its elements are non negative and its column sum is equal to one.

c. Deduce that 1 is an eigenvalue of R. (hint: one can show that it is an eigenvalue for R^{T}.)

d. Show that $\|R\| = 1$ for some matrix norm. Deduce that the sequence $(R^n)_n$ stays bounded and that the spectral radius of R is equal to 1.

e. Demonstrate that one can find an eigenvector for eigenvalue 1 with non-negative entries.
Hint: show that the sequence defined by $v_{k+1} = Av_k$ *and* $v_0 \geq 0$ *give non-negative vectors.*

The above questions have led you to prove parts of a fundamental theorem in matrix analysis called the *Perron-Frobenius* theorem.

Theorem 1.1 (Perron-Frobenius theorem)**.** *Let A be a non-negative $n \times n$ matrix. Then,*

(i) the spectral radius $\rho = \rho(A)$ is an eigenvalue;

(ii) there is a non-negative eigenvector $x \in \mathbb{R}^n_+$ such that $Ax = \rho x$;

(iii) if in addition A^k has all its entries (strictly) positive for some $k > 0$, then ρ is an eigenvalue of simple multiplicity and it is the only one of maximal modulus. Furthermore, there is a unique non-negative eigenvector x such that $Ax = \rho x$ and $\|x\|_1 = 1$. This vector is called the Perron vector.

The additional condition of (iii) is often called *primitivity*.

TUTORIAL 2 OPTIMIZATION

2.1 USING THE DEFINITIONS

Exercise 2.1 (Basic Differential calculus)**.** Use the composition lemma to compute the gradients of:

a. $f_1(x) = \|Ax - b\|_2^2$.

b. $f_2(x) = \|x\|_2$.

Exercise 2.2 (Fundamentals of convexity)**.** This exercise proves and illustrates some results seen in the course.

a. Let f and g be two convex functions. Show that $m(x) = \max(f(x), g(x))$ is convex.

b. Show that $f_1(x) = \max(x^2 - 1, 0)$ is convex.

c. Let f be a convex function and g be a convex, non-decreasing function. Show that $c(x) = g(f(x))$ is convex.

d. Show that $f_2(x) = \exp(x^2)$ is convex. What about $f_3(x) = \exp(-x^2)$

e. Justify why the 1-norm, the 2 norm, and the squared 2-norm are convex.

Exercise 2.3 (Strict and strong convexity)**.** A function $f : \mathbb{R}^n \to \mathbb{R}$ is said

- *strictly convex* if for any $x \neq y \in \mathbb{R}^n$ and any $\alpha \in]0, 1[$

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

- *strongly convex* if there exists $\beta > 0$ such that $f - \frac{\beta}{2}\|\cdot\|_2^2$ is convex.

a. For a strictly convex function f, show that the problem

$$\begin{cases} \min f(x) \\ x \in C \end{cases}$$

where C is a convex set admits at most one solution.

b. Show that a strongly convex function is also strictly convex.
Hint: use the identity $\|\alpha x + (1-\alpha)y\|^2 = \alpha\|x\|^2 + (1-\alpha)\|y\|^2 - \alpha(1-\alpha)\|x-y\|^2$.

c. Let f be a twice differentiable function. Show that f is strongly convex if and only if there exists $\beta > 0$ such that the eigenvalues of $\nabla^2 f(x)$ are larger than β for all x.

d. Discuss the strict and strong convexity of function f_1 and f_2 of Exercise 2.2.

Exercise 2.4 (Optimality conditions)**.** Let $f : \mathbb{R}^n \to \mathbb{R}$ be a twice differentiable function and $\bar{x} \in \mathbb{R}^n$. We suppose that f admits a local minimum at $\bar{x}$ that is $f(x) \geq f(\bar{x})$ for all x in a neighborhood[20] of $\bar{x}$.

[20] Formally, one would write $\forall x \in \mathbb{R}^n$ such that $\|x - \bar{x}\| \leq \varepsilon$ for $\varepsilon > 0$ and some norm $\|\cdot\|$.

a. For any direction $u \in \mathbb{R}^n$, we define the $\mathbb{R} \to \mathbb{R}$ function $q(t) = f(\bar{x} + tu)$. Compute $q'(t)$.

b. By using the first order Taylor expansion of q at 0, show that $\nabla f(\bar{x}) = 0$.

c. Compute $q''(t)$. By using the second order Taylor expansion of q at 0, show that $\nabla^2 f(\bar{x})$ is positive semi-definite.

2.2 Smoothness and Optimization

Exercise 2.5 (Descent lemma)**.** A function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be L-smooth if it is differentiable and its gradient ∇f is L-Lipchitz continuous, that is

$$\forall x, y \in \mathbb{R}^n, \quad \|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|.$$

The goal of the exercise is to prove that if $f : \mathbb{R}^n \to \mathbb{R}$ is L-smooth, then for all $x, y \in \mathbb{R}^n$,

$$f(x) \leq f(y) + (x - y)^{\mathrm{T}} \nabla f(y) + \frac{L}{2}\|x - y\|^2$$

a. Starting from fundamental theorem of calculus stating that for all $x, y \in \mathbb{R}^n$,

$$f(x) - f(y) = \int_0^1 (x - y)^{\mathrm{T}} \nabla f(y + t(x - y)) \mathrm{d}t$$

prove the descent lemma.

b. Give a function for which the inequality is tight and one for which it is not.

Exercise 2.6 (Smooth functions)**.** Consider the constant stepsize gradient algorithm $x_{k+1} = x_k - \gamma \nabla f(x_k)$ on an L-smooth function f with some minimizer (i.e. some $x^\star$ such that $f(x) \geq f(x^\star)$ for all x).

a. Use the *descent lemma* to prove convergence of the sequence $(f(x_k))$ when $\gamma \leq 2/L$.

b. Does the sequence (x_k) converge? To what?