

Lecture notes on proximal methods and regularization

Clément W. Royer

M2 IASD & M2 MASH - 2022/2023

- The last version of these notes can be found at:
<https://www.lamsade.dauphine.fr/~croyer/ensdocs/PGL/LectureNotesOML-PGL.pdf>.
- Comments, typos, etc, can be sent to `clement.royer@lamsade.dauphine.fr`.
Thanks to the students who have sent feedback.
- **Major updates of the document**
 - 2022.12.11: Full version of the notes.
 - 2022.11.10: First version of the notes.

Foreword

The purpose of these lectures is to discuss regularization, a topic of primary importance in learning problems. The goals are as follows:

- Discuss the use of regularization terms in a data science context;
- Describe the main ideas behind proximal gradient techniques;
- Review sparse regularizers, with a focus on the LASSO technique.

Notations

- Scalars (i.e. reals) are denoted by lowercase letters: $a, b, c, \alpha, \beta, \gamma$.
- Vectors are denoted by **bold** lowercase letters: $\mathbf{a}, \mathbf{b}, \mathbf{c}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}$.
- Matrices are denoted by **bold** uppercase letters: $\mathbf{A}, \mathbf{B}, \mathbf{C}$.
- Sets are denoted by **bold** uppercase cursive letters : $\mathcal{A}, \mathcal{B}, \mathcal{C}$.
- The set of natural numbers (nonnegative integers) is denoted by $\mathbb{N}$; the set of integers is denoted by $\mathbb{Z}$.
- The set of real numbers is denoted by $\mathbb{R}$. Our notations for the subset of nonnegative real numbers and the set of positive real numbers are $\mathbb{R}_+$ and $\mathbb{R}_{++}$, respectively.
- The notation $\mathbb{R}^d$ is used for the set of vectors with $d \in \mathbb{N}$ real components; although we may not explicitly indicate it in the rest of these notes, we always assume that $d \geq 1$.
- A vector $\mathbf{x} \in \mathbb{R}^d$ is thought as a column vector, with $x_i \in \mathbb{R}$ denoting its i -th coordinate in the canonical basis of $\mathbb{R}^d$. We thus write $\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix}$, or, in a compact form, $\mathbf{x} = [x_i]_{1 \leq i \leq d}$.
- Given a column vector $\mathbf{x} \in \mathbb{R}^d$, the corresponding row vector is denoted by $\mathbf{x}^T$, so that $\mathbf{x}^T = [x_1 \cdots x_d]$ and $[\mathbf{x}^T]^T = \mathbf{x}$. The scalar product between two vectors in $\mathbb{R}^d$ is defined as $\mathbf{x}^T \mathbf{y} = \mathbf{y}^T \mathbf{x} = \sum_{i=1}^d x_i y_i$.
- The Euclidean norm of a vector $\mathbf{x} \in \mathbb{R}^d$ is defined by $\|\mathbf{x}\|_2 = \sqrt{\mathbf{x}^T \mathbf{x}}$.
- We use $\mathbb{R}^{n \times d}$ to denote the set of real rectangular matrices with n rows and d columns, where n and d will always be assumed to be at least 1. If $n = d$, $\mathbb{R}^{d \times d}$ refers to the set of square matrices of size d .
- We identify a matrix in $\mathbb{R}^{d \times 1}$ with its corresponding column vector in $\mathbb{R}^d$.
- Given a matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$, $\mathbf{A}_{ij}$ refers to the coefficient from the i -th row and the j -th column of $\mathbf{A}$: the diagonal of $\mathbf{A}$ is given by the coefficients $\mathbf{A}_{ii}$. Provided this notation is not ambiguous, we use the notations $\mathbf{A}$, $[\mathbf{A}_{ij}]_{\substack{1 \leq i \leq n \\ 1 \leq j \leq d}}$ and $[\mathbf{A}_{ij}]$ interchangeably.
- For every $d \geq 1$, $\mathbf{I}_d$ refers to the identity matrix in $\mathbb{R}^{d \times d}$ (with 1s on the diagonal and 0s elsewhere).

Contents

1	Introduction	5
1.1	Optimization beyond $\mathcal{C}^1$ functions	5
1.2	Regularized problems	6
2	Subgradient and proximal gradient methods	8
2.1	Subgradient methods	8
2.1.1	Subgradients and subdifferentials	8
2.1.2	Subgradient algorithm	9
2.2	Proximal methods	10
2.2.1	Proximal point method and proximal operator	10
2.2.2	Proximal gradient methods	11
2.3	Conclusion of Chapter 2	12
3	Regularized optimization	13
3.1	The case of ℓ_2 regularization	13
3.1.1	Proximal gradient and ℓ_2 regularization	13
3.1.2	Ridge linear regression	14
3.2	The case of ℓ_1 regularization	15
3.2.1	Motivation : Sparse models	15
3.2.2	Proximal gradient and ℓ_1 regularization	16
3.2.3	The LASSO estimator	17
3.3	Conclusion of Chapter 3	17
4	Exercises	18

Chapter 1

Introduction

In these lectures, we focus on classical characteristics of optimization problems arising in machine learning, that preclude from applying (stochastic) gradient methods as seen in the previous parts of the course. The tools used to develop optimization methods applicable to these problems. We first provide examples of problems where gradients cannot be computed in Section 1.1, then describe a particular problem structure in Section 1.2.

1.1 Optimization beyond $\mathcal{C}^1$ functions

The most standard continuous optimization techniques are based on using derivatives of the function to be optimized in order to determine directions in which this function can be improved locally. In particular, assuming that a function is $\mathcal{C}^1$ implies that the gradient of this function can be computed at any point (and that the gradient changes in a continuous fashion). It is however common to encounter functions for which gradients cannot be defined at every point, in which case we say then that the function is *not differentiable* at these points. In the rest of these notes, we will use the following terminology.

Definition 1.1.1 (Nonsmooth functions) A function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is called **nonsmooth** if it is not differentiable everywhere.

Example 1.1.1 *Examples of nonsmooth functions*

- $w \mapsto |w|$ from $\mathbb{R}$ to $\mathbb{R}$;
- $w \mapsto \|w\|_1 = \sum_{i=1}^d |[w]_i|$ from $\mathbb{R}^d$ to $\mathbb{R}$;
- *ReLU activation*: $w \mapsto \max\{w, 0\}$ from $\mathbb{R}$ to $\mathbb{R}$.

Note that the functions above are continuous. The class of nonsmooth functions also includes discontinuous functions, such as

$$w \mapsto \begin{cases} 1 & \text{if } w \neq 0 \\ 0 & \text{if } w = 0. \end{cases}$$

Since nonsmooth functions are not differentiable everywhere, optimization problems that involve nonsmooth functions cannot a priori be solved via gradient-based methods. In Chapter 2, we will describe several tools used for optimizing certain classes of nonsmooth functions. Beforehand, we mention several other approaches for nonsmooth optimization.

Smooth reformulation One useful technique consists in reformulating a nonsmooth problem as a smooth one when possible. For instance, the problem $\min_{w \in \mathbb{R}} |w|$ is equivalent to

$$\underset{w, t^+, t^- \in \mathbb{R}}{\text{minimize}} \quad t^+ + t^- \quad \text{s. t.} \quad w = t^+ - t^-, t^+ \geq 0, t^- \geq 0.$$

This reformulation is a smooth problem involving only linear objective and constraints, which is easily solvable by state-of-the-art solvers.

Using gradients almost always works Another technique, frequently employed in practice, consists in working with functions that are nonsmooth but Lipschitz continuous¹ and using a gradient-based scheme. This approach is motivated by the following property, due to Rademacher.

Theorem 1.1.1 (Rademacher's theorem) *Let $f : \mathbb{R}^d \rightarrow \mathbb{R}$ be a Lipschitz continuous function. Then it is differentiable at almost every point in $\mathbb{R}^d$.*

Thanks to this theorem, it is theoretically possible to apply a gradient-type method to an optimization problem with a nonsmooth, Lipschitz continuous objective, as almost every query for a gradient would succeed. However, the points of interest in optimization (local and global minima) tend to be the points of nondifferentiability of a function, which this approach cannot identify.

For instance, the ReLU function is Lipschitz continuous and not differentiable at 0. thus most constructions involving ReLU (such as neural networks) would not be differentiable everywhere. However, most algorithms will operate under the assumption that the function is indeed differentiable. This is the case for most points (in fact, almost every point), but nonsmooth functions are likely to be non-differentiable at their minima, should they possess any.

Example 1.1.2 *Consider a shallow neural network of the form*

$$x \mapsto W_2 \sigma(W_1 x + b_1) + b_2,$$

where $W_1 \in \mathbb{R}^{h \times d}$, $b_1 \in \mathbb{R}^h$, $W_2 \in \mathbb{R}^{m \times h}$, $b_2 \in \mathbb{R}^m$ and σ denotes a ReLU activation applied componentwise, i.e.

$$\sigma(v) = [\max\{[v]_i, 0\}]_{i=1}^h \in \mathbb{R}^h.$$

The ReLU function $w \mapsto \max\{w, 0\}$ is 1-Lipschitz continuous and not differentiable at 0. As a result, any optimization problem based on the shallow neural network model (even with a smooth loss) will be a nonsmooth problem. This property is implicitly accounted for in current librairies such as PyTorch, JAX or Tensorflow by resorting to (numerical calculations of) generalized notions of gradients such as the one described in Chapter 2.

1.2 Regularized problems

A common practice in machine learning problems consists in enforcing a specific structure of the machine learning model **through the objective function** rather than by using constraints on the model parameters. Such regularized problems have the following form :

$$\underset{x \in \mathbb{R}^d}{\text{minimize}} \quad \underbrace{f(x)}_{\text{data-fitting term}} + \underbrace{\lambda \Omega(x)}_{\text{regularization term}}. \quad (1.2.1)$$

¹Recall that $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is L -Lipschitz continuous if $|f(w) - f(v)| \leq L\|w - v\|$ for any $(v, w) \in (\mathbb{R}^d)^2$.

where $\lambda > 0$ is called a regularization parameter. This parameter controls the weight of the regularization term compared to the data-fitting term. It is common for one of the terms in (1.2.1) (and sometimes both) to be nonsmooth. Such problems are called composite optimization problems.

When $\lambda \rightarrow 0$, the problem becomes essentially equivalent to minimizing the data-fitting term $f(\mathbf{x})$ only, and regularization no longer matters (numerically, this occurs for small values of λ). When $\lambda \rightarrow \infty$, the problem amounts to minimizing the regularization term, without accounting for the data-dependent term. A key issue in regularization consists in finding the right balance between data fitting and regularization.

Example 1.2.1 *The following functions are common choices for regularization terms:*

- ℓ_2 /ridge regularization: $\Omega(\mathbf{x}) = \frac{1}{2}\|\mathbf{x}\|_2^2$;
- ℓ_1 /LASSO regularization: $\Omega(\mathbf{x}) = \|\mathbf{x}\|_1$;
- Elastic net: $\Omega(\mathbf{x}) = \frac{\gamma}{2}\|\mathbf{x}\|_2^2 + \|\mathbf{x}\|_1$ for $\gamma > 0$;
- Group LASSO: Given a partition $\mathcal{G}$ of $\{1, \dots, d\}$, set $\Omega(\mathbf{x}) = \sum_{g \in \mathcal{G}} \|[x]_g\|_2$, where $[x]_g \in \mathbb{R}^{|g|}$ is the vector formed by the subset of coordinates of $\mathbf{x}$ corresponding to g .

At a broader level, regularization terms can be used to promote generalization capabilities of the solution, by reducing the sensitivity of the solution with respect to the problem data. They are also used to enforce sparsity in the problem solution. Finally, they can be used to improve the optimization landscape of a problem, for instance by guaranteeing that the regularized problem has a unique solution.

Constrained optimization and regularization The regularization approach bears a connection with the introduction of constraints on the problem variables. In fact, any set of constraints $\mathbf{x} \in \mathcal{X}$ can be encoded as a regularization function using the so-called indicator function

$$\Omega(\mathbf{x}) = \begin{cases} 0 & \text{if } \mathbf{x} \in \mathcal{X} \\ +\infty & \text{otherwise.} \end{cases}$$

Conversely, it is possible to describe the solution of a regularized problem as that of a constrained optimization problem. However, the philosophy behind regularization is to penalize points that do not satisfy a certain property instead of excluding those points like in a constrained formulation. The interest of regularization lies in the possibility of weighing the regularization term relatively to the data-fitting term.

Chapter 2

Subgradient and proximal gradient methods

In this chapter, we review the main algorithmic approaches that can be used to tackle problems involving nonsmooth functions as well as regularized optimization problems.

2.1 Subgradient methods

2.1.1 Subgradients and subdifferentials

In smooth optimization, the gradient is a vector that quantifies the rate of local change in the function. Generalized notions of gradient have been defined, especially in the case of convex, nonsmooth functions. We provide below the most common definition of a subgradient.

Definition 2.1.1 (Subgradient and subdifferential) Let $f : \mathbb{R}^d \rightarrow \mathbb{R}$ be a convex function. A vector $g \in \mathbb{R}^d$ is called a *subgradient* of f at $x \in \mathbb{R}^d$ if

$$\forall z \in \mathbb{R}^n, \quad f(z) \geq f(x) + g^T(z - x).$$

The set of all subgradients of f at x is called the *subdifferential* of f at x , and denoted by $\partial f(x)$.

Note that when the function f is differentiable at x , we have $\partial f(x) = \{\nabla f(x)\}$, thus the notion of subdifferential matches that of the gradient for differentiable functions. In addition, the subdifferential provides a characterization of the global minima of f , as shown by the following result.

Theorem 2.1.1 Let $f : \mathbb{R}^d \rightarrow \mathbb{R}$ be a convex function, and $\bar{x} \in \mathbb{R}^d$.

$$0 \in \partial f(\bar{x}) \quad \Leftrightarrow \quad \bar{x} \in \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} f(x).$$

Again, the result of Theorem 2.1.1 generalizes that of the smooth setting, since in that case the subdifferential consists in a single element.

Example 2.1.1 Let $f : \mathbb{R} \rightarrow \mathbb{R}$, $f(x) = |x|$.

$$\partial f(x) = \begin{cases} -1 & \text{if } x < 0 \\ 1 & \text{if } x > 0 \\ [-1, 1] & \text{if } x = 0. \end{cases}$$

The set $[-1, 1]$ contains 0, which confirms that $x^* = 0$ is the minimum of f .

Remark 2.1.1 Subgradients can also be defined for nonconvex functions, however in that case the subdifferential may be empty. This would typically be the case if the function happens not to be differentiable at a local maximum.

2.1.2 Subgradient algorithm

By analogy with gradient descent, we can design a subgradient method, as shown by Algorithm 1.

Algorithm 1: Generic subgradient method.

Initialization: $x_0 \in \mathbb{R}^d$.
for $k = 0, 1, \dots$ **do**
 1. Compute a subgradient $g_k \in \partial f(x_k)$.
 2. Compute a steplength $\alpha_k > 0$.
 3. Set $x_{k+1} = x_k - \alpha_k g_k$.
end

Such a method offers a flexibility in choosing the subgradient, which can be an issue. Moreover, choosing the stepsize is more difficult than for gradient descent, due to the nonsmooth nature of the problem. In fact, a subgradient can lead to increase in the function value for any stepsize, hence the choice of subgradient is critical to the success of this method. A convenient choice consists in setting g_k such that

$$g_k \in \operatorname{argmin}_{g \in \mathbb{R}^d} \{ \|g\| \mid g \in \partial f(x_k) \}. \quad (2.1.1)$$

With this choice, Algorithm 1 will not change the iterate once it reaches a point with $\mathbf{0}_{\mathbb{R}^d}$ in its subdifferential (i.e. a minimum for a convex function). Computing g_k according to (2.1.1) may however be an expensive procedure as it could require to compute the entire subdifferential.

As for the stepsize, it can be chosen constant or decreasing, similarly to the choices made in the case of stochastic gradient. Interestingly, theoretical guarantees can be obtained on the (auxiliary) sequence of averaged iterates

$$\bar{x}_k = \frac{1}{\sum_{j=0}^k \alpha_j} \sum_{j=0}^k \alpha_j x_j,$$

that has a smoother behavior than the original sequence. Similarly to the idea of iterate averaging for stochastic gradient methods, studying this sequence (in the presence of convexity) allows for theoretical convergence results.

Remark 2.1.2 In a finite-sum setting, stochastic subgradient estimates can be used in lieu of subgradients. Although the analysis of such methods becomes more intricate, these schemes are implicitly used in deep learning libraries when optimizing functions depending on nonsmooth losses or activation functions.

2.2 Proximal methods

Proximal methods are not restricted to nonsmooth optimization problems, but have proven particularly useful in the presence of nonsmoothness. Those techniques rely on the proximal operator, a mathematical object that is both defined as the solution to an optimization problem and used as a step in other optimization algorithms.

2.2.1 Proximal point method and proximal operator

In this section, we consider

$$\underset{x \in \mathbb{R}^d}{\text{minimize}} h(x) \quad (2.2.1)$$

where $h : \mathbb{R}^d \rightarrow \mathbb{R}$ is a convex function. Without assumptions on the differentiability of h , we cannot rely on a gradient-type method for optimizing this problem. Rather than using subgradients as in the previous section, we will instead compute a sequence of approximate solutions of problem (2.2.1).

Algorithm 2: Proximal point method.

Initialization: $x_0 \in \mathbb{R}^d$.

for $k = 0, 1, \dots$ **do**

1. Compute a steplength $\alpha_k > 0$.
2. Compute x_{k+1} such that

$$x_{k+1} \in \underset{z \in \mathbb{R}^d}{\operatorname{argmin}} \left\{ h(z) + \frac{1}{2\alpha_k} \|z - x_k\|_2^2 \right\}. \quad (2.2.2)$$

end

The process is given in Algorithm 2. At every iteration, the next iterate x_{k+1} is computed by solving

$$\underset{z \in \mathbb{R}^d}{\text{minimize}} \left\{ h(z) + \frac{1}{2\alpha_k} \|z - x_k\|_2^2 \right\}. \quad (2.2.3)$$

At first glance, this problem is not easier than the original one, because it involves the original objective h . However, the presence of the **proximal term** $\frac{1}{2\alpha_k} \|z - x_k\|_2^2$ actually makes the problem easier to solve. In particular, the subproblem (2.2.3) is strongly convex (and thus the update (2.2.2) is uniquely defined), and solving this subproblem always improves the objective.

Lemma 2.2.1 *At the k th iteration of Algorithm 2, we have*

$$h(x_{k+1}) \leq h(x_k) - \frac{1}{2\alpha_k} \|x_{k+1} - x_k\|_2^2. \quad (2.2.4)$$

Definition 2.2.1 *Let $\varphi : \mathbb{R}^d \rightarrow \mathbb{R}$ and $x \in \mathbb{R}^d$. The **proximal operator of φ at x** , denoted by $\operatorname{prox}[\varphi][x]$, is given by*

$$\operatorname{prox}_\varphi[x] := \underset{u \in \mathbb{R}^d}{\operatorname{argmin}} \left\{ \varphi(u) + \frac{1}{2} \|u - x\|^2 \right\}. \quad (2.2.5)$$

As stated, the proximal operator is a set-valued mapping, i.e. its value is a set of vectors (possibly empty or infinite!). However, for certain classes of functions, the value of the proximal operator is a singleton. It is then convenient to identify it with the corresponding vector. We make this identification below in the case of convex functions, which will be our focus throughout these notes.

Definition 2.2.2 Let $\varphi : \mathbb{R}^d \rightarrow \mathbb{R}$ be a convex function. Then the proximal operator of φ is defined as the function $\text{prox}_\varphi[\cdot] : \mathbb{R}^d \rightarrow \mathbb{R}^d$ such that

$$\forall \mathbf{x} \in \mathbb{R}^d, \text{prox}_\varphi[\mathbf{x}] := \mathbf{u}^*, \{\mathbf{u}^*\} = \underset{\mathbf{u} \in \mathbb{R}^d}{\text{argmin}} \left\{ \varphi(\mathbf{u}) + \frac{1}{2} \|\mathbf{u} - \mathbf{x}\|^2 \right\}. \quad (2.2.6)$$

Using the definition of the proximal operator, note that an iteration of Algorithm 2 can be rewritten as

$$\mathbf{x}_{k+1} = \text{prox}[\alpha_k h] [\mathbf{x}_k].$$

2.2.2 Proximal gradient methods

Following our introduction of regularized problems in Section 1.2, we now turn to a certain class of regularized called **composite optimization** problems.

Definition 2.2.3 (Composite optimization) A composite optimization problem is of the form:

$$\underset{\mathbf{x} \in \mathbb{R}^d}{\text{minimize}} f(\mathbf{x}) + \lambda \Omega(\mathbf{x}), \quad (2.2.7)$$

where $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is a smooth, C^1 function, $\lambda > 0$ and $\Omega : \mathbb{R}^d \rightarrow \mathbb{R}$ is a convex, nonsmooth regularizer.

Because of the nonsmoothness of Ω , the overall objective of problem (2.2.7) is nonsmooth, and one may then consider applying a subgradient method to tackle this problem. However, knowledge about the particular problem structure, and the fact that f is a smooth function, allows for deriving better methods for this problem, that do not require direct calculations of subgradients.

Proximal gradient techniques, that are described in Algorithm 3, exploit the smoothness of f to iteratively solve a sequence of subproblems approximating the original one.

The core of an iteration of Algorithm 3 includes a gradient calculation as well as solving an auxiliary optimization problem (2.2.8), called the **proximal subproblem**. Such a method is only practical if the subproblems are easier to solve than the original problem.

Connection with the proximal operator If $\Omega \equiv 0$ (i. e. Ω is the zero function and the problem is un-regularized), one can show that the solution of (2.2.8) is given by

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k).$$

We thus recognize the gradient descent iteration. For arbitrary Ω , the following formula applies:

$$\mathbf{x}_{k+1} = \text{prox}_{\alpha_k \lambda \Omega} [\mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k)].$$

As such, proximal gradient bears a close connection with the proximal operator.

Algorithm 3: Proximal gradient method.**Initialization:** $\mathbf{x}_0 \in \mathbb{R}^d$.**for** $k = 0, 1, \dots$ **do**

1. Compute the gradient of the smooth part $\nabla f(\mathbf{x}_k)$.
2. Compute a steplength $\alpha_k > 0$.
3. Compute $\mathbf{x}_{k+1}$ such that

$$\mathbf{x}_{k+1} \in \operatorname{argmin}_{\mathbf{x} \in \mathbb{R}^d} \left\{ f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^\top (\mathbf{x} - \mathbf{x}_k) + \frac{1}{2\alpha_k} \|\mathbf{x} - \mathbf{x}_k\|_2^2 + \lambda \Omega(\mathbf{x}) \right\}. \quad (2.2.8)$$

end

However, note that applying the proximal point to problem (2.2.7) yields a different iteration, even in the case $\Omega \equiv 0$! Indeed, in that case, and assuming convexity of f , the proximal point iteration can be written as

$$\mathbf{x}_{k+1} = \operatorname{prox}_{\alpha_k f} [\mathbf{x}_k] = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_{k+1}).$$

The proximal point method is thus an implicit method, in the sense that its iteration is defined implicitly by the above equation to be solved for $\mathbf{x}_{k+1}$.

Remark 2.2.1 *Proximal gradient methods can be designed using most of the tools that can be applied to gradient descent : this includes stepsize choices, acceleration as well as stochastic aspects. Moreover, theoretical guarantees have been established for both nonconvex and convex f , though the latter has attracted more attention in the optimization literature. Note that the analysis of such methods is performed by looking at the vector*

$$\frac{1}{\alpha_k} (\mathbf{x}_k - \operatorname{prox}_{\alpha_k \lambda \Omega} [\mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k)]),$$

that plays a similar role than the gradient in analyzing gradient descent (and reduces to the gradient when $\Omega \equiv 0$).

2.3 Conclusion of Chapter 2

Optimization of convex nonsmooth functions can be performed thanks to subgradients, that also characterize solutions of the problem. An alternative to using subgradients consists in applying the proximal operator of the function in order to smooth out the problem. The proximal point method is thus another possibility for tackling nonsmooth optimization.

Many nonsmooth optimization problems that arise in machine learning exhibit a composite optimization structure, wherein the objective function is expressed as the sum of a smooth and a convex, nonsmooth term. In that setting, the proximal gradient algorithm is a generic technique that combines gradient steps over the smooth part of the objective with an application of the proximal operator.

Chapter 3

Regularized optimization

In this chapter, we investigate examples of regularized optimization problems. We study both the effect of regularization and the result of proximal gradient when applied to such problems.

3.1 The case of ℓ_2 regularization

The first key example we consider is the use of the Euclidean, or ℓ_2 norm, as a regularizer. A **regularized problem with ℓ_2 regularization**, also known as **ridge regularized problem**, has the form

$$\underset{\mathbf{x} \in \mathbb{R}^d}{\text{minimize}} \quad f(\mathbf{x}) + \frac{\lambda}{2} \|\mathbf{x}\|_2^2, \quad (3.1.1)$$

where f is a continuously differentiable function.

The ridge regularizer $\mathbf{x} \mapsto \frac{\lambda}{2} \|\mathbf{x}\|_2^2$ has several interpretations. It effectively penalizes $\mathbf{x}$ s with large components, and can be shown to be equivalent to a constraint on the squared norm $\|\mathbf{w}\|_2^2$. In addition, a ridge regularizer has the effect to reduce the variance of the problem solution with respect to the data. Finally, when the regularizer $\lambda > 0$ is big enough, this often turns the objective function into a strongly convex one, with the positive implications in terms of convergence speed and uniqueness of the (global) minimum.

3.1.1 Proximal gradient and ℓ_2 regularization

Consider applying the proximal gradient framework of Algorithm 3 to problem (3.1.1). The proximal subproblem to be solved at the k th iteration is then

$$\underset{\mathbf{x} \in \mathbb{R}^d}{\text{minimize}} \quad \left\{ f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^T (\mathbf{x} - \mathbf{x}_k) + \frac{1}{2\alpha_k} \|\mathbf{x} - \mathbf{x}_k\|_2^2 + \frac{\lambda}{2} \|\mathbf{x}\|_2^2 \right\},$$

which is a (strongly convex) quadratic optimization problem. Its solution (which we denote by $\mathbf{x}_{k+1}$ for convenience) is

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{\alpha_k}{1 + \lambda\alpha_k} \nabla f(\mathbf{x}_k)$$

On this problem, proximal gradient thus corresponds to gradient descent with a different stepsize. When $\lambda \rightarrow 0$, we recover the standard gradient descent. As λ gets larger, however, the stepsize becomes smaller, and therefore the step becomes less influenced by the value of the gradient. In

that sense, the iterates of proximal gradient will be less sensitive to changes in f than the iterates of gradient descent applied to the problem without regularization.

Gradient descent on ℓ_2 -regularized problems Applying gradient descent to the regularized problem (3.1.1) yields the following iteration

$$\mathbf{x}_{k+1} = (1 - \lambda\alpha_k)\mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k),$$

sometimes referred to as weight decay in a machine learning setting, due to the fact that the components of the previous iterate are shrunk (when $\lambda\alpha_k$ is sufficiently small), then combined with a gradient step. It can be shown that this approach leads to a solution with reduced variance with respect to f .

3.1.2 Ridge linear regression

The most common instance of ℓ_2 -regularized problem is a linear least-squares loss is an important class of problems. Such problems have the form

$$\underset{\mathbf{x} \in \mathbb{R}^d}{\text{minimize}} \frac{1}{2n} \|\mathbf{A}\mathbf{x} - \mathbf{y}\|^2 + \frac{\lambda}{2} \|\mathbf{x}\|_2^2, \quad (3.1.2)$$

where $\mathbf{A} \in \mathbb{R}^{n \times d}$ and $\mathbf{y} \in \mathbb{R}^n$ form the problem data, and $\lambda > 0$.

The regularization parameter does not change the nature of the problem, in that it remains a linear least-squares problem. However, problem (3.1.2) is necessarily strongly convex, whereas the un-regularized problem $\underset{\mathbf{x} \in \mathbb{R}^d}{\text{minimize}} \frac{1}{2} \|\mathbf{A}\mathbf{x} - \mathbf{y}\|^2$ may not be. As mentioned above, bringing strong convexity is a motivation for using ℓ_2 regularization.

Theorem 3.1.1 Suppose that $d \leq n$. Then,

- a) If $\mathbf{A}^\top \mathbf{A}$ is not invertible, the un-regularized problem has infinitely many solutions.
- b) If $\mathbf{A}^\top \mathbf{A}$ is invertible, there exists a unique solution to the un-regularized problem given by

$$\hat{\mathbf{x}} = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{y}.$$

- c) For any $\lambda > 0$, problem (3.1.2) always has a single solution given by

$$\mathbf{x}^*(\lambda) = (\mathbf{A}^\top \mathbf{A} + \lambda n \mathbf{I}_d)^{-1} \mathbf{A}^\top \mathbf{y}$$

Note that the result for the unregularized problem is actually independent of the assumption on n and d .

Looking at the formulas for $\hat{\mathbf{x}}$ and $\mathbf{x}^*(\lambda)$, we see that $\mathbf{x}^*(\lambda) \rightarrow \hat{\mathbf{x}}$ as $\lambda \rightarrow 0$ and $\mathbf{x}^*(\lambda) \rightarrow \mathbf{0}_{\mathbb{R}^d}$ as $\lambda \rightarrow \infty$. As λ increases, the importance of $\mathbf{A}$ and $\mathbf{y}$ in the definition of the solution is gradually reduced. In that sense, the solution is less sensitive to changes in the data. Additionally, the regularization leads to a solution with coordinates that are of the same magnitude.

Example 3.1.1 Let $n = d = 2$, $A = \begin{bmatrix} 1 & 0 \\ 0 & \varepsilon \end{bmatrix}$ for some $\varepsilon > 0$ and $\mathbf{y} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$. Then, we have

$$\hat{\mathbf{x}} = \begin{bmatrix} 1 \\ \frac{1}{\varepsilon} \end{bmatrix},$$

whose norm is of order $\frac{1}{\varepsilon}$ for small ε . However, for any $\lambda > 0$, we obtain

$$\mathbf{x}^*(\lambda) = \begin{bmatrix} \frac{1}{1+2\lambda} \\ \frac{1}{\varepsilon+2\lambda} \end{bmatrix},$$

whose norm is $\mathcal{O}(\frac{1}{\lambda})$ for λ sufficiently large, regardless of the value of ε . In that sense, regularizing leads to a more stable solution.

3.2 The case of ℓ_1 regularization

3.2.1 Motivation : Sparse models

While computing a model to explain some data, we might want to compute a model that explains the data using as few features as possible¹. Mathematically speaking, if our model is parameterized by a vector $\mathbf{x} \in \mathbb{R}^d$, our goal is to compute a vector that explains the data with as few nonzero coordinates as possible.

There exists a regularizer that penalized vectors with nonzero components (not just large as opposed to the ridge regularizer), called the ℓ_0 norm². An ℓ_0 -regularized problem has the form

$$\underset{\mathbf{x} \in \mathbb{R}^d}{\text{minimize}} f(\mathbf{x}) + \lambda \|\mathbf{x}\|_0, \quad \|\mathbf{v}\|_0 = |\{i \mid [\mathbf{v}]_i \neq 0\}|.$$

However, this function is nonsmooth and discontinuous; its combinatorial nature also introduces more complexity to the original problem. As a result, researchers have turned to an intermediate regularization term, the ℓ_1 norm defined by

$$\|\mathbf{x}\|_1 = \sum_{i=1}^d |[\mathbf{x}]_i|. \quad (3.2.1)$$

This function is continuous and convex; moreover, it is a norm function, which endows it with many desirable properties. As a result, the class of problems of the form

$$\underset{\mathbf{x} \in \mathbb{R}^d}{\text{minimize}} f(\mathbf{x}) + \lambda \|\mathbf{x}\|_1, \quad (3.2.2)$$

called ℓ_1 regularized problems or **lasso regularized problems**, has emerged as a tractable alternative to the ℓ_0 regularized formulation.

¹The goal of this process is *feature selection*.

²Though technically this function defines a semi-norm.

3.2.2 Proximal gradient and ℓ_1 regularization

A natural way to tackle problems of the form 3.2.2 is through the proximal gradient framework of Algorithm 3. Indeed, unlike for general regularizers, one can obtain a closed-form solution of the subproblem (2.2.8). Indeed, the proximal subproblem, given by

$$\underset{\mathbf{x} \in \mathbb{R}^d}{\text{minimize}} \left\{ f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^T (\mathbf{x} - \mathbf{x}_k) + \frac{1}{2\alpha_k} \|\mathbf{x} - \mathbf{x}_k\|_2^2 + \lambda \|\mathbf{x}\|_1 \right\},$$

has a unique solution. To obtain it, one computes the usual gradient step $\mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k)$, then one applies the **soft-thresholding function** $s_{\alpha_k \lambda}(\bullet)$ to each component, where this function is given by

$$\forall \mu > 0, \forall t \in \mathbb{R}, \quad s_\mu(t) = \begin{cases} t + \mu & \text{if } t < -\mu \\ t - \mu & \text{if } t > \mu \\ 0 & \text{otherwise.} \end{cases}$$

(Note that this function encodes the proximal operator for the ℓ_1 norm.) As a result, the solution of the proximal subproblem is defined component-wise according to the components of the gradient step. The resulting update is at the heart of the corresponding proximal algorithm, popularized in signal and image processing under the name ISTA (Iterative Soft-Thresholding Algorithm). A description of ISTA is given in Algorithm 4.

Algorithm 4: ISTA: Iterative Soft-Thresholding Algorithm.

Initialization: $\mathbf{x}_0 \in \mathbb{R}^d$.

for $k = 0, 1, \dots$ **do**

1. Compute the gradient of the smooth par $\nabla f(\mathbf{x}_k)$.
2. Compute a steplength $\alpha_k > 0$.
3. Compute $\mathbf{x}_{k+1}$ component-wise through the following rule

$$[\mathbf{x}_{k+1}]_i = \begin{cases} [\mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k)]_i + \alpha_k \lambda & \text{if } [\mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k)]_i < -\alpha_k \lambda \\ [\mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k)]_i - \alpha_k \lambda & \text{if } [\mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k)]_i > \alpha_k \lambda \\ 0 & \text{if } [\mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k)]_i \in [-\alpha_k \lambda, \alpha_k \lambda]. \end{cases} \quad (3.2.3)$$

end

It can be shown that the use of the soft-thresholding function does promote zero components in the new iterates, which results in sparser solutions at the end of the algorithmic run.

Remark 3.2.1 A notable improvement on ISTA was the inclusion of momentum, which resulted in a new algorithm called FISTA (Fast ISTA): this method is now the most widely used instance of ISTA.

3.2.3 The LASSO estimator

As for ℓ_2 regularization, using ℓ_1 regularization together with a linear least-squares loss is an important class of problems. Such problems have the form

$$\underset{\mathbf{x} \in \mathbb{R}^d}{\text{minimize}} \frac{1}{2} \|\mathbf{Ax} - \mathbf{y}\|^2 + \lambda \|\mathbf{x}\|_1, \quad (3.2.4)$$

where $\mathbf{A} \in \mathbb{R}^{n \times d}$ and $\mathbf{y} \in \mathbb{R}^n$ form the problem data, and $\lambda > 0$. This problem is often referred to as the LASSO (Least Absolute Shrinkage and Selection Operator) problem, though other terminologies such as basis pursuit can be found in the literature.

The properties of the solution of problem (3.2.4) can be described according to the columns of $\mathbf{A}$. Every column contains all the observed values for a specific feature captured by the linear model.

Theorem 3.2.1 *Let $\mathbf{b}_1, \dots, \mathbf{b}_d$ denote the columns of $\mathbf{A}$, and let $\mathbf{x}^*$ be a solution of problem (3.2.4). Then, for any $j = 1, \dots, d$, we have*

$$\begin{cases} |\mathbf{b}_j^T(\mathbf{Ax}^* - \mathbf{y})| \leq \lambda n & \text{if } [\mathbf{x}^*]_j = 0 \\ \mathbf{b}_j^T(\mathbf{Ax}^* - \mathbf{y}) = -\lambda n \text{sgn}([\mathbf{x}^*]_j) & \text{otherwise,} \end{cases} \quad (3.2.5)$$

where $\text{sgn}(t) = 1$ if $t > 0$ and $\text{sgn}(t) = -1$ if $t < 0$.

The result of Theorem 3.2.1 implies that the larger λ is, the more components of $\mathbf{x}^*$ will be zero. In that sense, the use of ℓ_1 regularization leads to sparser solutions than in absence of regularization.

3.3 Conclusion of Chapter 3

The most classical regularization techniques in optimization and beyond are ℓ_2 and ℓ_1 regularization. In a data-related setting, the former is typically used to reduce the dependency of the solution with respect to the data, while the latter promotes solutions that have zero coordinates.

In both cases, the proximal gradient method can then be written in closed form, leading to algorithms that are implementable quite efficiently. In particular, the case of regularized linear least squares has been widely studied, and the resulting problems are now part of numerous statistics and signal processing toolboxes.

Chapter 4

Exercises

Note: See also the questions in the lab session:

<https://www.lamsade.dauphine.fr/~croyer/ensdocs/PGL/SourcesLabPGL.zip>

Exercise 1: Stratified model

Consider a dataset divided in two groups:

$$\mathbf{A}_1 \in \mathbb{R}^{n_1 \times d}, \mathbf{y}_1 \in \mathbb{R}^{n_1}, \quad \mathbf{A}_2 \in \mathbb{R}^{n_2 \times d}, \mathbf{y}_2 \in \mathbb{R}^{n_2}.$$

Such a division is typically the result of a striking difference between the examples (for instance, medical data for two age categories).

For each group, we seek a linear model that best fits the data, i.e. a vector $\mathbf{x}_1 \in \mathbb{R}^d$ such that $\mathbf{A}_1 \mathbf{x}_1 \approx \mathbf{y}_1$ and a vector $\mathbf{x}_2 \in \mathbb{R}^d$ such that $\mathbf{A}_2 \mathbf{x}_2 \approx \mathbf{y}_2$. Because of the similar nature of these data samples, we would like the models to be as close as possible, i.e. $\mathbf{x}_1 \approx \mathbf{x}_2$. All these modeling assumptions lead to the following problem:

$$\underset{\substack{\mathbf{x}_1 \in \mathbb{R}^d \\ \mathbf{x}_2 \in \mathbb{R}^d}}{\text{minimize}} \quad \frac{1}{2} \|\mathbf{A}_1 \mathbf{x}_1 - \mathbf{y}_1\|_2^2 + \frac{1}{2} \|\mathbf{A}_2 \mathbf{x}_2 - \mathbf{y}_2\|_2^2 + \frac{\lambda}{2} \|\mathbf{x}_1 - \mathbf{x}_2\|_2^2, \quad (4.0.1)$$

where $\lambda \geq 0$.

- a) Letting $\mathbf{x} := \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} \in \mathbb{R}^{2d}$, we can rewrite the objective function of problem (4.0.1) under the form:

$$\varphi(\mathbf{x}) := \frac{1}{2} \mathbf{x}^T \begin{bmatrix} \mathbf{A}_1^T \mathbf{A}_1 + \lambda \mathbf{I} & -\lambda \mathbf{I} \\ -\lambda \mathbf{I} & \mathbf{A}_2^T \mathbf{A}_2 + \lambda \mathbf{I} \end{bmatrix} \mathbf{x} + \begin{bmatrix} \mathbf{A}_1^T \mathbf{y}_1 \\ \mathbf{A}_2^T \mathbf{y}_2 \end{bmatrix}^T \mathbf{x} + \frac{1}{2} (\mathbf{y}_1^T \mathbf{y}_1 + \mathbf{y}_2^T \mathbf{y}_2),$$

where $\mathbf{I}_d \in \mathbb{R}^{d \times d}$ the identity matrix. What classical category of optimization problem does problem (4.0.1) belong to?

- b) Give the first-order optimality conditions for problem (4.0.1).
c) Consider the case $\lambda = 0$, where $\mathbf{x}_1$ and $\mathbf{x}_2$ are learned separately
i) Justify that the resulting problem is convex.
ii) Give a close form expression for a solution of problem (4.0.1) in that case.