

Optimization for large scale machine learning

ENS, M2 Info

Mathurin Massias
mathurin.massias@gmail.com

Contents

1	Motivation: gradient descent on ordinary least squares	3
1.1	Why does statistical learning need optimization?	4
1.2	Solving least squares with gradient descent	6
1.3	Numerical puzzles	7
1.4	Exercises	8
2	Reminder on convex analysis	10
2.1	Convexity and minimizers	10
2.2	Exercises	11
2.2.1	Convexity	11
2.2.2	Gradient	12
3	Gradient descent on convex functions	13
3.1	Basic properties and convergence rates	13
3.2	Exercises	17
3.2.1	Convexity inequalities	17
3.2.2	Gradient descent	17
4	Non-smooth convex optimization	19
4.1	Constrained optimization	19
4.2	Extended value functions	20
4.3	Subdifferential of convex functions	20
4.4	The proximal operator	22
4.5	The proximal point algorithm	23
4.6	The proximal gradient descent algorithm	24
4.7	Exercises	25
4.7.1	Subdifferential	25
4.7.2	Constrained optimization	25

5	Duality	26
5.1	The Fenchel transform	26
5.2	Fenchel-Rockafellar duality	28
5.3	Primal dual algorithms	29
5.4	Lagrangian duality	31
5.5	Exercises	33
6	Second order optimization: Newton method	34
6.1	Newton method	34
6.2	Quasi Newton methods	36
6.2.1	The Broyden/SR1 formula	36
6.2.2	BFGS	37
6.3	DFP	39
6.4	Exercises	39
7	The finite sum structure: stochastic methods	40
7.1	Stochastic gradient descent	40
7.2	Variance reduction	41
8	Continuous view: gradient flows and Nesterov	43
8.1	Gradient flow	43
8.2	An ODE modeling Nesterov acceleration	44
8.3	Exercises	44
9	Mirror descent	46
9.1	The mirror descent algorithm	46
9.2	Mirror descent on the simplex: the exponentiated gradient algorithm	47
9.3	Online learning and mirror descent	48
10	Implicit regularization	49
10.1	Implicit bias of GD on least squares	49
10.2	“Implicit” bias of mirror descent	49
10.3	Logistic regression on separable data	51
10.4	Matrix factorization: can everything be explained in terms of norms?	51
11	Automatic and implicit differentiation	51
11.1	Forward and backward automatic differentiation	51
12	Variational inequalities	53
12.1	Solving VIs: the extragradient method	55

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 calls 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:

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

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 than the smallest value taken by f , we are more interested in finding a minimizer, that is

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

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 represents 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

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

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 Gaussian 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:

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

Minimization of the negative log-likelihood in x gives

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

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:*

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

[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):

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

with the Huber loss defined as:

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

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 choose the best ρ .

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

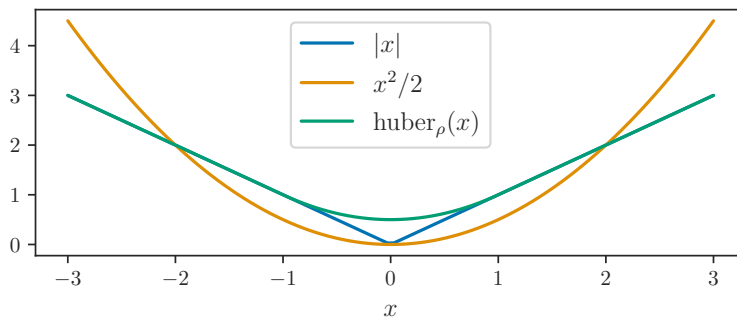


Figure 1: ℓ_1 , squared ℓ_2 and Huber ($\rho = 1$) losses

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 :

$$b_i \approx f_x(a_i) , \quad (1.9)$$

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

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

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.).

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

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

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:

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

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 $\mu \text{Id} \preceq A^\top A \preceq L \text{Id}$. 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:*

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

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

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

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

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

Observe that

$$(1 - \eta L) \text{Id} \preceq \text{Id} - \eta A^\top A \preceq (1 - \eta \mu) \text{Id} , \quad (1.16)$$

hence

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

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

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 rate¹. 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.*

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

We can also obtain a rate in objective value:

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

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

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

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:

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

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?

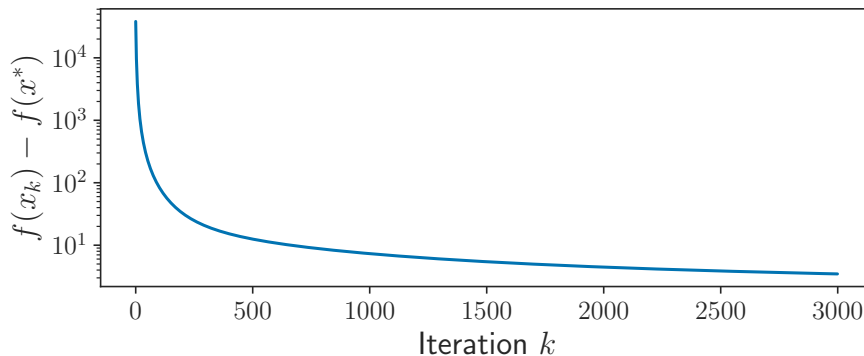


Figure 2: Objective convergence rate of GD on OLS on a random i.i.d. $A \in \mathbb{R}^{500 \times 500}$.

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?

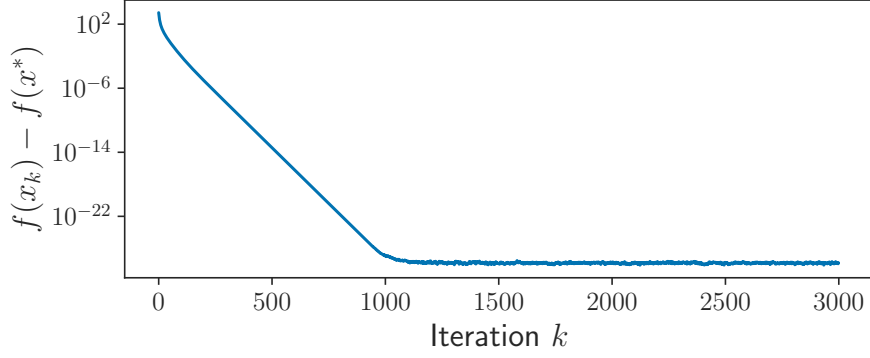


Figure 3: Objective convergence rate of GD on OLS on a random i.i.d. $A \in \mathbb{R}^{200 \times 300}$.

1.4 Exercises

Exercise 1.5. ☹️ Let $A \in \mathbb{R}^{n \times d}$. Show that $\text{Ker } A = \text{Ker } A^*A$.

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:

$$\min \frac{1}{2} \|Ax - b\|^2, \quad (1.22)$$

amounts to solving $A^*Ax = A^*b$ (aka the normal equations).

Show that the set of solutions is:

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

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,

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

is given by:

$$\hat{x} = \hat{x}_c, \quad (1.24)$$

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

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 number² of $A^\top A$ is equal to 1. Show that gradient descent with stepsize $1/L$ converges in a single iteration for the problem $\min \frac{1}{2} \|Ax - b\|^2$.

²i.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 aims at solving an optimization problem. For the Ordinary Least Squares objective function, we have proved two types of convergence results³:

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

Both results 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 $f : \mathbb{R}^d \rightarrow \mathbb{R}$. 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 $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is convex if and only if it lies below its cords:*

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

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 $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is μ -strongly convex if for all $x, y \in \mathbb{R}^d$,*

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

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 9.

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 is 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 $f : \mathbb{R}^d \rightarrow \mathbb{R}$ be a convex function. Then all local minimizers of f are also global.*

³Does one imply the other? Under which condition?

Proposition 2.5 (Above its tangents). *Let $f : \mathbb{R}^d \rightarrow \mathbb{R}$ 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 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*

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

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 $f : \mathcal{Y} \rightarrow \mathbb{R}$ be a convex function and $A : \mathcal{X} \rightarrow \mathcal{Y}$ 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 $f : \mathbb{R}^d \rightarrow \mathbb{R}$ be convex. Let $n \in \mathbb{N}$, $x_1, \dots, x_n \in \mathbb{R}^d$, and let $\lambda_1, \dots, \lambda_n$ 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 $f : \mathbb{R}^d \rightarrow \mathbb{R}$ 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 a 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 $A \in \mathbb{R}^{n \times d}$, $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and $g : \mathbb{R}^d \rightarrow \mathbb{R}$ defined as $g(x) = f(Ax)$ for all $x \in \mathbb{R}^d$. Show that

$$\begin{aligned}\nabla g(x) &= A^* \nabla f(Ax) \ , \\ \nabla^2 g(x) &= A^* \nabla^2 f(Ax) A \ .\end{aligned}$$

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 the 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 properties that influence the convergence rate we obtain?

3.1 Basic properties and convergence rates

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*

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

and

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

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 step size.
- 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,

$$f(x_k) - f(x^*) \leq \langle \nabla f(x_k), x_k - x^* \rangle \quad (3.3)$$

$$\leq \frac{1}{\eta} \langle x_k - x_{k+1}, x_k - x^* \rangle \quad (3.4)$$

$$\leq \frac{1}{2\eta} (\|x_k - x_{k+1}\|^2 + \|x_k - x^*\|^2 - \|x_{k+1} - x^*\|^2) . \quad (3.5)$$

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:

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

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

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. \square

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 $x, y \in \text{dom } f$,*

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\| . \quad (3.8)$$

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 $x, y \in \text{dom } f$,*

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

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:

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

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

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

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

Proposition 3.5 (Convergence rate of gradient descent on L -smooth functions). *Let $f : \mathbb{R}^d \rightarrow \mathbb{R}$ 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:*

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

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

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

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

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

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

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

We then use the convexity of f :

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

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

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

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

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

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

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

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

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

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

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

Proposition 3.6. *Let $f : \mathbb{R}^d \rightarrow \mathbb{R}$ 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:*

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

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

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

Combined with the descent lemma,

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

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

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

3.2 Exercises

3.2.1 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 $x \in \mathbb{R}^d$, $\nabla^2 f(x) \preceq L \text{Id}$.

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 $x \in \mathbb{R}^d$, $\mu \text{Id} \preceq \nabla^2 f(x)$.

Exercise 3.11 (Polyak-Łojasiewicz inequality). ☹☹ Let f be a μ -strongly-convex and differentiable function. Let $x^* = \text{argmin} f(x)$. 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) ,

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

3.2.2 Gradient descent

Exercise 3.14 (Exercise 8.4 in discrete time). ☹☹ Let $f : \mathbb{R}^d \rightarrow \mathbb{R}$ 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.

Exercise 3.17 (Gradient descent is equivariant to rotation). Let $f : \mathbb{R}^d \rightarrow \mathbb{R}$ be convex and Lipschitz differentiable (not required for the exercise, just to be in a setup where gradient descent is relevant). Let U be a rotation matrix ($U^\top U = \text{Id}$) and $\phi : \mathbb{R}^d \rightarrow \mathbb{R}$ a reparametrization of f defined as $\phi = f(U\cdot)$.

Show that gradient descent is equivariant to rotation: the iterates of GD of f and ϕ respectively, (x_k) and (y_k) , started at $x_0 = Uy_0$, satisfy $x_k = Uy_k$ for all $k \in \mathbb{N}$. How do the respective losses compare?

Show that studying the convergence of gradient descent on a quadratic can be reduced to studying gradient descent on a separable quadratic (meaning, with diagonal Hessian).

4 Non-smooth convex optimization

For a differentiable objective function $f : \mathbb{R}^d \rightarrow \mathbb{R}$, 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 value, 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 value, 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 the 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 want 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*

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

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:

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

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:

$$\min_{x \in [0,1]} x . \quad (4.3)$$

This problem is friendly: the objective is convex, and 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 with generalizations that are very well adapted. First, let us introduce a tool that allows removing 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:*

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

Note that this is different from the other indicator function that takes value 1 on the set and 0 outside⁴. The indicator function conveniently allows transforming all constrained minimization problems into unconstrained ones of course at the price of working with extended value functions:

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

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

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

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 :*

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

An element of the subdifferential is called a subgradient.

⁴this 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 than one subgradient. It may also be empty at some point ([Exercise 4.16](#)); the set of points where the subdifferential is not empty is called its *domain* (not to be confounded with for extended-value functions). As [Exercise 4.22](#) shows, if f is convex the domain of ∂f contains the interior of the domain of f .

The subdifferential 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:*

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

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

The subdifferential allows an elegant characterization of the minimizers of *all* convex functions, even the nondifferentiable ones.

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

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

The proof is 1 line and left to the reader.

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

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

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

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

$$\partial(f + g) \supset \partial f + \partial g , \quad (4.11)$$

(to be understood as the Minkowski sum). Equality does not hold in general, although the counterexamples 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 continuity.

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

$$f(x) \leq \liminf_{k \rightarrow \infty} f(x_k) . \quad (4.12)$$

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*

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

is closed.

Proposition 4.11 (Existence of minimizers, the direct method of calculus of variations). *Let $f : \mathbb{R}^d \rightarrow \overline{\mathbb{R}}$ 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, $f(x) \rightarrow_{\|x\| \rightarrow \infty} +\infty$. 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 can't 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 , $f(\tilde{x}) \leq \liminf_{k \rightarrow \infty} f(y_k) = -\infty$ 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 \tilde{x} satisfies $f(\tilde{x}) \leq f^*$ and thus $f(\tilde{x}) = f^*$. \square

Definition 4.12. *The function $f : \mathbb{R}^d \rightarrow \overline{\mathbb{R}}$ 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 $\iota_{\mathcal{C}} \Gamma_0$?

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

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

Why is it well-defined (why does the infimum exist? why is it attained? 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 $p = \text{prox}_f(x)$ if and only if $x - p \in \partial f(p)$.*

This justifies the frequent formulation $\text{prox}_f = (\text{Id} + \partial f)^{-1}$ (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, $\text{prox}_{\iota_{\mathcal{C}}}$ is the (well-defined) projection onto \mathcal{C} . For this and other reasons, proxs can be thought of as a generalization of projections.

4.5 The proximal point algorithm

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):

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

Suppose one wants to numerically approximate the solution to this equation. We pick a time discretization step $\eta > 0$, and 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:

$$x_{k+1} - x_k = -\eta \nabla f(x_k) ; \quad (4.16)$$

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):

$$x_{k+1} - x_k = -\eta \nabla f(x_{k+1}) . \quad (4.17)$$

Rewriting this as $(\text{Id} + \eta \nabla f)(x_{k+1}) = x_k$, it becomes visible that this scheme is 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 : $x^* = \text{prox}_f(x^*)$.

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

$$\|x - y\|^2 \leq \langle \text{prox}_f(x) - \text{prox}_f(y), x - y \rangle . \quad (4.18)$$

By Cauchy-Schwarz inequality, this immediately implies that proxs are non-expansive:

$$\|\text{prox}_f(x) - \text{prox}_f(y)\| \leq \|x - y\| . \quad (4.19)$$

Proof. Let $p_x = \text{prox}_f(x)$, $p_y = \text{prox}_f(y)$. The prox characterization for p_x is $x - p_x \in \partial f(p_x)$, so by definition of the subdifferential:

$$f(p_y) \geq f(p_x) + \langle x - p_x, p_y - p_x \rangle . \quad (4.20)$$

Do the same for p_y , sum the two inequations to cancel out the f 's, reorder. \square

TODO add results on firmly non expansive operators, composition, α -averaged operators.

4.6 The proximal gradient descent algorithm

$$x_{k+1} = \text{prox}_{\gamma g}(x_k - \gamma \nabla f(x_k)) \Leftrightarrow \frac{x_k - x_{k+1}}{\gamma} - \nabla f(x_k) \in \partial g(x_{k+1}) \quad (4.21)$$

Hence for all x ,

$$g(x_{k+1}) - g(x) \leq \langle \frac{x_k - x_{k+1}}{\gamma} - \nabla f(x_k), x_{k+1} - x \rangle \quad (4.22)$$

and by the descent lemma:

$$f(x_{k+1}) - f(x_k) \leq \langle \nabla f(x_k), x_{k+1} - x_k \rangle + \frac{L}{2} \|x_{k+1} - x_k\|^2 . \quad (4.23)$$

By using $x = x_k$ and summing, we get a first nice result: the proximal gradient descent method is a descent method for $0 < \gamma < 2/L$:

$$F(x_{k+1}) - F(x_k) \leq \left(\frac{L}{2} - \frac{1}{\gamma} \right) \|x_{k+1} - x_k\|^2 . \quad (4.24)$$

Back to our general proof, we sum [Equations \(4.22\)](#) and [\(4.23\)](#):

$$F(x) - F(x_{k+1}) - f(x) + f(x_k) \geq \frac{1}{\gamma} \langle x_{k+1} - x_k, x_{k+1} - x \rangle - \frac{L}{2} \|x_{k+1} - x_k\|^2 - \langle \nabla f(x_k), x_k - x \rangle$$

$$F(x) - F(x_{k+1}) \geq \frac{1}{\gamma} \langle x_{k+1} - x_k, x_{k+1} - x \rangle - \frac{L}{2} \|x_{k+1} - x_k\|^2 + D_f(x, x_k) \quad (4.25)$$

$$\geq \frac{L}{2} \|x - x_{k+1}\|^2 - \frac{L}{2} \|x_k - x\|^2 \quad (4.26)$$

TODO general γ ? Apply in $x = x^*$, sum and use the fact that suboptimality decreases at each iteration to obtain:

$$k(F(x_k) - F(x^*)) \leq \sum_{t=1}^k F(x_t) - F(x^*) \leq \frac{L}{2} \|x_0 - x^*\|^2 . \quad (4.27)$$

4.7 Exercises

4.7.1 Subdifferential

Exercise 4.16. ☹ Provide an example of convex function which has an empty subdifferential at some point of its domain.

Exercise 4.17. ☹ Let $f : \mathbb{R}^d \rightarrow \mathbb{R}$ be separable: $f(x) = \sum_1^d f_i(x_i)$ where the f_i 's are functions of the real variable.

Show that $\partial f(x) = \partial f_1(x_1) \times \dots \times \partial f_d(x_d)$.

Compute the subdifferential of the ℓ_1 -norm.

Exercise 4.18. ☹ Show that the subdifferential of a function at a point writes as an intersection of half-spaces. Show that it is convex and closed.

Exercise 4.19. ☹ Show that the subdifferential of ι_C at $x \in C$ is equal to the normal cone of C at x , that is:

$$\mathcal{N}_C(x) = \{u : \forall z \in C, \langle u, z - x \rangle \leq 0\}$$

Exercise 4.20. ☹☹ Compute the subdifferential of the Euclidean norm at any point in \mathbb{R}^d .

Exercise 4.21 (Exercise 4.25 revisited). ☹☹ Show that the first order optimality condition for differentiable convex constrained optimization rewrites:

$$x^* \in \underset{C}{\operatorname{argmin}} f(x) \Leftrightarrow -\nabla f(x^*) \in \mathcal{N}_C(x^*) .$$

Exercise 4.22 (Non emptiness of subdifferential). ☹☹☹ Let $f : \mathbb{R}^d \rightarrow \overline{\mathbb{R}}$ be convex. Show that if $x \in \operatorname{int}(\operatorname{dom} f)$, $\partial f(x)$ is non empty and compact.

Show that if x lies on the boundary of $\operatorname{dom} f$, $\partial f(x)$ is either empty or unbounded.

Exercise 4.23. ☹ This exercise is in dimension 2. Let \mathcal{C}_1 and \mathcal{C}_2 be two closed balls of strictly positive radius, that share a single point. Show that $\partial(\iota_{\mathcal{C}_1} + \iota_{\mathcal{C}_2})$ is a strict superset of $\partial\iota_{\mathcal{C}_1} + \iota_{\mathcal{C}_2}$ at this point.

4.7.2 Constrained optimization

Exercise 4.24. ☹ Show that the indicator function ι_C is convex (resp. lower semicontinuous, resp. proper) if C is convex (resp. closer, resp. nonempty).

Exercise 4.25 (Global optimality condition for constrained convex optimisation). ☹ Let $f : \mathbb{R}^d \rightarrow \mathbb{R}$ be a convex differentiable function, let C be a convex subset of \mathbb{R}^d . Show that $x^* \in \operatorname{argmin}_{x \in C} f(x)$ if and only if

$$\forall x \in C, \langle \nabla f(x^*), x - x^* \rangle \geq 0 .$$

Exercise 4.26. ☹☹ Let C be a non-empty closed convex set. Show that for all $x \in C, y \in \mathbb{R}^d$,

$$\|x - y\|^2 \geq \|x - \Pi_C(y)\|^2 + \|y - \Pi_C(y)\|^2 .$$

In particular this shows that $\|y - x\| \geq \|x - \Pi_C(y)\|$, which says that projection can only get you closer to optimum (taking $x = x^*$, if your current iterate is y).

5 Duality

One motivation (but not the single reason for this chapter to exist!) is the following question: when do we stop iterative algorithms? Usually we'd settle for $f(x_k) - f(x^*)$ less than some tolerance, but $f(x^*)$ is unknown.

5.1 The Fenchel transform

A fundamental tool in convex analysis is the Fenchel transform (sometimes called the Fenchel-Legendre or Legendre transform), which is, in many ways, akin to the Fourier transform in signal processing.

Definition 5.1 (Fenchel conjugate). *Let $f : \mathbb{R}^d \rightarrow \overline{\mathbb{R}}$. The Fenchel conjugate (or transform) of f is:*

$$f^*(u) = \sup_{x \in \mathbb{R}^d} \langle u, x \rangle - f(x) . \quad (5.1)$$

It is always convex, even if f is not ([Exercise 2.8](#)). To get familiar with it, compute the Fenchel transforms of $\|\cdot\|^2/2a$ ($a > 0$), $\|\cdot\|$ and $x \mapsto \sup_{y \in [-1,1]} x \cdot y$.

Geometrical interpretation of the Fenchel transform $f^*(u) \leq \alpha$ is equivalent to

$$\forall x \in \mathbb{R}^d, \quad \langle x, u \rangle - \alpha \leq f(x) , \quad (5.2)$$

meaning that $-\alpha$ is the intercept of a global affine minorant of f of slope u . $-f^*(u)$ is thus the biggest possible intercept of such minorants – beware, there may not exist such minorants ($f^*(u) = +\infty$).

Proposition 5.2 (Involution on Γ_0). *If $f \in \Gamma_0(\mathbb{R}^d)$, $f^{**} = f$.*

The proof technique is (to my knowledge) not really used in other optimization proofs, but it uses a beautiful argument.

Proof. By definition, $f^{**}(x) = \sup_u \langle u, x \rangle - f^*(u)$. Let us write $\phi_u(x) = \langle u, x \rangle - f^*(u)$. What are these functions? They are affine, and they globally lower bound f by Fenchel-Young inequality: $\forall x \in \mathbb{R}^d, \phi_u(x) \leq f(x)$. So

$$f^{**}(x) = \sup_u \phi_u(x) \leq \sup_{\substack{a \text{ affine} \\ a \leq f}} a(x) . \quad (5.3)$$

In fact, these two supremums are equal: take an affine function globally lower bounding f , with slope u . It must therefore be equal to $\langle x, u \rangle - c$ for some $c \in \mathbb{R}$. For all $x \in \mathbb{R}^d$, $-c + \langle x, u \rangle \leq f(x)$, so $c \geq \sup_x \langle x, u \rangle - f(x) = f^*(u)$. Therefore $\langle \cdot, u \rangle - c \leq \phi_u$.

Hence, the supremum over all affine functions lower bounding f is equal to the supremum over the “extremal” functions ϕ_u only:

$$f^{**}(x) = \sup_{\substack{a \text{ affine} \\ a \leq f}} a(x) . \quad (5.4)$$

This shows that $f^{**} \leq f$ all the time (no requirement on convexity).

Now we show equality when f is convex. Take $y \in \mathbb{R}, x \in \mathbb{R}^d$ such that $y < f(x)$ i.e. (y, x) not in the epigraph of f . The latter is convex, so by the celebrated Hahn-Banach theorem, we can find an hyperplane separating the two. This hyperplane corresponds to an affine function (TODO what if it is vertical?) that is above y at x , at globally lower bounds f . So $f^{**}(x) > y$. This concludes the proof.

TODO this also means that f^{**} is the best pointwise: if $f^{**}(x) < g(x)$ with g convex and $g \leq f$, then we can apply Hahn-Banach too and construct an affine a that is above $(x, f^{**}(x))$ and globally below f , which is impossible. □

Proposition 5.3 (Fenchel-Young inequality). *Let $f : \mathbb{R}^d \rightarrow \overline{\mathbb{R}}$. Then for all $x, u \in \mathbb{R}^d$,*

$$f(x) + f^*(u) \geq \langle x, u \rangle . \quad (5.5)$$

Equality holds if and only if $u \in \partial f(x)$.

Proof. The first part is trivial by definition of f^* . Then,

$$\begin{aligned} f^*(u) = \langle x, u \rangle - f(x) &\Leftrightarrow x \in \operatorname{argmin} f - \langle \cdot, u \rangle \\ &\Leftrightarrow 0 \in \partial(f - \langle \cdot, u \rangle)(x) \\ &\Leftrightarrow u \in \partial f(x) . \end{aligned}$$

□

Proposition 5.4. *If $f \in \Gamma_0(\mathbb{R}^d)$, $\partial f^* = (\partial f)^{-1}$, in the sense that:*

$$u \in \partial f(x) \Leftrightarrow x \in \partial f^*(u) . \quad (5.6)$$

Proof. Use then Fenchel-Young equality for f and then use $f^{**} = f$. □

Proposition 5.5 (Moreau decomposition formula). *Let $f \in \Gamma_0(\mathbb{R}^d)$. Then*

$$\operatorname{prox}_{\gamma f} + \gamma \operatorname{prox}_{\gamma^{-1} f^*}(\cdot / \gamma) = \operatorname{Id} \quad (5.7)$$

Proof.

$$p = \operatorname{prox}_{\gamma f}(x) \Leftrightarrow \frac{x - p}{\gamma} \in \partial f(p) \quad (5.8)$$

$$\Leftrightarrow p \in \partial f^*\left(\frac{x - p}{\gamma}\right) \quad (5.9)$$

$$\Leftrightarrow \frac{x}{\gamma} - \frac{x - p}{\gamma} \in \partial \frac{1}{\gamma} f^*\left(\frac{x - p}{\gamma}\right) \quad (5.10)$$

$$\Leftrightarrow \frac{x - p}{\gamma} = \operatorname{prox}_{f^*/\gamma}\left(\frac{x}{\gamma}\right) . \quad (5.11)$$

□

Proposition 5.6. *f is μ -strongly convex iff f^* is $\frac{1}{\mu}$ -smooth.*

5.2 Fenchel-Rockafellar duality

Many problems in Machine Learning have the following form:

$$\min_{x \in \mathbb{R}^d} f(Ax) + g(x) = P(x) , \quad (5.12)$$

with $A \in \mathbb{R}^{n \times d}$, $f \in \Gamma_0(\mathbb{R}^n)$ and $g \in \Gamma_0(\mathbb{R}^d)$.

Definition 5.7. *The Fenchel-Rockafellar dual problem of [Problem \(12\)](#) is:*

$$\min_{y \in \mathbb{R}^n} f^*(y) + g^*(-A^\top y) = D(y) . \quad (5.13)$$

It is often alternatively written as a concave maximization problem

$$\max_{y \in \mathbb{R}^n} -f^*(y) - g^*(-A^\top y) . \quad (5.14)$$

Where does this problem come from? It can be obtained by Lagrangian duality (??):

$$\inf_{x \in \mathbb{R}^d} f(Ax) + g(x) = \inf_{x \in \mathbb{R}^d, z \in \mathbb{R}^n} f(z) + g(x) \quad \text{s.t. } Ax = z \quad (5.15)$$

$$= \inf_{x \in \mathbb{R}^d, z \in \mathbb{R}^n} f(z) + g(x) + \iota_{\{z\}}(Ax) \quad (5.16)$$

$$= \inf_{x \in \mathbb{R}^d, z \in \mathbb{R}^n} f(z) + g(x) + \sup_{y \in \mathbb{R}^n} \langle y, Ax - z \rangle \quad (5.17)$$

$$= \inf_{x \in \mathbb{R}^d, z \in \mathbb{R}^n} \sup_{y \in \mathbb{R}^n} f(z) + g(x) + \langle y, Ax - z \rangle \quad (5.18)$$

Let's swap inf and sup. By doing so, we lose the equivalence ($\sup \inf \leq \inf \sup$, without equality in general). We have:

$$\sup_{y \in \mathbb{R}^n} \inf_{x \in \mathbb{R}^d, z \in \mathbb{R}^n} f(z) + g(x) + \langle y, Ax - z \rangle \quad (5.19)$$

$$= \sup_{y \in \mathbb{R}^n} \inf_{x \in \mathbb{R}^d} g(x) + \langle y, Ax \rangle + \inf_{z \in \mathbb{R}^n} f(z) - \langle y, z \rangle \quad (5.20)$$

$$= \sup_{y \in \mathbb{R}^n} - \sup_{x \in \mathbb{R}^d} -g(x) + \langle -A^\top y, x \rangle - \sup_{z \in \mathbb{R}^n} -f(z) + \langle y, z \rangle \quad (5.21)$$

$$= \sup_{y \in \mathbb{R}^n} -g^*(-A^\top y) - f^*(y) , \quad (5.22)$$

that is our dual problem.

Proposition 5.8 (Weak duality). *For all $x, y \in \mathbb{R}^d \times \mathbb{R}^n$, $P(x) + D(y) \geq 0$.*

Proof. Write the sum, group f with f^* and g with g^* , then use the Fenchel-Young inequality. \square

Proposition 5.9. *Let $x^*, y^* \in \mathbb{R}^d \times \mathbb{R}^n$. The following conditions are equivalent:*

1. $x^* \in \operatorname{argmin} P$, $y^* \in \operatorname{argmin} D(y)$, $\inf P + \inf D = 0$.
2. $P(x^*) + D(y^*) = 0$

3. $f(Ax^*) + f^*(y^*) = \langle Ax^*, y^* \rangle$ and $g(x^*) + g^*(-A^\top u^*) = -\langle x^*, A^\top u^* \rangle$,
4. $-A^\top u^* \in \partial g(x^*)$ and $y^* \in \partial f(Ax^*)$
5. $x^* \in \partial g^*(-A^\top u^*)$ and $Ax^* \in \partial f^*(y^*)$

Proof. 1 same as 2 is OK.

2 same as 3: Do the same as in the proof of [Proposition 5.8](#), and use that the sum of two positive terms is 0, if and only if both terms are 0.

3 same as 4: equality in Fenchel-Young case

4 same as 5: [Proposition 5.4](#) □

Definition 5.10 (Saddle point of the Lagrangian). *The pair $x^*, y^* \in \mathbb{R}^d \times \mathbb{R}^n$ is said to be a saddle-point of the Lagrangian if:*

$$\forall x, y \in \mathbb{R}^d \times \mathbb{R}^n, \mathcal{L}(x^*, y^*) \leq \mathcal{L}(x, y^*) \leq \mathcal{L}(x, y^*) . \quad (5.23)$$

Proposition 5.11. *If $x^*, y^* \in \mathbb{R}^d \times \mathbb{R}^n$ is a saddle-point of the Lagrangian, then x^* solves the primal and y^* solves the dual.*

Proof. Write the two inequalities stating that the pair is a saddle point, make subgradients appear, and use [Proposition 5.9](#). □

Definition 5.12 (Qualification condition). *The following inclusion is called a qualification condition:*

$$0 \in \text{int}(\text{dom } f - A \text{ dom } g) . \quad (5.24)$$

Where does it come from? It can be seen as a stronger version of “the primal objective P is proper”:

$$\begin{aligned} \text{dom } P \neq \emptyset &\Leftrightarrow \exists x \in \mathbb{R}^d, f(Ax) + g(x) < +\infty \\ &\Leftrightarrow \exists x \in \mathbb{R}^d, f(Ax) < +\infty, g(x) < +\infty \\ &\Leftrightarrow \exists x \in \mathbb{R}^d, y \in \mathbb{R}^n, y = Ax, f(y) < +\infty, g(x) < +\infty \\ &\Leftrightarrow \exists x \in \mathbb{R}^d, y \in \mathbb{R}^n, y = Ax, y \in \text{dom } f, x \in \text{dom } g \\ &\Leftrightarrow \exists x \in \text{dom } g, y \in \text{dom } f, 0 = y - Ax \\ &\Leftrightarrow 0 \in \text{dom } f - A \text{ dom } g . \end{aligned} \quad (5.25)$$

Proposition 5.13 (Strong duality). *Suppose that QC holds. Then there exists a saddle point of the Lagrangian, and $\inf P + \inf D = 0$ (the duality gap is 0 at optimum).*

Proof. The proof is not easy. □

5.3 Primal dual algorithms

In this section, we introduce more algorithms to deal with composite problems, based on duality/splitting.

Chambolle-Pock algorithm The Chambolle-Pock algorithm solves problems of the form:

$$\min_{x \in \mathbb{R}^d} f(Ax) + g(x), \quad (5.26)$$

where $f \in \Gamma_0(\mathbb{R}^n)$, $g \in \Gamma_0(\mathbb{R}^d)$, but both need not be smooth. It requires access to their proximal operators. Let σ and τ be positive stepsizes such that $\sigma\tau\|A\|_2^2 < 1$. Let $\theta \in]0, 1]$. The Chambolle-Pock iterations write:

$$\begin{cases} \bar{y}_k = y_k + \theta(y_k - y_{k-1}) \\ x_{k+1} = \text{prox}_{\tau g}(x_k - \tau A^\top \bar{y}_k) \\ y_{k+1} = \text{prox}_{\sigma f^*}(y_k + \sigma Ax_k) \end{cases} \quad (5.27)$$

Note: The interpolation parameter θ is most often taken equal to 1. The interpolation step can be performed on x instead of y , which yields a different version of the algorithm.

Vu-Condat algorithm Vu and Condat independently proposed an algorithm that handles an additional smooth term h in the objective:

$$\min_{x \in \mathbb{R}^d} f(Ax) + g(x) + h(x), \quad (5.28)$$

where $f \in \Gamma_0(\mathbb{R}^n)$, $g \in \Gamma_0(\mathbb{R}^d)$, $h \in \Gamma_0(\mathbb{R}^d)$, f and g 's proximal operators are known, and h is L_h -smooth⁵.

For stepsizes τ and σ satisfying⁶:

$$\frac{\tau\sigma}{\|A\|^2} + \frac{\tau L_h}{2} \leq 1, \quad (5.29)$$

the Condat-Vu algorithm reads:

$$\begin{cases} \bar{y}_k = y_k + \theta(y_k - y_{k-1}) \\ x_{k+1} = \text{prox}_{\tau g}(x_k - \tau \nabla h(x_k) \tau A^\top \bar{y}_k), \\ y_{k+1} = \text{prox}_{\sigma f^*}(y_k + \sigma Ax_k). \end{cases} \quad (5.30)$$

ADMM The Alternating Direction Method of Multipliers (ADMM method) is designed to solve problems of the form:

$$\min_{x \in \mathbb{R}^d, z \in \mathbb{R}^n} f(z) + g(x) \quad \text{s.t.} \quad Ax + Bz = c. \quad (5.31)$$

For simplicity and to match the setup of Chambolle-Pock we take $B = -\text{Id}$ and $c = 0$.

The augmented Lagrangian of parameter ρ is:

$$\mathcal{L}^\rho(x, y, z) = f(z) + g(x) + \langle y, Ax - z \rangle + \frac{\rho}{2} \|Ax - z\|^2 \quad (5.32)$$

The ADMM algorithm reads:

$$\begin{cases} x_{k+1} \in \text{argmin}_x \mathcal{L}^\rho(x, y_k, z_k) \\ y_{k+1} \in \text{argmin}_y \mathcal{L}^\rho(x_{k+1}, y, z_k) \\ y_{k+1} = y_k + r(Ax_{k+1} - z_{k+1}) \end{cases} \quad (5.33)$$

⁵for completeness let us mention that it can even handle the slightly more complex case where f is replaced by $f \square F$ with F strongly convex, leading to a second smooth term, F^* , in the saddle point formulation.

⁶this condition is in the case $\beta > 0$; otherwise inequality must be strict

5.4 Lagrangian duality

Fenchel-Rockafellar duality (Section 5.2) is a particular case of the more versatile *Lagrangian* duality. The latter is concerned with convex problems with affine constraints, that is problems of the form

$$\min_{x \in \mathcal{D}} f_0(x) \quad \text{s.t.} \quad f_i(x) \leq 0 \quad \forall i \in [m], \quad h_i(x) = 0 \quad \forall i \in [p] \quad (5.34)$$

where $\mathcal{D} \subset \mathbb{R}^d$ is convex, the functions f_0, \dots, f_m are convex and h_1, \dots, h_p are affine.

Definition 5.14 (Lagrangian, dual function, Lagrangian dual). *The Lagrangian associated to Problem (34) is the function*

$$\begin{aligned} \mathcal{L} : \mathcal{D} \times \mathbb{R}_+^n \times \mathbb{R}^p &\rightarrow \mathbb{R} \\ (x, \lambda, \mu) &\mapsto f_0(x) + \sum_{i=1}^n \lambda_i f_i(x) + \sum_{i=1}^p \mu_i h_i(x) ; \end{aligned} \quad (5.35)$$

Note that the variables λ_i associated with the inequalities are positive; the rationale is that, as soon as x does not satisfy one constraint, maximizing in λ_i or μ_i will make the Lagrangian go to $+\infty$. Also note that for any x feasible for Problem (34), the Lagrangian value for any λ, μ upper bounds $f_0(x)$.

Finally, the dual function is:

$$\begin{aligned} g : \mathbb{R}_+^n \times \mathbb{R}^p &\rightarrow \mathbb{R} \\ (\lambda, \mu) &\mapsto \min_{x \in \mathcal{D}} \mathcal{L}(x, \lambda, \mu) . \end{aligned} \quad (5.36)$$

and the Lagrange dual problem of Problem (34) is:

$$\max_{\lambda \in \mathbb{R}_+^n, \mu \in \mathbb{R}^p} g(\lambda, \mu) . \quad (5.37)$$

Proposition 5.15 (Weak duality). *Under the convexity and linearity assumptions on the f_i 's and h_i 's respectively, weak duality holds: the dual optimal value is smaller than the primal optimal value.*

Proof. Let $f : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}$, let $a_0, b \in \mathbb{R}^n \times \mathbb{R}^m$. Then:

$$\inf_{a \in \mathbb{R}^n} f(a, b) \leq f(a_0, b) \quad (5.38)$$

$$\sup_{b \in \mathbb{R}^m} \inf_{a \in \mathbb{R}^n} f(a, b) \leq \sup_{b \in \mathbb{R}^m} f(a_0, b) \quad (5.39)$$

The left-hand side is a constant, so the infimum of the right-hand side with respect to a_0 must be greater:

$$\sup_{b \in \mathbb{R}^m} \inf_{a \in \mathbb{R}^n} f(a, b) \leq \inf_{a \in \mathbb{R}^n} \sup_{b \in \mathbb{R}^m} f(a, b) . \quad (5.40)$$

So the sup of the infs is smaller than the inf of the sups, which concludes. \square

Under some conditions however, we have strong duality: the primal and dual values are equal. There are many such conditions; one of the most popular ones is Slater's condition.

Proposition 5.16 (Slater's condition). *If [Problem \(34\)](#) satisfies Slater's condition, meaning that it is strictly feasible:*

$$\exists x \in \mathcal{D}, \quad f_i(x) < 0 \quad \forall i \in [m], \quad h_i(x) = 0 \quad \forall i \in [p] \quad , \quad (5.41)$$

then strong duality holds: the primal and dual optimal values are equal.

5.5 Exercises

Exercise 5.17. ☞ Show that the Fenchel transform is order-reversing: if $f \leq g$, $g^* \leq f^*$.

Exercise 5.18. ☞☞ Compute the Fenchel transform of the ℓ_p -norm for $p \in [1, +\infty]$.

Exercise 5.19. ☞☞ Show that $x \mapsto \frac{1}{2}\|x\|^2$ is a fixed point of the Fenchel transform. Show that it is the only fixed point of the Fenchel transform.

Exercise 5.20. ☞ Let $f : \mathbb{R}^d \rightarrow \mathbb{R}$ be separable: $f(x) = \sum_1^d f_i(x_i)$ where the f_i 's are functions of the real variable. Show that $f^*(u) = (f_i^*(u_i))_{i \in [d]}$.

Exercise 5.21 (“Lipschitzing trick”). ☞☞ Let $f \in \Gamma_0(\mathbb{R}^d)$. Show that f is M -Lipschitz if and only if the domain of f^* is included in the Euclidean ball of center 0 and radius M .

Exercise 5.22 (Convolution smoothing). ☞ Let $f : \mathbb{R}^d \rightarrow \mathbb{R}$ be convex. Show that $f \square \frac{L}{2} \|\cdot\|^2$ is L -smooth.

Exercise 5.23 (Convolution Lipschitzing). ☞ Let $f : \mathbb{R}^d \rightarrow \mathbb{R}$ be convex. Show that $f \square L \|\cdot\|$ is L -Lipschitz.

Exercise 5.24. Let $\mathcal{D} = \mathbb{R} \times \mathbb{R}_+^*$, and $f : \mathcal{D} \rightarrow \mathbb{R}$ defined by $f(x, y) = e^{-x}$. Show that the problem

$$\min_{x, y \in \mathcal{D}} f(x, y) \quad \text{s.t.} \quad \frac{x^2}{y} \leq 0 \tag{5.42}$$

is convex. Compute its optimal value. Compute the Lagrangian dual and its optimal value. Does Slater's condition hold for this problem?

6 Second order optimization: Newton method

Let f be a twice differentiable function. We have seen that gradient descent for stepsize $1/L$ on a L smooth function can be interpreted as a Majorization-Minimization approach, using the descent lemma to upper bound f globally by an isotropic parabola (majorization step), then minimizing this upper bound (minimization step).

The issue with this majorization is that it is very coarse: in dimension 2, take $f(x) = x_1^2 + 0.001x^2$. For this function, $L = 1$ and $\mu = 0.001$: we get a linear convergence rate for gradient descent, but the constant is very poor (see also [Figure 2](#)), because the upper bounding parabolas that we construct have Hessian $L \text{Id}$, and $\mu \ll L$. In practice, this leads to very small steps being performed after the first one.

6.1 Newton method

We can try to preserve more information about the curvature, by using a 2nd order Taylor expansion of f – hence the name of “second order methods”:

$$f(y) \approx f(x) + \langle \nabla f(x), y - x \rangle + \frac{1}{2} \langle y - x, \nabla^2 f(x)(y - x) \rangle . \quad (6.1)$$

As in the MM interpretation of GD, we can derive an iterative algorithm for this: set $x = x_k$, and define x_{k+1} as the minimizer of the approximation. This yields

$$x_{k+1} = x_k - [\nabla^2 f(x_k)]^{-1} \nabla f(x_k) . \quad (6.2)$$

Compared to gradient descent, the scalar stepsize has been replaced with the application of a PSD matrix, that better captures the curvature of f .

Let us illustrate the possible behaviors for Newton’s method on a simple 1-dimensional convex function, $x \mapsto \sqrt{1+x^2}$. This function is \mathcal{C}^2 , with $f'(x) = \frac{x}{\sqrt{1+x^2}}$ and $f''(x) = \frac{1}{(1+x^2)^{3/2}}$, so Newton’s method iterates are given by

$$x_{k+1} = x_k - x_k(1 + x_k^2) = -x_k^3 . \quad (6.3)$$

There are therefore, depending on $|x_0|$, three possible behaviors:

- extremely fast convergence towards the minimizer if $|x_0| < 1$
- oscillations between 1 and -1 if $|x_0| = 1$
- extremely fast divergence if $|x_0| > 1$

This example illustrates a crucial fact about the vanilla Newton method: it only has *local* convergence properties – it converges if initialized close enough to the minimizer. But when it does, it enjoys a *superlinear* convergence rate. Let us formalize this and provide a proof.

Proposition 6.1. *If $x^* = \operatorname{argmin} f(x)$, f has M -Lipschitz Hessian, $\nabla^2 f(x^*) \succeq \mu \text{Id}_n$ ($\mu > 0$), and $\|x_0 - x^*\| \leq \frac{\mu}{2M}$, then:*

$$\|x_{k+1} - x^*\| \leq \frac{M}{\mu} \|x_k - x^*\|^2 . \quad (6.4)$$

It is hard to imagine a better rate than this one: the sequence $u_k = \frac{M}{\mu} \|x_k - x^*\|$ satisfies $u_{k+1} \leq u_k^2 \leq u_0^{2^k}$. If $u_0 < 1$, the number of iterations to reach $u_k < \varepsilon$ is of the order $\log(\log \varepsilon)$, that is, in practice, basically a constant!

Proof. By the fundamental theorem of calculus, for all x, h in \mathbb{R}^d :

$$\nabla f(x+h) - \nabla f(x) = \int_0^1 \nabla^2 f(x+th)h \, dt \quad (6.5)$$

Hence

$$\nabla f(x_k) = \int_0^1 \nabla^2 f(x^* + t(x_k - x^*))(x_k - x^*) \, dt \quad , \quad (6.6)$$

and thus

$$x_{k+1} - x^* = x_k - x^* - [\nabla^2 f(x_k)]^{-1} \int_0^1 \nabla^2 f(x^* + t(x_k - x^*))(x_k - x^*) \, dt \quad (6.7)$$

$$= [\nabla^2 f(x_k)]^{-1} \int_0^1 \left(\nabla^2 f(x_k) - \nabla^2 f(x^* + t(x_k - x^*)) \right) (x_k - x^*) \, dt \quad . \quad (6.8)$$

Let's control the operator norm of the matrix being integrated, using M -Lipschitzness of the Hessian

$$\int_0^1 \|\nabla^2 f(x_k) - \nabla^2 f(x^* + t(x_k - x^*))\| \, dt \leq \int_0^1 (1-t)M\|x_k - x^*\| \, dt = \frac{M}{2}\|x_k - x^*\| \quad (6.9)$$

Finally, let's bound the operator norm of $[\nabla^2 f(x_k)]^{-1}$:

$$\nabla^2 f(x_k) \succeq \nabla^2 f(x^*) - M\|x_k - x^*\| \text{Id}_n \succeq \left(\mu - M \underbrace{\|x_k - x^*\|}_{\leq \frac{\mu}{2M} \text{ (induction)}} \right) \text{Id}_n \succeq \frac{\mu}{2} \text{Id}_n \quad . \quad (6.10)$$

□

Despite the excellent convergence rate, Newton's method is not the “one algorithm to rule them all”:

- it may diverge if not started close enough to optimum
- the Hessian may not be invertible
- the Hessian is costly to compute
- the Hessian is even more costly to invert⁷ ($\mathcal{O}(d^3)$)

The principle of Quasi-Newton methods is to replace the inverse Hessian in [Equation \(6.2\)](#) by less costly approximations, in order to retain the very fast convergence rate while alleviating the computational burden.

⁷To compute $A^{-1}b$ numerically, one should solve the linear system $Ax = b$ with a LU factorization (Gaussian elimination) rather than invert A and apply to b : this is more stable and less costly. But the cost of the approach is still high

6.2 Quasi Newton methods

Algorithms in this section will have the following form:

$$\begin{cases} d_k = -B_k g_k \\ x_{k+1} = x_k + \rho_k d_k \end{cases} \quad \text{or} \quad \begin{cases} d_k = -H_k^{-1} g_k \\ x_{k+1} = x_k + \rho_k d_k \end{cases} \quad (6.11)$$

where $g_k = \nabla f(x_k)$, d_k is to be thought of as a descent direction, H_k (resp. B_k) is an approximation of the Hessian (resp. the inverse Hessian) of f at x_k , and ρ_k is a stepsize.

How to build such approximations? It is reasonable to impose that approximations should satisfy the *secant condition* or *Quasi-Newton relation*:

$$g_{k+1} - g_k = H_{k+1}(x_{k+1} - x_k) \quad \text{or} \quad x_{k+1} - x_k = B_{k+1}(g_{k+1} - g_k) \quad (6.12)$$

Where does this condition come from? Suppose we have finished iteration $k+1$ and, to compute x_{k+2} , we want to construct (then minimize) ϕ_{k+1} , a quadratic approximation of f having H_{k+1} as Hessian:

$$\phi_{k+1}(x) = f(x_{k+1}) + \langle g_{k+1}, x - x_{k+1} \rangle + \frac{1}{2} \langle x - x_{k+1}, H_{k+1}(x - x_{k+1}) \rangle . \quad (6.13)$$

This quadratic approximation of f is exact at x_{k+1} ($\phi(x_{k+1}) = f(x_{k+1})$), and tangent at x_{k+1} too ($\nabla \phi_{k+1}(x_{k+1}) = \nabla f(x_{k+1})$). To impose the secant condition is simply to impose that f and ϕ_{k+1} are also tangent at the previous iterate, x_k .

The approximations in this section are updated iteratively, using rank 1 or rank 2 “corrections” from B_k (resp. H_k) to B_{k+1} (resp. H_{k+1}).

Following the notation of Nocedal and Wright, we write

$$y_k = g_{k+1} - g_k = \nabla f(x_{k+1}) - \nabla f(x_k) , \quad (6.14)$$

$$s_k = x_{k+1} - x_k . \quad (6.15)$$

6.2.1 The Broyden/SR1 formula

First, we consider rank-one updates of the Hessian, i.e. updates of the form $H_{k+1} = H_k + \sigma v v^\top$. How should we set σ and v so that it satisfies the secant condition (6.12)? We want:

$$y_k = H_{k+1} s_k \quad (6.16)$$

$$= H_k s_k + \sigma v v^\top s_k \quad (6.17)$$

$$= H_k s_k + (\sigma v^\top s_k) v \quad (\text{rotation trick } ab^\top c = b^\top c a) , \quad (6.18)$$

This does not give us v , but it gives its direction: v is proportional to $y_k - H_k s_k$. So we know⁸ that there exists a scalar δ such that $v = \delta(y_k - H_k s_k)$, and we can plug it back in Equation (6.17) to solve for δ and obtain that

$$H_{k+1} = H_k + \frac{1}{s_k^\top (y_k - H_k s_k)} (y_k - H_k s_k)(y_k - H_k s_k)^\top . \quad (6.19)$$

⁸what if $v^\top s_k = 0$?

satisfies the secant condition. This is called the Broyden or SR1 formula.

This approximation removes the cost of computing the Hessian, and also its cost of storing (we just store the scalars $s_k^\top(y_k - H_k s_k)$ and vectors $(y_k - H_k s_k)$). But at first sight, we still have the $\mathcal{O}(d^3)$ cost of computing the update (linear system solving). However, since the update of the Hessian is rank-one, by the Sherman-Morrison formula (itself a special case of the Woodbury formula, see [Exercise 6.5](#)), so is the update of the inverse Hessian $B_k = H_k^{-1}$:

$$B_{k+1} = B_k - \frac{1}{(B_k y_k - s_k)^\top y_k} (B_k y_k - s_k)(B_k y_k - s_k)^\top. \quad (6.20)$$

Therefore if we start at a simple B_0 , say Id , then we only need to do, at each iteration, rank 1 updates that are cheap to store and apply ([Exercise 6.4](#))!

6.2.2 BFGS

The BFGS update is rank-two:

$$H_{k+1} = H_k + \frac{y_k y_k^\top}{y_k^\top s_k} - \frac{H_k s_k s_k^\top H_k}{s_k^\top H_k s_k}. \quad (6.21)$$

Proposition 6.2 (Inverse Hessian update for BFGS). *In terms of inverse Hessian B , the BFGS update (6.21) translates into*

$$B_{k+1} = \left(\text{Id} - \frac{s_k y_k^\top}{s_k^\top y_k} \right) B_k \left(\text{Id} - \frac{y_k s_k^\top}{s_k^\top y_k} \right) + \frac{s_k s_k^\top}{s_k^\top y_k}. \quad (6.22)$$

Proof. For lighter notation we write H , y and s for H_k , y_k , s_k . We apply the Sherman-Morrison formula twice in [Equation \(6.21\)](#). First

$$\left(H + \frac{y y^\top}{y^\top s} \right)^{-1} = H^{-1} - \frac{H^{-1} y y^\top H^{-1}}{s^\top y + y^\top H^{-1} y} \quad (6.23)$$

Then:

$$H_{k+1}^{-1} = H^{-1} - \underbrace{\frac{H^{-1} y y^\top H^{-1}}{s^\top y + y^\top H^{-1} y}}_{M_1} + \underbrace{\frac{\left(\text{Id} - \frac{H^{-1} y y^\top}{s^\top y + y^\top H^{-1} y} \right) s s^\top \left(\text{Id} - \frac{y y^\top H^{-1}}{s^\top y + y^\top H^{-1} y} \right)}{s^\top H s - s^\top H \left(H^{-1} - \frac{H^{-1} y y^\top H^{-1}}{s^\top y + y^\top H^{-1} y} \right) H s}}_{M_2} \quad (6.24)$$

The last term simplifies:

$$M_2 = \frac{s^\top y + y^\top H^{-1} y}{(s^\top y)^2} \left(\text{Id} - \frac{H^{-1} y y^\top}{s^\top y + y^\top H^{-1} y} \right) s s^\top \left(\text{Id} - \frac{y y^\top H^{-1}}{s^\top y + y^\top H^{-1} y} \right) \quad (6.25)$$

$$= \frac{1}{s^\top y} s s^\top + \frac{y^\top H^{-1} y}{(s^\top y)^2} s s^\top - \frac{1}{(s^\top y)^2} (H^{-1} y y^\top s s^\top + s s^\top y y^\top H^{-1}) + M_1 \quad (6.26)$$

$$= \frac{1}{s^\top y} s s^\top + \frac{y^\top H^{-1} y}{(s^\top y)^2} s s^\top - \frac{1}{s^\top y} (H^{-1} y s^\top + s y^\top H^{-1}) + M_1 \quad (6.27)$$

And so M_1 cancels out in Equation (6.24):

$$H_{k+1}^{-1} = H^{-1} + \frac{ss^\top}{s^\top y} + \frac{y^\top H^{-1} y s s^\top}{(s^\top y)^2} - \frac{sy^\top H^{-1} + H^{-1} y s^\top}{s^\top y} \quad (6.28)$$

$$= H^{-1} + \frac{ss^\top}{s^\top y} + \frac{sy^\top H^{-1} y s^\top}{(s^\top y)^2} - \frac{sy^\top H^{-1} + H^{-1} y s^\top}{s^\top y} \quad (6.29)$$

$$= \left(\text{Id} - \frac{sy^\top}{s^\top y} \right) H^{-1} \left(\text{Id} - \frac{ys^\top}{s^\top y} \right) + \frac{ss^\top}{s^\top y} \quad (6.30)$$

□

One very beautiful way to see this update is the following one.

Proposition 6.3 (BFGS as KL minimization). *The BFGS update Equation (6.21) is the solution of the following problem:*

$$\min_H -\log \det(H) + \langle H_k^{-1}, H \rangle \quad \text{subject to} \quad H = H^\top, Hs = y, \quad (6.31)$$

where the objective function is also known as the Stein loss, is equal to the KL divergence between two Gaussians of same means, aka the Bregman divergence induced by the negative logdet.

Proof. Introduce the Lagrangian:

$$\mathcal{L}(H, \lambda, \Theta) = -\log \det H + \langle H_k^{-1}, H \rangle + \langle \lambda, Hs - y \rangle + \langle \Theta, H - H^\top \rangle. \quad (6.32)$$

Notice that $\langle \lambda, Hs \rangle$ is a scalar, equal to its trace $\text{tr } \lambda^\top Hs = \text{tr } s \lambda^\top H = \langle \lambda s^\top, H \rangle$ where this time the scalar product is over matrices. Similarly, $\langle \Theta, H - H^\top \rangle = \langle \Theta, H \rangle - \langle \Theta, H^\top \rangle = \langle \Theta, H \rangle - \langle \Theta^\top, H \rangle = \langle \Theta - \Theta^\top, H \rangle$.

This gives us the two gradients we need to write the first order optimality condition over H . Using that the gradient of logdet is the inverse, we obtain:

$$\nabla_H \mathcal{L} = 0 \iff H^{-1} = H_k^{-1} + \lambda s^\top + \Theta - \Theta^\top. \quad (6.33)$$

First, we get rid of Θ : H must be symmetric hence H^{-1} too; H_k is symmetric (it's the approximate Hessian at the previous iteration) and this gives:

$$\Theta - \Theta^\top = \frac{1}{2}(s \lambda^\top - \lambda s^\top), \quad (6.34)$$

so

$$H^{-1} = H_k^{-1} + \frac{1}{2}(\lambda s^\top + s \lambda^\top). \quad (6.35)$$

To find λ , we use the other condition, $Hs = y$ aka $s = H^{-1}y$. This gives

$$\frac{1}{2}s^\top y \lambda = (s - H_k^{-1}y) - \frac{1}{2}\lambda^\top y s. \quad (6.36)$$

This may seem impossible to solve as λ appears twice, but it nevertheless tells us something: there exists a scalar a such that

$$\lambda = \frac{2}{s^\top y}(s - H_k^{-1}y) + as \quad . \quad (6.37)$$

Plug back this expression to obtain H^{-1} as a function of a , then use $H^{-1}y = s$ to solve in a (it's tedious, but a good exercise). Then plugging back a into λ we obtain

$$\lambda s^\top = \frac{2}{s^\top y}(s - H_k^{-1}y)s^\top + \left(\frac{1}{(s^\top y)^2}y^\top H_k^{-1}y - \frac{1}{s^\top y} \right) ss^\top \quad (6.38)$$

$$= -\frac{2}{s^\top y}H_k^{-1}ys^\top + \left(\frac{1}{(s^\top y)^2}y^\top H_k^{-1}y + \frac{1}{s^\top y} \right) ss^\top \quad . \quad (6.39)$$

To conclude let's be a little lazy: $\frac{1}{2}(\lambda s^\top + s\lambda^\top)$ is the symmetric part of λs^\top . As visible in Equation (6.39), the term in ss^\top is already symmetric, so we only need to symmetrize the other one:

$$\frac{1}{2}(\lambda s^\top + s\lambda^\top) = \left(\frac{1}{(s^\top y)^2}y^\top H_k^{-1}y + \frac{1}{s^\top y} \right) ss^\top - \frac{1}{s^\top y} (H_k^{-1}ys^\top + sy^\top H_k^{-1}) \quad . \quad (6.40)$$


and we recognize the computation we've been through in Equation (6.28), which allows to conclude invoking Proposition 6.2. \square

6.3 DFP

The DFP update is the same as BFGS, swapping the roles of H, y, s with B, s, y :

$$B_{k+1} = B_k + \frac{s_k s_k^\top}{y_k^\top s_k} - \frac{B_k y_k y_k^\top B_k}{y_k^\top B_k y_k} \quad . \quad (6.41)$$

6.4 Exercises

Exercise 6.4.  Show that any $n \times n$ matrix of rank 1 writes xy^\top with $x, y \in \mathbb{R}^n$. Show that the cost of multiplying by a rank one matrix is $2n$ instead of the usual n^2 .

Exercise 6.5 (Woodbury inverse formula).   Let $n, k \in \mathbb{N}$. Let $U \in \mathbb{R}^{n \times k}$ and $V \in \mathbb{R}^{k \times n}$. Show that

$$(\text{Id} + UV)^{-1} = \text{Id} - U(\text{Id} + VU)^{-1}V \quad . \quad (6.42)$$

Show that for any invertible A ,

$$(A + UV)^{-1} = A^{-1} - A^{-1}U(\text{Id} + VA^{-1}U)^{-1}VA^{-1} \quad . \quad (6.43)$$

When $k = 1$ and U and V respectively become column and row vectors u and v^\top , the formula is known as the Sherman-Morrison formula:

$$(A + uv^\top)^{-1} = A^{-1} - \frac{1}{1 + v^\top A^{-1}u} A^{-1}uv^\top A^{-1} \quad . \quad (6.44)$$

What is the naive cost of computing $A^{-1}uv^\top A^{-1}$? And the clever cost?

7 The finite sum structure: stochastic methods

7.1 Stochastic gradient descent

We consider problems with the *finite sum structure*:

$$\min_x f(x) = \sum_{i=1}^n f_i(x) . \quad (7.1)$$

Such problems notably arise from the maximum likelihood framework of [Section 1.1](#) where each f_i is the negative log-likelihood of a data point. Computing the gradient of f normally requires computing the gradient of each f_i , that scales with n . The principle of *stochastic* gradient descent is to replace $\nabla f(x_t)$ at iteration $t \in \mathbb{N}$ of gradient descent by $\nabla f_{i_t}(x_t)$ for some i_t sampled uniformly in $[n]$:

$$x_{t+1} = x_t - \eta \nabla f_{i_t}(x_t) . \quad (7.2)$$

The update direction is n times less costly to compute, and on average correct since:

$$\mathbb{E}_{i \sim \mathcal{U}([n])}[\nabla f_i(x)] = \sum_{i'=1}^n \mathbb{P}(i = i') \nabla f_{i'}(x) = \frac{1}{n} \sum_{i'=1}^n \nabla f_{i'}(x) = \nabla f(x) . \quad (7.3)$$

Let's make this rigorous.

Proposition 7.1. *Let f be a μ -strongly convex function with (unique) minimizer x^* . Let $0 < \eta \leq \frac{1}{\mu}$ and assume that the iterates of stochastic descent, x_t , have bounded gradients in expectation:*

$$\forall j \in [p], \forall t \in \mathbb{N}, \mathbb{E}[\|\nabla f_j(x_t)\|^2] \leq B^2 . \quad (7.4)$$

Then

$$\mathbb{E}[\|x_t - x^*\|^2] \leq (1 - \eta\mu)^t \|x_0 - x^*\|^2 + \frac{\eta}{\mu} B^2 . \quad (7.5)$$

As usual a few comments before moving into the proof:

- There's randomness so our rate is in expectation. We cannot hope for “absolute” rates without assumption since we could get very unlucky and sample always the same data point. But that is unlikely to happen: the same kind of rates with high probability can also be obtained.
- The expectation in (7.4) is over the indices (i_0, \dots, i_t) that are sampled.
- We have two bounds in the rate: an exponentially fast “forgetting” of the initial conditions, and an “irreducible” term: the upper bound does not go to 0 for fixed η . To make the second term small, we need η small, but this makes the first phase slower...
- The bound on the gradients is (this is my personal opinion) a circular reasoning (gradients are bounded if the algorithm converges, but it's needed to prove that the algorithm converges), and there's no real reason to think it holds a priori.

Proof.

$$\|x_{t+1} - x^*\|^2 = \|x_t - \eta \nabla f_{i_t}(x_t) - x^*\|^2 \quad (7.6)$$

$$= \|x_t - x^*\|^2 - 2\eta \langle \nabla f_{i_t}(x_t), x_t - x^* \rangle + \eta^2 \|\nabla f_{i_t}(x_t)\|^2. \quad (7.7)$$

We can take expectation with respect to (i_0, \dots, i_t) on both sides, and since x_t is independent of i_t ,

$$\mathbb{E}\|x_{t+1} - x^*\|^2 = \mathbb{E}\|x_t - x^*\|^2 - 2\eta \langle \nabla f(x_t), x_t - x^* \rangle + \eta^2 \mathbb{E}\|\nabla f_{i_t}(x_t)\|^2 \quad (7.8)$$

$$\leq \mathbb{E}\|x_t - x^*\|^2 - 2\eta \langle \nabla f(x_t), \mathbb{E}[x_t - x^*] \rangle + \eta^2 B^2. \quad (7.9)$$

Then since $\nabla f(x^*) = 0$, by μ -strong convexity of f (Exercise 3.9),

$$\mu \|x_t - x^*\|^2 \leq \langle \nabla f(x_t), x_t - x^* \rangle. \quad (7.10)$$

Thus

$$\mathbb{E}\|x_{t+1} - x^*\|^2 \leq (1 - 2\eta\mu) \mathbb{E}\|x_t - x^*\|^2 + \eta^2 B^2 \quad (7.11)$$

$$\leq (1 - 2\eta\mu)^{t+1} \mathbb{E}\|x_0 - x^*\|^2 + \eta^2 B^2 \sum_{t'=0}^t (1 - 2\eta\mu)^{t'} \quad (7.12)$$

$$\leq (1 - 2\eta\mu)^{t+1} \mathbb{E}\|x_0 - x^*\|^2 + \frac{1}{2\eta\mu} \eta^2 B^2 \quad (7.13)$$

$$(7.14)$$

□

7.2 Variance reduction

The problems of SGD: if you start at optimum, you move away from it. You need to reduce the stepsize to stop moving too much, but then there's a hard tradeoff because you still want to go fast to the minimizer.

f_i μ strongly convex and L -smooth,

GD: Fixed learning rate, $O(1/k)$ or linear, no problem structure exploitation. SGD: decreasing stepsize, $O(1/\sqrt{k})$ or $O(1/k)$. Can we get the best of both worlds? Variance is the issue.

	GD	SGD	Variance reduced
Cost per iteration	n	1	1
convex rate	sqrt	sqrt	sqrt
smooth convex rate	1 / k	1 / \sqrt{k}	1 / k
smooth strongly convex rate	linear	1 / k	linear

Idea of variance reduction: random variable X , estimate $\mathbb{E}X$ but with lower variance. Use a rv Y correlated with X with mean easier to estimate, introduce $\theta_\alpha = \alpha(X - y) + \mathbb{E}[Y]$ for $\alpha \in [0, 1]$. Unbiased if $\alpha = 1$ or $\mathbb{E}X = \mathbb{E}Y$.

Variance:

$$\mathbb{E}\theta_\alpha = \mathbb{E}[(\alpha(X - Y) - \alpha\mathbb{E}[(X - Y)])^2] \quad (7.15)$$

$$= \alpha^2 \mathbb{E}[(X - \mathbb{E}X - (Y - \mathbb{E}Y))^2] \quad (7.16)$$

$$= \alpha^2 (\text{Var } X + \text{Var } Y - 2 \text{cov}(X, Y)) \quad (7.17)$$

can be lower than $\text{Var } X$.

Application of idea: SAGA. X is the stochastic gradient, Y is the past stored gradient.

$$\begin{cases} j \sim \mathcal{U}(n) \\ g_j^k = \nabla f_j(x^k) \\ \text{thereststaysequal} \\ x^k = x^{k-1} - \gamma[g_j^k - g_j^{k-1} + \frac{1}{n} \sum_{i=1}^n g_i^{k-1}] \end{cases} \quad (7.18)$$

eEventually apply a prox

few comments: storage of gradients is expensive, for GLMs can be diminished. must be careful for update of the mean. sparse vs dense issue.

SAG: biased gradient update:

$$x^k = x^{k-1} - \gamma[\frac{g_j^k - g_j^{k-1}}{n} + \frac{1}{n} \sum_{i=1}^n g_i^{k-1}] \quad (7.19)$$

SVRG: snapshot, reference point \tilde{x} .

$$x^k = x^{k-1} - \gamma[\nabla f_j(x^k) - \nabla f_j^{\tilde{x}} + \frac{1}{n} \sum_{i=1}^n \nabla f_i^{\tilde{x}}] \quad (7.20)$$

update frequency of \tilde{x} ?

8 Continuous view: gradient flows and Nesterov

8.1 Gradient flow

In this section, we take the limit of algorithms as the stepsize goes to zero and times become continuous. Indeed gradient descent can be seen as a forward Euler discretization of the following Ordinary Differential Equation (ODE):

$$\dot{x}(t) = -\nabla f(x(t)) \quad (8.1)$$

while the proximal point algorithm algorithm is a backward/implicit discretization, since $\text{prox}_f = (\text{Id} + \nabla f)^{-1}$ for a differentiable function f .

Studying the gradient flow is often easier than studying its discretized counterpart, gradient descent: the same results can usually be obtained, but the proofs are much simpler and require less assumptions. In particular, stepsizes and Lipschitz constants are not involved.

Proposition 8.1. *The function value $f(x(t))$ decreases along the flow. If f is convex, the gradient norm $\|\nabla f(x(t))\|$ also decreases along the flow.*

Proof. A direct calculation shows:

$$\frac{d}{dt}f(x(t)) = \langle \nabla f(x(t)), \frac{d}{dt}x(t) \rangle = -\|\nabla f(x(t))\|^2 \leq 0 . \quad (8.2)$$

If f is convex, its Hessian is semidefinite positive⁹ and thus:

$$\frac{d}{dt}\|\nabla f(x(t))\|^2 = -\nabla f(x(t))\nabla^2 f(x(t))\nabla f(x(t)) \leq 0 . \quad (8.3)$$

□

Proposition 8.2 (Convergence rates). *Let x^* be a minimizer of f . Then*

$$f(x(t)) - f(x^*) \leq \frac{\|x(0) - x^*\|^2}{2t} \quad (8.4)$$

$$\|\nabla f(x(t))\|^2 \leq \frac{f(x(0))}{t} . \quad (8.5)$$

Proof. Again, differentiating a cleverly chosen functional gives:

$$\frac{d}{dt}\frac{1}{2}\|x(t) - x^*\|^2 = -\langle \nabla f(x(t)), x(t) - x^* \rangle \leq f(x^*) - f(x(t)) . \quad (8.6)$$

First, this shows that $x(t)$ gets closer to any minimizer as t increases since the RHS is negative. Second, since $f(x(t))$ is decreasing,

$$f(x(t)) - f(x^*) \leq \frac{1}{t} \int_0^t f(x(s)) - f(x^*) ds \leq \frac{1}{2t}\|x(0) - x^*\|^2 - \frac{1}{2t}\|x(t) - x^*\|^2 , \quad (8.7)$$

which proves the first result. For the second one, since the squared gradient norm decreases and is equal to $-\frac{d}{dt}f(x(t))$ by Equation (8.2),

$$\|\nabla f(x(t))\|^2 \leq -\frac{1}{t} \int_0^t \frac{d}{ds}f(x(s)) = \frac{1}{t}(f(x(0)) - f(x(t))) \leq \frac{1}{t}f(x(0)) . \quad (8.8)$$

□

⁹what if f is not twice differentiable?

8.2 An ODE modeling Nesterov acceleration

Can we do better than this $1/t$ rate? Su Boyd and Candès consider the following equation:

$$\ddot{x}(t) + \beta(t)\dot{x}(t) + \nabla(f(x(t))) = 0. \quad (8.9)$$

By letting $\epsilon(t) = \frac{A}{2}\|x - x^* + C\dot{x}\|^2 + B(f(x(t)) - f^*)$, they have

$$\dot{\epsilon}(t) = \frac{\dot{A}}{2}\|x - x^*\|^2 + (\dot{A}C + A(\dot{C} + 1 - C\beta))\langle x - x^*, \dot{x} \rangle - AC\langle x - x^*, \nabla f(x) \rangle \quad (8.10)$$

$$+ \left(\frac{\dot{A}C^2}{2} + AC(\dot{C} + 1 - C\beta)\right)\|\dot{x}\|^2 + (B - AC^2)\langle \dot{x}, \nabla f(x) \rangle + \dot{B}(f(x) - f^*) \quad (8.11)$$

To simplify the expression, they take many terms equal to 0 by choosing $B = AC^2$, $\dot{A}C^2 = 0$ so $\dot{A} = 0$, $\dot{C} + 1 - C\beta = 0$, so $\dot{C} = 1/2$.

yielding:

$$f(x(t)) - f(x^*) = \mathcal{O}\left(\frac{1}{B(t)}\right) = \mathcal{O}\left(\frac{1}{t^2}\right). \quad (8.12)$$

Discretization, Heavy ball vs Nesterov.

Possible discretization:

$$\frac{x_{k+1} - 2x_k + x_{k-1}}{h^2} + \frac{\alpha}{kh} \frac{x_{k+1} - x_k}{h} + \nabla f(x_k) = 0. \quad (8.13)$$

$$x_{k+1} = x_k + \left(1 - \frac{\alpha}{k}\right)(x_k - x_{k-1}) - h^2 \nabla f(x_k). \quad (8.14)$$

also $\frac{k-1}{k+2}$

Nesterov choice of momentum: $t_1 = 1$

$$\begin{cases} t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2} \\ x_{k+1} = y_k - \frac{1}{L} \nabla f(y_k) \\ y_{k+1} = x_k + \frac{t_k - 1}{t_{k+1}}(x_{k+1} - x_k) \end{cases} \quad (8.15)$$


The FISTA algorithm, same with prox.

Heavy Ball (different result?):

$$x_{k+1} = x_k - \alpha \nabla f(x_k) + \beta(x_k - x_{k+1}) \quad (8.16)$$

with choices $\alpha = \frac{4}{(\sqrt{L} + \sqrt{\mu})^2}$, $\beta = \frac{\sqrt{L} - \sqrt{\mu}}{\sqrt{L} + \sqrt{\mu}}$.

8.3 Exercises

Exercise 8.3 (Preliminaries).  Let $f : \mathcal{X} \rightarrow \mathbb{R}$ be twice differentiable. Let $\theta : \mathbb{R}_+ \rightarrow \mathcal{X}$ be differentiable. Show that

$$\begin{aligned} \frac{d}{dt} f(\theta(t)) &= \langle \nabla f(\theta(t)), \dot{\theta}(t) \rangle, \\ \frac{d}{dt} \nabla f(\theta(t)) &= \nabla^2 f(\theta(t)) \dot{\theta}(t). \end{aligned}$$

Exercise 8.4 (Everything decreases in gradient flow). ☕ Let $f : \mathbb{R}^d \rightarrow \mathbb{R}$ be a convex twice differentiable function. Let $x(t)$ be the gradient flow on f , defined as a solution of $\dot{x}(t) = -\nabla f(x(t))$.

Show that $f(x(t))$ decreases.

Show that $\|\nabla f(x(t))\|$ decreases.

Show that $\|x(t) - x^*\|$ decreases for any minimizer x^* of f .

Exercise 8.5. ☕ Do the results of [Exercise 8.4](#) hold when f is not convex?

9 Mirror descent

Nick Harvey lecture notes: <https://www.cs.ubc.ca/~nickhar/F18-531/Notes20.pdf>

Key to this algorithm is the concept of Bregman divergence.

Definition 9.1 (Bregman divergence). *Let J be a convex function. The Bregman divergence induced by J is $D : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ defined by:*

$$D(x, y) = J(x) - J(y) - \langle \nabla J(y), x - y \rangle \quad (9.1)$$

It is positive, it is usually not symmetric. It is point-separating if J is strictly convex. It is not like a distance, it is like a square distance. It is the difference at x between J and its linear upper bound at x . TODO graph.

Proposition 9.2 (Generalization of Pythagoras).

$$D(x, y) + D(z, x) - D(z, y) = \langle \nabla f(x) - \nabla f(y), x - z \rangle . \quad (9.2)$$

9.1 The mirror descent algorithm

Assumptions: J such that D is well defined, J ρ -strongly convex in some norm The mirror descent algorithm is given by the following equation: MD equations:

$$\nabla J(x_{k+1}) = \nabla J(x_k) - \eta \nabla f(x_k)$$

Proof of convergence rates, similar to that of subgradient descent:

$$\begin{aligned} f(x_k) - f(x^*) &\leq \langle \nabla f(x_k), x_k - x^* \rangle \\ &= \frac{1}{\eta} \langle J(x_k) - J(x_{k+1}), x_k - x^* \rangle \\ &= \frac{1}{\eta} [D(x_k, x_{k+1}) + D(x^*, x_k) - D(x^*, x_{k+1})] \end{aligned}$$

Sum to telescope:

$$\begin{aligned} \sum_1^t f(x_k) - f(x^*) &\leq \frac{1}{\eta} \left[\sum_1^t D(x_k, x_{k+1}) + D(x^*, x_1) - D(x^*, x_{t+1}) \right] \\ &\leq \frac{1}{\eta} \left[\sum_1^t D(x_k, x_{k+1}) + D(x^*, x_1) \right] \end{aligned}$$

It remains to bound the first term, for which we use strong convexity. Strong convexity means that $D(x_k, x_{k+1})$ is lower bounded, which is not very useful here. So we make the

opposite of a Bregman divergence appear:

$$\begin{aligned}
D(x_k, x_{k+1}) &= J(x_k) - J(x_{k+1}) - \langle \nabla J(x_{k+1}), x_k - x_{k+1} \rangle \\
&= \langle \nabla J(x_{k+1}) - \nabla J(x_k), x_{k+1} - x_k \rangle - D(x_{k+1}, x_k) \\
&= \langle -\eta \nabla f(x_k), x_{k+1} - x_k \rangle - D(x_{k+1}, x_k) \\
&= -\eta L \|x_{k+1} - x_k\| - \frac{\mu}{2} \|x_{k+1} - x_k\|^2 \\
&= \frac{\eta^2 L^2}{2\mu}
\end{aligned}$$

Note: handling projection is trivial because Bregman projection satisfies $D(x^*, x_{k+1}) \leq D(x^*, y_{k+1})$ where y_{k+1} is the intermediate step that gets projected into x_{k+1} .

Plug back and divide by t :

$$\frac{1}{t} \sum_1^t f(x_k) - f(x^*) \leq \eta \frac{L^2}{2\mu} + \frac{D(x^*, x_1)}{\eta t}$$

Minimize RHS in η gives $\eta = \sqrt{\frac{2\mu D}{L^2 t}}$ and an upper bound value of $\sqrt{\frac{2DL^2}{\mu t}}$.

Better rates: see also <https://www.cs.uic.edu/~zhangx/teaching/bregman.pdf>

9.2 Mirror descent on the simplex: the exponentiated gradient algorithm

The following potential, called *entropy*, is particularly adapted to optimization on the simplex:

$$J(x) = \sum_1^d x_i \log x_i . \quad (9.3)$$

It is 1-strongly convex with respect to the ℓ_1 norm (a result known as Pinsker's inequality), and its associated Bregman divergence is

$$D(x, y) = \sum_1^d x_i \log x_i - y_i \log y_i - (1 + \log y_i)(y_i - x_i) \quad (9.4)$$

$$= \sum_{i=1}^d x_i \log \frac{x_i}{y_i} + \sum_{i=1}^d x_i - \sum_{i=1}^d y_i \quad (9.5)$$

which is called the *Kullback-Leibler divergence* (usually considered for vectors x and y on the simplex, so the last two sums cancel). This Bregman divergence induces a nicely behaved projection onto the simplex. For $y \in \mathbb{R}_{++}^d$ (the case where y has vanishing entries is easy to handle similarly), let us compute

$$\operatorname{argmin}_{x \in \Delta_d} \sum_{i=1}^d x_i \log \frac{x_i}{y_i} + \sum_{i=1}^d x_i - \sum_{i=1}^d y_i \quad (9.6)$$

This problem can be solved by introducing the Lagrangian, with $\lambda \in \mathbb{R}$ and $\mu \in \mathbb{R}_+^d$

$$\mathcal{L}(x, \lambda, \mu) = \sum_{i=1}^d x_i \log \frac{x_i}{y_i} + \lambda \left(\sum_{i=1}^d x_i - 1 \right) - \sum_{i=1}^d \mu_i x_i \quad (9.7)$$

The saddle point conditions on \mathcal{L} yield

$$\begin{cases} \mu_i x_i = 0 \\ \lambda = \log \frac{x_i}{y_i} + 1 - \mu_i \text{ aka } x_i = y_i \exp(\lambda - 1 - \mu_i) \\ \sum_{i=1}^d x_i = 1 \end{cases} \quad (9.8)$$

The second equation shows that x_i cannot have value 0, hence $1 = \sum_{i=1}^d x_i = \exp(\lambda - 1) \sum_{i=1}^d y_i$, hence the solution is given by:

$$\operatorname{argmin}_{x \in \Delta_d} D(x, y) = \left(\frac{y_i}{\sum_{i=1}^d y_i} \right)_{i \in [d]} \quad (9.9)$$

Recall that the mirror descent iterates are given by $x_{k+1} = \nabla J^*(\nabla J(x_k) - \eta \nabla f(x_k))$, and we have

$$x = \nabla J^*(u) \Leftrightarrow u = \nabla J(x) \Leftrightarrow u = 1 + \log x \Leftrightarrow x = \exp(u - 1) \quad (9.10)$$

So, the mirror descent algorithm for the entropic potential is:

$$x_{k+1} = \exp(1 + \log x_k - \eta \nabla f(x_k) - 1) = x_k \exp(-\eta \nabla f(x_k)), \quad (9.11)$$

which is known as the exponentiated gradient algorithm. (TODO and normalization if projection)

9.3 Online learning and mirror descent

The online learning framework is the following: we don't minimize a function but, at time $t \in \mathbb{N}$, take a decision $x_t \in \mathbb{R}^d$, incur cost $f_t(x_t)$ (f_t possibly chosen in an adversarial fashion; we don't know it when picking x_t !), receive/observe $g_t \in \partial f_t(x_t)$ (information: how could we have done better for this stage), and can use it to take a new decision x_{t+1} .

How to measure the performance? The regret of an online algorithm

Definition 9.3 (Regret). *The regret at horizon $T \in \mathbb{N}$ of an algorithm/a sequence of actions x_t is:*

$$R(T) = \sum_{t=1}^T f_t(x_t) - \min_x \sum_{t=1}^T f_t(x) , \quad (9.12)$$

that is, the difference between the incurred cost and the minimal cost for a fixed strategy which knows all the f_t 's in hindsight.

Basic algorithm: subgradient descent. Works exactly as in the fixed f case under the same assumptions on f .

TODO example $f_t = x, 1 - x$ even/odd

10 Implicit regularization

Resources : Claire Boyer lecture notes: "This chapter heavily relies on the articles Hastie et al. (2019), Bartlett et al. (2021) and the lecture of Matus Telgarsky"

In classical statistical learning, it is often nice to control the norm of the solution in regression, or the margin in classification (Shalev Schwartz 2014 Understanding ML book). It is typically imposed by regularizing the objective function, using an additional square L2 term for example.

However, we have witnessed the advent of deep learning successes, where highly overparameterized, unregularized models which should have been prone to overfitting met stellar success. This has sparked new interest in the phenomenon of *implicit regularisation*, in which, amongst all solutions to an optimization problem, an algorithm always converges to a particular one – in this case, a “simple”, “good” minimizer of the empirical risk that generalizes well.

10.1 Implicit bias of GD on least squares

Consider the least square problem:

$$\min_x \frac{1}{2} \|Ax - b\|^2 \quad (10.1)$$

and the iterations of gradient descent on this problem, started at 0:

$$\begin{cases} x_0 = 0 \\ x_{k+1} = x_k - \eta A^\top (Ax_k - b) \end{cases} \quad (10.2)$$

A simple recursion shows that $x_k = \sum_0^k (\text{Id} - \eta A^\top A)^t \eta A^\top b$; if $\eta < 2/\|A\|^2$, the matrix series (called von Neumann series) converges¹⁰, to $(\text{Id} - (\text{Id} - \eta A^\top A)^\dagger)$ and so the iterates converge to $A^\dagger b$.

This solution is remarkable: it is the minimum norm solution to $A^\top Ax = A^\top b$ ¹¹.

There are many ways to show it (it can even be considered the definition of the pseudo inverse operator, I think). There are geometrical proofs and a nice proof by going to the dual TODO exercise.

Therefore, amongst all possible solutions to the linear system $A^\top Ax = A^\top b$, gradient descent “unknowingly” converges to the minimal norm one, without regularization.

What if one is interested in solutions that are minimal in other sense, e.g. minimize another functional than the Euclidean norm?

10.2 “Implicit” bias of mirror descent

J smooth and α -strongly convex, implicit bias of mirror descent on least squares? Mirror descent iterations:

¹⁰in the same way $\sum_0^{+\infty} \rho^k = (1 - \rho)^{-1}$

¹¹recall that in finite dimension there always exists such a solution, even when there is no solution to $Ax = b$; see [Exercise 1.5](#)

Proposition 10.1. *Consider the mirror descent iterations on least squares with potential smooth and strongly convex potential J :*

$$\nabla J(x_{k+1}) = \nabla J(x_k) - \eta A^*(Ax_k - b)$$

Then the iterates converge to the J minimal solution:

$$x_k \rightarrow \underset{x}{\operatorname{argmin}} J(x) \quad \text{s.t.} \quad Ax = b \quad (10.3)$$

Proof. Explicit “min- J solution” problem and its dual:

$$\text{Primal : } \hat{x} = \underset{x}{\operatorname{argmin}} J(x) = j(x) + \frac{\alpha}{2} \|x\|_2^2 \quad \text{s.t.} \quad Ax = b \quad (10.4)$$

$$\text{Dual : } \hat{\theta} = \underset{\theta}{\operatorname{argmin}} J^*(-A^*\theta) + \langle b, \theta \rangle \quad (10.5)$$

$$\text{Link : } -A^*\hat{\theta} = \alpha \hat{x} + \nabla j(\hat{x}) \quad \text{aka} \quad \hat{x} = \operatorname{prox}_{\frac{1}{\alpha}j}(-A^*\hat{\theta}/\alpha) \quad (10.6)$$

In fact, we only need strict convexity of J (in which case the interpretation as a proximal step for the primal variable x does not hold). J strictly convex makes J^* differentiable. J strongly convex so J^* smooth. Actually J^* Moreau envelope of j^* :

$$\begin{aligned} J^* &= (j + \frac{\alpha}{2} \|\cdot\|^2)^* \\ &= j^* \square \frac{1}{2\alpha} \|\cdot\|^2 \\ &= \inf_y j^*(y) + \frac{1}{2\alpha} \|\cdot - y\|^2 \quad (\text{Moreau envelope}) \end{aligned}$$

Since J is strongly convex, the dual is smooth and you can do gradient on it. The gradient of J^* writes as a prox in this case. Dual is smooth, do gradient descent on it:

$$\begin{aligned} \theta_{k+1} &= \theta_k - \eta [-A \nabla J^*(-A^*\theta_k) + b] \\ &= \theta_k - \eta [-A \operatorname{prox}_{\frac{1}{\alpha}j}(-A^*\theta_k) + b] \quad (\text{envelope thm.} + \text{Moreau decomposition}) \end{aligned}$$

Envelope theorem:

$$\begin{aligned} \nabla J^*(x) &= \frac{1}{\alpha} (x - \operatorname{prox}_{\alpha j}(x)) \\ &= \operatorname{prox}_{\frac{1}{\alpha}j}(x/\alpha) \quad (\text{Moreau decomposition formula}) \end{aligned}$$

The dual iterates θ_k solve the dual, the final step is to map them back to the primal using the primal-dual link equation. Inspired by link equation, let

$$x_k \triangleq \operatorname{prox}_{\frac{1}{\alpha}j}(-A^*\theta_k/\alpha), \text{ meaning } -A^*\theta_k = \alpha x_k + \nabla j(x_k) = \nabla J(x_k)$$

Plugging back x_k in dual GD iterates gives mirror descent:

$$\begin{aligned} -A^*\theta_{k+1} &= -A^*\theta_k + \eta A^*[-A \nabla J^*(-A^*\theta_k) + b] \\ \nabla J(x_{k+1}) &= \nabla J(x_k) - \eta A^*(Ax_k - b) \quad (\text{mirror descent}) \end{aligned}$$

Since $\theta_k \rightarrow \hat{\theta}$, $x_k \rightarrow \hat{x}$. □

10.3 Logistic regression on separable data

Motivation: controlling norm avoids overfitting, controls *stability* of solution. Typical example: Tikhonov, bound singular value.

$$(A^t A)^\dagger A^\top b = \sum_1^R \frac{1}{\sigma_i} v_i u_i^\top b$$

vs

$$(A^t A + \lambda \text{Id})^\dagger A^\top b = \sum_1^R \frac{\sigma_i}{(\sigma_i + \lambda)^2} v_i u_i^\top b$$

Iterative regularization Engl 1996

Implicit bias of GD on OLS. Generalization to Mirror descent

TODO give SVD as homework

1. the geometrical view with SVD for OLS: projection onto $\text{Span } A^\top$

Ref off the convex path: http://www.offconvex.org/2020/11/27/reg_dl_not_norm/

10.4 Matrix factorization: can everything be explained in terms of norms?

Deep linear nets for matrix factorization, ref <http://www.offconvex.org/2019/07/10/trajectories-linear-nets/>

Neyshabur thesis on implicit bias in DL: <https://arxiv.org/pdf/1709.01953.pdf>

11 Automatic and implicit differentiation

First order optimization methods require computing gradients of given functions. So far, all the functions were simple (least squares, logistic regression) and their gradient could be computed by hand.

We now turn to examples where the computation requires more advanced tools.

11.1 Forward and backward automatic differentiation

Note: AD acts on program, not on functions.

Consider the following scalar function:

$$f(x) = \exp(x^2 - 3x) \sin(x) + \cos(x^2 - 3x)x \quad (11.1)$$

It can be implemented through the following program in [Algorithm 1](#), which is itself representable by a Directed Acyclic Graph (DAG) in [Figure 4](#).

In this section we will introduce three ways to compute gradient/Jacobians of loss functions

Algorithm 1 Program implementing f

$$a = x^2$$

$$b = 3x$$

$$c = a + b$$

$$d = \exp(c)$$

$$e = \sin(x)$$

$$f = de$$

$$g = \cos(c)$$

$$h = gx$$

$$i = hf$$

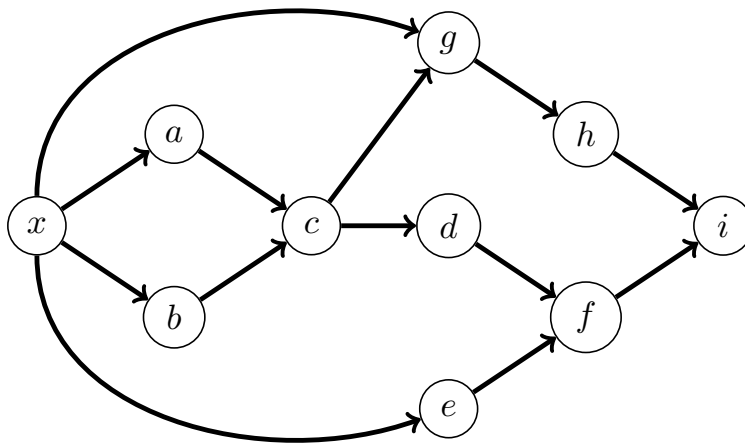


Figure 4: Directed acyclic graph representing the dependency graph of [Algorithm 1](#)

Tuning the weights of a neural network requires computing the Jacobian of the loss with respect to various parameters. Automatic differentiation, in particular Autodiff, backprop Autodiff for a one layer NN:

$$f(x) = W_2 \sigma(W_1 x) \tag{11.2}$$

12 Variational inequalities

In all this section K is a convex bounded subset of \mathbb{R}^d . The function $F : \mathbb{R}^d \rightarrow \mathbb{R}^d$ is called an operator.

Definition 12.1 (Strong and weak solutions to variational inequality problems). *We say that $x^* \in \mathbb{R}^d$ is a strong solution to the variational inequality problem, or simply to the variational inequality, associated to F if $x^* \in K$ and*

$$\forall y \in K, \langle F(x^*), x^* - y \rangle \leq 0 . \quad (12.1)$$

It is a weak solution if it belong to K and

$$\forall y \in K, \langle F(y), x^* - y \rangle \leq 0 . \quad (12.2)$$

In the unconstrained case, a strong solution is simply a point such that $F(x^*) = 0$. Variational inequalities generalize convex (constrained) minimization problems: to solve $\min_C f(x)$ is to find a strong solution to the VI associated to $F = \nabla f$.

Under certain conditions, strong and weak solutions are the same; these conditions involve a special property on F , monotonicity.

Definition 12.2 (Monotone operator). *The operator F is monotone if*

$$\forall x, y, \langle F(x) - F(y), x - y \rangle \geq 0 . \quad (12.3)$$

The most famous class of monotone operators is probably the one composed of the convex functions' gradients (see [Exercise 2.14](#)).

Proposition 12.3 (equivalence under monotonicity of F). *Let F be monotone. Then any strong solution to the VI is a weak solution to the VI. The converse is also true if F is continuous and K is convex, a result known as Minty's lemma¹².*

Proof. One direction is easy. To prove that any weak solution is a strong solution, let x^* be a weak solution, and let $y \in K$. Let $t \in]0, 1]$. Since K is convex, $x^* + t(y - x^*) \in K$, and so:

$$\langle F(x^* + t(x^* - y)), x^* - x^* + t(x^* - y) \rangle \leq 0 \quad (12.4)$$

$$t \langle F(x^* + t(x^* - y)), x^* - y \rangle \leq 0 \quad (12.5)$$

$$\langle F(x^* + t(x^* - y)), x^* - y \rangle \leq 0 . \quad (12.6)$$

Taking the limit as $t \rightarrow 0$, by continuity of F , yields that x^* is a strong solution since y was any point. \square

Variational inequalities generalize variational optimization problems (similarity with constrained optimization problems), but there is no objective anymore.

They handle more complex problems, for examples min-max problems arising in game theory.

¹²everything here is in finite dimension, but beware that the result may not hold in infinite dimension. Can you spot where this plays a role in the proof?

Example 12.4 (Rock paper scissors). *The rock-paper-scissors game is described by the payoff matrix $A = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & 1 \\ -1 & 1 & 0 \end{pmatrix}$. If Rock, Paper and Scissors choices are represented by 1, 2 and 3 respectively, then if player 1 plays i and player 2 plays j , player 1 receives payoff A_{ij} (and player 2 receives $-A_{ij}$).*

If Player 1 (P1) chooses a random strategy represented by a vector p in the 3-d simplex Δ_3 and Player 2 (P2) chooses $q \in \Delta_3$, then the expected payoff of P1 is:

$$p^\top Aq = \sum_{i,j=1}^3 p_i A_{ij} q_j . \quad (12.7)$$

The game for P1 is to solve $\max_p \min_q p^\top Aq$, while for P2 it is to solve $\max_q \min_p -p^\top Aq$, or equivalently $\min_q \max_p p^\top Aq$.

A result, due to von Neumann and reminiscent of [Section 5](#), states that the two objectives are the same:

$$\max_{p \in \Delta_3} \min_{q \in \Delta_3} p^\top Aq = \min_{q \in \Delta_3} \max_{p \in \Delta_3} p^\top Aq . \quad (12.8)$$

More generally, VIs are useful to model optimization problems of the form:

$$\min_u \max_v f(u, v) , \quad (12.9)$$

where f is differentiable, convex in u , and concave in v . Indeed, such a problem is equivalent to the VI $F = (\nabla_u f, -\nabla_v f)$.

To solve [Problem \(9\)](#), one may be tempted to perform gradient descent on u and gradient ascent on v simultaneously:

$$\begin{cases} u_{k+1} = u_k - \eta \nabla_u f(u_k) \\ v_{k+1} = v_k + \eta \nabla_v f(v_k) \end{cases} . \quad (12.10)$$

but unfortunately, this fails to converge even in very simple cases.

Example 12.5. *Consider the 2D problem $\min_{u \in \mathbb{R}} \max_{v \in \mathbb{R}} uv$, whose solution is $(0, 0)$. The iterates [\(12.10\)](#) on this problem read:*

$$\begin{cases} u_{k+1} = u_k - \eta v_k \\ v_{k+1} = v_k + \eta u_k \end{cases} . \quad (12.11)$$

It is easy to check that $\|(u_{k+1}, v_{k+1})\| = (1 + \eta)\|(u_k, v_k)\|$. Hence, the iterates will always diverge (worse: they get further away from the solution at every iteration). This is visible on [Figure 5](#).

We must therefore come up with a better method!

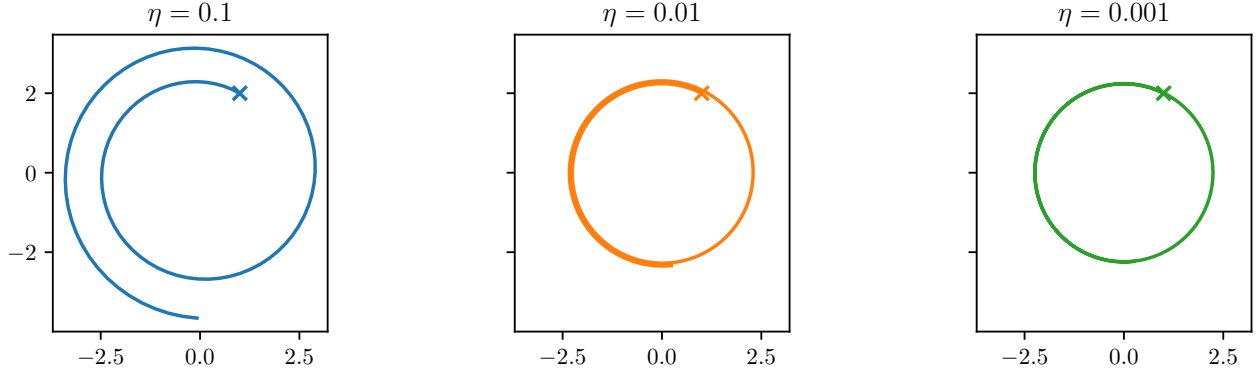


Figure 5: Gradient method for the saddle point problem $\min_{u \in \mathbb{R}} \max_{v \in \mathbb{R}} uv$, initialized at $(1, 1)$. No matter how small η , the iterates never converge, and in fact always go to infinity.

12.1 Solving VIs: the extragradient method

Suppose we want to solve a VI with an iterative algorithm. What is a relevant measure of error, since there is no objective involved? Guided by the definition of a weak solution, we will use the following value to quantify the quality of x :

$$\sup_{y \in K} \langle F(y), x - y \rangle. \quad (12.12)$$

Proposition 12.6 (Convergence rate of the extragradient method). *The extragradient method iterates are given by¹³:*

$$\begin{cases} x_k = z_{k-1} - \eta F(z_{k-1}) \\ z_k = z_{k-1} - \eta F(x_k) = z_{k-1} - \eta F(z_{k-1} - \eta F(z_{k-1})) \end{cases} \quad (12.13)$$

has the following convergence rates on the ergodic iterates $\bar{x}_t = \frac{1}{t} \sum_1^t x_k$:

- if F is bounded in norm by G and $\eta = \dots$, $\text{err } \bar{x}_t \leq \text{TODO}$
- if F is β Lipschitz and $\eta < 1/\beta$, rate is $/T$

Proof. In the whole proof, $y \in K$ is fixed. Let $t \in \mathbb{N}$, $k \in \llbracket 1, t \rrbracket$ By monotonicity of F ,

$$\langle F(y), x_k - y \rangle \leq \langle F(x_k), x_k - y \rangle. \quad (12.14)$$

Now, we want to use the definitions of x_k and z_k , and so we must make $x_{k+1} - x_k$ and TODO appears. We thus write:

$$\begin{aligned} \langle F(x_k), x_k - y \rangle &= \langle F(x_k), z_k - y \rangle + \langle F(z_{k-1}), x_k - z_k \rangle + \langle F(x_k) - F(z_{k-1}), x_k - z_k \rangle \\ &= \frac{1}{\eta} \langle z_{k+1} - z_k, z_k - y \rangle + \frac{1}{\eta} \langle z_{k-1} - x_k, x_k - z_k \rangle + \langle F(x_k) - F(z_{k-1}), x_k - z_k \rangle. \end{aligned} \quad (12.15)$$

¹³note: we cannot talk about extragradient *descent* since there is nothing being minimized here

We use the parallelogram identity on the two first terms of the right-hand side:

$$\langle z_{k+1} - z_k, z_k - y \rangle = \frac{1}{2} (\|z_{k+1} - y\|^2 - \|z_{k+1} - z_k\|^2 - \|z_k - y\|^2) \quad (12.16)$$

$$\langle z_{k-1} - x_k, x_k - z_k \rangle = \frac{1}{2} (\|z_{k+1} - z_k\|^2 + \|z_{k+1} - x_k\|^2 + \|x_k - z_k\|^2) \quad (12.17)$$

Summing, the second term of the RHS of the first line and the first term of the RHS of the second line cancel out. So the RHS in (12.15) is equal to:

$$\frac{1}{2\eta} (\|z_{k+1} - y\|^2 - \|z_k - y\|^2) - \frac{1}{2\eta} (\|z_{k+1} - x_k\|^2 + \|x_k - z_k\|^2) + \langle F(x_k) - F(z_{k-1}), x_k - z_k \rangle \quad (12.18)$$

The first term will telescope when we sum over k . It remains to handle the last one, involving F .

Bounded case: by applying Cauchy-Schwartz inequality, the triangular inequality, and Young inequality, we obtain

$$\langle F(x_k) - F(z_{k-1}), x_k - z_k \rangle \leq 2G\|x_k - z_k\| \leq 2\eta G^2 + \frac{\|x_k - z_k\|^2}{2\eta} . \quad (12.19)$$

Plug back in RHS, two terms cancel out. Summing over k from 1 to t , dividing by t and choosing η accordingly concludes the proof.

Lipschitz case:

$$\langle F(x_k) - F(z_{k-1}), x_k - z_k \rangle \leq \beta \|x_k - z_{k-1}\| \|x_k - z_k\| \leq \frac{\beta}{2} \|x_k - z_{k-1}\|^2 + \frac{\beta}{2} \|x_k - z_k\|^2 . \quad (12.20)$$

(12.18) is thus upper bounded by

$$\frac{1}{2\eta} (\|z_{k+1} - y\|^2 - \|z_k - y\|^2) + \left(\frac{1}{2\eta} - \frac{\beta}{2} \right) (\|z_{k+1} - x_k\|^2 + \|x_k - z_k\|^2) . \quad (12.21)$$

The second term is negative provided $\eta \leq 1/\beta$; summing, telescoping and dividing by t concludes similarly. \square