Dauphine | PSL
UNIVERSITÉ PARIS

Lecture notes on advanced gradient descent

Clément W. Royer

M2 IASD & M2 MASH - 2021/2022

- The last version of these notes can be found at:
  https://www.lamsade.dauphine.fr/~croyer/ensdocs/GD/LectureNotesOML-GD.pdf.
- Comments, typos, etc, can be sent to `clement.royer@dauphine.psl.eu`. *Thanks to the students who reached out.*
- **Major updates of the document**
  - 2021.12.31: Fixed numerical values in the solution of Exercise 3.
  - 2021.12.15: Added solutions of the exercises and fixed typos in Exercise 4.
  - 2021.11.02: Fixed typo in Theorem 1.1.8 and changed the statement of Exercise 1.
  - 2021.10.17: Revised the material, added Chapter 3 (nonconvex optimization lecture) and exercises.
  - 2021.10.15: Recording link for the second session.
  - 2021.10.13: Edited the recording link.
  - 2021.10.12: First version of Chapter 2 along with links to the recording. Added links and suggestion for background reading.
  - 2021.10.11: First version with introductory chapter.
- **Learning goals:**
  - Know the convergence rates and complexity bounds for gradient descent;
  - Understand the concept of momentum/acceleration under its various forms.
  - Identify the difficulties in dealing with nonconvex formulations, and what can be guaranteed for gradient-type methods then.

Foreword

The purpose of these lecture notes is to provide a modern view of gradient-type methods for smooth optimization. Like any presentation of such a large topic, these notes have their own biases. Those include that of the author as well as the reference they are mainly based upon [5]. Still, these lectures aim at achieving the following goals:

- Provide the main tools for deriving complexity results for gradient descent in the strongly convex, convex and nonconvex cases;
- Introduce the acceleration phenomenon and the concept of momentum in convex optimization, as well as the associated methods;
- Describe the difficulties of dealing with a nonconvex landscape in general, and discuss which nonconvex formulations from data science have better properties;
- Highlight recent results that support practice in running gradient descent on nonconvex problems.

Background reading material If some of the notations below (or some of the concepts introduced in the next chapter) are not familiar to the reader, it is recommended to look into additional material in these notions. In particular, the appendices of Sebastian Raschka's textbook are an excellent summary of mathematical topics in data science, among which differential calculus. The textbook of Boyd and Vandenberghe [1], freely available online, is a great reference for linear algebra (in particular, chapters 1 to 3 cover the very basics).

Notations

- Scalars (i.e. reals) are denoted by lowercase letters: $a, b, c, \alpha, \beta, \gamma$.
- Vectors are denoted by **bold** lowercase letters: $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}$.
- Matrices are denoted by **bold** uppercase letters: $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$.
- 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 $\boldsymbol{x} \in \mathbb{R}^d$ 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 $\boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix}$, or, in a compact form, $\boldsymbol{x} = [x_i]_{1 \leq 1 \leq d}$.
- Given a column vector $\boldsymbol{x} \in \mathbb{R}^d$, the corresponding row vector is denoted by $\boldsymbol{x}^{\mathrm{T}}$, so that $\boldsymbol{x}^{\mathrm{T}} = [x_1 \ \cdots \ x_d]$ and $[\boldsymbol{x}^{\mathrm{T}}]^{\mathrm{T}} = \boldsymbol{x}$. The scalar product between two vectors in $\mathbb{R}^n$ is defined as $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{y} = \boldsymbol{y}^{\mathrm{T}}\boldsymbol{x} = \sum_{i=1}^{d} x_i y_i$.
- The Euclidean norm of a vector $\boldsymbol{x} \in \mathbb{R}^d$ is defined by $\|\boldsymbol{x}\| = \sqrt{\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}}$. Note that for any vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ in $\mathbb{R}^d$, we have $|\boldsymbol{x}^{\mathrm{T}}\boldsymbol{y}| \leq \|\boldsymbol{x}\|\|\boldsymbol{y}\|$.
- 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 $\boldsymbol{A} \in \mathbb{R}^{n \times d}$, $\boldsymbol{A}_{ij}$ refers to the coefficient from the i-th row and the j-th column of $\boldsymbol{A}$: the diagonal of $\boldsymbol{A}$ is given by the coefficients $\boldsymbol{A}_{ii}$. Provided this notation is not ambiguous, we use the notations $\boldsymbol{A}$, $[\boldsymbol{A}_{ij}]_{\substack{1 \leq i \leq n \\ 1 \leq j \leq d}}$ and $[\boldsymbol{A}_{ij}]$ 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).
- For a matrix $\boldsymbol{A} \in \mathbb{R}^{d \times d}$, we define the spectral norm of $\boldsymbol{A}$ as $\|\boldsymbol{A}\| := \max_{\boldsymbol{v} \neq \boldsymbol{0}} \frac{\|\boldsymbol{A}\boldsymbol{v}\|}{\|\boldsymbol{v}\|}$. It follows that for any $\boldsymbol{v} \in \mathbb{R}^n$, we have $\|\boldsymbol{A}\boldsymbol{v}\| \leq \|\boldsymbol{A}\|\|\boldsymbol{v}\|$. When $\boldsymbol{A}$ is a symmetric matrix, $\|\boldsymbol{A}\|$ is equal to the largest absolute value of the eigenvalues of $\boldsymbol{A}$.

Chapter 1

Introduction

In this introductory chapter, we recall the key concepts that will be used throughout the rest of these notes. For a more detailed treatment of these aspects, the reader is encouraged to dive into the references given at the end of these notes.

1.1 About optimization problems

Optimization is the field of applied mathematics concerned with making the best decision out of a set of alternatives. In these notes, we are interested in **continuous** optimization problems, for which our problem variables (that characterize our decision) have a continuum of possible values, and the objective function (measuring how good a decision is) is continuous. Continuous optimization relies heavily on analysis and differential calculus to analyze problems and develop algorithms that provably reach a solution.

1.1.1 Mathematical background

Differential calculus We will mostly consider minimization problems involving a smooth objective function: the term "smooth" can be loosely defined in the optimization or learning literature, but generally means that the function is as regular as needed for the desired algorithms and analysis to be applicable. In these notes, we will consider that a smooth function is at least continuously differentiable, sometimes twice continuously differentiable. Those concepts are recalled below.

Definition 1.1.1 (Continuous function) *A function $f : \mathbb{R}^d \to \mathbb{R}^m$ is continuous at $\boldsymbol{w} \in \mathbb{R}^d$ if for every $\epsilon > 0$, it exists $\delta > 0$ such that*

$$\forall \boldsymbol{v} \in \mathbb{R}^d,\ \|\boldsymbol{v} - \boldsymbol{w}\| \leq \delta \implies \|f(\boldsymbol{v}) - f(\boldsymbol{w})\| \leq \epsilon.$$

Definition 1.1.2 (Lipschitz continuous function) *A function $f : \mathbb{R}^d \to \mathbb{R}^m$ is L-Lipschitz continuous over $\mathbb{R}^d$ if*

$$\forall (\boldsymbol{u}, \boldsymbol{v}) \in \left(\mathbb{R}^d\right)^2, \quad \|f(\boldsymbol{u}) - f(\boldsymbol{v})\| \leq L \|\boldsymbol{u} - \boldsymbol{v}\|,$$

where $L > 0$ is called a Lipschitz constant.

Lipschitz continuous functions can be sandwiched between two linear functions, which is particularly useful for optimization purposes. Note that every Lipschitz continuous function is continuous.

Derivatives are ubiquitous in continuous optimization, as they allow to characterize the local behavior of a function. We assume that the reader is familiar with the concept of derivative of a function from $\mathbb{R} \to \mathbb{R}$. A function $f : \mathbb{R}^d \to \mathbb{R}$ is called *differentiable* at $\boldsymbol{w} \in \mathbb{R}^d$ if all its partial derivatives at $\boldsymbol{w}$ exist.

Definition 1.1.3 (Classes of functions)

- *A function $f : \mathbb{R}^d \to \mathbb{R}$ is continuously differentiable if its first-order derivatives along every variable exist and are continuous. The set of continously differentiable functions is denoted by $\mathcal{C}^1(\mathbb{R}^d)$.*
- *A function $f : \mathbb{R}^d \to \mathbb{R}$ is twice continuously differentiable if $f \in \mathcal{C}^1(\mathbb{R}^d)$ and if its second-order derivative along every pair of variable exist and are continuous. The set of twice continously differentiable functions is denoted by $\mathcal{C}^2(\mathbb{R}^d)$.*

Definition 1.1.4 (First-order derivative) *Let $f \in \mathcal{C}^1(\mathbb{R}^d)$ be a continuously differentiable function. For any $\boldsymbol{w} \in \mathbb{R}^d$, the* **gradient of f at w** *is given by*

$$\nabla f(\boldsymbol{w}) := \left[\frac{\partial f}{\partial w_i}(\boldsymbol{w})\right]_{1 \leq i \leq d} \in \mathbb{R}^d.$$

Definition 1.1.5 (Second-order derivative) *Let $f \in \mathcal{C}^2(\mathbb{R}^d)$ be a twice continuously differentiable function. For any $\boldsymbol{w} \in \mathbb{R}^d$, the* **Hessian of f at w** *is given by*

$$\nabla^2 f(\boldsymbol{w}) := \left[\frac{\partial^2 f}{\partial w_i \partial w_j}(\boldsymbol{w})\right]_{1 \leq i,j \leq d} \in \mathbb{R}^{d \times d}.$$

The Hessian matrix is symmetric.

Finally, we define an important class of problems involving a Lipschitz continuity assumption.

Definition 1.1.6 (Smooth functions with Lipschitz derivatives)

- *Given $L > 0$, the set $\mathcal{C}_L^{1,1}(\mathbb{R}^d)$ represents the set of all functions $f : \mathbb{R}^d \to \mathbb{R}$ that belong to $\mathcal{C}^1(\mathbb{R}^d)$ such that ∇f is L-Lipschitz continuous.*
- *Given $L > 0$, the set $\mathcal{C}_L^{2,2}(\mathbb{R}^d)$ represents the set of all functions $f : \mathbb{R}^d \to \mathbb{R}$ that belong to $\mathcal{C}^2(\mathbb{R}^d)$ such that $\nabla^2 f$ is L-Lipschitz continuous.*

An important property of such functions is that one can derive upper approximations on their values, as shown by the following theorem.

Theorem 1.1.1 (First-order Taylor expansion) *Let $f \in C_L^{1,1}(\mathbb{R}^d)$ with $L > 0$. For any vectors $\boldsymbol{w}, \boldsymbol{z} \in \mathbb{R}^d$, one has:*

$$f(\boldsymbol{z}) \leq f(\boldsymbol{w}) + \nabla f(\boldsymbol{w})^{\mathrm{T}}(\boldsymbol{z} - \boldsymbol{w}) + \frac{L}{2}\|\boldsymbol{z} - \boldsymbol{w}\|^2. \tag{1.1.1}$$

This expansion is crucial in analyzing the performance of first-order algorithms, as we will do in the rest of these notes.

1.1.2 Solutions and optimality conditions

In the rest of this section, we will focus on the problem of minimizing a function $f : \mathbb{R}^d \to \mathbb{R}$ without constraints on the variables, which we represent as

$$\underset{\boldsymbol{w}\in\mathbb{R}^d}{\text{minimize}}\, f(\boldsymbol{w}). \tag{1.1.2}$$

The minimal value of problem (1.1.5) will be denoted by

$$\min_{\boldsymbol{w}\in\mathbb{R}^d} f(\boldsymbol{w}) \in \mathbb{R} \cup \{-\infty\}, \tag{1.1.3}$$

where the value $-\infty$ accounts for the possibility that f may be unbounded below (i.e. for any value $c \in \mathbb{R}$ there always exists a point $\boldsymbol{w}$ such that $f(\boldsymbol{w}) < c$), although this case is not of real interest for these notes. We also define the set of solutions of (1.1.2) by

$$\operatorname*{argmin}_{\boldsymbol{w}\in\mathbb{R}^d} f(\boldsymbol{w}) \subseteq \mathbb{R}^d. \tag{1.1.4}$$

When the function f is unbounded below, this set is empty. The set of solutions of the problem is the set of global minima of the problem, a notion defined below.

Definition 1.1.7 (Global minimum) *Given a function $f : \mathbb{R}^d \to \mathbb{R}$, a point $\boldsymbol{w}^* \in \mathbb{R}^d$ is called a global minimum of f if*

$$\forall \boldsymbol{w} \in \mathbb{R}^d, \qquad f(\boldsymbol{w}^*) \leq f(\boldsymbol{w}).$$

We also introduce a local approximation of the notion of solution: this concept is particularly important in nonconvex optimization.

Definition 1.1.8 (Local minimum) *Given a function $f : \mathbb{R}^d \to \mathbb{R}$, a point $\boldsymbol{w}^* \in \mathbb{R}^d$ is called a local minimum of the problem (1.1.2) if it possesses the lowest value of f within a neighborhood of itself, i.e. if there exists $\delta > 0$ such that*

$$\forall \boldsymbol{w} \in \mathbb{R}^d, \qquad [\|\boldsymbol{w} - \boldsymbol{w}^*\| \leq \delta] \quad \Longrightarrow \quad [f(\boldsymbol{w}^*) \leq f(\boldsymbol{w})]\,.$$

Optimality conditions In general, finding global or even local minima is a hard problem. For this reason, researchers in optimization have derived mathematical expressions that can be checked at a given point in finite time (unlike the conditions above), and help assess whether a given point is a local minimum or not.

In this introductory chapter, we will present these conditions in the context of an unconstrained optimization problem

$$\underset{\boldsymbol{w}\in\mathbb{R}^d}{\text{minimize}}\, f(\boldsymbol{w}). \tag{1.1.5}$$

Theorem 1.1.2 (First-order necessary condition) *Suppose that the objective function f in problem (1.1.5) belongs to $\mathcal{C}^1(\mathbb{R}^d)$. Then,*

$$[\,\boldsymbol{w}^* \text{ is a local minimum of } f\,] \implies \|\nabla f(\boldsymbol{w}^*)\| = 0. \tag{1.1.6}$$

Note that this condition is only necessary: there may exist points with zero gradient that are not local minima. Indeed, the set of points with zero gradient, called first-order stationary points or **first-order critical points**, may also include local maxima and saddle points[1]. We will investigate saddle points further in Chapter 3.

Provided we strengthen our smoothness requirements on f, we can establish stronger optimality conditions for problem (2.1.1).

Theorem 1.1.3 (Second-order necessary condition) *Suppose that the objective function f in problem (1.1.5) belongs to $\mathcal{C}^2(\mathbb{R}^d)$. Then,*

$$[\, \boldsymbol{w}^* \text{ is a local minimum of } f \,] \implies \left[\|\nabla f(\boldsymbol{w}^*)\| = 0 \quad \text{and} \quad \nabla^2 f(\boldsymbol{w}^*) \succeq \mathbf{0} \right]. \tag{1.1.7}$$

From Theorem 1.1.2, first-order stationary points that violate the condition $\nabla^2 f(\boldsymbol{w}^*) \succeq \mathbf{0}$ cannot be local minima: conversely, a stronger version of this property guarantees that we are in presence of a local minimum.

Theorem 1.1.4 (Second-order sufficient condition) *Suppose that the objective function f in problem (1.1.5) belongs to $\mathcal{C}^2(\mathbb{R}^d)$. Then,*

$$\left[\|\nabla f(\boldsymbol{w}^*)\| = 0 \quad \text{and} \quad \nabla^2 f(\boldsymbol{w}^*) \succ \mathbf{0} \right] \implies [\, \boldsymbol{w}^* \text{ is a local minimum of } f \,] \tag{1.1.8}$$

By exploiting the second-order derivative, it is thus possible to certify whether a point is a local minima (note that there could be local or even global minima such that $\nabla^2 f(\boldsymbol{w}^*) \succeq \mathbf{0}$). With further assumptions on the structure of the problem, these optimality conditions can be more informative about minima. This is the case when the objective function is convex: we detail this property in the next section.

1.1.3 Convexity

Convexity is at its core a geometric notion: before defining what a convex function is, we describe the corresponding property for a set.

Definition 1.1.9 (Convex set) *A set $\mathcal{C} \in \mathbb{R}^d$ is called* **convex** *if*

$$\forall (\boldsymbol{u}, \boldsymbol{v}) \in \mathcal{C}^2, \ \forall t \in [0,1], \qquad t\boldsymbol{u} + (1-t)\boldsymbol{v} \in \mathcal{C}.$$

Example 1.1.1 (Examples of convex sets) *The following sets are convex:*

- *The entire space $\mathbb{R}^d$;*
- *Every line segment of the form $\{t\boldsymbol{w} | t \in \mathbb{R}\}$ for some $\boldsymbol{w} \in \mathbb{R}^d$;*
- *Every (Euclidean) ball of the form $\left\{ \boldsymbol{w} \in \mathbb{R}^d \,\middle|\, \|\boldsymbol{w}\|^2 = \sum_{i=1}^d [\boldsymbol{w}]_i^2 \leq 1 \right\}$.*

We now provide the basic definition of a convex function.

[1] A vector is a saddle point of a function if it is a local minimum with respect to certain directions and a local maximum with respect to other directions of the space.

Definition 1.1.10 (Convex function) *A function $f : \mathbb{R}^d \to \mathbb{R}$ is* **convex** *if*

$$\forall (\boldsymbol{u}, \boldsymbol{v}) \in (\mathbb{R}^d)^2,\ \forall t \in [0,1], \qquad f(t\boldsymbol{u} + (1-t)\boldsymbol{v}) \leq t\, f(\boldsymbol{u}) + (1-t)\, f(\boldsymbol{v}).$$

Example 1.1.2 *The following functions are convex :*

- *Linear functions of the form $\boldsymbol{w} \mapsto \boldsymbol{a}^{\mathrm{T}}\boldsymbol{w} + b$, with $\boldsymbol{a} \in \mathbb{R}^d$ and $b \in \mathbb{R}$;*
- *Squared Euclidean norm: $\boldsymbol{w} \mapsto \|\boldsymbol{w}\|^2 = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}$.*

For differentiable functions, it is possible to characterize convexity using the derivatives of the function.

Theorem 1.1.5 *A function $f : \mathbb{R}^d \to \mathbb{R}$ in $\mathcal{C}^1(\mathbb{R}^d)$ is convex if and only if*

$$\forall \boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^d, \quad f(v) \geq f(u) + \nabla f(u)^{\mathrm{T}}(v-u). \tag{1.1.9}$$

The inequality (1.1.9) is fundamental in analyzing convex optimization algorithms, as it provides an underestimator for the variation of a (convex) objective function.

Convexity can also be characterized using the second-order derivative/the Hessian matrix (provided this derivative exists).

Theorem 1.1.6 *A function $f : \mathbb{R}^d \to \mathbb{R}$ in $\mathcal{C}^2(\mathbb{R}^d)$ is convex if and only if*

$$\forall \boldsymbol{w} \in \mathbb{R}^d, \quad \nabla^2 f(\boldsymbol{w}) \succeq \mathbf{0}. \tag{1.1.10}$$

Convex functions are particularly suitable for minimization problems as they satisfy the following property.

Theorem 1.1.7 *If f is a convex function, then every local minimum of f is a global minimum.*

If the function is differentiable, the optimality conditions as well as the characterization of convexity lead us to the following result.

Corollary 1.1.1 *If f is continuously differentiable, every point $\boldsymbol{w}^*$ such that $\|\nabla f(\boldsymbol{w}^*)\| = 0$ is a global minimum of f.*

Strong convexity The results above can be further improved by assuming that a convex function is strongly convex, as defined below.

Definition 1.1.11 (Strongly convex function) *A function $f : \mathbb{R}^d \to \mathbb{R}$ is μ-strongly convex (or* strongly convex of modulus $\mu > 0$*) if for all $(\boldsymbol{u}, \boldsymbol{v}) \in (\mathbb{R}^d)^2$ and $t \in [0,1]$,*

$$f(t\boldsymbol{u} + (1-t)\boldsymbol{v}) \ \leq\ t\, f(\boldsymbol{u}) + (1-t) f(\boldsymbol{v}) - \frac{\mu}{2} t(1-t)\|\boldsymbol{v} - \boldsymbol{u}\|^2.$$

Strong convexity leads to an even more desirable property in terms of optimization landscape: the global minimum (when it exists) is unique.

Theorem 1.1.8 *If f is a continuous, strongly convex function, it has a unique global minimum.*

Similarly to convex functions, it is possible to characterize strong convexity using first- and second-order derivatives.

Theorem 1.1.9 *A function $f : \mathbb{R}^d \to \mathbb{R}$ in $\mathcal{C}^1(\mathbb{R}^d)$ is μ-strongly convex if and only if*

$$\forall \boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^d, \quad f(\boldsymbol{v}) \geq f(\boldsymbol{u}) + \nabla f(\boldsymbol{u})^{\mathrm{T}}(\boldsymbol{v} - \boldsymbol{u}) + \frac{\mu}{2}\|\boldsymbol{v} - \boldsymbol{u}\|^2. \tag{1.1.11}$$

Theorem 1.1.10 *A function $f : \mathbb{R}^d \to \mathbb{R}$ in $\mathcal{C}^2(\mathbb{R}^d)$ is μ-strongly convex if and only if*

$$\forall \boldsymbol{w} \in \mathbb{R}^d, \quad \nabla^2 f(\boldsymbol{w}) \succeq \mu \mathbf{I}. \tag{1.1.12}$$

We end this section by giving two examples of strongly convex optimization problems.

Example 1.1.3 (Convex quadratic problems) *Consider*

$$\underset{\boldsymbol{w} \in \mathbb{R}^d}{\text{minimize}} \; f(\boldsymbol{w}) := \frac{1}{2}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{w} + \boldsymbol{b}^{\mathrm{T}}\boldsymbol{w}, \quad \boldsymbol{A} = \boldsymbol{A}^{\mathrm{T}} \succeq \mathbf{0}.$$

The function f belongs to $\mathcal{C}^2(\mathbb{R}^d)$, with $\nabla^2 f(\boldsymbol{w}) = \boldsymbol{A}$ for every $\boldsymbol{w} \in \mathbb{R}^d$. As a result, this function is convex. Moreover, if we assume that $\boldsymbol{A} \succ \mathbf{0}$, then the function is $\lambda_{\min}(\boldsymbol{A})$-strongly convex.

Example 1.1.4 (Projection onto a closed, convex set) *Let $\mathcal{X} \subseteq \mathbb{R}^d$ be a convex, closed[2] set, and $\boldsymbol{a} \in \mathbb{R}^d$. The problem of computing the projection of $\boldsymbol{a}$ onto $\mathcal{X}$ is formulated as*

$$\underset{\boldsymbol{w} \in \mathcal{X}}{\text{minimize}} \; \frac{1}{2}\|\boldsymbol{w} - \boldsymbol{a}\|^2.$$

The objective function of this problem is 1-strongly convex, which implies that the problem has a unique solution (i. e. the projection is unique).

1.2 About optimization algorithms

The field of optimization can be broadly divided into three categories:

- Mathematical optimization is concerned with the theoretical study of complex optimization formulations, and the proof of well-posedness of such problems (for instance, prove that there exist solutions to a given problem);
- Computational optimization deals with the development of software that can solve a family of optimization problems, through careful implementation of efficient methods;
- **Algorithmic** optimization lies in-between the previous two categories, and aims at proposing new algorithms that address a particular issue, with theoretical guarantees and/or validation of their practical interest.

These notes cover material from the third category of optimization activities. The design of optimization algorithms (also called methods, or schemes) is a particularly subtle process, as an algorithm must exploit the theoretical properties of the problem while being amenable to implementation on a computer.

[2] A set $\mathcal{X} \subseteq \mathbb{R}^d$ is closed if for every converging subsequence of $\{\boldsymbol{x}_n\}_n$, the limit of this sequence belongs to $\mathcal{X}$.

1.2.1 Basics of optimization methods

Most numerical optimization algorithms do not attempt to find a solution of a problem in a direct way, and rather proceed in an *iterative* fashion. Given a current point, that represents the current approximation to the solution, an optimization procedure attempts to move towards a (potentially) better point: to this end, the method generally requires a certain amount of calculation.

Suppose we apply such a process to the problem $\text{minimize}_{\boldsymbol{w}\in\mathbb{R}^d} f(\boldsymbol{w})$, resulting in a sequence of iterates $\{\boldsymbol{w}_k\}_k$. Ideally, these iterates obey one of the scenarios below:

1. The iterates get increasingly close to a solution, i. e.

$$\|\boldsymbol{w}_k - \boldsymbol{w}^*\| \to 0 \quad \text{when } k \to \infty.$$

   Although $\boldsymbol{w}^*$ is generally not known in practice, such results can be guaranteed by the theory, for instance on strongly convex problems.

2. The function values associated with the iterates get increasingly close to the optimum, i. e.

$$f(\boldsymbol{w}_k) \to f^* \quad \text{when } k \to \infty,$$

   As for the case above, f^* may not be known, but it can still be possible to prove convergence for certain algorithms and function classes (typically strongly convex, smooth functions).

3. The first-order optimality condition gets close to being satisfied, that is, $f \in \mathcal{C}^1(\mathbb{R}^d)$ and

$$\|\nabla f(\boldsymbol{w}_k)\| \to 0 \quad \text{when } k \to \infty.$$

Out of the three conditions, the last one is the easiest to track as the algorithm unfolds: it is, however, only a necessary condition, and does not guarantee convergence to a local minimum for generic, nonconvex functions. On the other hand, the first two conditions can only be measured approximately (by looking at the behavior of the iterates and enforcing decrease in the function values), but lead to stronger guarantees.

1.2.2 Convergence and convergence rates

The typical theoretical results that optimizers aim at proving for algorithms are asymptotic, as shown above: they only provide a guarantee in the limit. In practice, one may want to obtain more precise guarantees, that relate to a certain accuracy target that the practitioner would like to achieve. This led to the development of **global convergence rates**.

Example 1.2.1 (Global convergence rate for the gradient norm) *Given an algorithm applied to* $\text{minimize}_{\boldsymbol{w}\in\mathbb{R}^d} f(\boldsymbol{w})$ *that produces a sequence of iterates* $\{\boldsymbol{w}_k\}$*, we say that the method is* $\mathcal{O}(1/k)$ *for the gradient norm, or* $\|\nabla f(\boldsymbol{w}_k)\| = \mathcal{O}\left(\frac{1}{k}\right)$ *if*

$$\exists C > 0, \quad \|\nabla f(\boldsymbol{w}_k)\| \leq \frac{C}{k} \; \forall k.$$

Such rates allow to quantify how much effort (in terms of iterations) is needed to reach a certain target accuracy $\epsilon > 0$. This leads to the companion notion of **worst-case complexity bound**.

Example 1.2.2 (Worst-case complexity for the gradient norm) *Given an algorithm applied to* $\text{minimize}_{\boldsymbol{w}\in\mathbb{R}^d} f(\boldsymbol{w})$ *that produces a sequence of iterates* $\{\boldsymbol{w}_k\}$*, we say that the method has a worst-case complexity of* $\mathcal{O}\left(\epsilon^{-1}\right)$ *for the gradient norm if*

$$\exists C > 0,\ \|\nabla f(\boldsymbol{w}_k)\| \leq \epsilon \text{ when } k \geq \frac{C}{\epsilon}.$$

Such results are quite common in theoretical computer science or statistics, which partly explain their popularity in machine learning. In optimization, they have been developed for a number of years in the context of convex optimization but have only raised significant interest in general optimization over the last decade.

In the next chapters, we will analyze gradient descent through the prism of convergence rate analysis, and use the results to get insight on the practical performance of this method.

Chapter 2

Gradient descent and acceleration

In this chapter, we investigate the performance of gradient descent on convex optimization problems. After recalling the key results for gradient descent, we discuss other schemes that rely on gradient steps but possess better convergence guarantees. Although the intuition for such **accelerated** variants remains elusive, we will provide the mathematical details of the most famous method of this form.

2.1 Gradient descent on convex problems

In this chapter, we are interested in general optimization problems of the form

$$\underset{\boldsymbol{w}\in\mathbb{R}^d}{\text{minimize}}\, f(\boldsymbol{w}), \tag{2.1.1}$$

where we assume that $f \in \mathcal{C}^1(\mathbb{R}^d)$, therefore the gradient mapping for f exists, is continuous: we will also assume that it can be used in an algorithm. We will develop an algorithm that primarily relies on the use of gradient information, termed gradient descent.

2.1.1 Algorithm

Since we consider a problem with a continuously differentiable function, we know from the first-order optimality condition that for any local minimum $\boldsymbol{w}^*$, we necessarily have $\nabla f(\boldsymbol{w}^*) = 0$. As a result, given any point $\boldsymbol{w} \in \mathbb{R}^d$, only one of the two properties below holds:

1. Either $\nabla f(\boldsymbol{w}) = 0$, and $\boldsymbol{w}$ can be a local minimum;
2. Or $\nabla f(\boldsymbol{w}) \neq 0$ and the function f decreases *locally* from $\boldsymbol{w}$ in the direction of -$\nabla f(\boldsymbol{w})$.

Using this result, we can design the update rule

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha \nabla f(\boldsymbol{w}), \tag{2.1.2}$$

where $\alpha > 0$ is a stepsize parameter. If $\nabla f(\boldsymbol{w}) = 0$, the vector $\boldsymbol{w}$ does not change: this is consistent with the notion of first-order stationarity (we cannot get more information by using the gradient). On the contrary, when $\nabla f(\boldsymbol{w}) \neq 0$, we expect that there exists a range of values for $\alpha > 0$ for which such an update leads to a point with a lower objective value.

Remark 2.1.1 *When f is convex, we even have equivalence between $\|\nabla f(\boldsymbol{w})\| = 0$ and the fact that $\boldsymbol{w} \in \text{argmin}_{\boldsymbol{w}\in\mathbb{R}^d} f(\boldsymbol{w})$.*

Using the updating rule (2.1.2), we can design an algorithm for the minimization of the function f: this method is called **gradient descent**[1] and described in Algorithm 1.

Algorithm 1: Gradient descent algorithm.

Initialization: $\boldsymbol{w}_0 \in \mathbb{R}^d$.
for $k = 0, 1, \ldots$ **do**

1. Compute the gradient $\nabla f(\boldsymbol{w}_k)$.
2. Compute a steplength $\alpha_k > 0$.
3. Set $\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha_k \nabla f(\boldsymbol{w}_k)$.

end

A number of variants on Algorithm 1 can be obtained upon introducing a stopping criterion (typically a maximum number of iterations) and choosing a suitable rule for computing the sequence $\{\alpha_k\}_k$. The latter will be our focus in the next section.

2.1.2 Choosing the stepsize

There are numerous techniques used to select the stepsize[2]. We review the most general below, but point out that those are generally combined with knowledge about the problem in practice.

Constant stepsize One possible strategy is to maintain a constant step size throughout the entire algorithmic run, i. e. set $\alpha_k = \alpha > 0$. If the budget allows for it, several values of α can be tested for comparison. Under regularity assumptions on f, one can guarantee that there exists a value below which a constant stepsize will lead to complexity guarantees (see Section 2.1.3). For instance, when $f \in \mathcal{C}_L^{1,1}(\mathbb{R}^d)$, the choice

$$\alpha_k = \alpha = \frac{1}{L} \tag{2.1.3}$$

leads to such guarantees. Because of its dependence in L, this choice is tailored to the problem at hand. Note that the rule (2.1.3) requires knowledge of the Lipschitz constant, but this information may not be available in practice.

Decreasing stepsize Another popular choice consist in choosing the entire sequence $\{\alpha_k\}$ in advance so as to guarantee that $\alpha_k \to 0$ as $k \to \infty$. This also enables the derivation of theoretical results, under some conditions that can help designing the formula for the α_ks. However, this process forces the steps to get increasingly smaller, which may prevent fast progress towards the end of the algorithm.

[1] Although "gradient descent" is the most common terminology in data science, the historical name used in optimization is "steepest descent", because we move in the direction of the negative gradient is the direction of steepest change at a given point.

[2] Or *learning rate* in machine learning.

Adaptive choice with line search Line-search techniques have been widely used in continuous optimization: at every iteration, they aim at computing the value of α_k that leads to the largest decrease in the function value in the direction $-\nabla f(\boldsymbol{w}_k)$. In general, such exact line searches are not practical, and thus an inexact process is preferred. The most popular method is backtracking, that proceeds by testing a set of decreasing values: a simple version of a backtracking line search is described in Algorithm 1.

Algorithm 2: Basic backtracking line search in direction d.

Inputs: $\boldsymbol{w} \in \mathbb{R}^d$, $\boldsymbol{d} \in \mathbb{R}^d$, $\alpha^0 \in \mathbb{R}^d$.
Initialization: Set $\alpha = \alpha^0$ and $j = 0$.
while $f(\boldsymbol{w} + \alpha \boldsymbol{d}) > f(\boldsymbol{w})$ **do**
  Set $\alpha = \frac{\alpha}{2}$.
end
Output: α_j.

We can thus incorporate this line-search technique in step 2 of Algorithm 1 by calling the method with $\boldsymbol{w} = \boldsymbol{w}_k$, $\boldsymbol{d} = -\nabla f(\boldsymbol{w}_k)$ and (for instance) $\alpha_0 = 1$. Many variants can be built upon this simple framework. One drawback of line-search methods is that they require to evaluate the objective function, which can be deemed too expensive in certain data science contexts.

2.1.3 Convergence rate analysis of gradient descent

In this section, we present several convergence rates for gradient descent, in the case of a smooth objective function. We will eventually assume convexity of f, however note that the next result does not require f to be convex.

Proposition 2.1.1 (Descent property) *Consider the k-th iteration of Algorithm 1 applied to $f \in \mathcal{C}_L^{1,1}(\mathbb{R}^d)$, and suppose that $\nabla f(\boldsymbol{w}_k) \neq 0$. Then, if $0 < \alpha_k < \frac{2}{L}$, we have*

$$f(\boldsymbol{w}_k - \alpha_k \nabla f(\boldsymbol{w}_k)) < f(\boldsymbol{w}_k).$$

In particular, choosing $\alpha_k = \frac{1}{L}$ leads to

$$f(\boldsymbol{w}_k - \frac{1}{L}\nabla f(\boldsymbol{w}_k)) < f(\boldsymbol{w}_k) - \frac{1}{2L}\|\nabla f(\boldsymbol{w}_k)\|^2. \tag{2.1.4}$$

Proof. We use the inequality (1.1.1) with the vectors $(\boldsymbol{w}_k, \boldsymbol{w}_k - \alpha_k \nabla f(\boldsymbol{w}_k))$:

$$\begin{aligned} f(\boldsymbol{w}_k - \alpha_l \nabla f(\boldsymbol{w}_k)) &\leq f(\boldsymbol{w}_k) + \nabla f(\boldsymbol{w}_k)^{\mathrm{T}} [-\alpha_k \nabla f(\boldsymbol{w}_k)] + \frac{L}{2}\| - \alpha_k \nabla f(\boldsymbol{w}_k)\|^2 \\ &= f(\boldsymbol{w}_k) - \alpha_k \nabla f(\boldsymbol{w}_k)^{\mathrm{T}} \nabla f(\boldsymbol{w}_k) + \frac{L}{2}\alpha_k^2 \|\nabla f(\boldsymbol{w}_k)\|^2 \\ &= f(\boldsymbol{w}_k) + \left(-\alpha_k + \frac{L}{2}\alpha_k^2\right) \|\nabla f(\boldsymbol{w}_k)\|^2. \end{aligned}$$

If $-\alpha_k + \frac{L}{2}\alpha_k^2 < 0$, the second term on the right-hand side will be negative, thus we will have $f(\boldsymbol{w}_k - \alpha_l \nabla f(\boldsymbol{w}_k)) < f(\boldsymbol{w}_k)$. Since $-\alpha_k + \frac{L}{2}\alpha_k^2 < 0 \Leftrightarrow \alpha_k < \frac{2}{L}$ and $\alpha_k > 0$ by definition, this proves the first part of the result.

To obtain (2.1.4), one simply needs to use $\alpha_k = \frac{1}{L}$ in the series of equations above. □

The result of Proposition 2.1.1, sometimes called a descent lemma, is instrumental to obtain theoretical guarantees on Algorithm 1.

Assumption 2.1.1 *The objective function f belongs to $\mathcal{C}_L^{1,1}(\mathbb{R}^d)$ for $L > 0$ and there exists $f_{\text{low}} \in \mathbb{R}$ such that for every $\boldsymbol{w} \in \mathbb{R}^d$, $f(\boldsymbol{w}) \geq f_{\text{low}}$ (i. e. f is bounded below on $\mathbb{R}^d$).*

Our first result will be derived under the assumption that the function is convex.

Assumption 2.1.2 *The function f in problem 2.1.1 is convex. Moreover, it possesses a minimum $\boldsymbol{w}^*$ (i.e. $\operatorname{argmin}_{\boldsymbol{w}\in\mathbb{R}^d} f(\boldsymbol{w})$ is not empty) and we let $f^* = f(\boldsymbol{w}^*) = \min_{\boldsymbol{w}\in\mathbb{R}^d} f(\boldsymbol{w})$.*

We will now establish a convergence rate for gradient descent in the convex case.

Theorem 2.1.1 (Convergence rate of gradient descent for convex functions) *Let f be a function satisfying Assumptions 2.1.1 and 2.1.2. Suppose that Algorithm 1 is applied with $\alpha_k = \frac{1}{L}$. Then, for any $K \geq 1$, the iterate $\boldsymbol{w}_K$ satisfies*

$$f(\boldsymbol{w}_k) - f^* \leq \mathcal{O}\left(\frac{1}{K}\right). \tag{2.1.5}$$

Proof. Let K be an index such that for every $k = 0, \ldots, K-1$, $f(\boldsymbol{w}_k) - f^* > \epsilon$.
For any $k = 0, \ldots, K-1$, the characterization of convexity (1.1.9) at $\boldsymbol{w}_k$ and $\boldsymbol{w}^*$ gives

$$f(\boldsymbol{w}^*) \geq f(\boldsymbol{w}_k) + \nabla f(\boldsymbol{w}_k)^{\mathrm{T}}(\boldsymbol{w}^* - \boldsymbol{w}_k).$$

Combining this property with (2.1.4), we obtain:

$$\begin{aligned} f(\boldsymbol{w}_{k+1}) &\leq f(\boldsymbol{w}_k) - \frac{1}{2L}\|\nabla f(\boldsymbol{w}_k)\|^2 \\ &\leq f(\boldsymbol{w}^*) + \nabla f(\boldsymbol{w}_k)^{\mathrm{T}}(\boldsymbol{w}_k - \boldsymbol{w}^*) - \frac{1}{2L}\|\nabla f(\boldsymbol{w}_k)\|^2. \end{aligned}$$

To proceed onto the next step, one notices that

$$\nabla f(\boldsymbol{w}_k)^{\mathrm{T}}(\boldsymbol{w}_k - \boldsymbol{w}^*) - \frac{1}{2L}\|\nabla f(\boldsymbol{w}_k)\|^2 = \frac{L}{2}\left(\|\boldsymbol{w}_k - \boldsymbol{w}^*\|^2 - \|\boldsymbol{w}_k - \boldsymbol{w}^* - \frac{1}{L}\nabla f(\boldsymbol{w}_k)\|^2\right).$$

Thus, recalling that $\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \frac{1}{L}\nabla f(\boldsymbol{w}_k)$, we arrive at

$$\begin{aligned} f(\boldsymbol{w}_{k+1}) &\leq f(\boldsymbol{w}^*) + \frac{L}{2}\left(\|\boldsymbol{w}_k - \boldsymbol{w}^*\|^2 - \|\boldsymbol{w}_k - \boldsymbol{w}^* - \frac{1}{L}\nabla f(\boldsymbol{w}_k)\|^2\right) \\ &= f(\boldsymbol{w}^*) + \frac{L}{2}\left(\|\boldsymbol{w}_k - \boldsymbol{w}^*\|^2 - \|\boldsymbol{w}_{k+1} - \boldsymbol{w}^*\|^2\right). \end{aligned}$$

Hence,

$$f(\boldsymbol{w}_{k+1}) - f(\boldsymbol{w}^*) \leq \frac{L}{2}\left(\|\boldsymbol{w}_k - \boldsymbol{w}^*\|^2 - \|\boldsymbol{w}_{k+1} - \boldsymbol{w}^*\|^2\right). \tag{2.1.6}$$

By summing (2.1.6) on all indices k between 0 and $K-1$, we obtain

$$\sum_{k=0}^{K-1} f(\boldsymbol{w}_{k+1}) - f(\boldsymbol{w}^*) \leq \frac{L}{2}\left(\|\boldsymbol{w}_0 - \boldsymbol{w}^*\|^2 - \|\boldsymbol{w}_K - \boldsymbol{w}^*\|^2\right) \leq \frac{L}{2}\|\boldsymbol{w}_0 - \boldsymbol{w}^*\|^2.$$

Finally, using $f(\boldsymbol{w}_0) \geq f(\boldsymbol{w}_1) \geq ... \geq f(\boldsymbol{w}_K)$ (a consequence of Proposition 2.1.1, we obtain that

$$\sum_{k=0}^{K-1} f(\boldsymbol{w}_{k+1}) - f(\boldsymbol{w}^*) \geq K\left(f(\boldsymbol{w}_K) - f^*\right).$$

Injecting this formula into the previous equation finally yields the desired outcome:

$$f(\boldsymbol{w}_k) - f(\boldsymbol{w}^*) \leq \frac{L\|\boldsymbol{w}_0 - \boldsymbol{w}^*\|^2}{2}\frac{1}{K}.$$

□

Equivalently, we say that the worst-case complexity of gradient descent is $\mathcal{O}\left(\epsilon^{-1}\right)$, which means here that there exist a positive constant C (that depends on $\|\boldsymbol{w}_0 - \boldsymbol{w}^*\|$ and L) such that

$$f(\boldsymbol{w}_K) - f_{\text{low}} \leq \epsilon.$$

after at most $C\epsilon^{-1}$ iterations.

We now make a stronger assumption on f, namely, that it is strongly convex.

Assumption 2.1.3 *The function f in problem 2.1.1 is μ-strongly convex with $\mu > 0$. We denote its unique global minimum by $\boldsymbol{w}^*$ and we let $f^* = f(\boldsymbol{w}^*) = \min_{\boldsymbol{w}\in\mathbb{R}^d} f(\boldsymbol{w})$.*

Theorem 2.1.2 (Convergence of gradient descent for strongly convex functions) *Let f be a function satisfying Assumption 2.1.1 and 2.1.3. Suppose that Algorithm 1 is applied with $\alpha_k = \frac{1}{L}$ and let $\epsilon > 0$. Then, for any $K \in \mathbb{N}$, we have*

$$f(\boldsymbol{w}_k) - f^* \leq \left(1 - \frac{\mu}{L}\right)^k \left(f(\boldsymbol{w}_0) - f^*\right) = \mathcal{O}\left((1 - \tfrac{\mu}{L})^k\right). \tag{2.1.7}$$

Proof. We exploit the strong convexity property (1.1.11). For any $(\boldsymbol{u}, \boldsymbol{v}) \in (\mathbb{R}^d)^2$, we have

$$f(\boldsymbol{u}) \geq f(\boldsymbol{v}) + \nabla f(\boldsymbol{v})^{\mathrm{T}}(\boldsymbol{u} - \boldsymbol{v}) + \frac{\mu}{2}\|\boldsymbol{u} - \boldsymbol{v}\|^2.$$

Minimizing both sides with respect to $\boldsymbol{u}$ lead to $\boldsymbol{u} = \boldsymbol{w}^*$ on the left-hand side, and $\boldsymbol{u} = \boldsymbol{v} - \frac{1}{\mu}\nabla f(\boldsymbol{v})$ on the right-hand side.[3] As a result, we obtain for any $\boldsymbol{v}$ that

$$\begin{aligned} f^* &\geq f(\boldsymbol{v}) + \nabla f(\boldsymbol{v})^{\mathrm{T}}\left[-\frac{1}{\mu}\nabla f(\boldsymbol{v})\right] + \frac{\mu}{2}\| - \frac{1}{\mu}\nabla f(\boldsymbol{v})\|^2 \\ f^* &\geq f(\boldsymbol{v}) - \frac{1}{2\mu}\|\nabla f(\boldsymbol{v})\|^2. \end{aligned}$$

By re-arranging the terms, we arrive at

$$\|\nabla f(\boldsymbol{v})\|^2 \geq 2\mu\left[f(\boldsymbol{v}) - f^*\right], \tag{2.1.8}$$

which is valid for any $\boldsymbol{v} \in \mathbb{R}^d$. Using (2.1.8) together with (2.1.4) thus gives

$$f(\boldsymbol{w}_{k+1}) \leq f(\boldsymbol{w}_k) - \frac{1}{2L}\|\nabla f(\boldsymbol{w}_k)\|^2 \leq f(\boldsymbol{w}_k) - \frac{\mu}{L}(f(\boldsymbol{w}_k) - f^*).$$

[3] The proof of this result comes from minimization of strongly convex quadratics, and is left as an exercise.

This leads to

$$f(\boldsymbol{w}_{k+1}) - f^* \leq \left(1 - \frac{\mu}{L}\right)(f(\boldsymbol{w}_k) - f^*),$$

which we can iterate in order to obtain

$$f(\boldsymbol{w}_K) - f^* \leq \left(1 - \frac{\mu}{L}\right)^K (f(\boldsymbol{w}_0) - f^*).$$

It then suffices to note that the bound is also valid for $K = 0$. □

Similar results can be shown for the criterion $\|\boldsymbol{w}_k - \boldsymbol{w}^*\|$: in other words, the distance between the current iterate and the (unique) global optimum decreases at a rate $\mathcal{O}\left((1 - \frac{\mu}{L})^k\right)$.

Equivalently, we can show a worst-case complexity result: the method computes $\boldsymbol{w}_k$ such that $f(\boldsymbol{w}_k) - f^* \leq \epsilon$ (or $\|\boldsymbol{w}_k - \boldsymbol{w}^*\| \leq \epsilon$) in at most $\mathcal{O}\left(\frac{L}{\mu}\ln(\frac{1}{\epsilon})\right)$ iterations.

Remark 2.1.2 *The convergence rate of gradient descent in the strongly convex setting is called* **linear**, *because the logarithm of* $(1 - \mu/L)^K$ *is a linear function of* K *(in contrast to the rate for convex functions, called* **sublinear***). This terminology originates from the machine learning community, and is sometimes replaced by* **exponential** *and* **polynomial** *in optimization results (in that the bounds are exponential and polynomial in* $1/K$*).*

2.2 Acceleration techniques

2.2.1 Introduction: the concept of momentum

In Section 2.1.3, we derived complexity bounds for the gradient descent algorithm in the convex and strongly convex settings. These results are called *upper* complexity bounds, in the sense that they reflect the worst possible convergence rate that this algorithm could exhibit on a given problem. The issue of *lower* bounds, that show a rate that cannot be improved upon, has been the subject to a lot of attention, particularly in the convex optimization community.

For convex functions, the lower bound is actually $\mathcal{O}(\frac{1}{K^2})$, which is a sensible improvement over the bound in $\mathcal{O}(\frac{1}{K})$ of Theorem 2.1.1. There exist methods that can achieve this bound, thanks to an algorithmic principle called **acceleration**. The underlying idea of acceleration is that, at a given iteration and given the available information from previous iterations (in particular, the latest displacement), one can move along a better step than that given by the current gradient.

2.2.2 Nesterov's accelerated gradient method

Among the existing methods based on acceleration, the accelerated gradient algorithm proposed by Yurii Nesterov in 1983 is the most famous, to the point that it has been termed "Nesterov's algorithm".

Algorithm 3 provides a description of the method. Like the gradient descent method, it requires a single gradient calculation per iteration; however, unlike in gradient descent, the gradient is not evaluated at the current iterate $\boldsymbol{w}_k$, but at a combination of this iterate with the previous step $\boldsymbol{w}_k - \boldsymbol{w}_{k-1}$: this term is called the **momentum term**, and is key to the performance of accelerated gradient techniques.

Algorithm 3: Accelerated gradient method.

Initialization: $\boldsymbol{w}_0 \in \mathbb{R}^d$, $\boldsymbol{w}_{-1} = \boldsymbol{w}_0$.
for $k = 0, 1, \ldots$ **do**

1. Compute a steplength $\alpha_k > 0$ and a parameter $\beta_k > 0$.
2. Compute the new iterate as

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha_k \nabla f\left(\boldsymbol{w}_k + \beta_k(\boldsymbol{w}_k - \boldsymbol{w}_{k-1})\right) + \beta_k(\boldsymbol{w}_k - \boldsymbol{w}_{k-1}). \tag{2.2.1}$$

end

Another view of the accelerated gradient descent is that of a two-loop recursion: given $\boldsymbol{w}_0$ and $\boldsymbol{z}_0 = \boldsymbol{w}_0$, the update (2.2.1) can be rewritten as

$$\begin{cases} \boldsymbol{w}_{k+1} &= \boldsymbol{z}_k - \alpha_k \nabla f(\boldsymbol{z}_k) \\ \boldsymbol{z}_{k+1} &= \boldsymbol{w}_{k+1} + \beta_{k+1}(\boldsymbol{w}_{k+1} - \boldsymbol{w}_k). \end{cases} \tag{2.2.2}$$

This formulation decouples the two steps behind the accelerated gradient update: a gradient step on $\boldsymbol{z}_k$, combined with a momentum step on $\boldsymbol{w}_{k+1}$.

Choosing the parameters We now comment on the choice of the stepsize α_k and the momentum parameter β_k. The same techniques than those presented in Section 2.1.2 can be considered for the choice of α_k (stepsize parameter). As in the gradient descent case, the choice $\alpha_k = \frac{1}{L}$ is a standard one.

The choice of β_k is most crucial to obtaining the improved complexity bound. The standard values proposed by Nesterov depend on the nature of the objective function:

- If f is a μ-strongly convex, we set

$$\beta_k = \beta = \frac{\sqrt{L} - \sqrt{\mu}}{\sqrt{L} + \sqrt{\mu}} \tag{2.2.3}$$

  for every k. Note that this requires the knowledge of both the Lipschitz constant of the gradient and the strong convexity constant.

- For a general convex function f, β_k is computed in an adaptive way using two sequences, as follows:

$$t_{k+1} = \frac{1}{2}(1 + \sqrt{1 + 4t_k^2}), t_0 = 0, \quad \beta_k = \frac{t_k - 1}{t_{k+1}}. \tag{2.2.4}$$

The following informal theorem summarizes the complexity results that can be proven for Algorithm 3.

Theorem 2.2.1 *Consider Algorithm 3 applied to a function f satisfying Assumption 2.1.1, with $\alpha_k = \frac{1}{L}$, and let $\epsilon > 0$. Then, for any $K \geq 1$, the iterate $\boldsymbol{w}_K$ computed by Algorithm 3 satisfies*

i) $f(\boldsymbol{w}_K) - f^* \leq \mathcal{O}(\frac{1}{K^2})$ *if* f *also satisfies Assumption 2.1.2 (convex case), provided* β_k *is set according to the adaptive rule (2.2.4);*

ii) $f(\boldsymbol{w}_K) - f^* \leq \mathcal{O}\left((1 - \sqrt{\frac{\mu}{L}})^K\right)$ *if* f *also satisfies Assumption 2.1.3 (*μ*-strongly convex case), provided* β_k *is set to the constant value given by (2.2.3).*

Note that we can also derive worst-case complexity bounds for the accelerated gradient method, that show the same improvement. For instance, for strongly convex functions, we can establish that $f(\boldsymbol{w}_k) - f^* \leq \epsilon$ after at most $\mathcal{O}\left(\sqrt{\frac{L}{\mu}}\ln(\epsilon^{-1})\right)$ iterations, which improves over the $\mathcal{O}\left(\frac{L}{\mu}\ln(\epsilon^{-1})\right)$ bound of gradient descent.

2.2.3 Other accelerated methods

Heavy ball method The heavy ball method is a precursor of the accelerated gradient algorithm, that was proposed by Boris T. Polyak in 1964. Its k-th iteration can be written as

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \alpha\nabla f(\boldsymbol{w}_k) + \beta(\boldsymbol{w}_k - \boldsymbol{w}_{k-1}),$$

where the stepsize and momentum parameters are chosen to be constant values. The key difference between this iteration and Nesterov's lies in the gradient evaluation, which the heavy ball method performs at the current point: in that sense, the heavy ball method performs first the gradient update, then the momentum step, while Nesterov's method adopts the inverse approach. This method achieves the optimal rate of convergence on strongly convex quadratic functions in $\mathcal{O}\left(\left(1 - \sqrt{\frac{\mu}{L}}\right)^K\right)$, but can fail on general strongly convex functions. Note that the heavy-ball method can produce a nonmonotone sequence of objective values $\{f(\boldsymbol{w}_k)\}$.

Conjugate gradient The (linear) conjugate gradient method, proposed by Hestenes and Stiefel in 1952, has remained to this day one of the preferred methods to solve linear systems of equations and strongly convex quadratic minimization problems. Unlike Polyak's method, the conjugate gradient algorithm does not require knowledge of the Lipschitz constant L nor the parameter μ, because it exploits knowledge from the past iterations. The k-th iteration of conjugate gradient can be written as:

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \alpha_k \boldsymbol{p}_k, \quad \boldsymbol{p}_k = -\nabla f(\boldsymbol{w}_k) + \beta_k \boldsymbol{p}_{k-1}.$$

In a standard conjugate gradient algorithm, α_k and β_k are computed using formulas tailored to the problem: this contributes to their convergence rate analysis, which leads to a rate similar to that of accelerated gradient. However, unlike accelerated gradient, the conjugate gradient is guaranteed to terminate after d iterations on a d-dimensional problem. When d is very large, the bound for conjugate gradient matches that of the other methods, and in that sense does not depend on the problem dimension.

2.3 Conclusion

Part of the success of gradient-type methods is due to their inherent simplicity and their (relatively) low computational cost. These techniques also possess good convergence properties on convex and strongly convex problems. The gradient descent framework is the canonical example of a gradient-type algorithm: the choice of a stepsize defines the algorithmic variant at hand, and can have a significant impact in practice. Most stepsize choices are designed so as to satisfy convergence rate (or complexity) guarantees, that are particularly useful in the convex and strongly convex settings. Indeed, those quantify the performance of the method.

A natural question arising from these theoretical results is whether those are optimal. It turns out that gradient descent is not the fastest algorithm in the worst case: other approaches, often termed accelerated gradient techniques, provide better worst-case guarantees: these theoretical properties can reflect on the practical performance. The accelerated variants rely on the concept of momentum, which is also exploited in state-of-the-art stochastic algorithms used to learn complex models in machine learning.

Useful links The session on this topic has been recorded (unfortunately the video has been split in two six minutes before the end). The recordings can be accessed here (first part, about 3 hours) and there (second part, six minutes related to the notebook).

Chapter 3

Nonconvex optimization and gradient-type methods

In the second part of these notes, we focus on nonconvex optimization problems, and discuss the challenges in solving these problems efficiently. We first focus on the problems themselves, and identify undesirable features of nonconvex formulations. We then move to optimization algorithms, and highlight key results that can be obtained for gradient descent in a nonconvex setting.

3.1 Nonconvex formulations

This section is concerned with the properties of an optimization problem (no algorithm involved) when the objective function is nonconvex. Provided the function is sufficiently smooth (or differentiable), it is possible to provide precise characterizations of the local minima of the problem.

3.1.1 Problem and optimality conditions

We consider again a problem of the form

$$\underset{\boldsymbol{w}\in\mathbb{R}^d}{\text{minimize}}\, f(\boldsymbol{w}). \tag{3.1.1}$$

We will study this problem under the assumption that $f \in \mathcal{C}_L^{1,1}(\mathbb{R}^d)$, already made in Chapter 2. In addition, we will assume that the function is bounded below.

Assumption 3.1.1 *There exists a constant $f_{\text{low}} \in \mathbb{R}$ such that $f(\boldsymbol{w}) \geq f_{\text{low}}$ for all $\boldsymbol{w} \in \mathbb{R}^d$.*

Assumption 3.1.1 is particularly important in nonconvex optimization, where the lack of convexity can lead to functions that are unbounded below, such as $\boldsymbol{w} \mapsto -\|\boldsymbol{w}\|^2$. The existence of f_{low} means that the optimization procedure will at best converge to a point with optimal value f_{low}. In the context of optimization algorithms, Assumption 3.1.1 is often replaced by another property involving the initial point $\boldsymbol{w}_0$: it is assumed that f is bounded below on $\{\boldsymbol{w} \mid f(\boldsymbol{w}) \leq f(\boldsymbol{w}_0)\}$.

Unlike in convex optimization, nonconvex problems may have local minima that are not global minima. Solving these problems globally is thus much harder in general, and local minimization is often regarded as a more tractable alternative, in that algorithms can be designed to detect whether points are local minima (and if not, move on to better points). Still, such methods could converge

to local minima that possess significantly larger function values than the optimum: such points are called **spurious local minima**, and arise in highly nonconvex problems such as those posed by the training of neural networks.

Nonconvexity can also be an issue while attempting to find local minima. In Chapter 1, we saw that local solutions of problem 3.1.1 satisfy necessary optimality conditions. In particular, any local minimum $\boldsymbol{w}^*$ of f is a first-order critical point, in that $\|\nabla f(\boldsymbol{w}^*)\| = 0$. The set of first-order critical points is typically larger than the set of local minima for nonconvex functions. This leads to the following definitions.

Definition 3.1.1 *Consider a function $f \in \mathcal{C}^2(\mathbb{R}^d)$ and a point $\bar{\boldsymbol{w}} \in \mathbb{R}^d$ such as $\|\nabla f(\bar{\boldsymbol{w}})\| = 0$.*

- *If $\nabla^2 f(\bar{\boldsymbol{w}}) \prec \mathbf{0}$, this point is called a* **local maximum**.
- *If $\nabla^2 f(\bar{\boldsymbol{w}})$ has both positive and negative eigenvalues, the point $\bar{\boldsymbol{w}}$ is called a* **strict saddle point**.
- *If $\nabla^2 f(\bar{\boldsymbol{w}}) \succeq \mathbf{0}$ and $\bar{\boldsymbol{w}}$ is not a local minimum, then $\bar{\boldsymbol{w}}$ is called a* **high-order saddle point**.[1]

From this definition, it is clear that algorithms that aim at finding a first-order stationary point (with zero gradient) could be trapped at saddle points or even local maxima. Similarly, algorithms that can guarantee convergence to *second-order stationary points* (with zero gradient and positive semidefinite Hessian matrix) could converge to high-order saddle points. Most optimization algorithms used in practice aim at one of those two guarantees, which raises questions regarding their guarantees. Fortunately, in many nonconvex formulations arising in data science, the optimization landscape (i.e. the set of first-order critical points) is nice enough that applying techniques such as gradient descent can be appropriate.

3.1.2 Examples of nonconvex optimization problems

Example 3.1.1 (Principal component analysis and eigenvalue optimization) *Consider a data matrix $\boldsymbol{X} = [\boldsymbol{x}_i^{\mathrm{T}}]_{i=1}^n \in \mathbb{R}^{n \times d}$. We seek the first principal component of $\boldsymbol{X}$, that corresponds to the direction of maximum variability of the vectors $\boldsymbol{x}_i$. To this end, we form the empirical covariance matrix*

$$\boldsymbol{C} = \frac{1}{n}\sum_{i=1}^n (\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^{\mathrm{T}}, \quad \text{where} \quad \bar{\boldsymbol{x}} = \frac{1}{n}\sum_{i=1}^n \boldsymbol{x}_i.$$

and compute its maximum eigenvalue and an associated eigenvector. This amounts to solving the following minimization problem:

$$\underset{\boldsymbol{w} \in \mathbb{R}^d}{\text{minimize}} \, \frac{1}{2}\boldsymbol{w}^{\mathrm{T}}(-\boldsymbol{C})\boldsymbol{w} \quad \textit{subject to} \quad \|\boldsymbol{w}\|^2 = 1. \tag{3.1.2}$$

The formula for $\boldsymbol{C}$ guarantees that the matrix $-\boldsymbol{C}$ is negative semidefinite (denoted by $-\boldsymbol{C} \preceq \mathbf{0}$), thus the problem consists in minimizing a concave function over a nonconvex constraint. Despite this property, the local minima of the problem (3.1.2) are global minima, and all saddle points are strict when $-\boldsymbol{C} \prec \mathbf{0}$. Indeed, if $\boldsymbol{C} = \mathbf{0}$, all unit vectors are global minima. Assuming that $\boldsymbol{C}$ is

[1] The terminology is used in lieu of high-order critical point, which would be more rigorous. In fact, the definition could be stated using derivatives of arbitrary order, but most algorithms only consider first- and second-order derivatives and optimality conditions, as those can already lead to expensive calculations.

nonzero and thus has at least one positive eigenvalue, it can be shown that the set of critical points of the problem is the set of unit eigenvectors of $\boldsymbol{C}$. [2] *Among those points, only the eigenvectors corresponding to the largest eigenvalue of $\boldsymbol{C}$ (which corresponds to the smallest eigenvalue of $-\boldsymbol{C}$) are local minima with the same function value: hence these are global minima, and the problem does not have any spurious local minimum. The remaining eigenvectors are strict saddle points.*

Example 3.1.2 (Low-rank matrix completion) *Let $\boldsymbol{M} \in \mathbb{R}^{n \times d}$ be a data matrix, which is assumed to have low rank $r \ll \min\{n, d\}$.*[3] *Suppose that we sample a subset of the coefficients in $\boldsymbol{M}$, denoted by $\mathcal{S} \in \{1, \dots, n\} \times \{1, \dots, d\}$. Our goal is then to reconstruct the matrix $\boldsymbol{M}$ from the samples, hoping that this task can be performed with significantly less samples than $n \times d$ because of the low-rank property.*

Rather than formulating the problem using a matrix $\boldsymbol{W} \in \mathbb{R}^{n \times d}$ with an explicit low-rank constraint, we consider the so-called Burer-Monteiro approach in which the problem is solved over two smaller matrices $\boldsymbol{U} \in \mathbb{R}^{n \times r}$ and $\boldsymbol{V} \in \mathbb{R}^{d \times r}$, which can be quite cheaper in terms of problem variables ($(n+d)r$ versus nd). As a result, we consider:

$$\underset{\boldsymbol{U} \in \mathbb{R}^{n \times r}, \boldsymbol{V} \in \mathbb{R}^{d \times r}}{\text{minimize}} \ \frac{1}{2} \sum_{(i,j) \in \mathcal{S}} \left(\left[\boldsymbol{U} \boldsymbol{V}^{\mathrm{T}} \right]_{ij} - \boldsymbol{M}_{ij} \right)^2. \tag{3.1.3}$$

Problem (3.1.3) is nonconvex in the variables $\boldsymbol{U}$ and $\boldsymbol{V}$. Still, under appropriate assumptions on the data and the number of samples, it is possible to show that there are no spurious local minima, and that all saddle points are strict.

3.2 Gradient descent in the nonconvex setting

In this section, we return to the gradient descent algorithm investigated in Chapter 2. Our goal is first to show the versatility of this method, in that it can be applied to any problem with a continuously differentiable function and be endowed with convergence guarantees. However, the guarantees for nonconvex functions are weaker that what we established for convex and strongly convex functions. We will present first a standard analysis in terms of convergence rates, then highlight some recent results on the probabilistic behavior of gradient descent.

3.2.1 Convergence rate

In the nonconvex case, analyzing the behavior of gradient descent amounts to bounding the number of iterations required to drive the gradient norm below some threshold $\epsilon > 0$: this means that we should be able to show that the gradient norm actually goes below this threshold, which is a guarantee of convergence.

Theorem 3.2.1 (Complexity of gradient descent for nonconvex functions) *Let f be a nonconvex function satisfying Assumptions 2.1.1 and 3.1.1. Suppose that Algorithm 1 is applied to this problem with $\alpha_k = \frac{1}{L}$. Then, for any $K \geq 1$, we have*

$$\min_{0 \leq k \leq K-1} \|\nabla f(\boldsymbol{w}_k)\| \leq \mathcal{O}\left(\frac{1}{\sqrt{K}} \right). \tag{3.2.1}$$

[2] See the lecture by V. Duval on constrained optimization.

[3] For an arbitrary matrix $\boldsymbol{M} \in \mathbb{R}^{n \times d}$, the rank is defined as the number of nonzero eigenvalues of $\boldsymbol{M}^{\mathrm{T}} \boldsymbol{M}$.

Proof. From Proposition 2.1.1, we have that

$$\forall k = 0, \dots, K-1, \quad f(\boldsymbol{w}_{k+1}) \leq f(\boldsymbol{w}_k) - \frac{1}{2L}\|\nabla f(\boldsymbol{w}_k)\|^2 \leq f(\boldsymbol{w}_k) - \frac{1}{2L}\left(\min_{0\leq k\leq K-1} \|\nabla f(\boldsymbol{w}_k)\|\right)^2.$$

By summing across all such iterations, we obtain :

$$\sum_{k=0}^{K-1} f(\boldsymbol{w}_{k+1}) \leq \sum_{k=0}^{K-1} f(\boldsymbol{w}_k) - \frac{K}{2L}\left(\min_{0\leq k\leq K-1} \|\nabla f(\boldsymbol{w}_k)\|\right)^2.$$

Removing identical terms on both sides yields

$$f(\boldsymbol{w}_K) \leq f(\boldsymbol{w}_0) - \frac{K}{2L}\left(\min_{0\leq k\leq K-1} \|\nabla f(\boldsymbol{w}_k)\|\right)^2.$$

Using $f(\boldsymbol{w}_K) \geq f_{\text{low}}$ (which holds by Assumption 2.1.1) and re-arranging the terms leads to

$$\min_{0\leq k\leq K-1} \|\nabla f(\boldsymbol{w}_k)\| \leq \left[\frac{2L(f(\boldsymbol{w}_0) - f_{\text{low}})}{K}\right]^{1/2} = \mathcal{O}\left(\frac{1}{\sqrt{K}}\right).$$

□

Equivalently, we say that the worst-case complexity of gradient descent is $\mathcal{O}\left(\epsilon^{-2}\right)$, because for any $\epsilon > 0$, a reasoning similar to the proof of Theorem 3.2.1 guarantees that $\min_{0\leq k\leq K-1} \|\nabla f(\boldsymbol{w}_k)\| \leq \epsilon$ after at most

$$\left\lceil 2L(f(\boldsymbol{w}_0) - f_{\text{low}})\epsilon^{-2} \right\rceil = \mathcal{O}(\epsilon^{-2})$$

iterations.

Two observations can be made on the result of Theorem 3.2.1. First, the lack of convexity prevents from obtaining guarantees in terms of function values or iterates: instead, one obtains guarantees on the gradient norm. Those are enough to show asymptotic convergence to a point with zero gradient, but this point need not be a local minimum or a point close to a solution in terms of function value, as discussed in Section 3.1.1.

Secondly, the rate appearing in the bound (regardless of what it quantifies) is worse (slower) than that obtained in Chapter 2 for convex problems. In that sense, gradient descent is viewed as slower on nonconvex problems than on convex or strongly convex problems.

Remark 3.2.1 *For nonconvex optimization of $\mathcal{C}^1$ functions, it is known that there exists a function for which gradient descent converges exactly at the $\mathcal{O}(\frac{1}{\sqrt{K}})$ rate: in this case, the lower bound matches the upper bound, and can be obtained for a number of standard algorithms in unconstrained optimization. The situation is different for $\mathcal{C}^2$, where using second-order derivatives can improve the theoretical guarantees.*

3.2.2 Recent advances on gradient descent on nonconvex functions

Given the results established in the previous section, it might seem that gradient descent is not the ideal method to apply on nonconvex functions. Still, it remains a popular choice in practice, due to its simplicity and low computational cost. In many applications, it has been found to converge to local (and sometimes even global) minima of nonconvex functions. On the other hand, it is quite easy to construct counterexamples for which gradient descent will get stuck at saddle points or global maxima.

Example 3.2.1 *Run gradient descent on $f : \boldsymbol{w} \mapsto -\|\boldsymbol{w}\|^2$ with $\boldsymbol{w}_0 = \boldsymbol{0}$. Regardless of the stepsize choice, the method will remain stuck at the origin, which is a local maximum.*

The gap between these results has eluded optimization researchers for a number of years.A recent result, due to Lee et al. [3], provided a mathematical justification that reconciled the theoretical, pathological examples and the good behavior often observed in practice.

Theorem 3.2.2 ("Gradient descent almost always avoid saddle points") *Let f be a nonconvex, twice continuously differentiable function satisfying Assumption 2.1.1. Suppose that Algorithm 1 is run with a random initial point drawn according to the Lebesgue measure in $\mathbb{R}^d$, with $\alpha_k < \frac{1}{L}$ for every k. Then, with probability 1, the algorithm converges towards a second-order stationary point, i.e. a point with zero gradient and positive semidefinite Hessian matrix.*

The result of Theorem 3.2.2 guarantees that gradient descent will almost surely (i.e. with probability 1) converge to a critical point with nonnegative curvature. The randomness is taken on all possible choices for $\boldsymbol{w}_0$: the proof technique is based on the stable manifold theorem of differential geometry, and establishes that the set of $\boldsymbol{w}_0$s for which the method converges to a local maximum or a strict saddle point is of zero measure. In that sense, it is neglectible compared to the entire space, and by drawing $\boldsymbol{w}_0$ at random, one has a zero probability of drawing such an undesirable initial point.

Guaranteed convergence to a second-order stationary point is particularly valuable in the case of problems where all saddle points are strict, and when there are no spurious local minima. The result of Theorem 3.2.2 has thus attracted significant interest from the data science community, and a number of extensions. One relevant proposal is the variant on gradient descent proposed by Jin et al. [2]. Rather than putting the randomness on $\boldsymbol{w}_0$, these authors modify the gradient descent iterates by injecting noise. More precisely, the method starts by performing standard gradient descent iterations: if the k-th iterate is such that $\|\nabla f(\boldsymbol{w}_k)\|$ is small enough and no noise has been injected over a certain number of past iterations, the gradient descent update is replaced by

$$\boldsymbol{w}_{k+1} := \boldsymbol{w}_k + \xi_k, \qquad \xi_k \sim \mathbb{B}(\boldsymbol{0}, R), \tag{3.2.2}$$

i.e. the iterate is perturbed by a random vector uniformly distributed over the ball centered at the origin and of radius $R > 0$.

Perhaps surprisingly, this method can be equipped with a convergence rate that corresponds to better guarantees than that of standard gradient descent. The theorem below presents the result in the form of a complexity bound.

Theorem 3.2.3 *Suppose that the perturbed gradient descent method described above is applied to a function $f \in \mathcal{C}_\rho^{2,2}(\mathbb{R}^d) \cap \mathcal{C}_L^{1,1}(\mathbb{R}^d)$. Let $\epsilon > 0$ and $\delta \in (0,1)$: then, the method computes an iterate $\boldsymbol{w}_k$ such that*

$$\|\nabla f(\boldsymbol{w}_k)\| \leq \epsilon \quad \text{and} \quad \nabla^2 f(\boldsymbol{w}_k) \succeq -\sqrt{\rho\epsilon}$$

in at most

$$\mathcal{O}\left(\epsilon^{-2} \ln\left[\frac{d}{\epsilon\delta}\right]\right) \tag{3.2.3}$$

iterations with probability at least $1 - \delta$.

The result of Theorem 3.2.3 is probabilistic: there is a probability at least $1-\delta$ that this method satisfies the complexity bound (3.2.3), but there is also a probability at most δ that the bound does not hold. Such results are often termed *high-probability guarantees*, as δ is typically chosen to be small (but not zero, as the bound (3.2.3) would then be infinite).

Interestingly, this theorem shows that gradient descent can be equipped with second-order guarantees, despite being a first-order method in nature.[4] This is achieved thanks to the introduction of randomness, and is only one illustration of a growing trend in optimization: using randomized techniques to improve the performance of algorithms.

[4] Note, however, that the function must be twice continuously differentiable.

3.3 Conclusion

Minimizing a nonconvex function is significantly more challenging than in the convex case. Indeed, local minima cannot be identified with first-order derivatives, while second-order derivatives may only identify strict saddle points and local maxima. Nevertheless, many examples in data science exhibit a favorable landscape, where local minima are also global, and critical points that are not local minima can be escaped from.

When applied to such problems, gradient descent can be shown to converge to a first-order stationary point, but it is possible that this point is not a local minimum. Recent results indicate that it is almost certainly a second-order stationary point, which is a stronger guarantee than conventional wisdom would provide. Injecting randomness into the iterates (not just the initial point) also allows to establish complexity bounds in probability, a fact that renewed interest for randomized algorithms.

Useful link The session of this topic has been recorded. The recording for this course is available here.

Chapter 4

Exercises

4.1 Exercises for Chapter 2

Exercise 1: A strongly convex problem

Consider the minimization problem

$$\underset{\boldsymbol{w}\in\mathbb{R}^d}{\text{minimize}}\, f(\boldsymbol{w}) := \frac{1}{2}\|\boldsymbol{w}\|^2.$$

For any $\boldsymbol{w} \in \mathbb{R}^d$, we have $\nabla f(\boldsymbol{w}) = \boldsymbol{w}$ and $\nabla^2 f(\boldsymbol{w}) = \boldsymbol{I}_d$.

i) Justify that $f \in \mathcal{C}_1^{1,1}(\mathbb{R}^d)$ and that f is 1-strongly convex.

ii) Adapt the general convergence rates of gradient descent and accelerated gradient on strongly convex problems to this particular case. What do the results suggest about the problem?

iii) Write down the first iteration of gradient descent and accelerated gradient for the problem at hand using a stepsize of $\frac{1}{L}$: is the result in agreement with the convergence rates?

Exercise 2: Convergence rates

Let $f : \mathbb{R}^d \to \mathbb{R}$. Suppose that $f \in \mathcal{C}_L^{1,1}(\mathbb{R}^d)$ and that f is μ-strongly convex. Suppose that we apply gradient descent with a constant stepsize $\alpha_k = \frac{1}{\mu+L}$ to this problem.

a) Find a descent property similar to Proposition 2.1.1 for this variant of gradient descent.

b) Adapt the reasoning of the proof of Theorem 2.1.2 to show a convergence rate for this variant. Are additional assumptions on μ and/or L needed?

c) Compare the rate you obtained with the one of Theorem 2.1.2. Under which conditions is it better? Worse?

d) Can it be better than the rate of accelerated gradient on the same problem?

4.2 Exercises for Chapter 3

Exercise 3: Saddle points

Consider the two-dimensional function $f : \mathbb{R}^2 \to \mathbb{R}$ defined by

$$f(\boldsymbol{w}) := w_1^3(w_1 - 1) + w_2^4.$$

a) For every $\boldsymbol{w} \in \mathbb{R}^2$, we have $\nabla f(\boldsymbol{w}) = \begin{bmatrix} 4w_1^3 - 3w_1^2 \\ 4w_2^3 \end{bmatrix}$. Show that $\mathbf{0}$ (the zero vector in $\mathbb{R}^2$) is a first-order critical point of f. Are there others?

b) The formula for the Hessian matrix gives

$$\forall \boldsymbol{w} \in \mathbb{R}^d, \nabla^2 f(\boldsymbol{w}) = \begin{bmatrix} 12w_1^2 - 6w_1 & 0 \\ 0 & 12w_2^2 \end{bmatrix}.$$

How can the Hessian matrix inform about the nature of first-order critical points? What happens here for the points found in the previous question?

c) Show that $f(\boldsymbol{v}) < 0$ for any $\boldsymbol{v} \in \mathbb{R}^2$ such that $v_1 \in (0, 1)$ and $v_2 = 0$. Conclude on the nature of $\mathbf{0}$ for this problem.

Exercise 4: Gradient descent with line search

Consider the minimization of a nonconvex function $f : \mathbb{R}^d \to \mathbb{R}$, under Assumptions 2.1.1 and 3.1.1. Suppose that we use backtracking line search (see Algorithm 2) to compute a suitable stepsize.

a) For any $k \in \mathbb{N}$, suppose that $\|\nabla f(\boldsymbol{w}_k)\| > 0$. Show then that the line search terminates, and that the number of backtracking line-search iterations is bounded above by a constant, and that the line search terminates and outputs $\alpha_k \geq \frac{1}{L}$.

b) Adapt the reasoning of Theorem 3.2.1 to obtain a convergence rate for gradient descent in this setting, in terms of the number of iterations.

c) Turn this convergence rate into a complexity result.

d) Conclude by giving a worst-case complexity bound on the number of *function evaluations* required by the algorithm. How does this bound illustrate the additional cost of line-search techniques?

Bibliography

[1] S. Boyd and L. Vandenberghe. *Introduction to Applied Linear Algebra - Vectors, Matrices and Least Squares*. Cambridge University Press, Cambridge, United Kingdom, 2018.

[2] C. Jin, R. Ge, P. Netrapalli, S. M. Kakade, and M. I. Jordan. How to escape saddle points efficiently. In *Volume 70: International Conference on Machine Learning, 6-11 August 2017, International Convention Centre, Sydney, Australia*, pages 1724–1732. PMLR, 2017.

[3] J. D. Lee, I. Panageas, G. Piliouras, M. Simchowitz, M. I. Jordan, and B. Recht. First-order methods almost always avoid strict saddle points. *Math. Program.*, 176:311–337, 2019.

[4] Yu. Nesterov. *Lectures on convex optimization*. Springer International Publishing, second edition, 2018.

[5] S. J. Wright. Optimization algorithms for data analysis. In A. C. Gilbert M. W. Mahoney, J. C. Duchi, editor, *The mathematics of data*, number 25 in IAS/Park City Mathematics Series. AMS, IAS/Park City Mathematics Institute, and Society for Industrial and Applied Mathematics, Princeton, 2018.

Appendix A

Solutions of the exercises

A.1 Solutions of the exercises for Chapter 2

Solutions of Exercise 1: A strongly convex problem

i) The function f is quadratic in each variable, therefore it is (in particular) $\mathcal{C}^1$ and $\mathcal{C}^2$. Moreover, for any $(\boldsymbol{v}, \boldsymbol{w}) \in (\mathbb{R}^d)^2$, we have

$$\|\nabla f(\boldsymbol{v}) - \nabla f(\boldsymbol{w})\| = \|\boldsymbol{v} - \boldsymbol{w}\| \leq \|\boldsymbol{v} - \boldsymbol{w}\|,$$

therefore the gradient is 1-Lipschitz continuous. Finally, we have $\nabla^2 f(\boldsymbol{w}) = \boldsymbol{I}_d \geq 1 \times \boldsymbol{I}_d$, implying that f is 1-strongly convex.

ii) For a $\mathcal{C}^{1,1}_L$, μ-strongly convex function, the convergence rate of gradient descent is $\mathcal{O}((1-\frac{\mu}{L})^K)$, in that for any $K \geq 1$,

$$f(\boldsymbol{w}_K) - \min_{\boldsymbol{w} \in \mathbb{R}^d} f(\boldsymbol{w}) \leq \left(1 - \frac{\mu}{L}\right)^K \left(f(\boldsymbol{w}_0) - \min_{\boldsymbol{w} \in \mathbb{R}^d} f(\boldsymbol{w})\right).$$

For accelerated gradient, the rate is $\mathcal{O}((1 - \sqrt{\frac{\mu}{L}})^K)$. On our particular problem, we have $\mu = L = 1$: this leads to $f(\boldsymbol{w}_K) - \min_{\boldsymbol{w} \in \mathbb{R}^d} f(\boldsymbol{w}) = 0$ for $K \geq 1$, suggesting that gradient descent would converge in 1 iteration. The same observation can be made for accelerated gradient.

iii) We begin by pointing out that the first iterations of gradient descent and accelerated gradient are identical, and consist in taking a gradient step. With a stepsize of $\frac{1}{L}$, the first iteration is

$$\boldsymbol{w}_1 = \boldsymbol{w}_0 - \frac{1}{L}\nabla f(\boldsymbol{w}_0).$$

In our case, we have $L = 1$ and $\nabla f(\boldsymbol{w}_0) = \boldsymbol{w}0$, hence

$$\boldsymbol{w}_1 = \boldsymbol{w}_0 - \boldsymbol{w}_0 = \mathbf{0}_{\mathbb{R}^d}.$$

As a result, regardless of the choice $\boldsymbol{w}_0$, both gradient descent and accelerated gradient reach the minimum of the problem in one iteration : this confirms the result of the previous question.

Solutions of Exercice 2: Convergence rates

a) Using the Taylor expansion of f, we have

$$\begin{aligned} f\left(\boldsymbol{w}_k - \frac{1}{\mu+L}\nabla f(\boldsymbol{w}_k)\right) &\leq f(\boldsymbol{w}_k) - \frac{1}{\mu+L}\nabla f(\boldsymbol{w}_k)^{\mathrm{T}}\nabla f(\boldsymbol{w}_k) + \frac{L}{2(\mu+L)^2}\|\nabla f(\boldsymbol{w}_k)\|^2 \\ f\left(\boldsymbol{w}_k - \frac{1}{\mu+L}\nabla f(\boldsymbol{w}_k)\right) &\leq f(\boldsymbol{w}_k) + \left(\frac{L}{2(\mu+L)^2} - \frac{1}{\mu+L}\right)\|\nabla f(\boldsymbol{w}_k)\|^2 \\ f\left(\boldsymbol{w}_k - \frac{1}{\mu+L}\nabla f(\boldsymbol{w}_k)\right) &\leq f(\boldsymbol{w}_k) - \frac{2\mu+L}{2(\mu+L)^2}\|\nabla f(\boldsymbol{w}_k)\|^2. \end{aligned} \tag{A.1.1}$$

As $-\frac{2\mu+L}{2(\mu+L)^2} < 0$, the above inequality guarantees a decrease in the function value as long as $\|\nabla f(\boldsymbol{w}_k)\| > 0$. *Note : Proposition 2.1.1 immediately guarantees that latter property, since* $\frac{1}{\mu+L} < \frac{1}{L} < \frac{2}{L}$.

b) For any $k \geq 1$, using (2.1.8) gives $\|\nabla f(\boldsymbol{w}_{k-1})\| \geq 2\mu(f(\boldsymbol{w}_{k-1}) - f^*)$ with $f^* = \min_{\boldsymbol{w}\in\mathbb{R}^d} f(\boldsymbol{w})$. Injecting this inequality into (A.1.1) (applied at iteration $k-1$) gives

$$\begin{aligned} f(\boldsymbol{w}_k) &\leq f(\boldsymbol{w}_{k-1}) - \frac{2\mu+L}{2(\mu+L)^2}\|\nabla f(\boldsymbol{w}_{k-1})\|^2 \\ f(\boldsymbol{w}_k) &\leq f(\boldsymbol{w}_{k-1}) - \frac{\mu(2\mu+L)}{(\mu+L)^2}\left(f(\boldsymbol{w}_{k-1}) - f^*\right) \\ f(\boldsymbol{w}_k) - f^* &\leq \left(1 - \frac{\mu(2\mu+L)}{(\mu+L)^2}\right)\left(f(\boldsymbol{w}_{k-1}) - f^*\right). \end{aligned}$$

A recursive argument then yields

$$f(\boldsymbol{w}_k) - f^* \leq \left(1 - \frac{\mu(2\mu+L)}{(\mu+L)^2}\right)^k (f(\boldsymbol{w}_0) - f^*),$$

which also holds for $k = 0$. As a result, we have established a convergence rate in $\mathcal{O}\left(\left(1 - \frac{\mu(2\mu+L)}{(\mu+L)^2}\right)^k\right)$ for this variant of gradient descent. Note that $\frac{\mu(2\mu+L)}{(\mu+L)^2} < 1$ since

$$(\mu+L)^2 - \mu(2\mu+L) = L^2 - \mu^2 + L\mu > 0,$$

thus no additional assumption on L and μ is required.

c) In Theorem 2.1.2, the obtained rate was $\mathcal{O}\left(\left(1 - \frac{\mu}{L}\right)^k\right)$. Comparing the two ratios, we find that

$$\frac{\mu(2\mu+L)}{(\mu+L)^2} < \frac{\mu}{L},$$

hence the rate is slower for the approach proposed in this exercise, regardless of the value of μ and L. *Note : This can be explained by the fact that the stepsize* $\frac{1}{\mu+L}$ *is smaller than* $\frac{1}{L}$*, and can lead to a smaller decrease in the objective.*

d) The rate established in the previous questions is provably worse than that of gradient descent with stepsize $\frac{1}{L}$: since accelerated gradient achieves the optimal rate in $\mathcal{O}\left(\left(1 - \sqrt{frac\mu L}\right)^k\right)$, provably better than that of gradient descent, the proposed method cannot be better than accelerated gradient in terms of convergence rates.

A.2 Solutions of the exercises for Chapter 3

Solutions of Exercice 3: Saddle points

Consider the two-dimensional function $f : \mathbb{R}^2 \to \mathbb{R}$ defined by

$$f(\boldsymbol{w}) := w_1^3(w_1 - 1) + w_2^4.$$

a) To show that $\mathbf{0}$ is a first-order critical point, it suffices to show that its gradient is the zero vector. Applying the formula gives

$$\nabla f(\mathbf{0}) = \begin{bmatrix} 4 \times 0^3 - 3 \times 0^2 \\ 4 \times 0^3 \end{bmatrix},$$

hence $\mathbf{0}$ is indeed a first-order critical point.

Other critical points are determined by finding solutions of the system of equations $\nabla f(\boldsymbol{w}) = \mathbf{0}$, which corresponds to

$$\begin{cases} 4w_1^3 - 3w_1^2 &= 0 \\ 4w_2^3 &= 0. \end{cases}$$

The second equation immediately gives $w_2 = 0$. The first equation has three solutions, namely $w_1 \in \{0, -\frac{3}{4}, \frac{3}{4}\}$. As a result, the set of first-order critical points is

$$\left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} -\frac{3}{4} \\ 0 \end{bmatrix}, \begin{bmatrix} \frac{3}{4} \\ 0 \end{bmatrix} \right\}.$$

b) If the Hessian matrix of a first-order critical point is positive definite (resp. negative definite), we know that the point is a local minimum (resp. a local maximum). If it is positive semi-definite, the point is also a second-order critical point, but it may be a local minimum, a saddle point or a local maximum (the latter case being possible only when the Hessian matrix is the zero vector).

For the three critical points studied here, we find that

$$\begin{aligned} \nabla^2 f\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}\right) &= \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \\ \nabla^2 f\left(\begin{bmatrix} -\frac{3}{4} \\ 0 \end{bmatrix}\right) &= \begin{bmatrix} 45/4 & 0 \\ 0 & 0 \end{bmatrix} \succeq \mathbf{0} \\ \nabla^2 f\left(\begin{bmatrix} \frac{3}{4} \\ 0 \end{bmatrix}\right) &= \begin{bmatrix} 9/4 & 0 \\ 0 & 0 \end{bmatrix} \succeq \mathbf{0}. \end{aligned}$$

c) For any $\boldsymbol{v} \in \mathbb{R}^2$ such that $v_1 \in (0, 1)$ and $v_2 = 0$, we have :

$$f(\boldsymbol{v}) = v_1^3(v_1 - 1) + v_2^4 = v_1^3(v_1 - 1) < 0.$$

As a result, for any $\epsilon > 0$, the point $[\epsilon\ 0]^{\mathrm{T}}$ gives a better function value than $\mathbf{0}$. This implies that $\mathbf{0}$ cannot be a local minimum. The same reasoning shows that $f(\boldsymbol{u}) > 0$ for any $\boldsymbol{u} \in \mathbb{R}^2$ such that $v_2 \in (-1, 0)$ and $v_2 = 0$, so $\mathbf{0}$ cannot be a local maximum either. We thus conclude that $\mathbf{0}$ is a saddle point of the function.

Solutions of Exercise 4: Gradient descent with line search

a) Suppose that $\|\nabla f(\boldsymbol{w}_k)\| > 0$. By performing Algorithm 2, we seek the largest stepsize α_k such that

$$f(\boldsymbol{w}_k - \alpha_k \nabla f(\boldsymbol{w}_k)) < f(\boldsymbol{w}_k) \tag{A.2.1}$$

among the sequence $\{1, \frac{1}{2}, \frac{1}{4}, \dots\}$.

We know per Proposition 2.1.1 that any choice $\alpha < \frac{2}{L}$ satisfies the condition (A.2.1). As a result, for any value $\alpha = 2^{-j}$ with $j \in \mathbb{N}$ that does not satisfy (A.2.1), we must have

$$2^{-j} \geq \frac{2}{L} \quad \Leftrightarrow \quad j \leq \log_2\left(\frac{L}{2}\right).$$

Since we are testing every value of the form 2^{-j}, there will exist a value for which the inequality will be violated, implying that (A.2.1) will eventually be satisfied. More precisely, after no more than

$$\bar{j} = \left\lceil \log_2\left(\frac{L}{2}\right) \right\rceil$$

iterations of backtracking, the stepsize $\alpha = 2^{-\bar{j}}$ will satisfy $\alpha < \frac{2}{L}$ with $\alpha > \frac{1}{L}$ (otherwise the line-search would have terminated at the previous step of backtracking).

Overall, we have shown that the line-search terminates in a finite number of backtracking iterations (at most $\left\lceil \log_2\left(\frac{L}{2}\right)\right\rceil$) and outputs a stepsize α_k with $\alpha_k \geq \frac{1}{L}$.

b) To adapt the reasoning of Theorem 3.2.1, it suffices to observe that our bound $\alpha_k \geq \frac{1}{L}$ from the previous question leads to

$$f(\boldsymbol{w}_{k+1}) \leq f(\boldsymbol{w}_k) + \left(-\alpha_k + \frac{L}{2}\alpha_k^2\right)\|\nabla f(\boldsymbol{w}_k)\|^2 \leq f(\boldsymbol{w}_k) - \frac{1}{2L}\|\nabla f(\boldsymbol{w}_k)\|^2,$$

since $\alpha \mapsto -\alpha + \frac{L}{2}\alpha^2$ is increasing on $\left[\frac{1}{L}, 1\right]$. The rest of the proof follows by applying verbatim the arguments in the proof of Theorem 3.2.1, from which we obtain the same conclusion: namely, after $K \geq 1$ iterations, we have

$$\min_{0 \leq k \leq K-1} \|\nabla f(\boldsymbol{w}_k)\| \leq \frac{2L(f(\boldsymbol{w}_0) - f_{\text{low}})}{\sqrt{K}}.$$

c) A complexity result is obtained by defining $\epsilon > 0$ and bounding the number of iterations required to produce a point with gradient norm smaller than ϵ. Suppose that this has not happened by iteration $K \geq 1$: in that case, we have

$$\epsilon \leq \min_{0 \leq k \leq K-1} \|\nabla f(\boldsymbol{w}_k)\| \leq \frac{2L(f(\boldsymbol{w}_0) - f_{\text{low}})}{\sqrt{K}},$$

where the last inequality comes from the previous question. This inequality is only valid as long as

$$K \leq \frac{2L(f(\boldsymbol{w}_0) - f_{\text{low}})}{\epsilon^2}.$$

Therefore, the number of iterations needed to produce a point with gradient norm smaller than ϵ is at most [1]

$$\left\lceil \frac{2L(f(\boldsymbol{w}_0) - f_{\text{low}})}{\epsilon^2} \right\rceil .$$

We thus say that the complexity of gradient descent on nonconvex problems is $\mathcal{O}(\epsilon^{-2})$.

d) In the worst-case, every iteration requires $\left\lceil \log_2 \left(\frac{L}{2}\right) \right\rceil$ backtracking iterations/function evaluations, while there can be at most

$$\left\lceil \frac{2L(f(\boldsymbol{w}_0) - f_{\text{low}})}{\epsilon^2} \right\rceil$$

iterations. Overall, the total number of function evaluations is bounded above by

$$\left\lceil \log_2 \left(\frac{L}{2}\right) \right\rceil \times \left\lceil \frac{2L(f(\boldsymbol{w}_0) - f_{\text{low}})}{\epsilon^2} \right\rceil .$$

The cost of the line-search approach in terms of number of function evaluations is thus increased compared to that of the constant stepsize technique. *Note : The line-search approach does not require knowledge of* L.

[1] If we reach this number of iterations, then necessarily $\|\nabla f(\boldsymbol{w}_K)\| < \epsilon$.