Optimization for large scale machine learning
ENS, M2 Info

Mathurin Massias
mathurin.massias@gmail.com

Goals of the class

In this class we will study the resolution of optimization problems arising in Machine Learning. These problems are usually characterized by their large scale, which call for dedicated classes of algorithms. They also usually present some specific structure, that can be leveraged.

The main objectives are to:

1. develop a taxonomy of optimization problems and which algorithms can be used in which settings
2. understand convergence properties and functions properties that govern them
3. master theoretical tools and techniques to derive convergence proofs
4. get hands on practice to challenge theoretical results, in Python

I thank Cesare Molinari, Saverio Salzo and Silvia Villa for their graduate course on convex analysis.

General notational conventions

We almost always work in finite dimension and the optimization is performed over the set $\mathbb{R}^d$. In a Machine Learning context, the integer d will thus denote the dimension of the parameter space, while n will denote the number of samples (observed data points). We adopt the optimization convention, i.e. we write a linear system as $Ax = b$ (while statistics has $y = X\beta$, the inverse problems field writes $f = Au$, signal processing writes $y = \Phi x$...): the unknown is x; the iterates are x_k or x_t. As much as possible we write x^* for the minimizer of given problems, but due to the various meaning of star (Fenchel conjugation, adjoint), it happens that we also write $\hat{x}$ as in the statistical literature.

These conventions are meant to ease reading, but there can always be exceptions. It is anyways a good exercise to adapt to different notation.

1 Motivation: gradient descent on ordinary least squares

Most optimization problems studied in this class are of the following form:

$$\inf_{x\in\mathcal{C}} f(x) \ , \tag{1.1}$$

where $\mathcal{C}$ is a subset of $\mathbb{R}^d$ and f is a real-valued function on a subset of $\mathbb{R}^d$.

Formally, solving Problem (1) means to find the largest lower bound on all values $f(x)$ when x describes $\mathcal{C}$. This largest lower bound, call the *infimum*, may not exist, and if it exists it may not necessarily be attained, even for nicely-behaved f. The reader should mind that most of the time, we write $\min_{\mathcal{C}} f$ instead of $\inf_{\mathcal{C}} f$; this is an abuse of notation and does not mean that the infimum is attained (not even that it is finite).

In Machine Learning, rather that the smallest value taken by f, we are more interested in finding a minimizer, that is

$$x^* \in \operatorname*{argmin}_{x\in\mathcal{C}} f(x) \ , \tag{1.2}$$

meaning in finding $x^* \in \mathcal{C}$ such that for all $x \in \mathcal{C}$, $f(x^*) \leq f(x)$. In Machine Learning, x^* usually represent the parameters of a model, that once computed are used to achieve some statistical task.

1.1 Why does statistical learning need optimization?

Let $n \in \mathbb{N}$. Suppose that we have collected a dataset $(a_i, b_i)_{i\in[n]}$ of pairs observations-targets, and we assume that they are linearly related, i.e. there exists $x \in \mathbb{R}^d$ such that $b_i \approx a_i^\top x$.

This is very vague. To make it more precise and rigorous, it is extremely frequent to model the dataset as n independent, identically distributed realizations of two random variables A and B linked by

$$B \sim \mathcal{N}(A^\top x_{\text{true}}, \sigma^2) \ , \tag{1.3}$$

meaning that for all $i \in [n]$, $b_i = x_{\text{true}}^\top a_i + \varepsilon_i$, with $(\varepsilon_i)_{i\in[n]}$ n realizations of a centered Guassian variable of variance σ^2.

Given the collected observations, how can we infer x_{true}? From a statistical point of view, an answer is to maximize the conditional likelihood. Under model (1.3), the conditional likelihood is:

$$\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\|a_i^\top x - b_i\|^2}{2\sigma^2}\right) \ . \tag{1.4}$$

Minimization of the negative log likelihood in x gives

$$x^* \in \operatorname*{argmin}_{x\in\mathbb{R}^d} \sum_{1}^{n} (b_i - a_i^\top x)^2 = \operatorname{argmin} \frac{1}{2}\|Ax - b\|^2 \ , \tag{1.5}$$

where the **design matrix** $A \in \mathbb{R}^{n\times d}$ has i-th row a_i, and the **target vector** $b \in \mathbb{R}^n$ has i-th entry b_i. Problem (5) is called Ordinary Least Squares (OLS). Solving it gives an estimator for x_{true} and allows, for example, predicting a value b_{n+1} if one observes a new entry point a_{n+1}. This is also an illustration that the value of the infimum is not always very interesting (if $d \geq n$ and A has rank n it is 0), contrary to the minimizer.

How to compute the solution of Problem (5)? This is a classical problem in linear algebra and there exist direct ways to do it (Exercise 1.6). But in other settings there may not exist a closed-form solution: it is perfectly possible to postulate a different generative model than (1.3), or simply not impose a generative model, and find x by minimizing another cost/loss function. This is the setting of empirical risk minimization (ERM): one finds the parameters of a model by minimizing a cost function, often –but not always– arising from likelihood maximization.

Example 1.1 (ℓ_1 and Huber regression)**.** *Ordinary least squares are strongly influenced by outliers: the squared loss gives a lot of importance to large differences. In the objective function (5) a mismatch of 2 between the true observation b_i and the model fit $a_i^\top x$ costs $2^2/2 = 2$, but a mismatch of 10 costs $10^2/2 = 50$. A more* robust *solution, that is less dominated by large mismatches (or "outliers"), can be found by minimizing the sum of absolute errors:*

$$\operatorname*{argmin}_{x\in\mathbb{R}^d} \sum_{i=1}^{n} |a_i^\top x - b_i| = \operatorname*{argmin}_{x\in\mathbb{R}^d} \|Ax - b\|_1 \ . \tag{1.6}$$

Problem (6) is unfortunately more complicated to solve, as we shall see. It can be reformulated as a Linear Program, but the cost to solve it scales very badly with d. A hybrid of Problems (5) and (6) involves the Huber loss (Figure 1):

$$\operatorname*{argmin}_{x\in\mathbb{R}^d} \sum_{1}^{n} \mathrm{huber}_\rho(a_i^\top x - b_i) \ , \tag{1.7}$$

with the Huber loss defined as:

$$\mathrm{huber}_\rho(x) = \begin{cases} \frac{x^2}{2\rho} + \frac{\rho}{2} \ , & \text{if } |x| \le \rho \ , \\ |x| \ , & \text{otherwise.} \end{cases} \tag{1.8}$$

The Huber loss is less sensitive to outliers than the squared ℓ_2 loss, and more optimization friendly than the ℓ_1 loss, but there is no free lunch: it relies on a parameter $\rho \in \mathbb{R}_+$ that needs to be tuned, and there is no obvious and cheap way to chose the best ρ.

For both problems, there is no closed-form solution and we need optimization techniques to approximate the minimizer.

Finally, instead of a linear model, one may model the dependency between observation and target by a function belonging to some parametric class $\{f_x : \mathbb{R}^d \mapsto \mathbb{R} : x \in \mathbb{R}^p\}$, described by p parameters stored in x:

$$b_i \approx f_x(a_i) \ , \tag{1.9}$$

and, depending on what is meant by $\approx$, minimize a loss $L : \mathbb{R}\times\mathbb{R} \to \mathbb{R}$, yielding the following Empirical Risk Minimization (ERM) problem:

$$x^* \in \operatorname*{argmin}_{x\in\mathbb{R}^p} \sum_{i=1}^{n} L(f_x(a_i), b_i) \ . \tag{1.10}$$

As we have seen, the typical example is $f_x = x^\top \cdot$; but this framework also fits the Deep Learning framework (with f_x a composition of matrix vector multiplications followed by pointwise application of a non-linear function.).

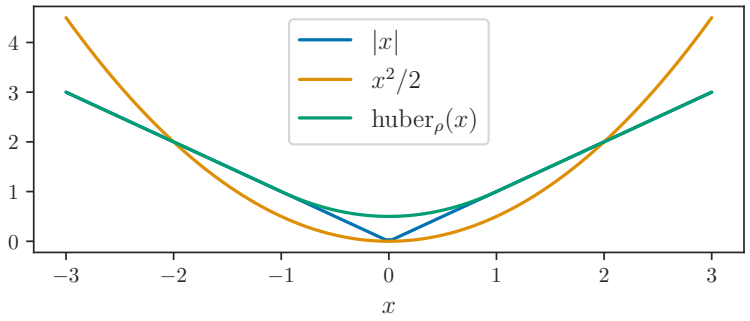

Figure 1: ℓ_1, squared ℓ_2 and Huber ($\rho = 1$) losses

1.2 Solving least squares with gradient descent

Problem (5) can be solved in closed-form with dedicated algorithms such as Cholesky, LU or QR decompositions (see Exercise 1.6); but ℓ_1, Huber and generic ERM models cannot. We need general purpose algorithms to solve classes of similar problems.

One of the arguably simplest minimization algorithm is gradient descent (GD). If it exists, the gradient of a function points in the direction of maximal increase, since the directional derivative $f'(x; v) = \lim_{\epsilon \to 0} \frac{f(x+\epsilon v) - f(x)}{\epsilon}$ is given by

$$f'(x; v) = \langle \nabla f(x), v \rangle \ . \tag{1.11}$$

To decrease the value of f as fast as possible, it makes sense to move in the opposite direction of the gradient. Taking a stepsize $\eta > 0$, this gives the gradient descent iterations:

$$\begin{aligned} &x_0 \in \mathbb{R}^d \ , \\ &x_{t+1} = x_t - \eta \nabla f(x_t) \ . \end{aligned} \tag{1.12}$$

Proposition 1.2 (Convergence rate of GD on OLS)**.** *Consider the OLS problem, and assume that there exist strictly positive scalars μ and L such that $\mu \operatorname{Id} \preceq A^\top A \preceq L \operatorname{Id}$. Let $\kappa = L/\mu$ denote the* condition number*. Then there exists a unique minimizer of least squares, x^*, and the iterates of gradient descent on ordinary least squares with stepsize $\eta = 1/L$ satisfy:*

$$\|x_t - x^*\| \leq \exp(-t/\kappa) \|x_0 - x^*\| \ . \tag{1.13}$$

Proof. The gradient of $x \mapsto \frac{1}{2}\|Ax - b\|^2$ is $A^\top(Ax - b)$, hence gradient descent on ordinary least squares read

$$x_{t+1} = x_t - \eta A^\top (A x_t - b) \ . \tag{1.14}$$

There exists a unique minimizer x^*, and it satisfies $A^\top A x^* = A^\top b$ (Exercises 1.5 and 1.6), so:

$$x_{t+1} - x^* = (\operatorname{Id} - \eta A^\top A)(x_t - x^*) \ . \tag{1.15}$$

Observe that

$$(1 - \eta L) \operatorname{Id} \preccurlyeq \operatorname{Id} - \eta A^\top A \preccurlyeq (1 - \eta \mu) \operatorname{Id} \ , \tag{1.16}$$

hence

$$\| \operatorname{Id} - \eta A^\top A \|_2 \leq q(\eta) \triangleq \max(|1 - \eta L|, |1 - \eta \mu|) \ . \tag{1.17}$$

Setting $\eta = 1/L$ yields $q(\eta) = 1 - \mu/L$, and using $(1-u)^t \leq \exp(-tu)$ concludes. □

Remark 1.3. *The convergence rate (1.13), is a very good rate in the sense that it decays fast towards 0. It is called a* linear *rate*[1]*. Suppose that we want to get ε-close to x^*, then we need at most $\kappa \log(\|x_0 - x^*\|/\varepsilon)$ iterations of GD, which grows quite slowly as ε goes to 0.*

We can also obtain a rate in objective value:

$$f(x_t) - f(x^*) = \langle \nabla f(x^*), x_t - x^* \rangle + \frac{1}{2}(x_t - x^*) A^\top A (x_t - x^*) \tag{1.18}$$

$$= \frac{1}{2}(x_t - x^*) A^\top A (x_t - x^*) \tag{1.19}$$

$$\leq \frac{L}{2} \|x_t - x^*\|^2 \ , \tag{1.20}$$

but as we've seen, such rates are usually less desirable than rates on x_t.

Remark 1.4 (Other choices of η.)**.** *We have actually proven the stronger result:*

$$\|x_t - x^*\| \leq q(\eta)^t \|x_0 - x^*\| \ . \tag{1.21}$$

This shows that any choice of η in $]0, 2/L[$ leads to convergence, because it results in $q(\eta) < 1$. We can tune η to minimize $q(\eta)$: doing so yields $\eta = \frac{2}{L+\mu}$ and $q(\eta) = \frac{L-\mu}{L+\mu} = \frac{\kappa - 1}{\kappa + 1}$.

Questions: Does there always exist an L as in the assumptions of Proposition 1.2? And a μ? To what do they correspond geometrically?

1.3 Numerical puzzles

On Figure 2 is the numerically observed convergence rate of GD on one instance of OLS. The matrix A is 500 by 500 with i.i.d. normal entries so it is full rank, $\mu > 0$ and Proposition 1.2 applies. Yet the rate we see is clearly not linear. Why?

On Figure 3, A is still random but its shape is 200×300, so $A^\top A$ is not full rank, $\mu = 0$. Yet we see a clear linear convergence rate (numerical floating point errors kick in at iteration 1000). Why?

[1] this is a terrible name: it is linear in log scale

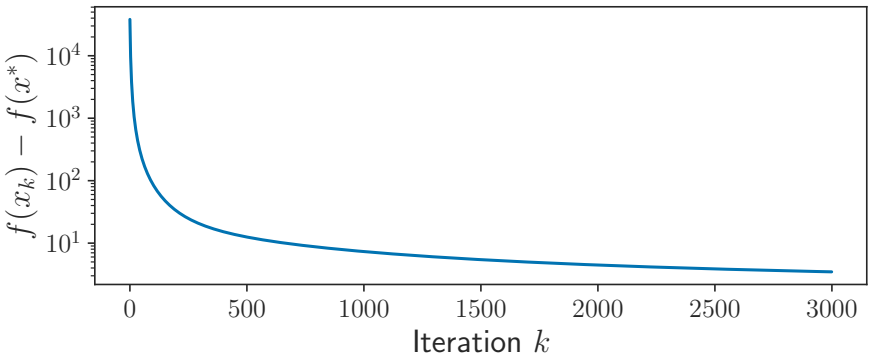

Figure 2: Objective convergence rate of GD on OLS on a random i.i.d. $A \in \mathbb{R}^{500\times500}$.

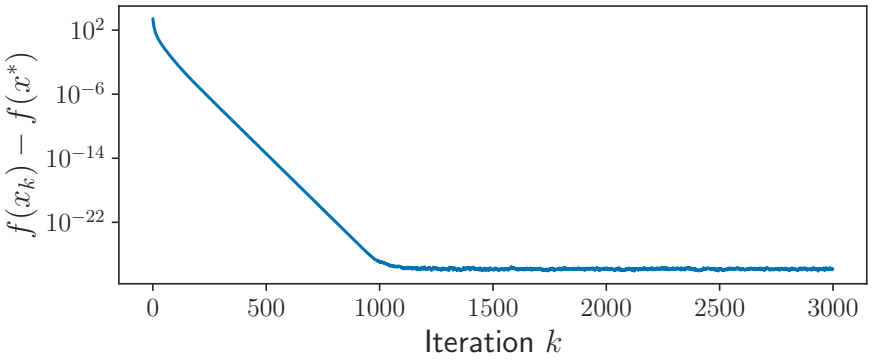

Figure 3: Objective convergence rate of GD on OLS on a random i.i.d. $A \in \mathbb{R}^{200\times300}$.

1.4 Exercises

Exercise 1.5. ☕ *Let $A \in \mathbb{R}^{n\times d}$. Show that* $\operatorname{Ker} A = \operatorname{Ker} A^*A$.
Show that for any $b \in \mathbb{R}^n$ there exist a solution to $A^*Ax = A^*b$.
☕☕ *Show that there does not always exist a solution to $A^*Ax = A^*b$ in the infinite dimensional case (when A is a bounded linear operator between infinite dimensional Hilbert spaces.)*

Exercise 1.6. ☕ *Let $A \in \mathbb{R}^{n\times d}$, $b \in \mathbb{R}^n$. Show that solving Ordinary Least Squares:*

$$\min \frac{1}{2}\|Ax - b\|^2 \ , \tag{1.22}$$

amounts to solving $A^*Ax = A^*b$ (aka the normal equations*).*
Show that the set of solutions is:

$$A^\dagger b + \operatorname{Ker} A \ .$$

Exercise 1.7. ☕ *When is $x \mapsto \|Ax - b\|^2$ strictly convex? Strongly convex?*

Exercise 1.8 (Least squares with intercept)**.** ☕☕ *An* intercept x_0 *is a constant scalar term in the linear prediction function, that becomes $a \mapsto a^\top x + x_0$. Fitting an intercept can be done by adding a column of 1s to A. Alternatively, show that the solution of least squares with intercept,*

$$(\hat{x}, \hat{x}_0) \in \underset{x\in\mathbb{R}^d, x_0\in\mathbb{R}}{\operatorname{argmin}} \frac{1}{2}\|Ax - b - x_0\mathbf{1}\|^2 \tag{1.23}$$

is given by:

$$\hat{x} = \hat{x}_c \ , \tag{1.24}$$

$$\hat{x}_0 = \frac{1}{n}\sum_1^n (a_i^\top \hat{x} - b_i) \ , \tag{1.25}$$

where $\hat{x}_c$ is the solution of least squares without intercept on centered data A_c and b_c (versions of A and b where the rowwise mean has been subtracted).

Exercise 1.9 (Gradient descent on isotropic parabola)**.** ☕ *Let $A \in \mathbb{R}^{n\times d}$ be such that the condition number*[2] *of $A^\top A$ is equal to 1. Show that gradient descent with stepsize $1/L$ converges in a single iteration for the problem $\min \frac{1}{2}\|Ax - b\|^2$.*

[2] i.e. the ratio between the largest and the smallest eigenvalues of $A^\top A$.

2 Reminder on convex analysis

Section 1 has shown us basic notions in optimization. We have studied gradient descent, an iterative algorithm producing a sequence of iterates x_k that aim at solving an optimization problem. For the Ordinary Least Squares objective function, we have proved two types of convergence results[3]:

- in iterates: $\|x_k - x^*\| \to 0$
- in function values: $f(x_k) - \inf f \to 0$.

Both result were *non-asymptotic*: we had information about the speed at which convergence happened.

In the sequel we want to minimize functions beyond least squares. For starters, we'll work with convex functions, because they're roughly the only ones we can hope to minimize globally.

2.1 Convexity and minimizers

Definition 2.1 (Global and local minimizers)**.** *Let $f : \mathbb{R}^d \to \mathbb{R}$. A global minimizer of f is a point x^* such that $f(x^*) \leq f(x)$ for all $x \in \mathbb{R}^d$. A local minimizer of f is a point x^* such that there exists a neighborhood V of x^* such that $f(x^*) \leq f(x)$ for all $x \in V$.*

Definition 2.2 (Convexity)**.** *A function $f : \mathbb{R}^d \to \mathbb{R}$ is convex if and only if it lies below its cords:*

$$\forall x, y \in \mathbb{R}^d, \forall \lambda \in [0,1], \quad f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y) \ . \tag{2.1}$$

It is strictly convex if the inequality (2.1) holds strictly (for $\lambda \in]0,1[$).

Definition 2.3 (Strong convexity)**.** *Let $\mu > 0$. A function $f : \mathbb{R}^d \to \mathbb{R}$ is μ-strongly convex if for all $x, y \in \mathbb{R}^d$,*

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y) - \frac{\mu}{2}\lambda(1-\lambda)\|x-y\|^2 \ . \tag{2.2}$$

Note: this is the only true definition of strong convexity. Other definitions are rather characterizations in special cases. In particular, many other definitions assume that the norm is the Euclidean one, a requirement that this definition does not have. This plays a role in Section 18.

When $\|\cdot\|$ is the Euclidean norm, then f is μ-strongly convex if and only if $f - \frac{\mu}{2}\|\cdot\|^2$ is convex (Exercise 2.16). In this sense, a strongly convex function it so convex that when you subtract a parabola, it remains convex.

Convex functions are amenable to optimization because of the nice "local to global" properties they enjoy.

Proposition 2.4 (Local minimizers are global minimizers)**.** *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex function. Then all local minimizers of f are also global.*

[3] Does one imply the other? Under which condition?

Proposition 2.5 (Above its tangents)**.** *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex differentiable function. Then*

$$\forall x, y \in \mathbb{R}^d, \quad f(x) - f(y) \leq \langle \nabla f(x), x - y \rangle \ .$$

Written as $f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle$, it tells that f lies above all of its tangents. From local information $\nabla f(x)$, we get an information about the whole behavior of f.

Convex differentiable functions are nice because it is easy to characterize their minimizers.

Proposition 2.6 (Optimality condition for convex differentiable functions)**.** *Let f be a convex differentiable function. Then*

$$x^* \in \operatorname*{argmin}_{x \in \mathbb{R}^d} f(x) \Leftrightarrow \nabla f(x^*) = 0 \ . \tag{2.3}$$

Equipped with this theoretical framework, we can move to our first convergence proofs on generic functions.

2.2 Exercises

2.2.1 Convexity

Exercise 2.7. ☕ *Show that local minimizers of convex functions are global minimizers.*

Exercise 2.8 (Pointwise supremum preserves convexity)**.** ☕ *Let $(f_i)_I$ be a family of convex functions (not necessarily countable). Show that $x \mapsto \sup_{i \in I} f_i(x)$ is convex.*

Exercise 2.9 (Precomposition by linear operator preserves convexity)**.** ☕ *Let $f : \mathcal{Y} \to \mathbb{R}$ be a convex function and $A : \mathcal{X} \to \mathcal{Y}$ a linear operator. Show that $f(A\cdot)$ is convex (on $\mathcal{X}$).*

Exercise 2.10 (Misconceptions on existence of minimizers)**.** ☕ *Provide an example of convex function which does not admit a minimizer.*
What if the function is continuous and lower bounded?

Exercise 2.11. ☕ *Show that a strictly convex function has at most one minimizer.*

Exercise 2.12 (Jensen's inequality)**.** ☕ *Let $f : \mathbb{R}^d \to \mathbb{R}$ be convex. Let $n \in \mathbb{N}$, $x_1, \ldots, x_n \in \mathbb{R}^d$, and let $\lambda_1, \ldots, \lambda_n$ be positive scalars summing to 1.*
Show that $f(\sum_{i=1}^n \lambda_i x_i) \leq \sum_{i=1}^n \lambda_i f(x_i)$.

Exercise 2.13. ☕ *Show that the sublevel sets of a convex function are convex. Find a function with convex sublevel sets which is not convex.*

Exercise 2.14 (First order characterization of convex functions)**.** ☕☕ *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a differentiable function. Show that the following are equivalent:*

1. *f is convex*
2. *f lies above its tangents: $\forall x, y \in \mathbb{R}^d, f(x) \geq f(y) + \langle \nabla f(y), x - y \rangle$*
3. *∇f is monotone: $\forall x, y \in \mathbb{R}^d, \langle \nabla f(x) - \nabla f(y), x - y \rangle \geq 0$*

Exercise 2.15 (Continuity)**.** ☕☕☕ *Show that a convex function is locally Lipschitz (hence continuous) on the interior of its domain.*

Exercise 2.16 (Characterization of strongly convex functions in the Euclidean case)**.** ☕ *Show that f is μ-strongly convex with respect to the Euclidean norm if and only if $f - \frac{\mu}{2}\|\cdot\|^2$ is convex.*

Exercise 2.17. ☕ *Show that a strongly convex function admits exactly one minimizer.*

2.2.2 Gradient

Exercise 2.18. ☕ *Provide an example of setting where the gradient is not equal to the vector of partial derivatives.*

Exercise 2.19. ☕ *Show that the gradient of a function is orthogonal to the level lines of that function.*

Exercise 2.20. ☕ *Compute the gradients and Hessians of $x \mapsto \|x\|^2$, $x \mapsto \|x\|$, $x \mapsto a^\top x$. Is it true that the gradient of $x \mapsto \frac{1}{2}x^\top Ax$ is equal to Ax?*

Exercise 2.21. ☕ *Let $A \in \mathbb{R}^{n\times d}$, $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^d \to \mathbb{R}$ defined as $g(x) = f(Ax)$ for all $x \in \mathbb{R}^d$. Show that*

$$\nabla g(x) = A^* \nabla f(Ax) \ ,$$
$$\nabla^2 g(x) = A^* \nabla^2 f(Ax) A \ .$$

Exercise 2.22. ☕☕ *Compute the gradient of the logdet function, $M \mapsto \log\det(M)$.*

3 Gradient descent on convex functions

In Section 1 we proved that for gradient descent on ordinary least squares we can get very fast (so-called "linear") convergence rates in terms of function values and iterates. Two numbers governed this convergence: global upper and lower bounds L and μ on the Hessian.

Our proof was very ad hoc, relying on the explicit expression of minimizer, the gradient and the Hessian. The questions we will try to answer in the sequel are:

- can we generalize the results of Proposition 1.2 to other objective functions?
- under which conditions?
- what are the objective's property that influence the convergence rate we obtain?

The first and weakest result we'll prove in this class concerns convex Lipschitz functions.

Proposition 3.1 (Gradient descent on Lipschitz convex functions)**.** *Let f be a convex, L-Lipschitz differentiable function admitting at least one minimizer x^*. For $t \in \mathbb{N}$, the iterates of gradient descent $x_{k+1} = x_k - \eta \nabla f(x_k)$ satisfy*

$$\min_{1 \leq k \leq t} f(x_k) - f(x^*) \leq \frac{L\|x_0 - x^*\|}{\sqrt{t}} \quad , \tag{3.1}$$

and

$$f(\tfrac{1}{t}\sum_{k=1}^{t} x_k) - f(x^*) \leq \frac{L\|x_0 - x^*\|}{\sqrt{t}} \quad , \tag{3.2}$$

if the stepsize η is taken as $\eta = \frac{\|x_0 - x^*\|}{L\sqrt{t}}$.

Let us make a few observations before the proof:

- This algorithm is a bit weird: the total number of iterations t must be known in advance to select the stepsize (in Exercise 3.16 we get rid of this up to a slight worsening in the rate, using a decaying stepsize $\eta_k \propto 1/\sqrt{k}$).
- The objective values are not necessarily decreasing, it is not a *descent* algorithm.
- The more iterations we do, the smaller we need to take the stepsize.
- The stepsize depends on the unknown quantity $\|x_0 - x^*\|$. We can get rid of this dependency by bounding $\|x_0 - x^*\|$ with e.g. the diameter of the domain.
- The kind of rate is not as strong as in the OLS case: we know nothing about the last iterate x_t and only have results on the ergodic (averaged) iterates or on the best iterate.

Proof. Let x^* be a minimizer of f. For $k \in \mathbb{N}$, using convexity of f, definition of x_{k+1} and the parallelogram identity,

$$\begin{aligned} f(x_k) - f(x^*) &\leq \langle \nabla f(x_k), x_k - x^* \rangle & (3.3)\\ &\leq \frac{1}{\eta}\langle x_k - x_{k+1}, x_k - x^* \rangle & (3.4)\\ &\leq \frac{1}{2\eta}\left(\|x_k - x_{k+1}\|^2 + \|x_k - x^*\|^2 - \|x_{k+1} - x^*\|^2\right) \quad . & (3.5) \end{aligned}$$

Summing, the telescopic series cancels out, and since f is L-Lipschitz we can bound $\|x_{k+1} - x_k\| = \eta\|\nabla f(x_k)\|$ by ηL, so:

$$\frac{1}{t}\sum_{k=1}^{t} f(x_k) - f(x^*) \leq \frac{1}{2t\eta}(t\eta^2 L^2 + \|x_0 - x^*\|^2 - \|x_{t+1} - x^*\|^2) \tag{3.6}$$

$$\leq \frac{\eta L^2}{2} + \frac{\|x_0 - x^*\|^2}{2t\eta} \tag{3.7}$$

Minimizing the RHS in η gives $\eta = \frac{R}{L\sqrt{t}}$ and the upper bound has value $\frac{RL}{\sqrt{t}}$. Jensen's inequality concludes. □

The $1/\sqrt{k}$ rate is quite poor: to approach the optimal value within ε, one needs $\mathcal{O}(1/\varepsilon^2)$ iterations. With more assumptions on f, it can be improved to $\mathcal{O}(1/t)$, as we shall see in Proposition 3.5.

Definition 3.2 (L-smoothness)**.** *A differentiable function f is L-smooth if its gradient is L-Lipschitz: for all $x, y \in \operatorname{dom} f$,*

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\| \ . \tag{3.8}$$

A widely used property of L-smooth functions is that they satisfy the so-called Descent lemma.

Lemma 3.3 (Descent lemma)**.** *Let f be a L-smooth function. Then for all $x, y \in \operatorname{dom} f$,*

$$f(x) \leq f(y) + \langle \nabla f(y), x - y\rangle + \frac{L}{2}\|x - y\|^2 \ . \tag{3.9}$$

Think of this inequality for a y fixed, and involving two functions of x, f and $\phi_y = f(y) + \langle \nabla f(y), \cdot - y\rangle + \frac{L}{2}\|\cdot - y\|^2$. The function ϕ_y is a convex, isotropic parabola. Equation (3.9) says that ϕ_y globally upper bounds f; in addition, the two functions are equal and tangent at y.

Proof. Notice that:

$$f(x) - f(y) = \int_0^1 \frac{d}{dt} f(y + t(x - y))dt = \int_0^1 \langle \nabla f(y + t(x - y)), x - y\rangle dt \ . \tag{3.10}$$

Subtract $\langle \nabla f(y), x - y\rangle$, use Cauchy-Schwarz and the hypothesis on f, conclude. □

Lemma 3.4 (Cocoercivity of the gradient)**.** *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a L-smooth function.*

$$\forall x, y \in \mathbb{R}^d, \quad \langle \nabla f(x) - \nabla f(y), x - y\rangle \geq \frac{1}{L}\|\nabla f(x) - \nabla f(y)\| \ . \tag{3.11}$$

Proposition 3.5 (Convergence rate of gradient descent on L-smooth functions)**.** *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex L-smooth function with non empty set of minimizers. Then the iterates of gradient descent with stepsize $1/L$ converge at the rate:*

$$f(x_k) - f(x^*) \leq \frac{2L\|x_0 - x^*\|}{k} \ . \tag{3.12}$$

Proof. Let x^* be a minimizer of f. First, we show that the distance of x_k to x^* decreases:

$$\|x_{k+1} - x^*\|^2 = \|x_k - x^*\|^2 + 2\langle x_k - x^*, x_{k+1} - x_k\rangle + \|x_{k+1} - x_k\|^2 \tag{3.13}$$

$$= \|x_k - x^*\|^2 - \frac{2}{L}\langle \nabla f(x_k), x_k - x^*\rangle + \frac{1}{L^2}\|\nabla f(x_k)\|^2 \ . \tag{3.14}$$

Lemma 3.4, combined with $\nabla f(x^*) = 0$, yields:

$$-\frac{2}{L}\langle \nabla f(x_k), x_k - x^*\rangle + \frac{1}{L^2}\|\nabla f(x_k)\|^2 \leq -\frac{1}{L^2}\|\nabla f(x_k)\|^2 \leq 0 \ , \tag{3.15}$$

which concludes: $\|x_{k+1} - x^*\| \leq \|x_k - x^*\|$.

We then use convexity of f:

$$f(x_k) - f(x^*) \leq \langle \nabla f(x_k), x_k - x^*\rangle \leq \|\nabla f(x_k)\|\|x_k - x^*\| \leq \|\nabla f(x_k)\|\|x_0 - x^*\| \ . \tag{3.16}$$

Finally the descent Lemma (Lemma 3.3) quantifies the decrease in objective at each iteration:

$$f(x_{k+1}) - f(x_k) \leq -\frac{1}{2L}\|\nabla f(x_k)\|^2 \ . \tag{3.17}$$

Introducing $\delta_k = f(x_k) - f(x^*)$, we have:

$$\delta_{k+1} - \delta_k \leq -\frac{1}{2L}\|\nabla f(x_k)\|^2 \tag{3.18}$$

$$\leq -\frac{1}{2L}\frac{\delta_k^2}{\|x_0 - x^*\|^2} \ . \tag{3.19}$$

The above manipulations are quite standard in optimization, but the following trick is a bit surprising the first time you see it: dividing by $\delta_k\delta_{k+1}$,

$$\frac{1}{\delta_k} - \frac{1}{\delta_{k+1}} \leq -\frac{1}{2L\|x_0 - x^*\|^2}\frac{\delta_k}{\delta_{k+1}} \tag{3.20}$$

$$\leq -\frac{1}{2L\|x_0 - x^*\|^2} \ , \tag{3.21}$$

since (δ_k) decreases (by (3.18)). Summing and telescoping concludes. □

If the function is even more regular, the rate of gradient descent can further be improved.

Proposition 3.6. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be L-smooth and μ-strongly convex. Then it admits a unique minimizer x^* (Exercise 2.17) and the iterates of gradient descent with stepsize $1/L$ converge linearly:*

$$f(x_k) - f(x^*) \leq \left(1 - \frac{\mu}{L}\right)^k (f(x_0) - f(x^*)) \ . \tag{3.22}$$

Proof. Let x^* be the minimizer of f. Being μ-strongly convex, f satisfies the Polyak-Łojasiewicz inequality (Exercise 3.11):

$$\forall x \in \mathbb{R}^d, f(x) - f(x^*) \leq \frac{1}{2\mu}\|\nabla f(x)\|^2 \ . \tag{3.23}$$

Combined with the descent lemma,

$$f(x_{k+1}) - f(x_k) \leq -\frac{1}{2L}\|\nabla f(x_k)\|^2 \tag{3.24}$$

$$f(x_{k+1}) - f(x^*) + f(x^*) - f(x_k) \leq -\frac{\mu}{L}(f(x_k) - f(x^*)) \ . \tag{3.25}$$

hence $f(x_{k+1}) - f(x^*) \leq (1 - \frac{\mu}{L})(f(x_k) - f(x^*))$. □

3.1 Exercises

3.1.1 Convexity inequalities

Exercise 3.7. ☕☕ *Let f be a convex and differentiable function. Let $L > 0$. Show that the following properties are equivalent:*

1. $\forall x, y,\ \|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$
2. $\forall x, y,\ f(x) \leq f(y) + \langle \nabla f(y), x - y\rangle + \frac{L}{2}\|x - y\|^2$
3. $\forall x, y,\ \frac{1}{L}\|\nabla f(x) - \nabla f(y)\|^2 \leq \langle x - y, \nabla f(x) - \nabla f(y)\rangle$

Exercise 3.8. ☕ *Let f be a twice differentiable L-smooth function. Show that for all $x \in \mathbb{R}^d$, $\nabla^2 f(x) \preccurlyeq L\,\mathrm{Id}$.*

Exercise 3.9. ☕ *Let f be a differentiable function. Show that the following properties are equivalent:*

1. *f is μ-strongly convex*
2. $\forall x, y,\ f(x) \geq f(y) + \langle \nabla f(y), x - y\rangle + \frac{\mu}{2}\|x - y\|^2$
3. $\forall x, y,\ \mu\|x - y\|^2 \leq \langle x - y, \nabla f(x) - \nabla f(y)\rangle$

Exercise 3.10. ☕ *Let f be a twice differentiable μ-strongly convex function. Show that for all $x \in \mathbb{R}^d$, $\mu\,\mathrm{Id} \preccurlyeq \nabla^2 f(x)$.*

Exercise 3.11 (Polyak-Łojasiewicz inequality)**.** ☕☕ *Let f be a μ-strongly-convex and differentiable function. Let $x^* = \operatorname{argmin} f(x)$. Show that f satisfies the Polyak-Łojasiewicz inequality:*

$$\mu(f(x) - f(x^*)) \leq \frac{1}{2}\|\nabla f(x)\|^2 \ .$$

Provide an example of function which is not strongly convex, but satisfies the inequality.

Exercise 3.12. ☕☕ *Let f be a L-smooth μ-strongly convex function. Show that for any x, y,*

$$\frac{\mu L}{\mu + L}\|x - y\|^2 + \frac{1}{L + \mu}\|\nabla f(x) - \nabla f(y)\|^2 \leq \langle x - y, \nabla f(x) - \nabla f(y)\rangle \ .$$

Exercise 3.13 (3 point descent lemma)**.** ☕ *Let f be convex and L-smooth. Show that for any triplet (x, y, z),*

$$f(x) \leq f(y) + \langle \nabla f(z), x - y\rangle + \frac{L}{2}\|x - z\| \ .$$

3.1.2 Gradient descent

Exercise 3.14 (**??** in discrete time)**.** ☕☕ *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a convex, twice differentiable L-smooth function.*
Show that the iterates of gradient descent on f with step size $0 < \alpha < 2/L$ have decreasing gradient norm.
Can you show it if f is not twice differentiable?
Is it still true when f is not convex?

Exercise 3.15. ☕ *Provide a finite dimensional example of convex L-smooth function f such that gradient descent with stepsize $< 2/L$ diverges.*

Exercise 3.16 (Gradient descent on Lipschitz function without knowing the horizon)**.** ☕☕ *For gradient descent on a Lipschitz differentiable objective, the classical proof assumes that the total number of iterations t is known in advance, to set the fixed stepsize $\eta \propto 1/\sqrt{t}$. Show that using a decreasing stepsize $\eta_k \propto 1/\sqrt{k}$ leads to a rate of order $\log k/\sqrt{k}$ on the ergodic or best iterate.*

4 Non smooth convex optimization

For a differentiable objective function $f : \mathbb{R}^d \to \mathbb{R}$, we have seen:

1. convexity: f is above its tangents
2. Lipschitzness
3. L-smoothness, implying the descent lemma (Lemma 3.3), meaning that at any point $y \in \mathbb{R}^d$ there exists an isotropic parabola of curvature L that globally **upper** bounds f and is tangent to f at y.
4. strong convexity: at any point $y \in \mathbb{R}^d$ there exists an isotropic parabola of curvature μ that globally **lower** bounds f and is tangent to f at y.

Depending on the properties satisfied by f, we have:

- 1 gives the necessary and sufficient condition for x to be a minimizer: $\nabla f(x^*) = 0$.
- $1 + 2$ gives a convergence rate of $1/\sqrt{k}$ under stepsize depending on the total number of iterations. The rate is in objective, for the best iterate or the averaged iterates.
- $1 + 3$ improves it to $1/k$ with stepsize $1/L$ not depending on the number of iterations. The rate is in objective, on the last iterate.
- $3 + 4$ sandwiches the functions between two isotropic parabolas of curvature μ and L and gives the excellent linear rate. The rate is on the distance from last iterate to the unique minimizer.

Now we want to consider non-differentiable functions f. Why do we need to handle these?

4.1 Constrained optimization

Optimizers sometimes wants the minimizer of their cost functions to belong to some set $\mathcal{C} \subset \mathbb{R}^d$.

For example, suppose that the variable x corresponds to an image, stored as a 2D array of pixels – an element of $\mathbb{R}^{d \times d}$. Then the values of the pixels are not just any real numbers: they should be in $[0, 1]$ if they correspond to gray levels for example.

If the optimization variable $x \in \mathbb{R}^d$ represents proportions (say, of various types of cells in a sane/cancerous tissue), it should have non-negative entries that sum to 1: $x \in \Delta_d$, where Δ_d is the d-dimensional *simplex*

$$\Delta_d = \{x \in \mathbb{R}^d : \forall i \in [d],\, x_i \geq 0, \sum_{i=1}^{d} x_i = 1\} \ . \tag{4.1}$$

Finally, suppose the practitioner is looking for the approximate solution to a linear system (using least squares), but wants a solution x^* that uses as few variables as possible. There are many ways to do this, but one popular (because convex) way to do so is to constrain the ℓ_1 norm of x:

$$\min_{x \in \mathbb{R}^d} \frac{1}{2}\|Ax - b\|^2 \quad \text{subject to} \quad \|x\|_1 \leq \tau \ .$$

All these examples lead us to consider constrained problems:

$$\min_{x \in \mathcal{C}} f(x) \ . \tag{4.2}$$

Because of the constraint, even if f is still smooth, the tools we have used so far no longer apply. Consider the basic one-dimensional problem:

$$\min_{x \in [0,1]} x \ . \tag{4.3}$$

This problem is friendly: the objective is convex, the constraint set $[0, 1]$ is convex and compact. The minimizer is 0, yet $\nabla f(0) = 1 \neq 0$ and so for constrained convex optimization, the global characterization of minimizers (Proposition 2.6) does not hold.

Fortunately, we can replace some concepts seen so far by generalizations that are very well adapted. First, let us introduce a tool that allows to remove the constraints.

4.2 Extended value functions

In optimization, it is often convenient to work with functions that take values in $\overline{\mathbb{R}} \triangleq \mathbb{R} \cup \{+\infty\}$. The prototypical example is the *indicator* function of a set.

Definition 4.1 (Indicator function)**.** *In convex analysis, the indicator function $\iota_{\mathcal{C}}$ of a subset $\mathcal{C}$ of $\mathbb{R}^d$ is:*

$$\begin{aligned} \iota_{\mathcal{C}} : \mathbb{R}^d &\to \overline{\mathbb{R}} \\ x &\mapsto \begin{cases} 0 \ , & \text{if } x \in \mathcal{C} \ , \\ +\infty \ , & \text{otherwise} \ . \end{cases} \end{aligned} \tag{4.4}$$

Note that this is different from the other indicator function that takes value 1 on the set and 0 outside[4]. The indicator function conveniently allows transforming all constrained minimization problems into unconstrained ones of course at the price of working with extended value functions:

$$\operatorname*{argmin}_{x \in \mathcal{C}} f(x) = \operatorname*{argmin}_{x \in \mathbb{R}^d} f(x) + \iota_{\mathcal{C}}(x) \ . \tag{4.5}$$

Definition 4.2. *The* domain *of a function $f : \mathbb{R}^d \to \overline{\mathbb{R}}$ is:*

$$\operatorname{dom} f \triangleq \{x \in \mathbb{R}^d, f(x) < +\infty\} \ . \tag{4.6}$$

The second tool to overcome non-differentiability is a substitute for the gradient.

4.3 Subdifferential of convex functions

Definition 4.3. *The subdifferential of f at x is the set of slopes of all affine minorants of f that are exact at x:*

$$\partial f(x) = \{u \in \mathbb{R}^d : \forall y \in \mathbb{R}^d, \ f(y) \geq f(x) + \langle u, x - y \rangle\} \ . \tag{4.7}$$

An element of the subdifferential is called a subgradient.

[4] this poor function is rarely convex

Note that contrary to the gradient, the subdifferential is a set-valued mapping: the subdifferential at one point may contain more that one subgradient. It may also be empty at some point (Exercise 4.16), but it's not on the interior of the domain if f is convex (Exercise 4.22).

It is a generalization of the gradient in the following sense.

Proposition 4.4. *Let f be a convex differentiable function. Then the subdifferential only contains the gradient:*

$$\partial f(x) = \{\nabla f(x)\} \ . \tag{4.8}$$

Note: the converse is true (if the subdifferential reduces to a point, f is differentiable at this point).

The subdifferential allows to elegantly characterize the minimizers of *all* convex functions, even the nondifferentiable ones.

Proposition 4.5 (Fermat's rule)**.** *Let f be convex. Then for all x,*

$$x \in \operatorname{argmin} f \Leftrightarrow 0 \in \partial f(x) \ . \tag{4.9}$$

The proof is 1 line and left to the reader.

Example 4.6. *The subdifferential of the absolute value is:*

$$\partial |\cdot|(x) = \begin{cases} \{x/|x|\}, & \text{if } x \neq 0 \ , \\ [-1, 1], & \text{otherwise} \ . \end{cases} \tag{4.10}$$

What is the subdifferential of $\iota_{[0,1]}$?

Proposition 4.7 (Subdifferential of sum)**.** *The subdifferential of the sum contains the sum of subdifferentials:*

$$\partial(f + g) \supset \partial f + \partial g \ , \tag{4.11}$$

(to be understood as the Minkowski sum). Equality does not hold in general, although the counter examples are pathological (Exercise 4.23).

Remark 4.8 (Subgradient descent)**.** *If you look again with nonsmooth eyes at Proposition 3.1 regarding gradient descent on Lipschitz convex functions, we actually never used the fact that f was differentiable: we only started with $f(x_k) - f(x^*) \leq \langle \nabla f(x_k), x_k - x^* \rangle$. Since a subgradient of f at x_k, by definition, also satisfies this inequality, it means we can replace $\nabla f(x_k)$ by any $g_k \in \partial f(x_k)$ and the proof still holds.*

This algorithm is called subgradient descent.

The last tool we need for nonsmooth optimization is the proximal operator.

4.4 The proximal operator

So far we have brushed away the existence of minimizers, and only assumed they exist. In classical analysis, the celebrated Weierstrass theorem says that a continuous function is bounded and reaches its minimum on any compact. We will provide a slight relaxation of this theorem that handles a slightly weaker property than convexity.

Definition 4.9 (Lower semi-conitnuity)**.** *The function $f : \mathbb{R}^d \to \overline{\mathbb{R}}$ is lower semicontinuous at $x \in \operatorname{dom} f$ if for any sequence (x_k) converging to x,*

$$f(x) \leq \liminf_{k\to\infty} f(x_k) \ . \tag{4.12}$$

Any continuous function is clearly lower semicontinuous.

Remark 4.10. *Lower semicontinuous functions are also referred to as* closed *functions. This is because a function is l.s.c. if and only if its epigraph*

$$\operatorname{epi} f = \{(x,t) : \ f(x) \leq t\} \subset \mathbb{R}^d \times \mathbb{R} \tag{4.13}$$

is closed.

Proposition 4.11 (Existence of minimizers, the direct method of calculus of variations)**.** *Let $f : \mathbb{R}^d \to \overline{\mathbb{R}}$ be a coercive lower semicontinuous function, with non empty domain. Then f admits at least one minimizer.*

Note: this result is only valid in finite dimension.

Proof. Let x_0 such that $f(x_0) < +\infty$. Since f is coercive, $f(x) \to_{\|x\|\to\infty} +\infty$. Let M such that $\|x\| \geq M \implies f(x) \geq f(x_0)$. Let $\mathcal{B}$ be the ball of center 0, radius M. Since we are in finite dimension, $\mathcal{B}$ is compact.

Let $f^* = \inf_{x\in\mathcal{B}} f(x)$. In general, we could have $f^* = -\infty$ (think $d = 1$, $M = 1$, $f(x) = 1/x$ except $f(0) = 0$). But we'll show that since f is l.s.c., it cant be the case.

Assume $f^* = -\infty$. We can construct a sequence (x_k) in $\mathcal{B}$ such that $\forall k \in \mathbb{N}, f(x_k) \leq -k$. But $\mathcal{B}$ is compact: we can extract a converging subsequence, that we call y_k, converging to $\tilde{x} \in \mathcal{B}$.

By lower semicontinuity of f, $f(\tilde{x}) \leq \liminf_{k\to\infty} f(y_k) = -\infty$ which is not possible since f does not take value $-\infty$.

Hence f^* is finite. Now we do the exact same reasoning to construct (x_k) such that $f(x_k) \leq f^* + 1/k$. Take a converging subsequence, use lower semicontinuity to show that the limit point satisfies $f(\tilde{x}) \leq f^*$ and thus $f(\tilde{x}) = f^*$. □

Definition 4.12. *The function $f : \mathbb{R}^d \to \overline{\mathbb{R}}$ is said to be (or more rigorously belong to) $\Gamma_0(\mathbb{R}^d)$ if it is:*

- *convex*
- *proper (its domain is not empty)*
- *lower semicontinuous*

When is $\iota_{\mathcal{C}}$ Γ_0?

Definition 4.13 (Proximal operator)**.** *Let $f \in \Gamma_0(\mathbb{R}^d)$. The proximal operator of f evaluated at x is*

$$\operatorname{prox}_f(x) = \operatorname*{argmin}_{y \in \mathbb{R}^d} f(y) + \frac{1}{2}\|x - y\|^2 \ . \tag{4.14}$$

Why is it well-defined (why does the argmin contain exactly one point)?

The following is a useful characterization of proxs.

Proposition 4.14. *Characterization of proximal operators Let $f \in \Gamma_0(\mathbb{R}^d)$. Then $p = \operatorname{prox}_f(x)$ if and only if $x - p \in \partial f(p)$.*

This justifies the frequent formulation $\operatorname{prox}_f = (\mathrm{Id} + \partial f)^{-1}$ (a.k.a. the prox is the resolvent of the subdifferential).

Proof. Apply Fermat's rule. □

Proximal operators may seem new, but without realizing it you know and have used some particular instances: if $\mathcal{C}$ is non empty, closed and convex, $\operatorname{prox}_{\iota_\mathcal{C}}$ is the (well-defined) projection onto $\mathcal{C}$. For this and other reasons, proxs can be thought of as generalization of projections.

Proposition 4.15 (Firm non expansivity of proximal operators)**.** *Proximal operators are firmly non expansive:*

4.5 The proximal point algorithm

By Fermat's rule and the characterization of proxs (Proposition 4.14), x^* is a minimizer of f if and only if it is a fixed point of prox_f: $x^* = \operatorname{prox}_f(x^*)$.

Let's jump ahead to the continuous time version of gradient descent that we'll see later, the gradient flow. The gradient flow is an Ordinary Differential Equation (ODE):

$$\begin{aligned} x(0) &= x_0, \\ \dot{x}(t) &= -\nabla f(x(t)) \ . \end{aligned} \tag{4.15}$$

Suppose one wants to numerically approximate the solution to this equation. We pick a time discretization step $\eta > 0$, define time instants $t_k = k\eta$ for $k \in \mathbb{N}$. We will construct a sequence x_k that should approximate the continuous version $x(t_k)$. We approximate the derivative $\dot{x}(t_k)$ by finite differences $\frac{x_{k+1} - x_k}{\eta}$.

Thus a discrete approximation of (4.15) is:

$$x_{k+1} - x_k = -\eta \nabla x_k \ ; \tag{4.16}$$

this is a *forward* Euler discretization scheme, that gives the iterations of gradient descent.

On the other hand, one could chose a *backward* Euler scheme, also called *implicit* (because it does not give an explicit value of x_k):

$$x_{k+1} - x_k = -\eta \nabla x_{k+1} \ . \tag{4.17}$$

Rewriting this as $(\mathrm{Id} + \eta \nabla f)(x_{k+1}) = x_k$, it becomes visible that this scheme is the proximal point algorithm!