Differentiable Optimisation: Theory and Algorithms – Part II: Algorithms

Andrea Simonetto

INSTITUT POLYTECHNIQUE DE PARIS

ENSTA

IP PARIS

OPT202 — Lecture Notes, AA 2023 / 2024

Differentiable Optimisation: Theory and Algorithms

Part II: Algorithms

Lecture notes, version at December 1, 2023

Andrea Simonetto

Forewords

Context and Aim

This course follows naturally OPT201, which covers the theory part of continuous optimisation. OPT201 focuses on optimality conditions, convexity, and duality. In OPT202, we will look at how to use these notions to build algorithms that *solve* the problems.

In particular, **the aim of the course is to be able to answer the questions,**

1. Given an optimisation problem, which algorithm do I use to solve it?
2. Which properties and theoretical guarantees does the algorithm that I have chosen have?
3. Conversely, if I want to use a certain algorithm, which characteristics does the optimisation problem need to have?

In order to answer to these three questions, we will need to build a theory of algorithms, and ultimately understand what we really mean by *solving an optimisation problem.*

Note: The sections or subsections marked with * contain optional advanced material, which is not covered in class.

References

We will use standard references in convex optimisation and algorithms, which you can refer to.

[YN] Y. Nesterov, *Introductory Lectures on Convex Programming*, Kluiver, 2004

[BV] S. Boyd, L. Vandenberghe, *Convex Optimization*, Cambridge, 2004

I will also point out a few other ones, chapter by chapter.

Palaiseau, December 1, 2023.

Contents

Chapter 1

First lecture

Introduction and unconstrained optimisation (I)

1.1 Introduction

In this course, we look at generic optimisation problems of the form

$$\begin{aligned} (\mathsf{P}) \quad \underset{\boldsymbol{x}\in X\subseteq\mathbf{R}^n}{\text{minimise}} \quad & f(\boldsymbol{x}) && (1.1)\\ \text{subject to} \quad & g(\boldsymbol{x}) \leqslant 0 && (1.2)\\ & h(\boldsymbol{x}) = 0, && (1.3) \end{aligned}$$

where the functions $f(\boldsymbol{x}) : \mathbf{R}^n \to \mathbf{R}$, $g(\boldsymbol{x}) : \mathbf{R}^n \to \mathbf{R}^p$, $h(\boldsymbol{x}) : \mathbf{R}^n \to \mathbf{R}^q$. In addition, X represents a generic closed set. As it appears, we look at continuous optimisation problems for which $\boldsymbol{x} \in \mathbf{R}^n$. We indicate with $\boldsymbol{x}^*$ a (global) optimizer of the original problem, and by $f^* = f(\boldsymbol{x}^*)$ the unique (global) minimum. We also indicate the Euclidean inner product as $\langle v, u\rangle = v^\top u = u^\top v$ for two vectors $u, v \in \mathbf{R}^n$.

As we go along, we will answer the following questions:

- How do we solve the above? And what do we mean by solving?
- Can we set up **algorithms** to solve the above?
- Can we distinguish easy and hard problems in the algorithmic perspective, beyond the general divide between convex and nonconvex problems?

It is very non trivial to ask oneself the question: *what do we mean by solving optimisation problems?*. Surely, on one side, you may remember that continuous optimisation problems may come with conditions that optimal points need to verify. These are the so-called KKT conditions. We will recall what they are in due course, but for now think of them as a system of non-linear equations and inequalities.

On the other side, even equipped with these KKT, how can we solve for the optimal points when we have thousands, or millions variables? What is the computational complexity associated with this solution?

As we can already see, we need to build some theory around the notion of algorithm to be able to characterise its complexity and the its inner workings. This is what we will do next.

1.2 Oracles, algorithms, and an impossibility theorem

We start by defining the main players of our new theory of algorithms. For a given optimisation problem, we denote its class as $\mathcal{F}$. The class is the collection of properties that that problem has. For example, $\mathcal{F}$ can represent the class of convex problems.

We then define an oracle, denoted by $\mathcal{O}$, as a computational entity that can generate answers to our questions. An example is a gradient oracle, which can generate the value of the gradient of a function $\nabla_{\boldsymbol{x}} f(\boldsymbol{x})$, whenever we give it $\boldsymbol{x}$.

And finally, we denote by $\mathcal{M}$ the method, that is the algorithm that we are going to employ to solve our optimisation problem.

Definition 1.1 (Performance) *We call performance or* analytical complexity *of a method $\mathcal{M}$ for problem class $\mathcal{F}$ using the oracle $\mathcal{O}$, the number of calls to the oracle to find an approximate solution to $\mathcal{F}$ with an accuracy $\epsilon > 0$.*

The definition may be not very formal, but it will give us a clear direction for our theory. In particular we are interested in knowing how many iterations (i.e., how long do we have to wait) to obtain an approximate solution for a given optimisation problem. This question is both extremely pragmatic and practical: you want to know if you have to wait one minute, or three days. But the question is also very theoretical: we want to find theoretical results that are valid for whole problem classes.

So, how does a typical method look like? Well, a typical method is an iterative algorithm, which start with an initial guess $\boldsymbol{x}_0$, and for all subsequent times $k \in \mathbf{N}$, does:

1. Calls the oracle $\mathcal{O}$ at $\boldsymbol{x}_k$;
2. Applies the method $\mathcal{M}$ to the information up to time k and form the new test point $\boldsymbol{x}_{k+1}$;
3. Stops if a stopping criterion is verified (the accuracy is reached), otherwise $k \leftarrow k + 1$.

Let's give some examples to fix the ideas on these new notions.

Example 1.1 *The quadratic problem*

$$\underset{\boldsymbol{x}\in\mathbf{R}^n}{\text{minimise}} \ \frac{1}{2}\boldsymbol{x}^\mathsf{T}\mathbf{Q}\boldsymbol{x} + \boldsymbol{c}^\mathsf{T}\boldsymbol{x}, \tag{1.4}$$

has optimality conditions: $\mathbf{Q}\boldsymbol{x} = -\boldsymbol{c}$. If $\mathbf{Q}$ is positive semidefinite, these conditions are necessary and sufficient. Otherwise just necessary.

Suppose $\mathbf{Q}$ is positive semidefinite. Then to find an optimal point $\boldsymbol{x}^*$, we could use a linear system solver. In this case we need an $\mathcal{O}$ that delivers gradient and Hessian information, and we use the linear solver as a method $\mathcal{M}$. Which linear solvers do you know and which performance do they have?

Example 1.2 *The generic differentiable problem*

$$\underset{\boldsymbol{x}\in\mathbf{R}^n}{\text{minimise}} \ f(\boldsymbol{x}), \tag{1.5}$$

has necessary optimality conditions: $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = 0$.

We could use the gradient method with stepsize $\alpha_k > 0$

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \nabla_{\boldsymbol{x}} f(\boldsymbol{x}_k), \tag{1.6}$$

as a method $\mathcal{M}$, with a gradient oracle $\mathcal{O}$, and a stopping criterion $\|\nabla_{\boldsymbol{x}} f(\boldsymbol{x})\| \leqslant \epsilon$.

Let's get back to the development of our theory. We remind that the only information available for the method $\mathcal{M}$ are the answers of the oracle $\mathcal{O}$. We distinguish, zero-order oracles, if they only answer the value of the cost $f(\boldsymbol{x})$; first-order oracle, if they only answer the value $f(\boldsymbol{x})$ and the gradient $\nabla_{\boldsymbol{x}} f(\boldsymbol{x})$; and second-order oracles, if they answer the value $f(\boldsymbol{x})$, the gradient $\nabla_{\boldsymbol{x}} f(\boldsymbol{x})$ and the Hessian $\nabla^2_{\boldsymbol{x}\boldsymbol{x}} f(\boldsymbol{x})$.

In this context, the oracle of the first example was a second-order oracle, while the one of the second example was a first-order oracle.

Let us consider now a concrete optimisation problem and let us try to apply the formal language that we have introduced. Consider,

$$\text{(P')} \qquad \underset{\boldsymbol{x}\in[0,1]^n}{\text{minimise}} \ f(\boldsymbol{x}). \tag{1.7}$$

If $f(\boldsymbol{x})$ is defined by a complicated procedure, you may have only access to a zero-order oracle $\mathcal{O}$. This is typically the case if $f(\boldsymbol{x})$ is generated by a simulation.

Suppose $f(\boldsymbol{x})$ is of a particular Lipschitz class, for which there exists a scalar $L > 0$ that:

$$\forall \boldsymbol{x}, \boldsymbol{y} \in [0,1]^n : \quad |f(\boldsymbol{x}) - f(\boldsymbol{y})| \leqslant L\|\boldsymbol{x} - \boldsymbol{y}\|_\infty.$$

Lipschitz classes will play a big role in the following, but for now just accept the definition.

Remember that an accuracy ϵ means finding a point $\bar{\boldsymbol{x}}$ for which,

$$|f(\bar{\boldsymbol{x}}) - f(\boldsymbol{x}^*)| \leqslant \epsilon,$$

or, since we know that by optimality $f(\bar{\boldsymbol{x}}) - f(\boldsymbol{x}^*) \geqslant 0$, then we can also just focus on $f(\bar{\boldsymbol{x}}) - f(\boldsymbol{x}^*) \leqslant \epsilon$.

So, let's try a zero-order oracle $\mathcal{O}$ on problem (P') with functional class above.

The only method $\mathcal{M}$ to obtain the accuracy we require is a gridding of the feasible space. In particular, a uniform gridding. It obtains an accuracy ϵ if $\|\boldsymbol{x} - \boldsymbol{y}\|_\infty \leqslant \frac{\epsilon}{L}$, since

$$f(\bar{\boldsymbol{x}}) - f(\boldsymbol{x}^*) = |f(\bar{\boldsymbol{x}}) - f(\boldsymbol{x}^*)| \leqslant L\|\boldsymbol{x} - \boldsymbol{y}\|_\infty,$$

which implies that we need $\lceil \frac{L}{\epsilon} \rceil^n$ grid points. This brings us to the first key message of this course.

Theorem 1.1 (Informal, adapted from Nesterov) *General optimisation problems are unsolvable.*

Proof. *The problem we just saw is an embodiment of a generic optimisation problem.*

To reach an accuracy ϵ, you need $\lceil \frac{L}{\epsilon} \rceil^n$ grid points.

For a very simple problem $L = 4, n = 10, \epsilon = 0.01$, you need $> 10^{26}$ calls of the oracle. Even with very performant machines you can do 10^{14} calls per second. Then you need 10^{12} seconds and therefore 32000 years. ♣

This is important, most of the problem you will write, they will be unsolvable. optimisation is hard.

In this course, we will see how to we solve problems with $n \sim 10^3 - 10^6$ and $\epsilon \sim 10^{-6}$. These problems will not be the most general ones, but they will cover a good portion of the relevant problems for society. We will see how the structure and class of the problem will play an essential role, and not just its convexity.

We start by looking at unconstrained optimisation.

1.3 Unconstrained optimisation

1.3.1 Setting

We consider now the unconstrained problem

$$\mathsf{(PU)} \qquad \underset{\boldsymbol{x} \in \mathbf{R}^n}{\text{minimise}} \; f(\boldsymbol{x}), \tag{1.8}$$

where the cost function $f : \mathbf{R}^n \to \mathbf{R}$ is not necessarily convex or differentiable, unless specified. We indicate by $\mathcal{C}^k(\mathbf{R}^n)$ the class of functions in $\mathbf{R}^n$ that are at least k-times differentiable.

For Problem (PU), we already know that,

- If $f \in \mathcal{C}^1(\mathbf{R}^n)$, then $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = 0$ is a necessary condition for optimality;
- If $f \in \mathcal{C}^1(\mathbf{R}^n)$ and f is convex, then $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = 0$ is a necessary and sufficient condition for optimality.

These two points are very important. Convexity defines the landscape but it does not necessarily says how hard is to solve for $\boldsymbol{x}^*$. In particular, when we have convexity, we have conditions that, if we are able to solve, deliver the optimal points. When we don't have convexity, we don't have even that!

In general, for nonconvex problems, we are happy to find stationary points.

1.3.2 Lipschitz conditions

It turns out that convexity is not the only important element for the theory of algorithms. Lipschitz continuity plays an equal (or even bigger) role too. Let us define it properly.

Definition 1.2 *We label with $\mathcal{C}_L^{k,p}(\mathbf{R}^n)$, $p \leqslant k$, a k-differentiable function which has its p derivative L-Lipschitz continuous, i.e.,*

$$\forall \boldsymbol{x}, \boldsymbol{y} : \|\nabla^p f(\boldsymbol{x}) - \nabla^p f(\boldsymbol{y})\| \leqslant L\|\boldsymbol{x} - \boldsymbol{y}\|.$$

Lipschitz continuity has a very intuitive geometric interpretation as a condition on growth, as we will see shortly.

An important class for us will be the class $\mathcal{C}_L^{1,1}$, that is the functions that are at least once differentiable and whose gradient is Lipschitz continuous. For these, we have the following result.

Lemma 1.2 (YN, Lemma 1.2.3) *Let $f \in \mathcal{C}_L^{1,1}$. Then for any $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{R}^n$ we have:*

$$|f(\boldsymbol{y}) - f(\boldsymbol{x}) - \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x}\rangle| \leqslant \frac{L}{2}\|\boldsymbol{y} - \boldsymbol{x}\|^2.$$

Proof. *For all $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{R}^n$, we have,*

$$f(\boldsymbol{y}) = f(\boldsymbol{x}) + \int_0^1 \langle \nabla f(\boldsymbol{x} + \tau(\boldsymbol{y} - \boldsymbol{x})), \boldsymbol{y} - \boldsymbol{x}\rangle \mathrm{d}\tau.$$

Hence, we can obtain the following chain of relations,

$$\begin{aligned}
|f(\boldsymbol{y}) - f(\boldsymbol{x}) - \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x}\rangle| &= \left| \int_0^1 \langle \nabla f(\boldsymbol{x} + \tau(\boldsymbol{y} - \boldsymbol{x})) - \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x}\rangle \,\mathrm{d}\tau \right| \\
&\leqslant \int_0^1 |\langle \nabla f(\boldsymbol{x} + \tau(\boldsymbol{y} - \boldsymbol{x})) - \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x}\rangle| \,\mathrm{d}\tau \\
\textit{by Cauchy–Schwarz} \quad &\leqslant \int_0^1 \|\nabla f(\boldsymbol{x} + \tau(\boldsymbol{y} - \boldsymbol{x})) - \nabla f(\boldsymbol{x})\| \|\boldsymbol{y} - \boldsymbol{x}\| \mathrm{d}\tau \\
\textit{by Lipschitz} \quad &\leqslant \int_0^1 \tau \mathrm{d}\tau L\|\boldsymbol{y} - \boldsymbol{x}\|^2 = \frac{L}{2}\|\boldsymbol{y} - \boldsymbol{x}\|^2,
\end{aligned}$$

from which the thesis follows. ♣

Geometrically, Lemma 1.2 means the following. Consider a function $f \in \mathcal{C}_L^{1,1}$. Let us fix some $x_0 \in \mathbf{R}^n$ and form two quadratic functions

$$\varphi_1(\boldsymbol{x}) = f(\boldsymbol{x}_0) + \langle \nabla f(\boldsymbol{x}_0), \boldsymbol{x} - \boldsymbol{x}_0\rangle + \frac{L}{2}\|\boldsymbol{x} - \boldsymbol{x}_0\|^2 \tag{1.9}$$

$$\varphi_2(\boldsymbol{x}) = f(\boldsymbol{x}_0) + \langle \nabla f(\boldsymbol{x}_0), \boldsymbol{x} - \boldsymbol{x}_0\rangle - \frac{L}{2}\|\boldsymbol{x} - \boldsymbol{x}_0\|^2 \tag{1.10}$$

Then, the graph of the function f is located between the graphs of $\varphi_1(\boldsymbol{x})$ and $\varphi_2(\boldsymbol{x})$.

$$\varphi_1(\boldsymbol{x}) \geqslant f(\boldsymbol{x}) \geqslant \varphi_2(\boldsymbol{x}), \quad \forall \boldsymbol{x} \in \mathbf{R}^n.$$

This is exactly the condition of growth that we were talking about, which will make the algorithms easier or more difficult: we know upper and lower bounds on how the function changes. And these bounds are easy quadratic functions. This will be very handy.

Let's dig a bit deeper in Lipschitz conditions, since we are on the right track. We now look at the Hessian of a function $f \in \mathcal{C}_L^{2,1}(\mathbf{R}^n)$, which we indicated with $\nabla^2 f(\boldsymbol{x})$. For real-valued symmetric matrices like the Hessian, we can defined an induced Euclidean norm, for which we know that,

$$\|\nabla^2 f(\boldsymbol{x})\| \leqslant L \iff \max_i |\mathrm{eig}(\nabla^2 f(\boldsymbol{x}))| \leqslant L \iff -LI_n \preceq \nabla^2 f(\boldsymbol{x}) \preceq LI_n.$$

This chain of relations needs some explanations. Imposing a bound on the norm (left-side) is equivalent to impose bounds on the absolute values of the eigenvalues (eig) of the matrix (middle), which is equivalent to say that the matrix is bounded with conic inequalities $\preceq$ (right-side).

In particular, we say that $A \succeq B$ for two squared symmetric matrices, if and only if $A - B \succeq 0$, that is, the matrix $A - B$ is positive semidefinite.

We now have the following result.

Lemma 1.3 *Function f belongs to $\mathcal{C}_L^{2,1}(\mathbf{R}^n)$ if and only if*

$$\|\nabla^2 f(\boldsymbol{x})\| \leqslant L, \quad \forall x \in \mathbf{R}^n,$$

where $\nabla^2 f(\boldsymbol{x})$ is the Hessian of f.

Proof. *Homework. (Try also the generic $\mathcal{C}_L^{p+1,p}(\mathbf{R}^n)$).* ♣

Lemma 1.3 is not very surprising and it is quite intuitive, given the quadratic growth condition we have established just before in Lemma 1.2.

1.4 The gradient method

We are ready to formulate and analyse our first algorithm: the gradient method. We recall that the gradient of a function is an ascent direction, so taking the opposite will bring us towards lower function values.

The **simplest gradient descent method** would use a stepsize $\alpha_k > 0$, and,

Gradient descent method

- *Start with* $\boldsymbol{x}_0 \in \mathbf{R}^n$
- *Iterate* $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k), \quad k = 0, 1, \ldots.$

There are different methods to choose the stepsize α_k (either a priori or online):

- Constant: $\alpha_k = \alpha$
- Vanishing, for example: $\alpha_k = \frac{\alpha}{\sqrt{k+1}}$
- Fully optimised: $\alpha_k = \arg\min_{\alpha \geqslant 0} f(\boldsymbol{x}_k - \alpha \nabla f(\boldsymbol{x}_k))$
- Backtracking: set $\alpha_k = 1$, $\tau \in (0,1)$, $\beta \in (0, 1/2)$:

  While $f(\boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k)) > f(\boldsymbol{x}_k) - \beta \alpha_k \|\nabla f(\boldsymbol{x}_k)\|^2$, set: $\alpha_k \leftarrow \tau \alpha_k$.

Different choices of stepsize will give rise to different properties of the gradient method. Some will not work in general, and some (like the fully optimised one) will be too expensive computationally. One that works well in most scenarios is the backtracking strategy. In this solution, we start with a large α and we refine it, so to guarantee at least a $\beta \alpha_k \|\nabla f(\boldsymbol{x}_k)\|^2$ reduction in cost.

What about convergence? And what is convergence anyway?

Now that we have defined an algorithm, it is important to talk about convergence, convergence rate, and ultimately performance. We know that performance means how many iterations we need to reach a certain accuracy. But what about convergence? An algorithm, like the gradient method, will deliver a sequence of points $\{\boldsymbol{x}_k\}_{k \in \mathbf{N}}$, and it is natural to ask whether this sequence

will converge to a limit point $\bar{\boldsymbol{x}}$ (possibly an optimal point). Then, it is also natural to ask about how fast this convergence goes (at least asymptotically), meaning evaluating the ratio,

$$\frac{\|\boldsymbol{x}_k - \bar{\boldsymbol{x}}\|}{\|\boldsymbol{x}_{k-1} - \bar{\boldsymbol{x}}\|},$$

or related metrics.

Let's characterise these notions for the gradient method.

Theorem 1.4 *Assuming f bounded below, the* **gradient descent method** $\mathcal{M}$ *on unconstrained problems with cost $f \in \mathcal{C}^1(\mathbf{R}^n)$ with fully optimized α_k will generate a sequence $\{\boldsymbol{x}_k\}$ that converges to a stationary point $\bar{\boldsymbol{x}}$, so that $\nabla f(\bar{\boldsymbol{x}}) = 0$.*

Proof. *We have $f(\boldsymbol{x}_{k+1}) = f(\boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k)) < f(\boldsymbol{x}_k)$, with equality only if $\nabla f(\boldsymbol{x}_k) = 0$, and since f is bounded below $f(\boldsymbol{x}_k)$ reaches a limit point for which $\nabla f(\boldsymbol{x}_k) = 0$. So the sequence $\{\boldsymbol{x}_k\} \to \bar{\boldsymbol{x}}$.* ♣

This is hardly satisfactory: typically the stepsize will vanish, i.e., $\alpha_k \to 0$, which makes the overall method very slow. Also it seems we don't have a notion of performance readily available, and not even a convergence rate result.

This is of course the worst scenario. You don't want to be in this scenario. However, in a lot of cases, for example, when dealing with training neural networks with non-differentiable activation functions (like ReLu), then you are in this case. Take it as warning: when training non-smooth machine learning models, you can wait for a long time for the result, and it can be just a stationary point, which may be quite bad.

Can we do something **much** better? Let's consider Lipschitz functions.

Theorem 1.5 *Assuming f bounded below, the* **gradient descent method** $\mathcal{M}$ *on unconstrained problems with cost $f \in \mathcal{C}^{1,1}_L(\mathbf{R}^n)$ with backtracking α_k will generate a sequence $\{\boldsymbol{x}_k\}$ that converges to a stationary point as*

$$\min_{k=0,\ldots,t} \|\nabla f(\boldsymbol{x}_k)\| \leqslant \frac{1}{\sqrt{t+1}} \sqrt{\frac{L}{\tau\beta}(f(\boldsymbol{x}_0) - f^*)} \quad . \tag{1.11}$$

Proof. *Use Lipschitz and the fact that $\boldsymbol{x}_{k+1} - \boldsymbol{x}_k = -\alpha_k \nabla f(\boldsymbol{x}_k)$:*

$$\begin{aligned} f(\boldsymbol{x}_{k+1}) &\leqslant f(\boldsymbol{x}_k) + \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x}_{k+1} - \boldsymbol{x}_k \rangle + \frac{L}{2}\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\|^2 & (1.12)\\ &\leqslant f(\boldsymbol{x}_k) - \alpha_k \left(1 - \frac{\alpha_k}{2} L\right) \|\nabla f(\boldsymbol{x}_k)\|^2, & (1.13) \end{aligned}$$

for which we know we need $\alpha_k < 2/L$. Which guarantees convergence to f^*.

We could stop here, but we want the rate. For backtracking line search, we have,

$$f(\boldsymbol{x}_{k+1}) \leqslant f(\boldsymbol{x}_k) - \alpha_k \beta \|\nabla f(\boldsymbol{x}_k)\|^2. \tag{1.14}$$

Furthermore $\alpha_k = 1/L \leqslant 1$ is a valid choice since $\beta < 1/2$, so at least $\alpha_k \geqslant \frac{\tau}{L}$.

As such,

$$f(\boldsymbol{x}_{k+1}) \leqslant f(\boldsymbol{x}_k) - \alpha_k \beta \|\nabla f(\boldsymbol{x}_k)\|^2 \leqslant f(\boldsymbol{x}_k) - \frac{\tau}{L}\beta \|\nabla f(\boldsymbol{x}_k)\|^2. \tag{1.15}$$

Now, summing over $k = 0, \ldots, t$:

$$\sum_{k=0}^{t} \frac{\tau}{L}\beta \|\nabla f(\boldsymbol{x}_k)\|^2 \leqslant \sum_{k=0}^{t} f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1}) = f(\boldsymbol{x}_0) - f(\boldsymbol{x}_{t+1}) \leqslant f(\boldsymbol{x}_0) - f^*, \tag{1.16}$$

and finally, since $\min_k \|\nabla f(\boldsymbol{x}_k)\|^2 \leqslant \frac{1}{t+1}\sum_{k=0}^{t} \|\nabla f(\boldsymbol{x}_k)\|^2$, the thesis follows. ♣

What is the meaning of this result? We have obtained a global (starting from any point $\boldsymbol{x}_0$) convergence in terms of $\|\nabla f(\boldsymbol{x})\|$. The result implies that the sequence $\{\min \|\nabla f(\boldsymbol{x})\|\}$ converges to 0 with a certain rate.

In general for nonconvex problems it is hard to obtain global convergence in terms of objective $f(\boldsymbol{x}_k) - f^*$, or distance $\|\boldsymbol{x}_k - \boldsymbol{x}^*\|$.

In addition, by using an first-order oracle $\mathcal{O}$ on method $\mathcal{M}$ (gradient), for problem (PU) of class $\mathcal{C}_L^{1,1}$, we have obtain a **convergence rate** estimate of $O(\frac{1}{\sqrt{t+1}})$.

This means that to obtain an accuracy $\|\nabla f(\boldsymbol{x})\| \leqslant \epsilon$, we need

$$t \geqslant (f(\boldsymbol{x}_0) - f^*)\frac{L}{\tau\beta\epsilon^2} = O\left(\frac{1}{\epsilon^2}\right) \tag{1.17}$$

calls to our oracle. This is the result we wanted, a performance result, sometimes referred to as a performance certificate. This is also good: it does not depend on the problem dimension n. Look back at Theorem 1.1 and the impossibility result: that one depended on the problem dimension.

Before continuing, it is good to introduce some general notation and definitions. In general, we call results as in Eq. (1.11) as convergence certificates.

Formally, we define,

Definition 1.3 (Convergence certificate) *A convergence certificate is an inequality that upper bounds a convergence metric as a function of the number of iterations.*

Definition 1.4 (Local and global convergence) *A convergence is local if it is achieved starting close enough to the limit point. A convergence is global if it is achieved starting from any point.*

Definition 1.5 (Big-O, little-o notation) *Informally, the notation $O(\cdot)$ indicates the asymptotical behaviour (for big t's) up to irrelevant constants. Formally, for two functions $f, g : \mathbf{R} \to \mathbf{R}$ with $g(t) > 0$ for all t, we say $f = O(g)$ iff, $\lim_{t\to\infty} \frac{|f(t)|}{g(t)} = \text{const}$.*

Similarly, we define the notation $o(\cdot)$, and we say that $f = o(g)$, if g grows much faster than f asymptotically, so that, for two functions $f, g : \mathbf{R} \to \mathbf{R}$ with $g(t) > 0$ for all t, we say $f = o(g)$ iff, $\lim_{t\to\infty} \frac{f(t)}{g(t)} = 0$.

So, in Eq. (1.11), the metric is $\min_k \|\nabla f(\boldsymbol{x}_k)\|$, the number of iterations is t, and the upper bound is $O(1/\sqrt{t+1})$.

Let us get back to the convergence certificate of Eq. (1.11). In general a rate of $O(\frac{1}{\sqrt{t}})$ is not great. We can do better if we look at functions, for which around an optimal point, the following extra property is verified,

$$mI_n \preceq \nabla^2 f(\boldsymbol{x}) \preceq LI_n, \qquad \forall \boldsymbol{x} : \|\boldsymbol{x} - \boldsymbol{x}^*\| \leqslant R, \tag{1.18}$$

for a given $R > 0$.

Note that the left inequality is true for strongly convex functions for all $\boldsymbol{x}$. The right inequality follows from $f \in \mathcal{C}_L^{2,1}(\mathbf{R}^n)$, so redundant. Here however we do not talk about convex functions, but only locally strongly convex ones.

We now have the result we wanted.

Theorem 1.6 *The gradient descent method $\mathcal{M}$ on unconstrained problems, with constant stepsize $\alpha < 2/L$, for cost functions $f \in \mathcal{C}_L^{2,1}(\mathbf{R}^n)$ verifying the property (1.18) above has a local (starting at $\|\boldsymbol{x}_0 - \boldsymbol{x}^*\| \leqslant R$) convergence certificate of*

$$\|\boldsymbol{x}_t - \boldsymbol{x}^*\| \leqslant \rho^t \|\boldsymbol{x}_0 - \boldsymbol{x}^*\| \leqslant \rho^t R.$$

for $\rho = \max\{|1 - \alpha m|, |1 - \alpha L|\} < 1$, and t iterations.

Proof. *Local means starting close enough to a local minimizer, so pick* $\|\boldsymbol{x}_0 - \boldsymbol{x}^*\| \leqslant R$. *For this,*

$$\begin{aligned} \|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\| &= \|\boldsymbol{x}_k - \alpha\nabla f(\boldsymbol{x}_k) - \boldsymbol{x}^*\| && (1.19)\\ &= \|\boldsymbol{x}_k - \alpha\nabla f(\boldsymbol{x}_k) - \boldsymbol{x}^* - \alpha\nabla f(\boldsymbol{x}^*)\| && (1.20)\end{aligned}$$

Set $g(\boldsymbol{x}) := \boldsymbol{x} - \alpha\nabla f(\boldsymbol{x})$, *for* $\|\boldsymbol{x}_k - \boldsymbol{x}^*\| \leqslant R$ *property* (1.18) *is valid and* $\|\nabla g(\boldsymbol{x})\| \leqslant \rho$.

In fact,

$$\begin{aligned}\|\nabla g(\boldsymbol{x})\| = \|I - \alpha\nabla^2 f(\boldsymbol{x})\| = \max|\mathrm{eig}(I - \alpha\nabla^2 f(\boldsymbol{x}))| = \\ = \max|1 - \alpha\mathrm{eig}(\nabla^2 f(\boldsymbol{x}))| \leqslant \max\{|1 - \alpha m|, |1 - \alpha L|\}. \quad (1.21)\end{aligned}$$

For Lemma 1.3 we have also that,

$$\|g(\boldsymbol{x}) - g(\boldsymbol{y})\| \leqslant \rho\|\boldsymbol{x} - \boldsymbol{y}\|,$$

for which, $\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\| \leqslant \rho\|\boldsymbol{x}_k - \boldsymbol{x}^*\|$. *The choice* $\alpha < 2/L$ *guarantees that* $\rho < 1$ *and therefore, once in* $\mathcal{R}(\boldsymbol{x}) := \{\boldsymbol{x} | \|\boldsymbol{x} - \boldsymbol{x}^*\| \leqslant R\}$ *we stay in* $\mathcal{R}(\boldsymbol{x})$. *So then, we can iterate the above inequality for* t *iterations, and obtain the claim.* ♣

Let us analyse the result. We have a local convergence for special functions in terms of $\|\boldsymbol{x}_k - \boldsymbol{x}^*\|$ with constant stepsize. The convergence rate is exponential, or as we say in optimisation: it is a linear convergence. This nomenclature is due to the fact that in this case,

$$\lim_{k\to\infty} \frac{\|\boldsymbol{x}_k - \boldsymbol{x}^*\|}{\|\boldsymbol{x}_{k-1} - \boldsymbol{x}^*\|} = \rho,$$

so the ratio is constant.

In addition, for an accuracy $\|\boldsymbol{x}_t - \boldsymbol{x}^*\| \leqslant \epsilon$, we need

$$t \geqslant \frac{\log R + \log(1/\epsilon)}{\log(1/\rho)} = O\left(\log\left(\frac{1}{\epsilon}\right)\right)$$

calls of the oracle.

We can pick the best $\alpha = 2/(m + L)$, leading to the best $\rho = \frac{L-m}{L+m} = \frac{\kappa-1}{\kappa+1}$. The number κ is the condition number $\kappa := L/m$, and for badly conditioned functions (κ large), the gradient will not work well, since $\rho \approx 1$, and $t \to \infty$.

A linear convergence is much better than a $O(1/\sqrt{t})$ convergence. And it also turns out this is the fastest convergence result we can expect for gradient methods (we will look at these type of claims in the second class).

However, if we allow ourself the chance to use second-order oracles, we may do better.

1.5 Newton's method

We introduce now the Newton's method for functions at least in $\mathcal{C}^2(\mathbf{R}^n)$, which is a second-order (oracle) method.

Newton's method

- *Start with* $\boldsymbol{x}_0 \in \mathbf{R}^n$
- *Iterate: solve the linear system* $\nabla^2 f(\boldsymbol{x}_k)\boldsymbol{d}_k = \nabla f(\boldsymbol{x}_k)$, $\quad \boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \boldsymbol{d}_k$, $\quad k = 0, 1, \ldots$.

The method involves solving a linear system of equations

$$\nabla^2 f(\boldsymbol{x}_k)\boldsymbol{d}_k = \nabla f(\boldsymbol{x}_k),$$

where the unknown is the Newton's step $\boldsymbol{d}_k$. Different methods will be generated by different choices of how you solve the linear system. We recall that the worst-case complexity of solving a quadratic linear system of n dimension is $O(n^3)$.

In the most naive (just in theory) version, we can set,

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - [\nabla^2 f(\boldsymbol{x}_k)]^{-1} \nabla f(\boldsymbol{x}_k).$$

The Newton's method is more computationally complex per iteration than the gradient method, which performs only vector additions, but what can we say in terms of convergence, and performance?

Consider functions $f \in \mathcal{C}^{2,2}_M(\mathbf{R}^n)$ with the additional property that

$$\nabla^2 f(\boldsymbol{x}) \succeq m I_n, \qquad \forall \boldsymbol{x} : \|\boldsymbol{x} - \boldsymbol{x}^*\| \leqslant R, \tag{1.22}$$

for a local region $R > 0$. Then we have the following theorem.

Theorem 1.7 The Newton method *applied to unconstrained problems, for cost functions $f \in \mathcal{C}^{2,2}_M(\mathbf{R}^n)$ verifying property (1.22) above, for any $R > 2m/M$, and starting in $\|\boldsymbol{x}_0 - \boldsymbol{x}^*\| < \min\{1, 2m/M\}$ has a local convergence certificate of*

$$\frac{M}{2m}\|\boldsymbol{x}_t - \boldsymbol{x}^*\| \leqslant \left(\frac{M}{2m}\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|\right)^{2^t} \leqslant C^{2^t},$$

when run for t iterations.

Proof. *Consider the update,*

$$\begin{aligned} r_{k+1} := \|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\| &= \|\boldsymbol{x}_k - \boldsymbol{x}^* - [\nabla^2 f(\boldsymbol{x}_k)]^{-1} \nabla f(\boldsymbol{x}_k)\| \\ &= \|\boldsymbol{x}_k - \boldsymbol{x}^* - [\nabla^2 f(\boldsymbol{x}_k)]^{-1} \int_0^1 \nabla^2 f(\boldsymbol{x}^* + \tau(\boldsymbol{x}_k - \boldsymbol{x}^*))(\boldsymbol{x}_k - \boldsymbol{x}^*) \mathrm{d}\tau\| \\ &= \|[\nabla^2 f(\boldsymbol{x}_k)]^{-1} \times \int_0^1 [\nabla^2 f(\boldsymbol{x}_k) - \nabla^2 f(\boldsymbol{x}^* + \tau(\boldsymbol{x}_k - \boldsymbol{x}^*))] \mathrm{d}\tau \, (\boldsymbol{x}_k - \boldsymbol{x}^*)\| \end{aligned}$$

The first line is due to:

$$\nabla f(\boldsymbol{x}_k) = \nabla f(\boldsymbol{x}^*) + \int_0^1 \nabla^2 f(\boldsymbol{x}^* + \tau(\boldsymbol{x}_k - \boldsymbol{x}^*))(\boldsymbol{x}_k - \boldsymbol{x}^*) \mathrm{d}\tau.$$

The second line is due to:

$$\boldsymbol{x}_k - \boldsymbol{x}^* = [\nabla^2 f(\boldsymbol{x}_k)]^{-1} [\nabla^2 f(\boldsymbol{x}_k)](\boldsymbol{x}_k - \boldsymbol{x}^*).$$

Use now Cauchy–Schwarz, the property $f \in \mathcal{C}^{2,2}_M(\mathbf{R}^n)$, and the local property $\nabla^2 f(\boldsymbol{x}) \succeq m I_n$ to say,

$$\begin{aligned} r_{k+1} &\leqslant \|[\nabla^2 f(\boldsymbol{x}_k)]^{-1} \times \int_0^1 [\nabla^2 f(\boldsymbol{x}_k) - \nabla^2 f(\boldsymbol{x}^* + \tau(\boldsymbol{x}_k - \boldsymbol{x}^*))] \mathrm{d}\tau \, (\boldsymbol{x}_k - \boldsymbol{x}^*)\| \\ &\leqslant \frac{M}{m} \int_0^1 (1 - \tau) \mathrm{d}\tau \, \|\boldsymbol{x}_k - \boldsymbol{x}^*\|^2 \leqslant \frac{M}{2m} r_k^2. \end{aligned}$$

If $r_0 < 2m/M < R$, then $M/2m \, r_0 < 1$ and $r_1 < r_0 < 1$, so recursively $r_{k+1} < r_k$. The thesis follows considering the metric $\frac{M}{2m} r_k < 1$:

$$\frac{M}{2m} r_{k+1} \leqslant \left(\frac{M}{2m} r_k\right)^2 \leqslant \cdots \leqslant \left(\frac{M}{2m} r_0\right)^{2^{k+1}}.$$

♣

Let us, once again, analyse this result. First, the theorem offers a local convergence for special functions in terms of distance to the optimiser: $\|\boldsymbol{x}_k - \boldsymbol{x}^*\|$. The convergence is doubly exponential,

or as we say in optimisation: it is a quadratic convergence. In particular, to obtain an accuracy of $\|\boldsymbol{x}_t - \boldsymbol{x}^*\| \leqslant \epsilon$, we need,

$$t \gtrsim \log\log \frac{1}{\epsilon}.$$

calls to the oracle.

In practice this is very fast: to reach a very good accuracy $\epsilon \sim 10^{-20}$, typically 5-6 steps of Newton are enough; and this is basically independently of the problem conditioning κ. So the Newton's method works also for problems that are very badly conditioned. We will see how this is an essential feature for obtaining solvers to a large variety of problems.

As a corollary, for quadratic functions, the constant $M = 0$, so convergence is global, i.e., starting from anywhere, and it is achieved in exactly one iteration.

As for the disadvantages of the Newton's method we can cite two things. The first item is that the computational complexity per iteration is typically high. Solving a linear system involves $O(n^3)$ operations, and it may be not affordable in large-scale machine learning problem, or (in general) for any optimisation problems of reasonable size $n > 10^3$. This can be tackled in part by employing *quasi-Newton's* methods, which approximate the solution of the linear system or the Hessian. We will not discuss them here, but it is good to know that they exist.

An important message to remember here is that Newton's method is not the answer to all our problems, since it may be too computational complex, so we may need to use only first-order methods.

The second disadvantage of the Newton's method is that it is not a descent method, even in the convex case, and you have local convergence only. This is to say that, the method may diverge if not initalised properly, even in the convex case. This is however not very difficult to fix, as we see next.

1.6 Damped Newton's method

A way to fix the non-descending nature of the Newton's method is to add a stepsize to it, or as we say, we add damping. The method looks like the following,

Damped Newton's method

- *Start with* $\boldsymbol{x}_0 \in \mathbf{R}^n$
- *Iterate:*
  - *Solve the linear system:* $\nabla^2 f(\boldsymbol{x}_k)\boldsymbol{d}_k = \nabla f(\boldsymbol{x}_k)$,
  - *Backtracking: set* $\alpha_k = 1$, $\tau \in (0,1), \beta \in (0,1/2)$*:*
    While $f(\boldsymbol{x}_k - \alpha_k \boldsymbol{d}_k)) > f(\boldsymbol{x}_k) - \beta\alpha_k \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{d}_k \rangle$, *set:* $\alpha_k \leftarrow \tau\alpha_k$.
  - $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \boldsymbol{d}_k$.

The backtracking step chooses a stepsize α_k to guarantee that the cost decreases of at least certain amount. In fact, when the backtracking condition is verified, then,

$$f(\boldsymbol{x}_k - \alpha_k \boldsymbol{d}_k)) \leqslant f(\boldsymbol{x}_k) - \beta\alpha_k \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{d}_k \rangle.$$

With this new relationship in place, and asking also for a Lipschitz gradient, we can establish convergence of the scheme and the fact that, after a few steps, $\alpha_k = 1$ and we get back to the standard Newton's method. This latter fact is important, and it means that damping does not alter the local quadratic convergence of the method.

By using damping, we render the Newton's method globally convergent (in the convex case we will show it in the next lesson), but it is still local in the nonconvex case and yet eventually quadratic.

While we will get back to these points in a few lessons, we close this lesson with some comparisons between gradient and Newton's.

1.7 Gradient vs. Newton's

We can summarise the methods as follows.

- For the gradient method we have derived sub-linear convergence rates $O(1/\sqrt{t})$ and linear ones $O(\rho^t)$ for different problem classes; for gradients the latter is the best you get, but you need only a first-order oracle: less computations per iteration!
- For the Newton's method we have derived quadratic convergence rates! Which is extremely fast. However Newton's require a second-order oracle and it is only a local method. We have shown how to overcome some of its drawbacks with damping.

For Newton's method, we have also seen that its convergence is nearly independent on the condition numbering κ, reaching good accuracy with a number of steps that is virtually independent from the problem. A way to peek into this property is to understand that Newton's method is **affine invariant**.

Let T be a non-singular squared matrix and define the change of coordinates $\tilde{f}(\boldsymbol{y}) = f(T\boldsymbol{y})$, $\boldsymbol{x} = T\boldsymbol{y}$. Then,

$$\nabla \tilde{f}(\boldsymbol{y}) = T^{\mathsf{T}} \nabla f(\boldsymbol{x}), \qquad \nabla^2 \tilde{f}(\boldsymbol{y}) = T^{\mathsf{T}} \nabla^2 f(\boldsymbol{x}) T.$$

and the Newton's direction,

$$T\boldsymbol{d}_{\boldsymbol{y}} = T[\nabla^2 \tilde{f}(\boldsymbol{y})]^{-1} \nabla \tilde{f}(\boldsymbol{y}) = \boldsymbol{d}_{\boldsymbol{x}},$$

so $T(\boldsymbol{y}_k - \boldsymbol{d}_{\boldsymbol{y}}) = \boldsymbol{x}_k - \boldsymbol{d}_{\boldsymbol{x}}$, meaning that "reconditioning" of the problem does not affect how Newton's behaves.

1.8 Least-squares problems

We close this lecture with an example of unconstrained problem which you will encounter over and over. For instance, when training a neural network.

Suppose you have some data that tells you the label $y_i \in \mathbf{R}$ of some feature $x_i \in \mathbf{R}^n$. You can imagine you have m of such data points (x_i, y_i). You then build an neural network architecture, which is nothing but a series of interconnections and functions. At the end, the network is a map $h_i(\boldsymbol{x}; w) : x_i \mapsto y_i$, with some weights $w \in \mathbf{R}^p$ to train. The training problem is then,

$$\underset{w \in \mathbf{R}^p}{\text{minimise}} \ \frac{1}{2} \sum_{i=1}^{m} \|y_i - h_i(\boldsymbol{x}; w)\|_2^2,$$

and depending on the nature of h_i, the problem is easier or more complex. Typically, h_i is highly nonconvex and you have many local minima. Also, when the weights become big, your landscape can become very flat, making the progress of any method challenging.

The problem above is an example of a least-squares problem, first introduced by Gauss (aged 18) and then used by him to determine the position of the moon Ceres in ∼1800.

1.9 References

- YN: Introduction and Chapter 1;
- BV: Chapter 9.

◇ ◇ ◆

1.10 Exercises

Exercise 1.1 (Exam 2023) *Consider the unconstrained problem,*

$$\min_{x \in \mathbf{R}^n} f(x)$$

for a function $f : \mathbf{R}^n \to \mathbf{R}$, which is $\mathcal{C}^1(\mathbf{R}^n)$ and its gradient is L-Lipschitz continuous. Note that the function is **not** *convex.*

1. *Consider the gradient method with variable stepsize $\alpha_k > 0$ to find the minimum f^* of f. Which convergence certificates can one expect (that is: which quantity converges and with which convergence rate)?*
2. *Let f now have the additional property that,*

$$\frac{1}{2}\|\nabla f(x)\|^2 \geqslant \eta(f(x) - f^*), \qquad \forall x \in \mathbf{R}^n,$$

for $L > \eta > 0$.

   (a) *Prove that all the stationary points of $f(x)$, say $\bar{x}$, are global minimizers, i.e., $f(\bar{x}) = f^*$.*

   (b) *Prove that the gradient method with constant stepsize $\alpha = \frac{1}{L}$ offers a convergence certificate of the form,*

$$f(x_k) - f^* \leqslant \rho^k(f(x_0) - f^*),$$

for $\rho = (1 - \eta/L) < 1$, and sequence $\{x_k\}_{k\in\mathbf{N}}$ generated by said gradient method.

   (c) *How is this convergence called?*

Exercise 1.2 *Consider the problem*

$$\min_{x\in\mathbf{R}^n} f(x),$$

for $f \in \mathcal{C}_L^{1,1}(\mathbf{R}^n)$. Consider the gradient method to find a local minimum.

Theorem. *Assuming f bounded below, the gradient method on problem with $f \in \mathcal{C}_L^{1,1}(\mathbf{R}^n)$ with constant $\alpha < 2/L$ will converge to a stationary point as*

$$\min_{k=0,\ldots,t} \|\nabla f(x_k)\| \leqslant \frac{1}{\sqrt{t+1}} \sqrt{\frac{1}{\alpha(1-\alpha L/2)}(f(x_0) - f^*)} \quad .$$

□

Prove the theorem.

Exercise 1.3 *Consider the problem*

$$\min_{x\in\mathbf{R}^n} f(x).$$

We study the convergence of Newton's method in the decrement,

$$\lambda(x) = (\nabla^\top f(x)\nabla^2 f(x)^{-1}\nabla f(x))^{1/2}$$

for special strongly convex functions f such that for $v = -\nabla^2 f(x)^{-1}\nabla f(x)$, $t \geqslant 0$:

$$(1 - t\lambda(x))^2\nabla^2 f(x) \preceq \nabla^2 f(x + tv) \preceq \frac{1}{(1 - t\lambda(x))^2}\nabla^2 f(x).$$

(These functions are self-concordant, and they will be important in Class 4).

Suppose $\lambda(x_k) < 1$, and define $x_{k+1} = x_k - \nabla^2 f(x_k)^{-1}\nabla f(x_k) = x_k - v_k$. Prove that

$$\lambda(x_{k+1}) \leqslant \frac{\lambda^2(x_k)}{(1 - \lambda(x_k))^2}.$$

Furthermore if $\lambda(x_0) < 1/4$, prove that we can achieve quadratic convergence, with,

$$2\lambda(x_{k+1}) \leqslant (2\lambda(x_k))^2.$$

Hint: in both cases, assume without loss of generality that $\nabla^2 f(x_k) = I_n$.

Chapter 2

Second lecture

Unconstrained optimisation (II): convex case

2.1 The convex problem class

We now look at convex problems to understand whether they help to design algorithms that are better in terms of convergence, convergence rates, and generally computational complexity. In the interest of time, we only look here at simple methods, like the gradient (and small variations thereof), but our findings can be extended to fancier methods you will encounter in machine learning or in other fields.

We will see that convexity per se is not the only ingredient to obtain "better" results. Lipschitz continuity plays a hugely important part as well. Even though convexity will help in obtaining global optimality certificates, rather than stationarity.

Before starting, I recall that whenever we talk about convergence, we will need to be careful in saying what is converging to what exactly. In optimisation problems, we will distinguish at least three types of convergence results. The first being the fixed point residual, typically $\|\nabla_{\boldsymbol{x}} f(\boldsymbol{x}_k)\|$ converging to 0. The second it the objective convergence, $|f(\boldsymbol{x}_k) - f^*| \to 0$, and the third is the distance to the optimiser $\|\boldsymbol{x}_k - \boldsymbol{x}^*\| \to 0$. The former is weaker (it needs typically less assumptions and it is not a strong result), the latter is a stronger result, but harder to prove in general.

In fact, the convergence $\|\boldsymbol{x}_k - \boldsymbol{x}^*\| \to 0$ implies $|f(\boldsymbol{x}_k) - f^*| \to 0$, which in turn implies $\|\nabla_{\boldsymbol{x}} f(\boldsymbol{x}_k)\| \to 0$. But the converse is not true. For instance you may have multiple optimisers $\boldsymbol{x}^*$ and, even though $|f(\boldsymbol{x}_k) - f^*| \to 0$, the sequence $\boldsymbol{x}_k$ may not have a limit point. Also, you may have $\|\nabla_{\boldsymbol{x}} f(\boldsymbol{x}_k)\| \to 0$, but you are at a stationary point, not an optimiser, so $|f(\boldsymbol{x}_k) - f^*|$ does not converge to 0.

We are now ready to start.

2.1.1 Some necessary definitions

Definition 2.1 (Convex function) *A function $f : X \subseteq \mathbf{R}^n \to \mathbf{R}$ is convex iff X is convex and*

$$(\mathsf{C1}) \quad \forall \boldsymbol{x}, \boldsymbol{y} \in X, \lambda \in [0,1] : \quad f(\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}) \leqslant \lambda f(\boldsymbol{x}) + (1-\lambda) f(\boldsymbol{y}).$$

Multiple definitions exist if we have additional properties, for example:

$$(\mathsf{C1}) + f \in \mathcal{C}^1(X) \iff \forall \boldsymbol{x}, \boldsymbol{y} \in X,\ f(\boldsymbol{x}) \geqslant f(\boldsymbol{y}) + \langle \nabla f(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y} \rangle; \tag{2.1}$$

$$(\mathsf{C1}) + f \in \mathcal{C}^2(X) \iff \forall \boldsymbol{x}, \boldsymbol{y} \in X,\ \nabla^2 f(\boldsymbol{x}) \succeq 0. \tag{2.2}$$

A proof of the relations above can be found in YN, by combining his Definition 2.1.1, with his Theorems 2.1.2 and 2.1.4.

Definition 2.2 (Strongly convex function) *A convex function $f : X \subseteq \mathbf{R}^n \to \mathbf{R}$ is m-strongly convex iff*

$$(\mathsf{SC}) \quad f(\boldsymbol{x}) - \frac{m}{2}\|\boldsymbol{x}\|^2 \text{ is convex.}$$

We note here that f does not need to be differentiable to be strongly convex. Here too, multiple definitions exist if we have additional properties, for example:

$$(\mathsf{SC}) + f \in \mathcal{C}^1(X) \iff \forall \boldsymbol{x}, \boldsymbol{y} \in X, \langle \nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y} \rangle \geqslant m\|\boldsymbol{x} - \boldsymbol{y}\|^2 \tag{2.3}$$

$$(\mathsf{SC}) + f \in \mathcal{C}^2(X) \iff \forall \boldsymbol{x}, \boldsymbol{y} \in X, \nabla^2 f(\boldsymbol{x}) \succeq mI_n \tag{2.4}$$

A proof of the relations above can be found in YN, by using Theorem 2.1.8, 2.1.9, and 2.1.10.

Definition 2.3 (Smooth function) *A convex function $f : X \subseteq \mathbf{R}^n \to \mathbf{R}$ is L-smooth iff*

$$(\mathsf{LC}) \quad \frac{L}{2}\|\boldsymbol{x}\|^2 - f(\boldsymbol{x}) \text{ is convex.}$$

Importantly $(\mathsf{LC}) \implies f \in \mathcal{C}^1(X)$, that is a condition on the function implies its differentiability. Since this is rather important, we give a short proof here.

Proof. *Define $g(x) = \frac{L}{2}\|x\|^2 - f(x)$, by the definition of subgradient of a convex function:*

$$g(x) - g(x') \geqslant u^\top (x - x'), \quad \forall x, x', u \in \partial g,$$

where ∂g is the subdifferential of g. Furthermore, by definition of g and the convexity of f,

$$\frac{L}{2}\|x\|^2 - \frac{L}{2}\|x'\|^2 \geqslant f(x) - f(x') + u^\top (x - x') \geqslant (u + v)^\top (x - x'), \quad \forall x, x', u \in \partial g, v \in \partial f,$$

where we have indicated with ∂f the subdifferential of f.

Since the function $\frac{L}{2}\|x\|^2$ is convex and differentiable, then we know that its subdifferential at x' is a singleton and it is equivalent to Lx'. Then, $u+v$ is a singleton and $u+v = Lx'$. Furthermore since u and v are elements of the subdifferentials and can be chosen independently, then u and v separately are also singletons. That is, f is differentiable. So $(LC) \implies f \in \mathcal{C}^1(\mathbf{R}^n)$. ♣

As before, depending on additional properties, multiple definitions of smoothness exist, for example:

$$(\mathsf{LC}) \iff \forall \boldsymbol{x}, \boldsymbol{y} \in X, \|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\| \leqslant L\|\boldsymbol{x} - \boldsymbol{y}\| \tag{2.5}$$

$$(\mathsf{LC}) \iff \forall \boldsymbol{x}, \boldsymbol{y} \in X, \frac{1}{L}\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\|^2 \leqslant \langle \nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y} \rangle \tag{2.6}$$

$$(\mathsf{LC}) + f \in \mathcal{C}^2(X) \iff \forall \boldsymbol{x}, \boldsymbol{y} \in X, 0 \preceq \nabla^2 f(\boldsymbol{x}) \preceq LI_n \tag{2.7}$$

We see already that (2.5) is a Lipschitz condition on the gradient of f. Since our definition is equivalent to (2.5), often it is the latter than is used as a definition of smoothness.

The proof of the relations above can be found in YN, e.g., Theorem 2.1.5 and related.

We look here at the m-strongly convex L-smooth class, which is at least one time differentiable. To echo the class $\mathcal{C}_L^{p,q}$, we refer to the convex classes as

$$f \in \mathcal{S}_{m,L}^{p,q} \iff \{f \in \mathcal{C}_L^{p,q}\} \cap \{f \text{is } m\text{-strongly convex}\}.$$

We are especially interested in the function class $f \in \mathcal{S}_{m,L}^{1,1}$. If $f \in \mathcal{S}_{0,L}^{1,1}$, the function is just convex. We also use a slight abuse of notation to indicate with $f \in \mathcal{S}_{m,+\infty}^{1,1}$ when the function is not Lipschitz and not differentiable. However, at least here, unless explicitly said, $0 < m \leqslant L < +\infty$. Finally, we remind that $m \leqslant L$ always and $\kappa := L/m \geqslant 1$.

Even though there are plenty of definitions, for this course, you will need to remember at least the following,

$$\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbf{R}^n, \langle \nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y} \rangle \geqslant m\|\boldsymbol{x} - \boldsymbol{y}\|^2 \tag{2.8}$$

$$\boldsymbol{x}, \boldsymbol{y} \in \mathbf{R}^n, \|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\| \leqslant L\|\boldsymbol{x} - \boldsymbol{y}\|, \tag{2.9}$$

which will carry you a long way.

Geometrically, a function in the class $\mathcal{S}^{1,1}_{m,L}$ is a function that in all its domain grows faster than a quadratic function with concavity m but slower than a quadratic of concavity L. This is rather easy to see for doubly differentiable functions, for which $mI_n \preceq \nabla^2 f(\boldsymbol{x}) \preceq LI_n$, but it is also valid in general. In this context, the class $\mathcal{S}^{1,1}_{m,L}$ impose conditions on growth, which will be key in establishing efficient algorithms.

We have now all the tools to characterise convergence of our methods.

2.2 Standard gradient methods

2.2.1 Case I: Gradient for $f \in \mathcal{S}^{1,1}_{m,L}$

We look at the unconstrained problem

$$\underset{\boldsymbol{x} \in \mathbf{R}^n}{\text{minimise}}\, f(\boldsymbol{x}),$$

when $f \in \mathcal{S}^{1,1}_{m,L}$ $(0 < m \leqslant L < +\infty)$, and we use the gradient method with constant stepsize $\alpha > 0$.

We already have a result for it (Theorem 1.6), and in particular:

Theorem 2.1 *The gradient descent method on unconstrained problems with constant stepsize $\alpha < 2/L$, for functions $f \in \mathcal{S}^{1,1}_{m,L}$ has a (global) convergence certificate of*

$$\|\boldsymbol{x}_t - \boldsymbol{x}^*\| \leqslant \rho^t \|\boldsymbol{x}_0 - \boldsymbol{x}^*\| \tag{2.10}$$

for $\rho = \max\{|1 - \alpha m|, |1 - \alpha L|\}$, and t iterations.

Proof. *The proof follows from the proof of Theorem 1.6, substituting the fact that the local property (1.18) is now valid for the whole space. Particular attention has to be put on the fact that f is now not doubly differentiable, so one has to prove $\|g(\boldsymbol{x}) - g(\boldsymbol{y})\| \leqslant \rho\|\boldsymbol{x} - \boldsymbol{y}\|$ by other means. How?* ♣

Can we do better in the convex case in terms of rate, more than just global? No we can't. The reasoning is rather technical, but there exist a function $g \in \mathcal{S}^{1,1}_{m,L}$ for which we have

$$\|\boldsymbol{x}_t - \boldsymbol{x}^*\| \geqslant \omega^t \|\boldsymbol{x}_0 - \boldsymbol{x}^*\|, \qquad 0 < \omega \leqslant \rho.$$

This result is very strong: it tells you that in the worst case, linear convergence for $f \in \mathcal{S}^{1,1}_{m,L}$ is the best you can achieve. We will not prove this result here, but the interested reader can have a look at YN.

We can derive how other metrics (objective, fixed point) converge as well, and you will prove them in the exercises. In particular, for $f \in \mathcal{S}^{1,1}_{m,L}$, you will also derive similar linear converging conditions for $f(\boldsymbol{x}_t) - f(\boldsymbol{x}^*)$ and $\|\nabla f(\boldsymbol{x}_t)\|$, completing the family picture for the gradient method.

Furthermore, one can show that there exist optimal stepsize and rates, which are, $\alpha^* = \frac{2}{m+L}$ and $\rho^* = \frac{\kappa - 1}{\kappa + 1}$.

2.2.2 Case II: Gradient for $f \in \mathcal{S}^{1,1}_{0,L}$

Let's move on to a non-strongly convex function class. In this case, there may be multiple solutions in terms of $\boldsymbol{x}^*$, all equally optimal.

Theorem 2.2 *The gradient descent method on unconstrained problems with constant stepsize $\alpha < 2/L$, for functions $f \in \mathcal{S}^{1,1}_{0,L}$ has an ergodic (global) convergence certificate of*

$$f(\bar{\boldsymbol{x}}_t) - f(\boldsymbol{x}^*) \leqslant \frac{C}{t+1}\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|^2, \tag{2.11}$$

for $\bar{\boldsymbol{x}}_t = \frac{1}{t+1}\sum_{k=0}^{t} \boldsymbol{x}_k$, constant $C > 0$, and t iterations.

Proof. *Start with the update rule,*

$$\begin{aligned} \|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\|^2 &= \|\boldsymbol{x}_k - \alpha\nabla f(\boldsymbol{x}_k) - \boldsymbol{x}^*\|^2 \\ &= \|\boldsymbol{x}_k - \boldsymbol{x}^*\|^2 - 2\alpha\langle\nabla f(\boldsymbol{x}_k), \boldsymbol{x}_k - \boldsymbol{x}^*\rangle + \|\alpha\nabla f(\boldsymbol{x}_k)\|^2 \end{aligned}$$

Use $\nabla f(\boldsymbol{x}^*) = 0$ to say that the right-hand side (RHS) is

$$\|\boldsymbol{x}_k - \boldsymbol{x}^*\|^2 - 2\alpha\langle\nabla f(\boldsymbol{x}_k) - \nabla f(\boldsymbol{x}^*), \boldsymbol{x}_k - \boldsymbol{x}^*\rangle + \alpha^2\|\nabla f(\boldsymbol{x}_k) - \nabla f(\boldsymbol{x}^*)\|^2.$$

Now use Eq. (2.6) to say,

$$RHS \leqslant \|\boldsymbol{x}_k - \boldsymbol{x}^*\|^2 - \alpha(2-\alpha L)\langle\nabla f(\boldsymbol{x}_k) - \nabla f(\boldsymbol{x}^*), \boldsymbol{x}_k - \boldsymbol{x}^*\rangle.$$

For $\alpha < 2/L$ the term $(2-\alpha L)$ is positive and the error $\|\boldsymbol{x}_k - \boldsymbol{x}^*\|^2$ decreases.

This gives convergence, but not the rate.

Use convexity to say $f(\boldsymbol{x}^*) \geqslant f(\boldsymbol{x}_k) + \langle\nabla f(\boldsymbol{x}_k), \boldsymbol{x}^* - \boldsymbol{x}_k\rangle$ and hence: $f(\boldsymbol{x}_k) - f(\boldsymbol{x}^*) \leqslant \langle\nabla f(\boldsymbol{x}_k), \boldsymbol{x}_k - \boldsymbol{x}^*\rangle$.

Therefore,

$$RHS \leqslant \|\boldsymbol{x}_k - \boldsymbol{x}^*\|^2 - \alpha(2-\alpha L)[f(\boldsymbol{x}_k) - f(\boldsymbol{x}^*)].$$

Sum over k to arrive at

$$\sum_{k=0}^{t} f(\boldsymbol{x}_k) - f(\boldsymbol{x}^*) \leqslant \frac{1}{\alpha(2-\alpha L)}\left(\sum_{k=0}^{t} -\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\|^2 + \|\boldsymbol{x}_k - \boldsymbol{x}^*\|^2\right) \leqslant \underbrace{\frac{1}{\alpha(2-\alpha L)}}_{=:C}\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|^2.$$

Now we use Jensen's inequality valid for convex f: $f(\sum_i \beta\boldsymbol{x}_i) \leqslant \sum_i \beta_i f(\boldsymbol{x}_i)$ for $\sum_i \beta_i = 1$, to say,

$$f(\bar{\boldsymbol{x}}_t) - f(\boldsymbol{x}^*) \leqslant \frac{1}{t+1}\sum_{k=0}^{t} f(\boldsymbol{x}_k) - f(\boldsymbol{x}^*) \leqslant \frac{C}{t+1}\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|^2,$$

from which the thesis. ♣

Let us analyse the result. Theorem 2.2 gives us a global convergence result in terms of ergodic objective, with a rate of $O(1/t)$. An ergodic objective, or ergodic mean, is the fact that we are using $\bar{\boldsymbol{x}}_t$ instead of $\boldsymbol{x}_t$. Sometimes this helps to render the proofs easier. With a little more effort, we could obtain an objective convergence $f(\boldsymbol{x}_t) - f^* \leqslant O(1/t)$, but we do not look into that here.

An objective convergence of this type in the convex case is better than in the nonconvex case, for which we had a fixed point convergence to a stationary point with rate $O(1/\sqrt{t})$ (imposing Lipschitz).

Can we do better? Yes, actually the attainable complexity bound for the class $\mathcal{S}^{1,1}_{0,L}$ is $O(1/t^2)$. But how? The answer to this question will make us study one of the most important algorithms in optimisation of the past forty years: Nesterov's method.

2.3 Nesterov's accelerated gradient

Let's set the stage by defining some supporting sequences, $\{\lambda_k\}$ and $\{\gamma_k\}$ defined as,

$$\lambda_0 = 0,\ \lambda_k = \frac{1+\sqrt{1+4\lambda_{k-1}^2}}{2},\ \text{and}\ \gamma_k = \frac{1-\lambda_k}{\lambda_{k+1}}. \tag{2.12}$$

Note already that $\gamma_k \leqslant 0$, $\lambda_1 = 1$, and $\lambda_k > 1$ for $k > 1$.

Now the algorithm is defined by the following equations,

Nesterov's accelerated gradient descent

- *Start with* $\boldsymbol{x}_0 = \boldsymbol{y}_0 \in \mathbf{R}^n$ *and the sequences* (2.12)
- *Iterate:* $\boldsymbol{y}_{k+1} = \boldsymbol{x}_k - \alpha\nabla f(\boldsymbol{x}_k), \qquad \boldsymbol{x}_{k+1} = (1-\gamma_k)\boldsymbol{y}_{k+1} + \gamma_k \boldsymbol{y}_k, \quad k = 0, 1, \ldots.$

In other words, Nesterov's accelerated gradient descent performs a simple step of gradient descent to go from $\boldsymbol{x}_k$ to $\boldsymbol{y}_{k+1}$, and then it 'slides' a little bit further than $\boldsymbol{y}_{k+1}$ in the direction given by the previous point $\boldsymbol{y}_k$.

Theorem 2.3 *Nesterov's accelerated gradient method on unconstrained problems, with constant stepsize $\alpha = 1/L$, for functions $f \in \mathcal{S}_{0,L}^{1,1}$ has a (global) convergence certificate of*

$$f(\boldsymbol{x}_t) - f(\boldsymbol{x}^*) \leqslant \frac{2L}{t^2}\|\boldsymbol{x}_1 - \boldsymbol{x}^*\|^2, \tag{2.13}$$

for t iterations.

Proof. *Start from convexity, Lipschitz, and $\alpha = 1/L$:*

$$\begin{aligned}
&f\left(\boldsymbol{x} - \alpha\nabla f(\boldsymbol{x})\right) - f(\boldsymbol{y}) \\
&\leqslant f\left(\boldsymbol{x} - \alpha\nabla f(\boldsymbol{x})\right) - f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^{\mathsf{T}}(\boldsymbol{x} - \boldsymbol{y}) \\
&\leqslant \nabla f(\boldsymbol{x})^{\mathsf{T}}\left(\boldsymbol{x} - \alpha\nabla f(\boldsymbol{x}) - \boldsymbol{x}\right) + \frac{L}{2}\left\|\boldsymbol{x} - \alpha\nabla f(\boldsymbol{x}) - \boldsymbol{x}\right\|^2 + \nabla f(\boldsymbol{x})^{\mathsf{T}}(\boldsymbol{x} - \boldsymbol{y}) \\
&= -\frac{1}{2L}\|\nabla f(\boldsymbol{x})\|^2 + \nabla f(\boldsymbol{x})^{\mathsf{T}}(\boldsymbol{x} - \boldsymbol{y}).
\end{aligned}$$

Now let us apply the inequality to $\boldsymbol{x} = \boldsymbol{x}_k$ and $\boldsymbol{y} = \boldsymbol{y}_k$, which gives

$$\begin{aligned}
f(\boldsymbol{y}_{k+1}) - f(\boldsymbol{y}_k) &= f\left(\boldsymbol{x}_k - \alpha\nabla f(\boldsymbol{x}_k)\right) - f(\boldsymbol{y}_k) \\
&\leqslant -\frac{1}{2L}\|\nabla f(\boldsymbol{x}_k)\|^2 + \nabla f(\boldsymbol{x}_k)^{\mathsf{T}}(\boldsymbol{x}_k - \boldsymbol{y}_k) \\
&= -\frac{L}{2}\|\boldsymbol{y}_{k+1} - \boldsymbol{x}_k\|^2 - L(\boldsymbol{y}_{k+1} - \boldsymbol{x}_k)^{\mathsf{T}}(\boldsymbol{x}_k - \boldsymbol{y}_k).
\end{aligned}$$

Similarly we apply it to $\boldsymbol{x} = \boldsymbol{x}_k$ and $\boldsymbol{y} = \boldsymbol{x}^*$ which gives

$$f(\boldsymbol{y}_{k+1}) - f(\boldsymbol{x}^*) \leqslant -\frac{L}{2}\|\boldsymbol{y}_{k+1} - \boldsymbol{x}_k\|^2 - L(\boldsymbol{y}_{k+1} - \boldsymbol{x}_k)^{\mathsf{T}}(\boldsymbol{x}_k - \boldsymbol{x}^*).$$

Now multiplying the first by $(\lambda_k - 1)$ which is positive for $k > 2$ and adding the result to the second, one obtains with $\delta_k = f(\boldsymbol{y}_k) - f(\boldsymbol{x}^*)$,

$$\lambda_k\delta_{k+1} - (\lambda_k - 1)\delta_k \leqslant -\frac{L}{2}\lambda_k\|\boldsymbol{y}_{k+1} - \boldsymbol{x}_k\|^2 - L(\boldsymbol{y}_{k+1} - \boldsymbol{x}_k)^{\mathsf{T}}(\lambda_k\boldsymbol{x}_k - (\lambda_k - 1)\boldsymbol{y}_k - \boldsymbol{x}^*).$$

Multiplying the last inequality by $\lambda_k > 0$ and by using that by definition $\lambda_{k-1}^2 = \lambda_k^2 - \lambda_k$ one obtains

$$\begin{aligned}
&\lambda_k^2\delta_{k+1} - \lambda_{k-1}^2\delta_k \\
&\leqslant -\frac{L}{2}\left(\|\lambda_k(\boldsymbol{y}_{k+1} - \boldsymbol{x}_k)\|^2 + 2\lambda_k(\boldsymbol{y}_{k+1} - \boldsymbol{x}_k)^{\mathsf{T}}(\lambda_k\boldsymbol{x}_k - (\lambda_k - 1)\boldsymbol{y}_k - \boldsymbol{x}^*)\right).
\end{aligned}$$

We remark that $\lambda_{k-1}^2 = \lambda_k^2 - \lambda_k$ *follows from* $\lambda_k = \frac{1+\sqrt{1+4\lambda_{k-1}^2}}{2}$*, in fact the latter implies,*

$$(2\lambda_k - 1)^2 = 1 + 4\lambda_{k-1}^2 \iff \lambda_{k-1}^2 = \lambda_k^2 - \lambda_k.$$

Now one can verify that

$$\begin{aligned}&\|\lambda_k(\boldsymbol{y}_{k+1} - \boldsymbol{x}_k)\|^2 + 2\lambda_k(\boldsymbol{y}_{k+1} - \boldsymbol{x}_k)^\mathsf{T}(\lambda_k\boldsymbol{x}_k - (\lambda_k - 1)\boldsymbol{y}_k - \boldsymbol{x}^*)\\ &= \|\lambda_k\boldsymbol{y}_{k+1} - (\lambda_k - 1)\boldsymbol{y}_k - \boldsymbol{x}^*\|^2 - \|\lambda_k\boldsymbol{x}_k - (\lambda_k - 1)\boldsymbol{y}_k - \boldsymbol{x}^*\|^2.\end{aligned}$$

Next remark that, by definition of γ_k*, one has*

$$\begin{aligned}&\boldsymbol{x}_{k+1} = \boldsymbol{y}_{k+1} + \gamma_k(\boldsymbol{y}_k - \boldsymbol{y}_{k+1})\\ &\Leftrightarrow \lambda_{k+1}\boldsymbol{x}_{k+1} = \lambda_{k+1}\boldsymbol{y}_{k+1} + (1 - \lambda_k)(\boldsymbol{y}_k - \boldsymbol{y}_{k+1})\\ &\Leftrightarrow \lambda_{k+1}\boldsymbol{x}_{k+1} - (\lambda_{k+1} - 1)\boldsymbol{y}_{k+1} = \lambda_k\boldsymbol{y}_{k+1} - (\lambda_k - 1)\boldsymbol{y}_k.\end{aligned}$$

Putting together the previous relationships one gets with $u_k = \lambda_k\boldsymbol{x}_k - (\lambda_k - 1)\boldsymbol{y}_k - \boldsymbol{x}^*$*,*

$$\lambda_k^2\delta_{k+1} - \lambda_{k-1}^2\delta_k \leqslant \frac{L}{2}\Big(\|u_k\|^2 - \|u_{k+1}\|^2\Big).$$

Summing these inequalities from $k = 1$ *to* $k = t - 1$ *(rem:* $\lambda_0 = 0, \lambda_1 = 1$*) one obtains:*

$$\delta_t \leqslant \frac{L}{2\lambda_{t-1}^2}\|u_1\|^2.$$

By induction: $\lambda_{t-1} \geqslant \frac{t}{2}$ *which concludes the proof.* ♣

What is the meaning of this convergence result? We have a global convergence result in terms of objective, with a rate of $O(1/t^2)$, which matches the lower bound. In this context, we say that Nesterov's accelerated method is an optimal method. This is way better than in the nonconvex case, for which we had a fixed point convergence to a stationary point with rate $O(1/\sqrt{t})$.

If one is attentive, one can see that we have proved a result for a specific $\alpha = 1/L$. However, a one-line proof can show that we can extend the theorem to any $\alpha \leqslant 1/L$, how?

Corollary 2.1 *Theorem 2.3 is valid for* $\alpha \leqslant 1/L$*.*

Proof. *Homework.* ♣

2.3.1 Nesterov's alternative formulations*

Nesterov's accelerated scheme is only one of the few accelerations that are possible for solving smooth convex problems. Moreover, Nesterov's acceleration may be presented in a few alternative (but almost equivalent) forms. In fact, in the proof of the method, we use the fact that the coefficients $\lambda_{k-1}^2 = \lambda_k^2 - \lambda_k$. However, for the rate to be proven, one would only need the looser condition, $-\lambda_{k-1}^2 \leqslant -(\lambda_k^2 - \lambda_k)$, which leads to $\lambda_{k-1}^2 \geqslant \lambda_k^2 - \lambda_k$. This yields a few different alternatives.

A popular choice is to set $\lambda_k = \frac{k}{2}$, this verifies,

$$\lambda_{k-1}^2 \geqslant \lambda_k^2 - \lambda_k \implies (k-1)^2 \geqslant k^2 - 2k \iff 1 \geqslant 0.$$

With this choice $\gamma_k = \frac{2-k}{k+1}$. So that the method is presented as follows.

Nesterov's accelerated gradient descent (variant)

- *Start with* $\boldsymbol{x}_0 = \boldsymbol{y}_0 \in \mathbf{R}^n$
- *Iterate:* $\boldsymbol{y}_{k+1} = \boldsymbol{x}_k - \alpha\nabla f(\boldsymbol{x}_k), \qquad \boldsymbol{x}_{k+1} = \boldsymbol{y}_{k+1} + \frac{k-2}{k+1}(\boldsymbol{y}_{k+1} - \boldsymbol{y}_k), \quad k = 0, 1, \ldots.$

2.4 The subgradient method

Let's now look at the case in which we lose differentiability. We let $f \in \mathcal{S}^{1,1}_{m,+\infty}$. In this case, since $\nabla f(\boldsymbol{x})$ does not exist, we need another notion, the one of subgradient $\partial f(\boldsymbol{x})$.

Definition 2.4 (Subdifferential) *We define the subdifferential at $\boldsymbol{x}$ as the set*

$$\partial f(\boldsymbol{x}) := \{\boldsymbol{g} \in \mathbf{R}^n | f(\boldsymbol{y}) \geqslant f(\boldsymbol{x}) + \langle \boldsymbol{g}, \boldsymbol{y} - \boldsymbol{x} \rangle, \, \forall \boldsymbol{y} \in \mathbf{R}^n\}.$$

You have studied in details the subdifferential and the subgradient in OPT201. For the sake of this course, it is useful to see the subdifferential as a set of all the vectors that generate hyperplanes that stay "below" the function at any point. You can also see the parallel between the definition of subdifferential and the definition of convex function. In particular, if the function f is convex and differentiable, then, the subdifferential is a singleton: $\partial f(\boldsymbol{x}) = \{\nabla f(\boldsymbol{x})\}$.

You may encounter many definitions of subdifferential in the nonconvex domain. Luckily for us, in the convex domain they are equivalent to our definition. Finally, in many algorithms, we don't even need to get the whole subdifferential set, just one element $\boldsymbol{g} \in \partial f(\boldsymbol{x})$ will be enough!

With this new notion in place, we are ready for the subgradient method as follows.

Select a stepsize $\alpha_k > 0$:

Subgradient method

- *Start with* $\boldsymbol{x}_0 \in \mathbf{R}^n$
- *Iterate: find a* $\boldsymbol{g}_k \in \partial f(\boldsymbol{x}_k)$, *compute* $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \boldsymbol{g}_k, \quad k = 0, 1, \ldots.$

We are now also ready for its convergence certificate under the additional condition that the subgradient is bounded, as

$$f \in \mathcal{S}^{0,0}_{0,G}, \text{ or equivalently } \|\boldsymbol{g}\| \leqslant G, \text{ or equivalently } \|f(\boldsymbol{x}) - f(\boldsymbol{y})\| \leqslant G\|\boldsymbol{x} - \boldsymbol{y}\| \qquad \forall \boldsymbol{x}, \boldsymbol{y}.$$

Theorem 2.4 *The subgradient method on unconstrained problems with stepsize satisfying the conditions,*

$$\lim_{k\to\infty} \alpha_k \to 0, \quad \sum_{k=0}^{\infty} \alpha_k = \infty,$$

for functions $f \in \mathcal{S}^{0,0}_{0,G}$ has a (global) convergence certificate of

$$\lim_{t\to\infty} \min_{k=0,\ldots,t} f(\boldsymbol{x}_t) - f(\boldsymbol{x}^*) = 0. \tag{2.14}$$

Furthermore, at best, $\min_{k=0,\ldots,t} f(\boldsymbol{x}_t) - f(\boldsymbol{x}^*) = O(1/\sqrt{t})$.

Valid conditions on α_k are, e.g., $\alpha_k = \frac{1}{k+1}$, $\alpha_k = \frac{1}{\sqrt{k+1}}$,...

Proof. *[for the choice of $\alpha_k = \frac{1}{k+1}$] We start from the algorithm definition and convexity,*

$$\begin{aligned} \|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\|^2 &= \|\boldsymbol{x}_k - \boldsymbol{x}^*\|^2 - 2\alpha_k \langle \boldsymbol{g}_k, \boldsymbol{x}_k - \boldsymbol{x}^* \rangle + \alpha_k^2 \|\boldsymbol{g}_k\|^2 \\ &\leqslant \|\boldsymbol{x}_k - \boldsymbol{x}^*\|^2 - 2\alpha_k (f(\boldsymbol{x}_k) - f(\boldsymbol{x}^*)) + \alpha_k^2 G^2. \end{aligned}$$

Summing over k, we obtain,

$$2 \sum_k \alpha_k (f(\boldsymbol{x}_k) - f(\boldsymbol{x}^*)) \leqslant \|\boldsymbol{x}_0 - \boldsymbol{x}^*\|^2 + \sum_k \alpha_k^2 G^2.$$

Since $\sum_k \alpha_k (f(\boldsymbol{x}_k) - f(\boldsymbol{x}^*)) \geqslant (\sum_k \alpha_k) \min_k (f(\boldsymbol{x}_k) - f(\boldsymbol{x}^*))$, then,

$$\min_k (f(\boldsymbol{x}_k) - f(\boldsymbol{x}^*)) \leqslant \frac{1}{2\sum_k \alpha_k} \|\boldsymbol{x}_0 - \boldsymbol{x}^*\|^2 + \frac{\sum_k \alpha_k^2}{2\sum_k \alpha_k} G^2 \to 0.$$

The $\to 0$ is not immediately evident and requires some extra work. It is obvious if, e.g., $\alpha_k = \frac{1}{k+1}$.

In particular, the selection $\alpha_k = \frac{1}{k+1}$, we obtain the lowest rate bound,

$$\min_k (f(\boldsymbol{x}_k) - f(\boldsymbol{x}^*)) \sim O(\frac{1}{\log t}) \gtrsim O(\frac{1}{\sqrt{t}})$$

where we recall that $O(\log t) < O(\sqrt{t})$. ♣

We have a global convergence result in terms of best objective, with a rate of $O(1/\sqrt{t})$. This is the rate of nonconvex fixed point residual in the Lipschitz case, so it's better in the convex case, but it's very slow compared to the convex Lipschitz case, for which we found $O(1/t^2)$.

In the next class, we will see how to avoid the use of the subgradient method (in some cases) by the use of splitting operator and restore reasonable convergence rates.

2.4.1 Adding strong convexity*

One may want to see if adding strong convexity to a non-differentiable cost f can change the convergence certificates we obtain. This is not the case, since adding strong convexity does not change the differentiability of the cost. So in general, you still have a $O(1/\sqrt{t})$ convergence guarantee.

However, adding strong convexity, that is, considering functions $f \in \mathcal{S}^{1,1}_{m,+\infty}$, can help the subgradient method to converge linearly to an error ball. In this sense, the subgradient method obtains two phases, a fast phase up to an error ball, and a slow phase, to the true optimizer.

Let's see how to show that. First, we need a improved strong convexity characterisation.

Lemma 2.5 *A function $f : \mathbf{R}^n \to \mathbf{R}$ is m-strongly convex iff $f(\boldsymbol{x}) - \frac{m}{2}\|\boldsymbol{x}\|^2$ is convex,* or equivalently

$$(\boldsymbol{v} - \boldsymbol{w})^\top (\boldsymbol{x} - \boldsymbol{y}) \geqslant m\|\boldsymbol{x} - \boldsymbol{y}\|^2, \qquad \forall \boldsymbol{v} \in \partial f(\boldsymbol{x}), \boldsymbol{w} \in \partial f(\boldsymbol{y}),$$

for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{R}^n$.

You can find the proofs in many standard references, such as E. K. Ryu and S. Boyd, *Primer on Monotone Operator Methods*, 2016.

Then, we can prove that,

Theorem 2.6 *The subgradient method on unconstrained problems with constant stepsize $\alpha < 1/2m$ for functions $f \in \mathcal{S}^{0,0}_{m,G}$ has a (global) convergence certificate of*

$$\lim_{k\to\infty} \|\boldsymbol{x}_k - \boldsymbol{x}^*\| = \sqrt{\frac{\alpha}{2m}} G. \tag{2.15}$$

Proof. *The function is strongly convex, so we have an unique optimiser $\boldsymbol{x}^*$. We start from the algorithm definition,*

$$\begin{aligned}
\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\|^2 &= \|\boldsymbol{x}_k - \boldsymbol{x}^*\|^2 - 2\alpha\langle \boldsymbol{g}_k, \boldsymbol{x}_k - \boldsymbol{x}^*\rangle + \alpha^2\|\boldsymbol{g}_k\|^2 \\
&= \|\boldsymbol{x}_k - \boldsymbol{x}^*\|^2 - 2\alpha\langle \boldsymbol{g}_k - \boldsymbol{g}^*, \boldsymbol{x}_k - \boldsymbol{x}^*\rangle + \alpha^2 G^2 \\
&\qquad \textit{since } \boldsymbol{g}^* \in \partial f(\boldsymbol{x}^*), \textit{ and we can take } \boldsymbol{g}^* = 0 \\
&\leqslant (1 - 2m\alpha)\|\boldsymbol{x}_k - \boldsymbol{x}^*\|^2 + \alpha^2 G^2 \qquad \textit{by strong convexity.}
\end{aligned}$$

Then, since $\alpha < 1/2m$, by geometric recursion,

$$\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\|^2 \leqslant (1 - 2m\alpha)^k \|\boldsymbol{x}_0 - \boldsymbol{x}^*\|^2 + \frac{\alpha G^2}{2m},$$

from which the thesis follows. ♣

From the proof, we can see how the error $\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\|^2$ decreases exponentially, up to an error floor $\frac{\alpha G^2}{2m}$. We can also see that if we want to reduce the error floor, we need to take α small, but this renders $(1 - 2m\alpha) \approx 1$, which damps convergence. This is a rather typical behaviour.

2.5 Damped Newton's method in the convex case

To close this chapter, we revisit the damped Newton's method we have seen in the previous class, and we provide a convergence certificate via the following theorem.

Theorem 2.7 *Apply the damped Newton's method on unconstrained problems, for cost functions $f \in \mathcal{S}^{2,2}_{m,M}$, and generate the sequence $\{\boldsymbol{x}_k\}$. Then, there exist two positive scalars η and γ with $0 < \eta \leqslant m^2/M$ and $\gamma > 0$ such that*

- *If $\|\nabla f(\boldsymbol{x}_k)\| \geqslant \eta$, then*
$$f(\boldsymbol{x}_{k+1}) - f(\boldsymbol{x}_k) \leqslant -\gamma.$$
- *If $\|\nabla f(\boldsymbol{x}_k)\| < \eta$, then the backtracking line search selects $\alpha_k = 1$ and*
$$\frac{M}{2m^2}\|\nabla f(\boldsymbol{x}_{k+1})\| \leqslant \left(\frac{M}{2m^2}\|\nabla f(\boldsymbol{x}_k)\|\right)^2.$$

As such, we obtain an accuracy of ϵ, i.e., $f(\boldsymbol{x}_k) - f^* \leqslant \epsilon$, in,
$$\frac{f(\boldsymbol{x}_0) - f^*}{\gamma} + \log_2\log_2\left(\frac{2m^3}{M^2\epsilon}\right)$$
iterations.

Proof. *The proof can be found in [BV] Section 9.5.3.* ♣

Theorem 2.7 tells us that the application of the damping strategy renders the Newton's method global. It can therefore converge from any initial points. The initial converge can be slow, but it is at least constant:
$$f(\boldsymbol{x}_{k+1}) - f(\boldsymbol{x}_k) \leqslant -\gamma,$$
for all k's. Then, when we are close to an optimiser, the quadratic convergence phase kicks in, and we converge very fast.

2.6 Main messages

We finish this lesson with a recap table and a figure. The table captures all the results we have presented in this chapter. Your goal is to learn this table and understand what each entry means. In particular, for unconstrained problems, you should know which method to use in which situations, with which convergence certificate and performance.

For the sake of completeness, we present here also a figure to give more tangible intuitions on how slow or how fast certain methods can be.

Figure 2.1 depicts the different order of convergence rate if we could put them all on the same plot, for the same error measure and number of iterations. This graph tells you that $O(1/\sqrt{t})$ is unbelievably slow, reducing the error of one order of magnitude each *two* orders of magnitude iterations. Read it again. To get one more digit right, you need two order of magnitude more iterations!

On the other hand of the spectrum, the Newton's method is incredibly fast: the number of correct digits of your optimiser roughly *doubles* at each step! Needless to say: Go Newton!

2.7 References

- YN: Introduction and Chapter 2
- BV: Chapter 9
- S. Bubeck's blog *I'm a bandit — The complexities of optimization*, post 12, 13, 14, 15. [link to blog]

◇ ◆ ◇

Table 2.1. *Summary of the certificates results obtained for nonconvex and convex unconstrained problems depending on the type of method and function class.*

Function type	generic	$\mathcal{C}_L^{1,1}, \mathcal{S}_{0,L}^{1,1}$	$\mathcal{C}_{m,L}^{1,1}, \mathcal{S}_{m,L}^{1,1}$	$\mathcal{C}_{m,M}^{2,2}, \mathcal{S}_{m,M}^{2,2}$
	First-order			Second-order
nonconvex	convergence $f \in \mathcal{C}^1$	$O(1/\sqrt{t})$	locally $O(\rho^t)$	locally $O(\rho^{2^t})$
	Theorem 1.4	Theorem 1.5	Theorem 1.6	Theorem 1.7
Convex	$O(1/\sqrt{t})$	$O(1/t)$, $O(1/t^2)$	$O(\rho^t)$	globally: const.,
		$\rightsquigarrow$ Nesterov's		locally $O(\rho_2^{2^t})$
	Theorem 2.4	Theorem 2.2-2.3	Theorem 2.1	Theorem 2.7
			+ Exercise 2.2	

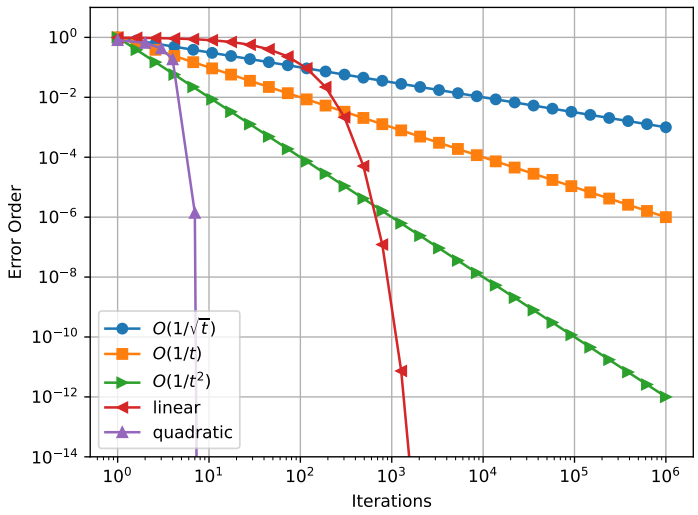

Figure 2.1. *The intuition behind the convergence rates.*

2.8 Exercises

Exercise 2.1 (Resit 2023) *We aim at minimizing the following quadratic problem,*

$$\min_{x\in\mathbf{R}^n} \frac{1}{2}x^\mathsf{T}Ax + b^\mathsf{T}x + c,$$

for A symmetric and positive definite, b a vector, and c a scalar. Assume that $\ell I_n \preceq A \preceq LI_n$, for two scalars $0 < \ell \leqslant L < +\infty$.

1. *Is the problem convex? Is the problem smooth? Is the problem strongly convex? Let $x^\star$ be an optimizer of the above problem. Prove that it is unique and find its value.*
2. *To solve the problem above, we look at the following variation of a gradient scheme. Start with random x_1, x_0 and define two stepsizes α, $\beta > 0$, then for $k > 1$ do:*

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

Prove that,

$$\begin{bmatrix} x_{k+1} - x^\star \\ x_k - x^\star \end{bmatrix} = \underbrace{\begin{bmatrix} (1+\beta)I_n - \alpha A & -\beta I_n \\ I_n & 0 \end{bmatrix}}_{=:T} \begin{bmatrix} x_k - x^\star \\ x_{k-1} - x^\star \end{bmatrix}$$

3. *Prove that the above algorithm converges (in the sense that $\|x_k - x^\star\| \to 0$) if the eigenvalues of T are inside the unit ball, meaning they are all in modulus less than 1.*
4. *Let $n = 1$, $A = \ell$, $\beta = 0$. Find the allowed range for the stepsize α to ensure convergence. In this case, determine also the rate of convergence.*

Exercise 2.2 *The aim of this exercise is to derive linear convergence guarantees.*

Consider the problem,

$$x^* \in \arg\min_{x\in\mathbf{R}^n} f(x),$$

for a μ-strongly convex and L-smooth function.

Consider the gradient method with constant stepsize $\alpha > 0$ to find the unique minimizer of f.

In class, we have see that,

$$\|x_k - x^*\| \leqslant \rho^k \|x_0 - x^*\|,$$

for all $\alpha < 2/L$. Start from it and,

- *Prove that,*

$$|f(x_k) - f(x^*)| \leqslant \kappa\rho^{2k}|f(x_0) - f(x^*)|,$$

- *Prove also that,*

$$\|\nabla f(x_k)\| \leqslant \kappa\rho^k\|\nabla f(x_0)\|, \qquad \kappa = L/m,$$

  which shows the effect of the condition number explicitly.

Exercise 2.3 *The aim of this exercise is to make you aware that different notion of convergences may not be necessarily better than others.*

Consider the problem,

$$x^* \in \arg\min_{x\in\mathbf{R}^n} f(x),$$

for a μ-strongly convex and L-smooth function.

Consider the gradient method with constant stepsize $\alpha > 0$ to find the unique minimizer of f. We have seen that the method converges as

$$\|x_k - x^*\| \leqslant \rho^k \|x_0 - x^*\|,$$

for all $\alpha < 2/L$. The convergence above is called linear convergence. Linear convergence of x_k is always better than $O(1/k)$ convergence of the same quantity. However, that may not be true for other quantities!

- *Prove that the gradient method in this case is also converging in an ergodic sense as,*

$$\|\bar{x}_k - x^*\| \leqslant \frac{1}{k+1}\frac{1}{1-\rho}\|x_0 - x^*\|, \qquad \bar{x}_k = \frac{1}{k+1}\sum_{i=0}^{k} x_i$$

for all $\alpha < 2/L$. In particular, show that linear convergence of x_k implies ergodic convergence as $O(1/k)$.

Take-home message: when you have ergodic *convergence as $O(1/k)$ it may not be so bad after all. And remember always to spell out convergence of what and with respect to what.*

Chapter 3

Third lecture

Constrained optimisation (I): first-order methods

3.1 Setting

We look at generic convex optimisation problems of the form

$$\begin{aligned} (\mathsf{PC}) \quad \underset{\boldsymbol{x}\in X\subseteq\mathbf{R}^n}{\text{minimise}} \quad & f(\boldsymbol{x}) && (3.1)\\ \text{subject to} \quad & g(\boldsymbol{x}) \leqslant 0 && (3.2)\\ & A\boldsymbol{x} = b, && (3.3) \end{aligned}$$

for which, we consider a convex cost function $f(\boldsymbol{x}) : \mathbf{R}^n \to \mathbf{R}$, convex inequality constraints $g(\boldsymbol{x}) : \mathbf{R}^n \to \mathbf{R}^l$, affine equality constraints, $h(\boldsymbol{x}) : \mathbf{R}^n \to \mathbf{R}^p \equiv A\boldsymbol{x} - b$, as well as a convex feasible set $X \subseteq \mathbf{R}^n$. From previous courses, see also the appendix, we know that (PC) is then a convex problem, for which necessary and sufficient conditions exist for finding its optimal points.

In this first class about the subject, we will focus on a specific version on (PC) and we will look at first-order methods (such as the gradient). In particular, we will look at proximal methods, projected gradient, dual ascent, and primal-dual methods.

I will give for granted some notions like KKT conditions and duality, but you can refer to the appendix for some recap.

3.1.1 Setting: simplified

We look at a specific convex optimisation problems of the form

$$(\mathsf{PC}) \quad \underset{\boldsymbol{x}\in X\subseteq\mathbf{R}^n}{\text{minimise}} \quad f(\boldsymbol{x}), \tag{3.4}$$

with a convex cost $f(\boldsymbol{x}) : \mathbf{R}^n \to \mathbf{R}$ and a convex feasible set $X \subseteq \mathbf{R}^n$. Whereby here X is a "simple" set.

The notion of a "simple" set is rather vague, but we can loosely characterise it as follows: a set is simple if and only if one can easily project over it (e.g., a box constraint, the semidefinite cone). This will be clearer later. For now think of X as a subset of all the possible convex sets.

We talk here of projection, since a typical methods for our simply constrained optimisation problem is the projected gradient method, which performs the iterations,

$$\boldsymbol{x}_{k+1} = \mathsf{P}_X[\boldsymbol{x}_k - \alpha\nabla f(\boldsymbol{x}_k)], \qquad \mathsf{P}_X(\boldsymbol{v}) := \arg\min_{\boldsymbol{x}\in X} \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{v}\|^2.$$

Here $\mathsf{P}_X(\boldsymbol{v})$ is the Euclidean projection of the vector $\boldsymbol{v}$ onto the convex set X. And therefore you can already see why I want the set X to be simple: I want its projection to be carried over with as little complexity as possible, ideally in closed-form.

All these notions will be cleared later, when properly formalised. To do that, we can start with a little more general setting than (PC). We will start from

$$(\mathsf{PS}) \quad \underset{\boldsymbol{x}\in\mathbf{R}^n}{\text{minimise}} \quad f_1(\boldsymbol{x}) + f_2(\boldsymbol{x}), \tag{3.5}$$

for convex costs $f_1(\boldsymbol{x}), f_2(\boldsymbol{x}) : \mathbf{R}^n \to \mathbf{R}$. This seems rather surprising, but this setting will be very fertile in terms of results and reach. Let's see how.

3.2 Splitting methods

Let's go back to our new cost in terms of $f_1(\boldsymbol{x}) + f_2(\boldsymbol{x})$. We consider problems for which,

- f_1 is convex and at least $f_1 \in \mathcal{S}^{1,1}_{0,L}$. In this context, f_1 will carry the favourable aspects of the optimisation problem;
- f_2 is the "rest". In general f_2 is non-differentiable and "just" closed convex proper (CCP).

We recall that,

Definition 3.1 (Closed convex proper functions) *The extended real line is $\overline{\mathbf{R}} := \mathbf{R} \cup \{+\infty\}$. An extended real convex function $f : \mathbf{R}^n \to \overline{\mathbf{R}}$ is proper iff is $\forall \boldsymbol{x} : f(\boldsymbol{x}) > -\infty$, and $\exists \boldsymbol{x}_0 : f(\boldsymbol{x}_0) < +\infty$. A proper convex function is closed iff it is lower semi-continuous (l.s.c.). A proper, l.s.c., convex function is indicated with $f \in \Gamma(\mathbf{R}^n)$.*

Example 3.1 (Indicator function) *Define the indicator function of a set X as the function $\iota_X : \mathbf{R}^n \to \overline{\mathbf{R}}$,*

$$\iota_X(\boldsymbol{x}) : \begin{cases} 0 & \text{if } x \in X, \\ +\infty & \text{otherwise.} \end{cases}$$

Then, the indicator function $\iota_X(\boldsymbol{x})$ of a closed and non-empty convex set X is $\Gamma(\mathbf{R}^n)$.

We know show that (PC) is a special case of (PS).

Proposition. (PC) and (PS) are have the same optimisers and optimal value iff f_2 is the indicator function for X.

Proof. *Consider*

$$\textit{(PC)} \quad \min_{\boldsymbol{x} \in X} f(\boldsymbol{x}), \qquad \textit{(PS')} \quad \min_{\boldsymbol{x} \in \mathbf{R}^n} f(\boldsymbol{x}) + \iota_{\boldsymbol{X}}(\boldsymbol{x})$$

and look at the optimality conditions

$$\textit{(PC-o)} \quad \nabla f(\boldsymbol{x}) + N_X(\boldsymbol{x}) \ni 0,$$

$$\textit{(PS'-o)} \quad \nabla f(\boldsymbol{x}) + \partial \iota_X(\boldsymbol{x}) \ni 0,$$

where $\partial \iota_X(\boldsymbol{x})$ is the subdifferential of $\iota_X(\boldsymbol{x})$. One can show (try it) that: $\partial \iota_X(\boldsymbol{x})$ is nothing else than the normal cone of X, i.e., $\partial \iota_X(\boldsymbol{x}) = N_X(\boldsymbol{x})$, for which the claim follows. ♣

Having established that our $f_1 + f_2$ is more general than the constrained setting, let us look at what we can do with it.

3.2.1 Forward-backward splitting

The first question we may ask is how do we find the optimal points of $f_1 + f_2$? If we studied well the second class, we know that if we were to use the subgradient method we would have a $O(1/\sqrt{t})$ convergence rate. This is not very enticing. On the other hand, since f_2 may be non-differentiable, we are not equipped with any better methods so far.

The following theorem may offer us some hints on how to proceed.

Theorem 3.1 *Let $\mathcal{A} = \nabla f_1$ and $\mathcal{B} = \partial f_2$. Then the following conditions are equivalent (for any constant $\alpha > 0$),*

$$\textit{(a)} \quad (\mathcal{A} + \mathcal{B})(\boldsymbol{x}) \ni 0 \tag{3.6}$$

$$\textit{(b)} \quad \boldsymbol{x} = \boldsymbol{x} - \alpha(\mathcal{A}(\boldsymbol{x}) + \boldsymbol{y}),\ \boldsymbol{y} \in \mathcal{B}(\boldsymbol{x}) \tag{3.7}$$

$$\textit{(c)} \quad \boldsymbol{x} + \alpha\mathcal{B}(\boldsymbol{x}) \ni \boldsymbol{x} - \alpha\mathcal{A}(\boldsymbol{x}) \tag{3.8}$$

$$\textit{(d)} \quad (I + \alpha\mathcal{B})(\boldsymbol{x}) \ni (I - \alpha\mathcal{A})(\boldsymbol{x})\ . \tag{3.9}$$

Proof. *Immediate.* ♣

Let's look at these conditions very carefully. Condition (a) is just the optimality condition. Condition (b) describes a fixed point iteration: the subgradient method. When the subgradient method is converging, then (b) is verified and therefore we are at optimality. So far nothing new.

Conditions (c-d) describe a new fixed point iteration: the proximal method. The proximal method is an implicit method, in fact we can write:

$$\boldsymbol{x} + \alpha\mathcal{B}(\boldsymbol{x}) \ni \boldsymbol{x} - \alpha\mathcal{A}(\boldsymbol{x}) \iff \boldsymbol{x} \in (I + \alpha\mathcal{B})^{-1}(I - \alpha\mathcal{A})\boldsymbol{x}. \tag{3.10}$$

Let us stop for a moment. We have split the optimality condition into two parts $\mathcal{A}$ and $\mathcal{B}$, then applied a forward pass $I - \alpha\mathcal{A}$ on $\boldsymbol{x}$, and subsequently a backward pass $(I + \alpha\mathcal{B})^{-1}$. This gives the name of what we are doing: a forward-backward splitting. If $\mathcal{A}$ and $\mathcal{B}$ are not just any operators, but (as in our case) gradient and subgradient of some functions, then the splitting is also named the proximal method.

How do we compute $(I + \alpha\mathcal{B})^{-1}$ and what is it? Let us first define the proximal operator.

Definition 3.2 (Proximal operator) *Given a function $\varphi \in \Gamma(\mathbf{R}^n)$ and a positive constant $\alpha > 0$, the proximal operator $\mathsf{prox}_{\alpha\varphi} : \mathbf{R}^n \to \mathbf{R}^n$ is defined as,*

$$\mathsf{prox}_{\alpha\varphi}(\boldsymbol{y}) := \arg\min_{\boldsymbol{x}\in\mathbf{R}^n} \left\{ \frac{1}{2\alpha}\|\boldsymbol{x} - \boldsymbol{y}\|^2 + \varphi(\boldsymbol{x}) \right\}.$$

The proximal operator is the result of an optimisation problem. Since the cost function of the proximal operator is strongly convex for any positive α, then the solution is unique and the operator is well-defined. We will see later its properties, for now let's accept this definition and move on. How do we interpret $(I + \alpha\mathcal{B})^{-1}(I - \alpha\mathcal{A})\boldsymbol{x}$ in an algorithmic fashion?

Let us consider the step,

$$\boldsymbol{x} + \alpha\partial f_2(\boldsymbol{x}) \ni \boldsymbol{x} - \alpha\nabla f_1(\boldsymbol{x}), \tag{3.11}$$

which represents $\boldsymbol{x} + \alpha\mathcal{B}(\boldsymbol{x}) \ni \boldsymbol{x} - \alpha\mathcal{A}(\boldsymbol{x})$ with our choice of $\mathcal{A}$ and $\mathcal{B}$. Consider now the following line of reasoning: let $\boldsymbol{y}$ represent the right-hand side,

$$\begin{aligned} \boldsymbol{x} + \alpha\partial f_2(\boldsymbol{x}) &\ni \boldsymbol{x} - \alpha\nabla f_1(\boldsymbol{x}) := \boldsymbol{y} \\ \boldsymbol{x} + \alpha\partial f_2(\boldsymbol{x}) - \boldsymbol{y} &\ni 0. \end{aligned}$$

Now, the last inclusion can be interpreted as the optimality condition of an optimisation problem, which one? This one:

$$\begin{aligned} \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{y}\|^2 + \alpha f_2(\boldsymbol{x}) &\to \min \\ \boldsymbol{x} = \arg\min_{\boldsymbol{w}\in\mathbf{R}^n} \left\{ \frac{1}{2\alpha}\|\boldsymbol{w} - \boldsymbol{y}\|^2 + f_2(\boldsymbol{w}) \right\}&, \end{aligned}$$

which reminds us directly the definition of a proximal operator. In fact, the last equation is equivalent to,

$$\begin{aligned} \boldsymbol{x} &= \mathsf{prox}_{\alpha f_2}(\boldsymbol{y}) \qquad \text{by definition} \\ \boldsymbol{x} &= \mathsf{prox}_{\alpha f_2}(\boldsymbol{x} - \alpha\nabla f_1(\boldsymbol{x})). \end{aligned}$$

So, finally, we have interpreted the forward-backward step as a gradient and proximal step respectively, i.e., $(I + \alpha\mathcal{B})^{-1}(I - \alpha\mathcal{A})\boldsymbol{x}$ is $\boldsymbol{x} = \mathsf{prox}_{\alpha f_2}(\boldsymbol{x} - \alpha\nabla f_1(\boldsymbol{x}))$.

We are now fully ready for the proximal gradient method.

Remark 1 *Note, many splitting methods exist for different "structures". The proximal gradient method is one of the most used, e.g., in machine learning.*

3.3 The proximal gradient method

The method reads as follows, for a stepsize selection $\alpha_k > 0$

Proximal gradient method

- *Start with* $\boldsymbol{x}_0 \in \mathbf{R}^n$
- *Iterate* $\boldsymbol{x}_{k+1} = \mathsf{prox}_{\alpha_k f_2}(\boldsymbol{x}_k - \alpha_k \nabla f_1(\boldsymbol{x}_k)), \quad k = 0, 1, \ldots.$

The method is an upgrade of the gradient method, where we split the cost into a "nice" cost f_1 and the "rest", and we take a forward step on the nice part and a backward step on the rest.

When f_2 is the indicator function of a closed convex set ι_X, then the proximal gradient is equivalent to a projected gradient method, in fact,

$$\mathsf{prox}_{\alpha_k \iota_X}(\boldsymbol{y}) = \arg\min_{\boldsymbol{x}\in\mathbf{R}^n} \left\{ \frac{1}{2\alpha}\|\boldsymbol{x}-\boldsymbol{y}\|^2 + \iota_X(\boldsymbol{x}) \right\} = \arg\min_{\boldsymbol{x}\in X} \left\{ \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{y}\|^2 \right\} = \mathsf{P}_X[\boldsymbol{y}],$$

which is the Euclidean projection on X. So, once again, we are saying that the splitting strategy is a generalisation of the constrained problem (PC).

3.3.1 Prox-friendly functions

As you may already imagine, the proximal method is computationally reasonable when taking the proximal operator is computationally reasonable. To say it in another way, f_2 needs to be prox-friendly, otherwise the computations that we need to solve the optimisation problem in the definition of the proximal operator may exceed the computations to solve the whole problem.

If f_2 is prox-friendly, then the proximal gradient method can be applied. There are a few prox-friendly functions. Below two simple examples.

Example 3.2 (Prox-friendly functions 1) *If $f_2 = \iota_X$, when X is a box constraint, then taking the proximal can be done in closed-form. For example, when $X = [0,1]^n$, then,*

$$\boldsymbol{y}_{k+1} = (\boldsymbol{x}_k - \alpha_k \nabla f_1(\boldsymbol{x}_k)), \quad \boldsymbol{x}_{k+1} = \mathsf{P}_{[0,1]^n}(\boldsymbol{y}_{k+1}),$$

and, each component i of the projection can be computed as,

$$[\mathsf{P}_{[0,1]^n}(\boldsymbol{y}_{k+1})]_i = \max\{0, \min\{1, [\boldsymbol{y}_{k+1}]_i\}\},$$

where $[\bullet]_i$ is the i-component of the vector.

Proof. The proximal gradient reads,

$$x_{k+1} = \mathsf{P}_{[0,1]^n}(x_k - \alpha \nabla f(x_k))$$

Now the projection operator reads,

$$\mathsf{P}_{[0,1]^n}(y_{k+1}) = \arg\min_{z\in[0,1]^n} \frac{1}{2}\|z - y_{k+1}\|^2 = \arg\min_{z\in[0,1]^n} \sum_{i=1}^n \frac{1}{2}([z]_i - [y_{k+1}]_i)^2.$$

The above problem is separable into n optimisation problems, i.e.,

$$\arg\min_{z\in[0,1]^n} \sum_{i=1}^n \frac{1}{2}([z]_i - [y_{k+1}]_i)^2 = \sum_{i=1}^n \frac{1}{2} \arg\min_{[z]_i\in[0,1]} ([z]_i - [y_{k+1}]_i)^2,$$

whose solution is

$$[x_{k+1}]_i = [z]_i^* = \max\{0, \min\{1, [y_{k+1}]_i\}\}.$$

♣

Example 3.3 (Prox-friendly functions 2) *For $f_2(\boldsymbol{x}) = \|\boldsymbol{x}\|_1$, then the proximal operator is the soft-thresholding operator:*

$$[\mathsf{prox}_{\alpha\|\bullet\|_1}(\boldsymbol{y})]_i \equiv \text{sign}([\boldsymbol{y}]_i)(|[\boldsymbol{y}]_i| - \alpha)_+,$$

where $[\bullet]_i$ is the i-th component of the vector, sign is the sign operator, and $(\bullet)_+$ is $\max\{\bullet, 0\}$.

Proof. The proximal gradient reads,

$$x_{k+1} = \text{prox}_{\alpha|\bullet|_1}(x_k - \alpha\nabla f(x_k)) = \arg\min_{z\in\mathbf{R}^n}(\|z\|_1 + \frac{1}{2\alpha}\|z - x_k + \alpha\nabla f(x_k)\|^2)$$

Solving the proximal step calling $y_{k+1} = x_k + \alpha\nabla f(x_k)$,

$$\arg\min_{z\in\mathbf{R}^n}(\|z\|_1 + \frac{1}{2\alpha}\|z - y_{k+1}\|^2) \iff \alpha\partial\|z\|_1 + (z - y_{k+1}) = 0$$

The last set of equations reads, for all i:

$$\alpha\partial|[z]_i| + [z]_i = [y_{k+1}]_i$$

We have now several possibilities,

$$[z]_i > 0 \implies [z]_i = [y_{k+1}]_i - \alpha \quad \text{(which is valid then for } [y_{k+1}]_i > \alpha)$$

$$[z]_i < 0 \implies [z]_i = [y_{k+1}]_i + \alpha \quad \text{(which is valid then for } [y_{k+1}]_i < -\alpha)$$

$$[z]_i = 0 \implies \partial|[z]_i| \in [-1, 1] \text{ and } [y_{k+1}]_i \in [-\alpha, \alpha].$$

Reversing,

$$[x_{k+1}]_i = [z]_i^* = \begin{cases} [y_{k+1}]_i - \alpha & \text{for } [y_{k+1}]_i > \alpha \\ [y_{k+1}]_i + \alpha & \text{for } [y_{k+1}]_i < -\alpha \\ 0 & \text{otherwise} \end{cases}$$

and compactifying: $[x_{k+1}]_i = \mathsf{sign}([y_{k+1}]_i)(|[y_{k+1}]_i| - \alpha)_+$. ♣

3.3.2 Proximal properties

The proximal operator has a number of useful properties that we can give and leverage to prove the convergence of the proximal gradient method. We focus on one in this course.

Theorem 3.2 *The proximal operator of a function $f_2 \in \Gamma(\mathbf{R}^n)$ is non-expansive, meaning that:*

$$\forall\alpha > 0, \forall\boldsymbol{y}', \boldsymbol{y} \in \mathbf{R}^n : \|\mathsf{prox}_{\alpha f_2}(\boldsymbol{y}') - \mathsf{prox}_{\alpha f_2}(\boldsymbol{y})\| \leqslant \|\boldsymbol{y}' - \boldsymbol{y}\|. \tag{3.12}$$

Proof. *It follows from the implicit function theorem, and it is omitted here.* ♣

We are now ready for the convergence results.

3.3.3 Convergence of the proximal gradient method

We have the following.

Theorem 3.3 *Let $f_1 \in \mathcal{S}^{1,1}_{m,L}$ and $f_2 \in \Gamma(\mathbf{R}^n)$. Choose a constant stepsize $\alpha < 2/L$. Then the proximal gradient method on (PS) has the (global) convergence certificate of,*

$$\|\boldsymbol{x}_t - \boldsymbol{x}^*\| \leqslant \rho^t\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|, \tag{3.13}$$

for $\rho = \max\{|1 - \alpha m|, |1 - \alpha L|\}$, and t iterations.

Furthermore, let $f_1 \in \mathcal{S}^{1,1}_{0,L}$ and $f_2 \in \Gamma(\mathbf{R}^n)$. Choose a constant stepsize $\alpha < 2/L$. Then the proximal gradient method on (PS) has the (global) convergence certificate of,

$$|f_1(\boldsymbol{x}_t) + f_2(\boldsymbol{x}_t) - f_1(\boldsymbol{x}^*) - f_2(\boldsymbol{x}^*)| \leqslant \frac{\omega}{t+1}\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|^2, \tag{3.14}$$

for a constant $\omega > 0$, and t iterations.

Consider now the following accelerated variant of the proximal gradient method. Define the supporting sequences, as before, in Equation (2.12), and the **proximal accelerated gradient method** as

Proximal accelerated gradient method

- *Start with* $\boldsymbol{x}_0 = \boldsymbol{y}_0 \in \mathbf{R}^n$ *and the sequences* (2.12)
- *Iterate:* $\boldsymbol{y}_{k+1} = \mathsf{prox}_{\alpha_k f_2}(\boldsymbol{x}_k - \alpha \nabla f_1(\boldsymbol{x}_k))$, $\boldsymbol{x}_{k+1} = (1-\gamma_k)\boldsymbol{y}_{k+1} + \gamma_k \boldsymbol{y}_k$, $k = 0, 1, \dots$.

Then,

Theorem 3.4 *Let* $f_1 \in \mathcal{S}^{1,1}_{0,L}$ *and* $f_2 \in \Gamma(\mathbf{R}^n)$. *Choose a constant stepsize* $\alpha < 1/L$. *Then the proximal accelerated gradient method on (PS) has the (global) convergence certificate of,*

$$|f_1(\boldsymbol{x}_t) + f_2(\boldsymbol{x}_t) - f_1(\boldsymbol{x}^*) - f_2(\boldsymbol{x}^*)| \leqslant \frac{\omega}{t^2}\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|^2, \tag{3.15}$$

for a constant $\omega > 0$, *and* t *iterations.*

Proof. *To prove the first statement of Theorem 3.3, we use the non-expansivity property. Start from the algorithm update,*

$$\begin{aligned}\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\| = \|\mathsf{prox}_{\alpha f_2}(\boldsymbol{x}_k - \alpha\nabla f_1(\boldsymbol{x}_k)) - \mathsf{prox}_{\alpha f_2}(\boldsymbol{x}^* - \alpha\nabla f_1(\boldsymbol{x}^*))\| \leqslant \\ \leqslant \|\boldsymbol{x}_k - \alpha\nabla f_1(\boldsymbol{x}_k) - \boldsymbol{x}^* + \alpha\nabla f_1(\boldsymbol{x}^*)\|. \end{aligned} \tag{3.16}$$

Then, the rest follows as in the proof of Theorem 2.1.

The remaining results ($f_1 \in \mathcal{S}^{1,1}_{0,L}$ *standard and accelerated) are more involved and omitted here.* ♣

3.3.4 Interpretation of the results

The results in Theorem 3.3 and 3.4 are quite impressive. If compared to the results we obtained in Theorem 2.1 and 2.3 for the unconstrained case, they say that the proximal gradient method has exactly the same guarantees of an unconstrained algorithm that does not have the complicated f_2 part.

This is brilliant. Whenever f_2 is a prox-friendly function, we can just apply the proximal gradient method and its accelerated variant and forget it exists for the convergence analysis.

In particular for strongly convex and smooth f_1 costs, we obtain a linear convergence rate, while for just smooth f_1 costs, we obtain a $O(1/k)$ and $O(1/k^2)$ rate. The latter in case of a Nesterov's acceleration.

Incidentally, in the case $f_1 \equiv 0$, then we obtain a better method than the subgradient method, for the case of prox-friendly f_2, called proximal point algorithm.

In this section, we have see that (PC) can be generalized into (PS) and we have introduced your first splitting method to solve it (the proximal gradient algorithm). Albeit the method works best when f_2 and therefore the set X is prox-friendly, the proximal gradient is one of the workhorses of machine learning and modern optimisation as first-order algorithms are concerned.

In fact, researchers are very good at finding prox-friendly f_2's.

The proximal gradient splits the objective in a nice part and in a not-so nice part, and then convergence goes as if only the nice part existed!

This brings us to the cases in which we can't avoid looking at non-prox-friendly constraints.

3.3.5 Numerical example

Before continuing, it is good to have a look at a specific numerical example, which you will explore in the second numerical project. We consider an image deblurring task: given an blurred image, our task is to retrieve the original, sharp one.

The task can be formulated in different ways. One possibility is to write an optimisation problem of the form,

$$x^* \in \arg\min_{x \in \mathbf{R}^n} \frac{1}{2}\|Ax - b\|_2^2 + \epsilon\|x\|_1, \tag{3.17}$$

for given matrix $A \in \mathbf{R}^{m \times n}$, vector $b \in \mathbf{R}^m$ and regularisation parameter $\epsilon > 0$. The regularisation is added, since typically the matrix A has very few rows $m \ll n$, and the problem can be ill-conditioned.

The cost function is convex, but non-differentiable, nor strongly convex. At face value, we should use the subgradient method. However, considering the splitting, $f_1(x) = \frac{1}{2}\|Ax - b\|_2^2$ and $f_2(x) = \epsilon\|x\|_1$, we can use the proximal gradient. Since f_1 is not just differentiable, but also smooth, we can also use the accelerated proximal gradient. The proximal gradient for this ℓ_1-regularised problem is traditionally called ISTA (i.e., iterative soft-thresholding algorithm), while the accelerated variant is called FISTA (for fast ISTA), since we are using the soft-thresholding operator to solve the proximal step (see Example 3.3).

In Figure 3.1, you can see the convergence curves for the three methods, and appreciate the added value of the acceleration. Also, in Figure 3.2, you can see the end result. Note that in this example $m = 92416, n = 131072$.

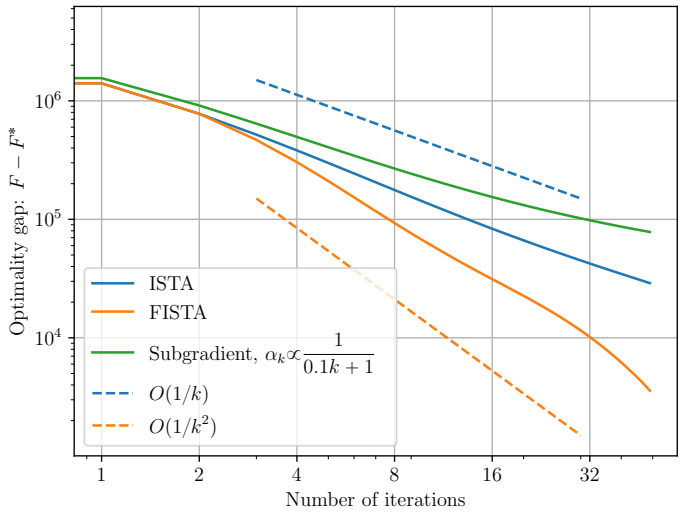

Figure 3.1. *Convergence plots for Problem* (3.17).

3.4 Duality

3.4.1 Equality constrained problems

We look now at a more explicit setting, for not so simple linear equality constraints,

$$\text{(PE)} \qquad \underset{\boldsymbol{x} \in \mathbf{R}^n}{\text{minimise}} \quad f(\boldsymbol{x}) \tag{3.18}$$

$$\text{subject to} \quad A\boldsymbol{x} = b, \tag{3.19}$$

where A is a $\mathbf{R}^{p \times n}$ real-valued matrix, and $b \in \mathbf{R}^p$ with b in the image of A for feasibility. Here $f : \mathbf{R}^n \to \mathbf{R}$ is a convex function, as usual.

Figure 3.2. *Image results for Problem (3.17). Original file by Camille Enlart, reproduced under Creative Commons Attribution-Share Alike 4.0 International license.*

We will look at first-order dual methods, so we will need the Lagrangian function for the problem above, which reads,

$$\mathcal{L}(\boldsymbol{x}, \lambda) = f(\boldsymbol{x}) + \langle \lambda, A\boldsymbol{x} - b \rangle = f(\boldsymbol{x}) + \lambda^\top (A\boldsymbol{x} - b).$$

We also need the dual problem,

$$\max_{\lambda \in \mathbf{R}^p} q(\lambda), \qquad q(\lambda) := \inf_{\boldsymbol{x} \in \mathbf{R}^n} \mathcal{L}(\boldsymbol{x}, \lambda),$$

where $q(\lambda) : \mathbf{R}^p \to \mathbf{R}$ is the so-called dual function.

Dual problems are useful, since when there is no duality gap then the maximum of the dual problem coincides with the minimum of the primal problem; so if the former is easier to solve, we can just solve it instead of the latter.

Here, you should know that when f convex and constraints are linear (and feasible), constraint qualification holds, and there is no duality gap. If you don't remember the details, please refer to OPT201, or the appendix.

3.4.2 The dual ascent method

If we look at the dual problem, we see that it is an unconstrained problem in λ, so we could optimise it via a gradient method. Indeed, the method (renamed Dual Ascent/ Uzawa's method), look like the following:

- Start with $\lambda_0 \in \mathbf{R}^p$
- Iterate $\lambda_{k+1} = \lambda_k + \alpha_k \boldsymbol{y}, \quad \boldsymbol{y} \in \partial q(\lambda_k) \quad k = 0, 1, \ldots.$

We already remark the following important point: in general we don't know if the dual function q is differentiable, so we may need to use the subgradient of the dual function.

You may have already noticed, the properties of convergence and rate of the dual ascent method are linked to the properties of the dual function q. In particular, most of the theorems we have already proved in the second lesson will apply here, once we can determine the properties of q *in terms of* properties of the original cost f and constraints A.

Remark 2 *We give for granted some standard linear algebra results, like the notion of singular values and how to compute them. If you are not familiar with that, please revise Appendix A.5 of [BV].*

We have the following key result.

Theorem 3.5 *Consider the problem (PE). The dual function $q(\lambda)$ is always concave. Furthermore, if we label the singular values of A as $\sigma_{\min} \leqslant \cdots \leqslant \sigma_{\max}$, then the following statements hold.*

(a) If $f \in \mathcal{S}^{1,1}_{m,+\infty}, \sigma_{\max} > 0$, then $-q \in \mathcal{S}^{1,1}_{0,L'}$ with $L' = \sigma^2_{\max}/m$.

(b) If $f \in \mathcal{S}^{1,1}_{0,L}, \sigma_{\min} > 0$, then $-q \in \mathcal{S}^{1,1}_{m',+\infty}$ with $m' = \sigma^2_{\min}/L$.

(c) The converse is true if f is convex.

Proof. *We prove only the second part of the theorem, since you should know that the dual function is always concave from OPT201.*

We start by (a). If $f \in \mathcal{S}^{1,1}_{m,+\infty}, \sigma_{\max} > 0$, then the function,

$$\boldsymbol{x}^*(\lambda) = \arg\min_{\boldsymbol{x}} \; f(\boldsymbol{x}) + \lambda^\top (A\boldsymbol{x} - b)$$

is single-valued, since the solution is unique (due to strong convexity of the cost). Optimality conditions state:

$$\nabla f(\boldsymbol{x}^*(\lambda)) = -A^\top \lambda.$$

Strong convexity of $f(\boldsymbol{x})$ also implies,

$$\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\| \|\boldsymbol{x} - \boldsymbol{y}\| \geqslant \langle \nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y} \rangle \geqslant m \|\boldsymbol{x} - \boldsymbol{y}\|^2, \qquad \forall \boldsymbol{x}, \boldsymbol{y},$$

hence:

$$\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\| \geqslant m \|\boldsymbol{x} - \boldsymbol{y}\|, \qquad \forall \boldsymbol{x}, \boldsymbol{y},$$

and setting $\boldsymbol{x} = \boldsymbol{x}^*(\lambda)$, $\boldsymbol{y} = \boldsymbol{x}^*(\lambda')$, we obtain

$$\frac{1}{m} \|A^\top \lambda' - A^\top \lambda\| \geqslant \|\boldsymbol{x}^*(\lambda) - \boldsymbol{x}^*(\lambda')\|. \tag{3.20}$$

Consider now the dual function's gradient along any direction $e \in \mathbf{R}^p$,

$$\begin{aligned} e^\top \nabla q(\lambda) &= \lim_{\epsilon \to 0} \frac{\min_{\boldsymbol{x}} \left(f(\boldsymbol{x}) + (\lambda^\top + \epsilon e^\top)(A\boldsymbol{x} - b) \right) - \min_x \left(f(\boldsymbol{x}) + \lambda^\top (A\boldsymbol{x} - b) \right)}{\epsilon} \\ &= \lim_{\epsilon \to 0} \frac{\left(f(\boldsymbol{x}^*(\lambda + \epsilon e)) + (\lambda^\top + \epsilon e^\top)(A\boldsymbol{x}^*(\lambda + \epsilon e) - b) \right) - \left(f(\boldsymbol{x}^*(\lambda)) + \lambda^\top (A\boldsymbol{x}^*(\lambda) - b) \right)}{\epsilon} \end{aligned}$$

Use then Taylor,

$$\begin{aligned} f(\boldsymbol{x}^*(\lambda + \epsilon e)) = f(\boldsymbol{x}^*(\lambda)) + \nabla f(\boldsymbol{x}^*(\lambda))(\boldsymbol{x}^*(\lambda + \epsilon e) - \boldsymbol{x}^*(\lambda)) + O(\|\epsilon\|^2) = \\ f(\boldsymbol{x}^*(\lambda)) - \lambda^\top A(\boldsymbol{x}^*(\lambda + \epsilon e) - \boldsymbol{x}^*(\lambda)) + O(\|\epsilon\|^2), \end{aligned}$$

where we have used the Lipschitz property of $\boldsymbol{x}^*(\lambda)$ for $O(.)$ (Eq. (3.20)) and the optimality condition.

Substituting the latter into the gradient of the dual,

$$\begin{aligned} e^\top \nabla q(\lambda) &= \lim_{\epsilon \to 0} \frac{-\lambda^\top A(\boldsymbol{x}^*(\lambda + \epsilon e) - \boldsymbol{x}^*(\lambda)) + (\lambda^\top + \epsilon e^\top)(A\boldsymbol{x}^*(\lambda + \epsilon e) - b) - \lambda^\top (A\boldsymbol{x}^*(\lambda) - b)}{\epsilon} \\ &= e^\top (A\boldsymbol{x}^*(\lambda) - b) \end{aligned}$$

and therefore $\nabla q(\lambda) = A\boldsymbol{x}^*(\lambda) - b$.

Lipschitz continuity of the gradient means analyzing:

$$\|\nabla q(\lambda) - \nabla q(\lambda')\| = \|A\boldsymbol{x}^*(\lambda) - A\boldsymbol{x}^*(\lambda')\| \leqslant \frac{\sigma^2_{\max}}{m} \|\lambda - \lambda'\|,$$

where we have used the Lipschitz property of $\boldsymbol{x}^*(\lambda)$ (Eq. (3.20)), which proves the claim.

We continue by proving (b). If $f \in \mathcal{S}^{1,1}_{0,L}, \sigma_{\min} > 0$, we need to prove strong convexity of $-q$. Here we cheat a bit by allowing f to be strictly convex, so q is differentiable, but the proof can be

extended to the general case with a little more work. So, with the cheat, we need a lower bound as,

$$\langle -\nabla q(\lambda) + \nabla q(\lambda'), \lambda - \lambda' \rangle \geqslant \kappa \|\lambda - \lambda'\|^2,$$

but,

$$\begin{aligned} \langle -\nabla q(\lambda) + \nabla q(\lambda'), \lambda - \lambda' \rangle &= \langle -A\boldsymbol{x}^*(\lambda) + A\boldsymbol{x}^*(\lambda'), \lambda - \lambda' \rangle \\ &= \langle -\boldsymbol{x}^*(\lambda) + \boldsymbol{x}^*(\lambda'), A^\top \lambda - A^\top \lambda' \rangle \\ &= \langle -\boldsymbol{x}^*(\lambda) + \boldsymbol{x}^*(\lambda'), \nabla f(\boldsymbol{x}^*(\lambda')) - \nabla f(\boldsymbol{x}^*(\lambda)) \rangle \end{aligned}$$

For L-smoothness of f,

$$\langle -\boldsymbol{x}^*(\lambda) + \boldsymbol{x}^*(\lambda'), \nabla f(\boldsymbol{x}^*(\lambda')) - \nabla f(\boldsymbol{x}^*(\lambda)) \rangle \geqslant \frac{1}{L} \|\nabla f(\boldsymbol{x}^*(\lambda')) - \nabla f(\boldsymbol{x}^*(\lambda))\|^2,$$

and using the optimality conditions,

$$\langle -\nabla q(\lambda) + \nabla q(\lambda'), \lambda - \lambda' \rangle \geqslant \frac{\sigma_{\min}^2}{L} \|\lambda - \lambda'\|^2,$$

as required.

We leave (c) for homework. ♣

This theorem is key: it relates the properties of f to the properties (i.e., the functional class) of q. In particular,

- If f is strongly convex and A is not identically zero, then the negative of the dual function $-q$ is smooth. I can then apply a gradient method, and its accelerated variant with a convergence rate of $O(1/t)$ and $O(1/t^2)$, respectively. I can also compute the Lipschitz constant, and therefore I know the stepsize upper bound.
- If f is smooth and A is full row rank, then the negative of the dual function $-q$ is strongly convex, but not necessarily differentiable. I will have to apply a subgradient method, with a convergence rate of $O(1/\sqrt{t})$.
- If f is both strongly convex and smooth, and A is full row rank, then $-q$ is also strongly convex and smooth, and I can apply a gradient method with linear convergence guarantees.

This is all quite good. However, we need to compute the gradient or the subgradient of $-q$. How do we proceed?

Theorem 3.6 *Consider (PE) and $f \in \Gamma(\mathbf{R}^n)$. Indicate with* conv *the convex hull, then:*

$$\partial q(\lambda) = \mathsf{conv}(A\boldsymbol{x}^*(\lambda) - b),$$

where, $\boldsymbol{x}^*(\lambda) \in \arg\min_{\boldsymbol{x} \in \mathbf{R}^n} \mathcal{L}(\boldsymbol{x}, \lambda)$, which is a set in general.

If f is strictly convex, then $\boldsymbol{x}^*(\lambda)$ is a singleton for every λ, and $\partial q(\lambda) \equiv \nabla q(\lambda)$, i.e., q is differentiable.

Proof. *The proof follows from the proof of Theorem 3.5 and the definition of subdifferential. It is also know as Danskin's theorem, and an alternate proof can be found in Bestekas, Nonlinear Programming, Section B.25.* ♣

Armed with our gradient definition, we can rewrite the dual ascent method as,

Dual (sub)-gradient ascent method

- *Start with $\lambda_0 \in \mathbf{R}^p$*
- *Iterate:*
  - *Solve the primal optimisation problem: $\boldsymbol{x}^*(\lambda_k) \in \arg\min_{\boldsymbol{x} \in \mathbf{R}^n} \mathcal{L}(\boldsymbol{x}, \lambda_k)$,*
  - *Update: $\lambda_{k+1} = \lambda_k + \alpha_k(A\boldsymbol{x}^*(\lambda_k) - b) \quad k = 0, 1, \ldots$*

In the algorithm, we have already indicated the gradient of the dual function as $A\boldsymbol{x}^*(\lambda_k) - b$. When the dual function is not differentiable, the algorithm does not change and we take an

element of the subdifferential of q, i.e., an optimiser $\boldsymbol{x}^*(\lambda_k)$. We recall that we do not need to take the whole subdifferential set.

If q is smooth, then we can accelerate the gradient via a Nesterov's acceleration in pretty much the same way we discussed in the previous lectures.

The convergence and rates of the (sub)-gradient ascent method and its accelerated variant, in terms of the sequence $\{\lambda_k\}$ as well as the cost $q(\lambda_k)$ follow readily from the discussion we had above, and in particular, let's make it a theorem.

Theorem 3.7 *Consider problem (PE). By using the (sub)-gradient dual ascent method we generate a sequence $\{\lambda_k\}$, which has the following convergence certificates,*

- *If $f \in \mathcal{S}^{1,1}_{m,+\infty}, \sigma_{\max} > 0$, and the stepsize is chosen constant as $\alpha < 2m/\sigma^2_{\max}$ then,*

$$q^* - q(\lambda_k) \leqslant O(1/t),$$

  for a number of iterations t.

- *If $f \in \mathcal{S}^{1,1}_{0,L}, \sigma_{\min} > 0$, then choosing the stepsize as in Theorem 2.4, at best,*

$$\min_{k=0,\dots,t} q^* - q(\lambda_k) \leqslant O(1/\sqrt{t}),$$

  for a number of iterations t.

- *If $f \in \mathcal{S}^{1,1}_{m,L}, \sigma_{\min} > 0$, then we can choose the stepsize as $\alpha < 2m/\sigma^2_{\max}$ to obtain,*

$$\|\lambda_k - \lambda^*\| \leqslant O(\rho^t),$$

  for a number of iterations t and $\rho = \max\{|1 - \alpha\sigma^2_{\min}/L|, |1 - \alpha\sigma^2_{\max}/m|\} < 1$.

Bu using a Nesterov's accelerated version, then, if $f \in \mathcal{S}^{1,1}_{m,+\infty}, \sigma_{\max} > 0$, and the stepsize is chosen constant as $\alpha < m/\sigma^2_{\max}$ then,

$$q^* - q(\lambda_k) \leqslant O(1/t^2),$$

for a number of iterations t.

Proof. *The proof follows from the discussion above.* ♣

The theorem tells us how to construct dual sequences that convergence in a pertinent sense. We recall that strong duality holds, and as such $f^* = q^*$, so we are converging to the minimum of our problem. However, no mention is given to the primal variable $\boldsymbol{x}^*$. That is, given a dual sequence, how can we recover the corresponding primal variable? This is important, right? At the end of the day, you want to solve (PE) and you don't necessarily care about its dual variant.

Theorem 3.8 (Primal recovery) *Consider problem (PE). If $f \in \mathcal{S}^{1,1}_{m,L}$, $L \leqslant +\infty$, then we can set*

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x} \in \mathbf{R}^n} \mathcal{L}(\boldsymbol{x}, \lambda^*).$$

Otherwise, construct the auxiliary ergodic variable,

$$\bar{\boldsymbol{x}}_k = \frac{1}{k+1} \sum_{i=0}^{k} \boldsymbol{x}_i, \quad \boldsymbol{x}_i := \boldsymbol{x}^*(\lambda_i).$$

Then, $\|\bar{\boldsymbol{x}}_k - \boldsymbol{x}^*\| \to 0$.

Proof. *The first part of the proof is obvious, since $\boldsymbol{x}^*$ is unique. The second part is more involved and left here.* ♣

Summarising: We have transformed (PE) into an unconstrained dual problem, which we can fully characterise and solve as we did in the second class.

Dual ascent is only one of the possible dual algorithms, and not even the best performing. Many algorithms out there, but all of them follow the basic principles that we have looked at here. In particular, you use duality to derive properties of the dual problem; then you use your basic unconstrained theory to derive convergence certificates. Finally, you can say something about primal recovery in some more complex cases.

Homework. Can you think of a dual Newton's method? Can you think of a dual proximal method? When would you apply these methods and with which guarantees?

Before moving on, stop a bit and reflect. In the proximal setup, convergence certificates were very easy to obtain, and the disadvantage was to have prox-friendly costs. In the dual setup, convergence certificates require a bit more work, and now we need that the gradient of the dual function is easy to compute. This latter requirement means that we can compute

$$\boldsymbol{x}^*(\lambda_k) = \arg\min_{\boldsymbol{x}\in\mathbf{R}^n} \mathcal{L}(\boldsymbol{x}, \lambda_k)$$

in an easier way than solving the full, constrained, problem. What if it is not the case? Let's look at primal-dual methods.

3.4.3 Inequality and equality constrained problems

When we play with dual ascent methods, you may be tempted to try out different modifications. As we have just discussed, you may find the optimisation $\arg\min_{\boldsymbol{x}\in\mathbf{R}^n} \mathcal{L}(\boldsymbol{x}, \lambda_k)$ awkward, and tempted to substitute it with one or a few steps a gradient method,

$$\boldsymbol{x}_{\kappa+1} = \boldsymbol{x}_\kappa - \alpha_\kappa \nabla_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}_\kappa, \lambda_k), \quad \kappa = 0, 1, \ldots, T,$$

where you could start with $\boldsymbol{x}_\kappa = \boldsymbol{x}_k$, and let $\boldsymbol{x}^*(\lambda_k) \approx \boldsymbol{x}_{T+1}$. This is of course a naive modification that may (or not) work. This type of modifications goes under the name of primal-dual methods, since we both change the primal and the dual at the same time. Let's look at a more general setting and then specify our results for a few algorithms.

We look now at the complete setting

$$\begin{aligned} (\mathsf{PI}) \qquad \underset{\boldsymbol{x}\in X\subseteq\mathbf{R}^n}{\text{minimise}} \quad & f(\boldsymbol{x}) && (3.21)\\ \text{subject to} \quad & A\boldsymbol{x} = b,\ g(\boldsymbol{x}) \leqslant 0, && (3.22) \end{aligned}$$

where A is a $\mathbf{R}^{p\times n}$ real-valued matrix, and $b \in \mathbf{R}^p$ with b in the image of A for feasibility. The functions $f : \mathbf{R}^n \to \mathbf{R}$ and $g : \mathbf{R}^n \to \mathbf{R}^l$ are convex functions, and X is a convex set. Assume that Slater's constraint qualification holds (i.e. $\exists \bar{\boldsymbol{x}} : \bar{\boldsymbol{x}} \in \mathsf{relint}(X), g(\bar{\boldsymbol{x}}) < 0$). In this context, problem (PI) is convex and there is no duality gap.

Notice that (PE) $\subset$ (PI).

We look here at primal-dual methods, whose convergence is harder and more technical to characterise, and mostly open in many settings. But, we will see what we can say, at least in a simple case.

3.4.4 A primal-dual method

Start by considering f, g to be differentiable. A simple (perhaps naive) primal gradient descent - dual ascent method is the following.

Form the Lagrangian

$$\mathcal{L}(\boldsymbol{x}, \lambda, \nu) = f(\boldsymbol{x}) + \lambda^\mathsf{T}(A\boldsymbol{x} - b) + \nu^\mathsf{T} g(\boldsymbol{x}),$$

then look at the saddle-point problem,

$$\min_{\boldsymbol{x}\in X} \max_{\lambda\in\mathbf{R}^p, \nu\in\mathbf{R}^l_+} \mathcal{L}(\boldsymbol{x}, \lambda, \nu). \tag{3.23}$$

We recall here that the dual variable associated to the inequality constraint $g(\boldsymbol{x}) \leqslant 0$ has to be constrained in the positive orthant, i.e., $\nu \in \mathbf{R}^l_+$. This comes from the KKT conditions.

Then, one can ask how to solve (3.23), for instance iterating as follows (where we indicate with P the projection operator).

(Basic) Primal-dual descent-ascent method

- *Start with* $\boldsymbol{x}_0, \lambda_0, \nu_0$ *and stepsizes* $\alpha, \beta, \gamma > 0$,
- *Iterate for* $k = 0, 1, \ldots$:

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \mathsf{P}_X[\boldsymbol{x}_k - \alpha \nabla_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}_k, \lambda_k, \nu_k)] \\ \lambda_{k+1} &= \lambda_k + \beta \nabla_{\lambda} \mathcal{L}(\boldsymbol{x}_k, \lambda_k, \nu_k) \\ \nu_{k+1} &= \mathsf{P}_{\mathbf{R}^l_{\geqslant 0}}[\nu_k + \gamma \nabla_{\nu} \mathcal{L}(\boldsymbol{x}_k, \lambda_k, \nu_k)] \end{aligned}$$

This primal-dual method (a method which alternates between a primal update and a dual update) is rather basic. Its convergence is hard to characterise in general (since $\mathcal{L}$ is not strictly concave in λ, ν). Better and more involved methods exists that tune the stepsize or take more or less steps in the primal and in the dual. Also, you can find methods that add penalisation or regularisation terms to help convergence. We do not explore this here, but it is useful to understand to which we are converging to, in general.

Convergence is to a saddle point, i.e., a point $(\boldsymbol{x}^*, \lambda^*, \nu^*)$ such that:

$$\forall \boldsymbol{x} \in X, \lambda, \nu \in \mathbf{R}^l_{\geqslant 0} \qquad \mathcal{L}(\boldsymbol{x}^*, \lambda, \nu) \leqslant \mathcal{L}(\boldsymbol{x}^*, \lambda^*, \nu^*) \leqslant \mathcal{L}(\boldsymbol{x}, \lambda^*, \nu^*).$$

This convergence is hard to characterise since you are not descending, not ascending with a certain rate, but you are alternating. So convergence involves determining how far you are from the saddle point, and how much you are violating your constraints.

I stop here, but let's have a look at one interesting example.

3.4.5 Convergence: an example

Let's go back to our (PE) and try to solve it with our basic primal-dual method. In this case $g \equiv 0$, and we don't need ν. We have the following theorem.

Theorem 3.9 *Let* $f \in \mathcal{S}^{1,1}_{m,L}$ *and* $g \equiv 0$. *Let* A *be full row rank. Then, the primal-dual algorithm with* $\alpha = \beta < 1/L$ *delivers a sequence* $\{\boldsymbol{x}_k, \lambda_k\}_{k \in \mathbf{N}}$ *such that,*

$$\{\boldsymbol{x}_k, \lambda_k\}_{k \in \mathbf{N}} \to (\boldsymbol{x}^*, \lambda^*)$$

linearly.

Proof. *The proof is interesting since we are going to use a different tool: Energy decrease! By the assumptions on the function and problem, we know that the saddle point* $(\boldsymbol{x}^*, \lambda^*)$ *exists and is unique.*

Consider then the variable: $\boldsymbol{z}_k = \mathrm{col}(\boldsymbol{x}_k - \boldsymbol{x}^*, \lambda_k - \lambda^*)$, *and the energy function* $V_k = \frac{1}{2}\|\boldsymbol{z}_k\|^2$. *If we show that the energy decreases with* k, *then* $\|\boldsymbol{z}_k\| \to 0$. *The energy function* V_k *is the so-called Lyapunov function, for the students who know what I am talking about.*

First, we can write

$$\boldsymbol{z}_{k+1} = \boldsymbol{z}_k + \alpha \begin{bmatrix} -\nabla_{\boldsymbol{x}} f(\boldsymbol{x}_k) - A^\mathsf{T} \lambda_k + \nabla_{\boldsymbol{x}} f(\boldsymbol{x}^*) + A^\mathsf{T} \lambda^* \\ A\boldsymbol{x}_k - b - A\boldsymbol{x}^* + b \end{bmatrix}.$$

Then, by using Taylor expansion and the mean value theorem,

$$\nabla_{\boldsymbol{x}} f(\boldsymbol{x}_k) - \nabla_{\boldsymbol{x}} f(\boldsymbol{x}^*) = \nabla_{\boldsymbol{x}\boldsymbol{x}} f(\boldsymbol{\xi}_k)(\boldsymbol{x}_k - \boldsymbol{x}^*),$$

for a certain $\boldsymbol{\xi}_k$ *on the line with end point* $\boldsymbol{x}_k$ *and* $\boldsymbol{x}^*$. *In particular, let* $\mathbf{Q}_k = \nabla_{\boldsymbol{x}\boldsymbol{x}} f(\boldsymbol{\xi}_k)$. *Then,*

$$\boldsymbol{z}_{k+1} = \boldsymbol{z}_k - \alpha \begin{bmatrix} \mathbf{Q}_k & A^\mathsf{T} \\ -A & \mathbf{0} \end{bmatrix} \boldsymbol{z}_k = \begin{bmatrix} \mathbf{I} - \alpha \mathbf{Q}_k & -\alpha A^\mathsf{T} \\ \alpha A & \mathbf{I} \end{bmatrix} \boldsymbol{z}_k.$$

This is a dynamical system. Convergence to zero requires the system to be stable, therefore the system matrix to have eigenvalues in the unit circle. If the matrix $[\mathbf{Q}_k, A^\mathsf{T}; -A, \mathbf{0}]$ *has all the eigenvalues in the positive half plane and its maximum eigenvalue is bounded, then we can make the system matrix to have eigenvalues in the unit circle by choosing a small enough* α*.*

In order to check for the eigenvalues of $[\mathbf{Q}_k, A^\mathsf{T}; -A, \mathbf{0}]$ *to be positive, we need to verify that*

$$\boldsymbol{v}^\mathsf{T} \begin{bmatrix} \mathbf{Q}_k & A^\mathsf{T} \\ -A & \mathbf{0} \end{bmatrix} \boldsymbol{v} > 0, \quad \forall \boldsymbol{v},$$

which is trivially verified. In particular $\sigma_{\max}([\mathbf{Q}_k, A^\mathsf{T}; -A, \mathbf{0}]) = \sigma_{\max}(\mathbf{Q}_k) = \|\mathbf{Q}_k\| \leqslant L$*. Thus, choosing* $\alpha < 1/L$ *leads stability of the dynamical system and convergence of the primal-dual method. In particular,* $\boldsymbol{z}_k \to 0$*, linearly.*

3.4.6 Summarising

We have transformed (PI) into a simply-constrained saddle-point problem, which however it is more complex to characterise and solve than what we have seen for (PE) with purely dual methods. In fact, the research in primal-dual methods is very active and we do not have a complete picture yet.

As we have seen, when moving from simply-constrained problems to general problems, we find more and more general algorithms, which are more and more difficult to handle in theory. While the scope here is just to give you the tools to understand the methods, don't forget that there are many many methods out there, even if the basic ingredients are all the same!

3.5 References

- E. K. Ryu and S. Boyd, *Primer on Monotone Operator Methods*, 2016
- N. Parikh and S. Boyd, *Proximal Algorithms*, 2013
- J. Eckstein, *Splitting methods for monotone operators with applications to parallel optimisation*, 1989

3.6 Exercises

Exercise 3.1 (Exam 2023) *Consider the following quadratic optimisation problem in two scalar variables,*

$$\min_{x_1 \in \mathbf{R}, x_2 \in \mathbf{R}} \frac{1}{2}(x_1 - x_2)^2 + 4(x_1 - 1) + a(x_1^2 + 2x_2^2), \qquad \text{subject to } (x_1, x_2) \in X,$$

where a *is a real scalar, and* X *a generic set.*

1. *Prove that it is sufficient that* $a \geqslant 0$ *and* X *convex for the problem above to be convex.*
2. *Consider now* $a = 0$ *and* $X = \{x_1, x_2 \,|\, x_1 \in [0,1], x_2 \in [0,1]\}$*, give an example of a first-order algorithm that achieves the optimal convergence rate for this class of problems. Write the algorithm explicitly (that is, write every steps of the algorithm and detail all the operations involved including the allowed numerical values for the stepsize) and provide its optimal convergence certificate in terms of distance to* f^**.*
3. *Consider now* $a = 1$ *and* $X = \{x_1, x_2 \,|\, x_1 - x_2 = 1\}$*:*
   (a) *Derive the dual function. Which functional properties does this dual function have? (Be as precise as you can be: for example if the function is* L*-smooth, derive the numerical value for the constant* L*, and so forth).*

(b) Derive the dual problem. Which first-order algorithm would solve the dual problem with the fastest convergence rate? Write the algorithm explicitly (that is, write every steps of the algorithm and detail all the operations involved including the allowed numerical values for the stepsize), and provide a convergence certificate for it.

Exercise 3.2 (Resit 2023) *Consider the problem,*

$$(P) \qquad \min_{x \in \mathbf{R}} f(x) + \varphi(x),$$

where, $f(x) = \frac{1}{2}px^2$*, with* $p > 0$ *and,*

$$\varphi(x) = \begin{cases} \frac{1}{2}x^2 & \text{if } x \in [-1, 1] \\ |x| - \frac{1}{2} & \text{otherwise} \end{cases}.$$

1. *Is the problem convex? Strongly convex?*
2. *Prove that*
$$\boldsymbol{prox}_{\alpha\varphi}(x) = \begin{cases} \frac{x}{1+\alpha} & \text{if } |x| \leqslant 1 + \alpha \\ x - \alpha\, \boldsymbol{sign}(x) & \text{otherwise} \end{cases}.$$
3. *Write the proximal gradient method and its accelerated variant to solve* (P)*. What are the convergence guarantees that you can give? What are the conditions on* α *to ensure convergence?*
4. *Considering the convergence* rate *alone. Is the accelerated variant better?*

Exercise 3.3 (Exam 2023) *Consider the set of real symmetric matrices of dimension* $n \times n$*, and indicate it as* $\mathbb{S}(\mathbf{R}^n)$*. Consider their eigenvalue decomposition as* $X = U\Lambda U^\top$*, where* $X \in \mathbb{S}(\mathbf{R}^n)$*, and* Λ *is the matrix collecting on the diagonal the eigenvalues of* X*. Let the eigenvalues be* $\lambda_1, \ldots, \lambda_n$*, and collect them in a vector* $\lambda \in \mathbf{R}^n$*, such that,* $\Lambda = \mathbf{diag}(\lambda)$*.*

Consider now special convex functions $g : \mathbb{S}(\mathbf{R}^n) \to \mathbf{R} \cup \{+\infty\}$*, the so-called spectral functions, which can be defined by considering eigenvalue functions alone. For example, the trace is a spectral function, since* $\mathbf{trace}(X) = \mathbf{1}^\top\lambda$*. Other such functions are the indicator function of the sets* $\mathcal{X}_1 = \{X \in \mathbb{S} | X \succeq 0 \equiv \lambda \geqslant 0\}$*, or* $\mathcal{X}_2 = \{X \in \mathbb{S} | \|X\|_F^2 \equiv \mathsf{trace}(X^\top X) \leqslant 1 \equiv \|\lambda\|_2^2 \leqslant 1\}$*.*

For these spectral functions, then the proximal map simplifies as

$$\mathsf{prox}_g(X) = U\mathbf{diag}[\mathsf{prox}_g(\lambda)]U^\top,$$

where with a slight abuse of notation we use the function g *both for matrix* X *and its eigenvalues* λ*.*

Now, consider the problem,

$$(\star) \qquad \min_{X \in \mathcal{X} \subseteq \mathbb{S}(\mathbf{R}^n)} f(X) + f_2(X),$$

for a differentiable convex function $f : \mathbb{S}(\mathbf{R}^n) \to \mathbf{R}$*, such that* $\nabla f(X) \in \mathbb{S}(\mathbf{R}^n)$*, and a generic convex function* $f_2 : \mathbb{S}(\mathbf{R}^n) \to \mathbf{R} \cup \{+\infty\}$*.*

1. *Write the standard proximal gradient method for the problem* $(\star)$ *for the set* $\mathcal{X} = \mathcal{X}_1$ *and* $f_2 \equiv 0$*. Give the proximal operator in closed-form.*
2. *Write the standard proximal gradient method for the problem* $(\star)$ *for the set* $\mathcal{X} = \mathcal{X}_2$ *and* $f_2 \equiv 0$*. Give the proximal operator in closed-form.*
3. *Write the standard proximal gradient method for the problem* $(\star)$ *for the set* $\mathcal{X} = \mathbb{S}(\mathbf{R}^n)$ *and* $f_2 = \mathbf{trace}(X)$*. Give the proximal operator in closed-form.*
4. *Consider problem* $(\star)$ *for* $f(X) = \exp(X)$*,* $\mathcal{X} = \mathcal{X}_1$*, and* $f_2 \equiv 0$*. With the theory we have studied in class, which convergence guarantees has the standard proximal gradient method applied to* $(\star)$ *for* $f(X) = \exp(X)$*? (You could consider* $n = 1$ *for simplicity, if you need).*

Chapter 4

Fourth lecture

Constrained optimisation (II): second-order methods, i.e., interior-point method

4.1 The full picture

We look at the complete explicit convex setting,

$$\begin{aligned} (\mathsf{PI}) \qquad & \underset{\boldsymbol{x}\in X\subseteq\mathbf{R}^n}{\text{minimise}} \quad f(\boldsymbol{x}) && (4.1)\\ & \text{subject to} \quad Ax = b,\, g(\boldsymbol{x}) \leqslant 0. && (4.2) \end{aligned}$$

where A is a $\mathbf{R}^{p\times n}$ real-valued matrix, and $b \in \mathbf{R}^p$ with b in the image of A for feasibility. Furthermore, both $f(\boldsymbol{x}) : \mathbf{R}^n \to \mathbf{R}$ and $g : \mathbf{R}^n \to \mathbf{R}^l$ are convex functions, and X is a convex set. Assume that Slater's constraint qualification holds (i.e. $\exists \bar{\boldsymbol{x}} : \bar{\boldsymbol{x}} \in \mathsf{relint}(X), g(\bar{\boldsymbol{x}}) < 0$). Remark that (PE) $\subset$ (PI).

In this lecture, we are interested in second-order algorithms that can deliver very fast convergence, since they will use variants of the Newton's algorithm in some inner loops. We are mainly concerned with interior-point methods, which are the state of the art, if one can afford the computations to run them. For example, you can see them implemented in off-the-shelf solvers like `cvxpy` in python.

To develop the methods, we will assume that both the cost f and the constraint function g are at least double differentiable, so that $f, g \in \mathcal{C}^2$. This is reasonable, since we will need their Hessians for constructing a second-order oracle.

We will then make use of some properties of the Newton's algorithm (namely its affine invariance) to build polynomial-time algorithms to a large class of optimisation problems. As far as computational complexity and time go, polynomial-time algorithms are the "good guys", meaning that they allow to solve optimisation problems in a time which is bounded by a polynomial function of the inputs. This is in contrast with exponential-time algorithms, which are in general the "bad guys". We will discuss this a bit further in the lecture, but keep in mind that the polynomial-time algorithm that we will study made the front page of the New York Times in 1984, when it was discovered! Polynomial-time algorithms are the stars in optimisation.

And, finally, we will use our star method to gauge which convex problems are easy (i.e., can be solved in polynomial time), and which not. Please, remember that convex problems are still in general very hard to solve, even though their optimality conditions can be written down.

4.1.1 Barrier functions and penalised problems

We are now ready to introduce some constructions and notions to help us develop the algorithm. Assume without loss of generality that $X \equiv \mathbf{R}^n$. This is without loss of generality, since you can load g with the constraints represented by X, as well.

Suppose now that we can construct a convex l.s.c. barrier function

$$\Phi_g(\boldsymbol{x}) : \begin{cases} \approx 0 & \forall \boldsymbol{x} : g(\boldsymbol{x}) \leqslant 0 \\ +\infty & \text{otherwise.} \end{cases} \tag{4.3}$$

As a typical example, consider the constraint $x \leqslant 0$, and the barrier

$$\Phi_{x\leqslant 0}(x) = -\log(-x).$$

This logarithmic barrier is $+\infty$ if $x \to 0$, and stays small for small negative x. As a generalisation, for the convex function $g(\boldsymbol{x}) : \mathbf{R}^n \to \mathbf{R}^l$ and the constraint $g(\boldsymbol{x}) \leqslant 0$, we can consider each constraint g_i for $i = 1, \ldots, l$, and then build a barrier as,

$$\Phi_g(\boldsymbol{x}) = -\sum_{i=1}^{l} \log(-g_i(\boldsymbol{x})), \tag{4.4}$$

both barrier functions extended outside the domain by $+\infty$.

Having now constructed a barrier, instead of the original problem, consider the following penalised version

$$\begin{aligned} (\mathsf{PIP}) \qquad \underset{\boldsymbol{x}\in\mathbf{R}^n}{\text{minimise}} \qquad & tf(\boldsymbol{x}) + \Phi_g(\boldsymbol{x}) && (4.5)\\ \text{subject to} \qquad & Ax = b, && (4.6) \end{aligned}$$

for $t \geqslant 0$.

Since the barrier function $\Phi_g(\boldsymbol{x})$ is convex, and $t \geqslant 0$, then the penalised problem is still convex. We define as $\boldsymbol{x}^*(t)$ a solution of this penalised problem and we further call the curve $\{\boldsymbol{x}^*(t)\}_{t\geqslant 0}$ the central path. The central path is then the curve generated by a solution of (PIP) by varying t. Moreover, we notice that $\boldsymbol{x}^*(0)$ is a particular feasible point, called the analytical center of the constraint function $g(\boldsymbol{x})$.

By simple inspection, we can immediately see that if we let $\lim_{t\to+\infty} \boldsymbol{x}^*(t)$ we obtain $\boldsymbol{x}^*$, a minimiser of the original problem PI, that is the central path converges to the solution of our original problem.

The idea is to move along the central path, using some basic algorithm to do small steps along the central path. The "basic algorithm" that is used in interior-point methods is the Newton's method, because of its very fast local convergence.

The technical details are for which classes of problems the (PIP) problems do not become harder and impossibly hard as $t \to \infty$, and how to select the increment of t such that we stay in the local convergence zone.

Since gradient methods depend on the condition number and they would become very slow for large values of t, we can safely say that they are useless in this lecture.

4.1.2 Interpretation of the penalised problem

To understand a bit better the idea behind the penalised problem, let us have a look at the KKT conditions for it. In particular, consider a logarithmic barrier function, then the KKT conditions for (PIP) are

$$\begin{aligned} t\nabla f(\boldsymbol{x}) + A^{\mathsf{T}}\lambda + \sum_{i=1}^{l} \frac{1}{-g_i(\boldsymbol{x})}\nabla g_i(\boldsymbol{x}) &= 0 && (4.7)\\ A\boldsymbol{x} = b, \quad g_i(\boldsymbol{x}) < 0 \qquad & i = 1, \ldots, l. && (4.8) \end{aligned}$$

Or equivalently, introducing the new variables $\nu_i = -1/(tg_i(\boldsymbol{x}))$,

$$\begin{aligned} \nabla f(\boldsymbol{x}) + A^{\mathsf{T}}\lambda + \sum_{i=1}^{l} \nu_i \nabla g_i(\boldsymbol{x}) &= 0 && (4.9)\\ A\boldsymbol{x} = b, \quad g_i(\boldsymbol{x}) < 0 \quad \nu_i > 0 \qquad & i = 1, \ldots, l && (4.10)\\ -\nu_i g_i(\boldsymbol{x}) = 1/t \qquad & i = 1, \ldots, l. && (4.11) \end{aligned}$$

By the second set of equation, we can see that the (PIP) approximates the complementary slackness condition, and as $t \to \infty$, we regain the original problem. Excellent.

4.2 A Newton's step for the penalised problem

We are now ready to develop a Newton's step/method for the penalised problem, which is an equality-constraint convex optimisation problem. The developments here are rather general and can be applied at any equality-constraint convex optimisation problems.

Let's start by constructing the Lagrangian function for (PIP) as,

$$F_t(\boldsymbol{x}, \lambda) = tf(\boldsymbol{x}) + \lambda^\mathsf{T}(A\boldsymbol{x} - b) - \sum_{i=1}^{l} \log(-g_i(\boldsymbol{x})), \tag{4.12}$$

where we have indicated with λ the dual variable associated to the constraint $A\boldsymbol{x} = b$.

We can further write the optimality conditions for (PIP) as,

$$\nabla_{\boldsymbol{x}} F_t(\boldsymbol{x}, \lambda) = 0 \qquad A\boldsymbol{x} = b. \tag{4.13}$$

To build a Newton's method, we need to expand the optimality conditions by using a Taylor approximation around a point $(\boldsymbol{x}_k, \lambda_k)$, i.e.,

$$\nabla_{\boldsymbol{x}} F_t(\boldsymbol{x}_k, \lambda_k) + \nabla^2_{\boldsymbol{x}\boldsymbol{x}} F_t(\boldsymbol{x}_k, \lambda_k)\Delta\boldsymbol{x} + \nabla^2_{\lambda\boldsymbol{x}} F_t(\boldsymbol{x}_k, \lambda_k)\Delta\lambda = 0 \tag{4.14}$$

$$A\boldsymbol{x}_k - b + A\Delta\boldsymbol{x} = 0. \tag{4.15}$$

We can now build the linear system for the Newton's increment $(\Delta\boldsymbol{x}, \Delta\lambda)$ as,

$$\underbrace{\begin{bmatrix} \nabla^2_{\boldsymbol{x}\boldsymbol{x}} F_t(\boldsymbol{x}_k, \lambda_k) & A^\mathsf{T} \\ A & 0 \end{bmatrix}}_{=:H_k} \begin{bmatrix} \Delta\boldsymbol{x} \\ \Delta\lambda \end{bmatrix} = \underbrace{\begin{bmatrix} -\nabla_{\boldsymbol{x}} F_t(\boldsymbol{x}_k, \lambda_k) \\ -A\boldsymbol{x}_k + b \end{bmatrix}}_{=:r_k}, \tag{4.16}$$

and solve for $(\Delta\boldsymbol{x}, \Delta\lambda)$.

Finally, we can use a damped strategy for the update, $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k\Delta\boldsymbol{x}$, $\lambda_{k+1} = \lambda_k + \alpha_k\Delta\lambda$, where $\alpha_k > 0$ is the stepsize and it is based on the chosen damped strategy.

4.2.1 Damping and the full method

We focus here on a special choice of damping to fix the ideas, namely the backtracking strategy. Since now the problem is constrained, we cannot use a strategy based on the cost, as done in previous lectures. It is much better to use instead a strategy based on the residual $\|r_k\|$, meaning that you would like to reduce it at least by a given amount per iteration.

The full method then looks like the following.

Damped Newton's method for equality-constrained problems

- *Start with* $\boldsymbol{x}_0, \lambda_0 \in \mathbf{R}^n$ *(Need:* $\boldsymbol{x}_0 : g_i(\boldsymbol{x}_0) < 0$ *if a penalised problem)*
- *Iterate:*
  - *Solve the linear system:* $H_k \begin{bmatrix} \Delta\boldsymbol{x} \\ \Delta\lambda \end{bmatrix} = r_k$,
  - *Backtracking: set* $\alpha_k = 1$, $\tau \in (0,1), \beta \in (0, 1/2)$*:*
    While $\|r(\boldsymbol{x}_k + \alpha_k\Delta\boldsymbol{x}, \lambda_k + \alpha_k\Delta\lambda)\| > (1 - \beta\alpha_k)\|r_k\|$, *set:* $\alpha_k \leftarrow \tau\alpha_k$.
  - $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k\Delta\boldsymbol{x}$, $\lambda_{k+1} = \lambda_k + \alpha_k\Delta\lambda$.

As long as the generalised function F_t is strongly convex and had a Lipschitz Hessian in $\boldsymbol{x}$, i.e., it is $\in \mathcal{S}^{2,2}_{m,M}$, then convergence goes as in Theorem 2.7, and in particular it is locally quadratic. The assumptions on F_t may be reasonable in some situations, but in our case, they may be awkward. In fact the logarithmic barrier is not strongly convex, and it may even have problems to have a Lipschitz Hessian.

In the next section, we will see how to modify the analysis and to change the assumption for a more general one, which will help us with the convergence proof of the interior-point method, independently of the conditioning parameter t.

4.3 Self concordance

We start with defining a new function class: the self-concordance class as follows.

Definition 4.1 (Self-concordance on R) *A convex function $f : \mathbf{R} \to \mathbf{R}$ is self-concordant iff*

$$|f'''(x)| \leqslant 2f''(x)^{3/2}$$

for all $x \in dom(f)$. Where $'''$ indicated the third order derivative and $''$ the second order one.

Definition 4.2 (Self-concordance on $\mathbf{R}^n$) *A convex function $f : \mathbf{R}^n \to \mathbf{R}$ is self-concordant iff it is self-concordant along every line in its domain, i.e., if the function $\tilde{f}(\mu) = f(\boldsymbol{x} + \mu v)$ is a self-concordant function of μ for all $\boldsymbol{x} \in dom(f)$ and for all v.*

Self concordance will be indicated as $\mathcal{SC}$. At first, this definition may sound a mathematical weirdness, but we will see how it can help us a great deal, and it encompass a good deal of functions!

Let's look at some examples and properties.

4.3.1 Examples and properties

Let us look at $|f'''(x)| \leqslant 2f''(x)^{3/2}$ in the scalar case. We notice immediately that, linear and positive quadratic functions are self-concordant since convex and $f'''(x) = 0$ and $f''(x) \geqslant 0$.

We also notice that, $-\log(x)$ is self-concordant on its domain: this key for us, since it will be the basis for our barrier functions, which are not strongly convex. Since this is the key step, let's make it a theorem.

Theorem 4.1 (Self concordance of the logarithmic function) *The function $-\log(x)$ is self-concordant on its domain.*

Proof. *Homework.* ♣

Some useful properties of the $\mathcal{SC}$ class are the following ones, whose proofs are immediate.

- If $f_1 \in \mathcal{SC}, f_2 \in \mathcal{SC}$ then, $f_1 + f_2 \in \mathcal{SC}$
- If $f \in \mathcal{SC}$ and $a \geqslant 1$, then, $af \in \mathcal{SC}$
- If $f \in \mathcal{SC}$ then $f(Ax + b) \in \mathcal{SC}$

4.3.2 Matrix extensions

In some cases, we also look at problems defined over the semi-definite cone. In particular, we look at problems defined over symmetric real-valued matrix variables $X \in \mathbb{S}(\mathbf{R}^n)$, such as,

$$\underset{X \in \mathbb{S}(\mathbf{R}^n)}{\text{minimise}} \quad F(X) \tag{4.17}$$

$$\text{subsect to} \quad X \succeq 0, \tag{4.18}$$

where $F : \mathbb{S}(\mathbf{R}^n) \to \mathbf{R}^n$ is a convex function. These types of problems are often referred to as semi-definite programs, and they are useful in many domains.

First of all, it should be immediate to prove that the above is a convex problem, since $X \succeq 0$ is a convex set (try it out).

Then, we need two things. First an inner product, which is the trace. Second, a self-concordant barrier.

Theorem 4.2 *An allowed barrier function for the constraint $X \succeq 0$ is $\phi(X) = -\log\det(X)$.*

Proof. *We start by proving that $\phi(X)$ is convex.*

For the function $\phi(X)$, $X \succ 0$, we can verify convexity by considering an arbitrary line, given by $X = Z + tV$, where Z, V are symmetric matrices of dimension n. We define $g(t) = \phi(Z + tV)$, and restrict g to the interval of values of t for which $Z + tV \succ 0$. Without loss of generality, we can assume that $t = 0$ is inside this interval, i.e., $Z \succ 0$.

We now show that $g''(t) \geqslant 0$ (g'' is the Hessian), and conclude that $\phi(X)$ is a convex function.

For the defined $g(t)$,

$$\begin{aligned} g(t) &= -\log\det(Z + tV) \\ &= -\log\det(Z^{1/2}(I + tZ^{-1/2}VZ^{-1/2})Z^{1/2}) \\ &= -\sum_{i=1}^{n}\log(1 + t\lambda_i) - \log\det Z, \end{aligned}$$

where $\lambda_1, \ldots, \lambda_n$ are the eigenvalues of $Z^{-1/2}VZ^{-1/2}$, therefore for the scalar function $g(t)$

$$g'(t) = \frac{dg}{dt} = -\sum_{i=1}^{n}\frac{\lambda_i}{1 + t\lambda_i}, \qquad g''(t) = \frac{d^2g}{dt^2} = \sum_{i=1}^{n}\frac{\lambda_i^2}{(1 + t\lambda_i)^2} \geqslant 0.$$

Note, we used the fact that $det(A) = \prod_{i=1}^{n}\lambda_i(A)$, that $det(AB) = det(A)det(B)$, and the usual $\log(ab) = \log(a) + \log(b)$.

We continue by proving the self-concordance of $\phi(X)$. Let us compute the derivatives along a line $X + tV$, for $X \succ 0$ and V symmetric.

$$\left.\frac{d\phi(X + tV)}{dt}\right|_{t=0} = -\left.\frac{d}{dt}\right|_{t=0}\{\log\det(X) + \log\det(I + tX^{-1}V)\} = -\mathbf{trace}(X^{-1}V),$$

$$\left.\frac{d^2\phi(X + tV)}{dt^2}\right|_{t=0} = -\left.\frac{d}{dt}\right|_{t=0}\mathbf{trace}((X + tV)^{-1}V) = \mathbf{trace}(X^{-1}VX^{-1}V),$$

$$\left.\frac{d^3\phi(X + tV)}{dt^3}\right|_{t=0} = \left.\frac{d}{dt}\right|_{t=0}\mathbf{trace}((X + tV)^{-1}V(X + tV)^{-1}V) = -2\mathbf{trace}(X^{-1}VX^{-1}VX^{-1}V).$$

Let $\tilde{V} = X^{-1/2}VX^{-1/2}$, then,

$$\left.\frac{d^2\phi(X + tV)}{dt^2}\right|_{t=0} = \mathbf{trace}(\tilde{V}^2) > 0, \qquad \left.\frac{d^3\phi(X + tV)}{dt^3}\right|_{t=0} = -2\mathbf{trace}(\tilde{V}^3).$$

Hence,

$$\left|\left.\frac{d^3\phi(X + tV)}{dt^3}\right|_{t=0}\right| = \left|-2\mathbf{trace}(\tilde{V}^3)\right| \leqslant 2\mathbf{trace}(\tilde{V}^2)^{3/2} = 2\left(\left.\frac{d^2\phi(X + tV)}{dt^2}\right|_{t=0}\right)^{3/2},$$

from which the thesis. ♣

4.3.3 Newton's method for self-concordant functions

We are now ready for the analysis of the damped Newton's method for self-concordant functions.

Theorem 4.3 *Damped Newton's method convergence globally for strictly convex and self-concordant functions on the problem $\min_{\boldsymbol{x}\in\mathbf{R}^n} f(\boldsymbol{x})$. In particular, there exists a $\gamma > 0$ depending on the backtracking method, for which it obtains a solution $\boldsymbol{x}_k$ for which $f(\boldsymbol{x}_k) - f(\boldsymbol{x}^*) < \epsilon$ in*

$$\frac{f(\boldsymbol{x}_0) - f(\boldsymbol{x}^*)}{\gamma} + \log\log(1/\epsilon)$$

iterations.

Typically $1/\gamma \sim 100$, while $\log\log(1/\epsilon) \approx 6$. Note that the accuracy does not depend on the conditioning of the function or on its strong convexity.

Proof. *The proof is rather involved and left to the reader to check out in [BV] book.* ♣

The theorem can be extended to equality-constrained Newton's methods and the result stay the same. With this result in place, let's look at interior-point methods.

4.4 The interior-point method

Recall the damped Newton's method for equality-constrained problems and consider the following method to solve (PI) with a sequence of (PIP) problems,

Interior-point method

- *Start with* $\boldsymbol{x}_0$ *such that* $g(\boldsymbol{x}_0) < 0$, $t > 0, \mu > 1$ *and required accuracy* ϵ. *Rem:* $g : \mathbf{R}^n \to \mathbf{R}^l$
- *Iterate* $\kappa = 0, 1 \ldots$
  - *Solve (PIP) starting from* $\boldsymbol{x}_\kappa$ *with a damped Newton's method. Let the solution be* $\boldsymbol{x}_{\kappa+1}$
  - *Quit if* $l/t < \epsilon$ *(remember complementary slackness:* $-\nu_i g_i(\boldsymbol{x}) = 1/t$*)*
  - *Update:* $t \leftarrow \mu t$

Remark 3 *You can't start with* t *satisfying* $l/t < \epsilon$, *because* t *would be too big and we may be far from the local convergence zone.*

Our aim is now to show that if we use damped Newton's method for each (PIP) the number of required iterations is bounded by a polynomial function of the inputs (e.g., number of constraints), hence, the interior-point method for self-concordant functions and barriers has a polynomial complexity and can solve (some) convex problems efficiently.

4.4.1 Computational complexity analysis: outer

First, we look at the outer steps (updates on t). Starting for a certain t, say t_0 for short, each steps we obtain an accuracy of

$$\frac{l}{\mu^\kappa t_0}.$$

Therefore for a desired accuracy ϵ, we need

$$\left\lceil \frac{\log(l/\epsilon t_0)}{\log \mu} \right\rceil$$

steps.

4.4.2 Computational complexity analysis: inner

Theorem 4.4 *Consider the (PIP)s generated by an interior-point method. Assume the technical requirements that,*

- *The function* $tf(\boldsymbol{x}) - \sum_{i=1}^{l} \log(-g_i(\boldsymbol{x}))$ *is self-concordant for all* $t > 0$*;*
- *The (PIP)s have bounded sub-level sets (i.e., they are solvable).*

Then, the number of required inner iterations of the damped Newton's method to achieve an accuracy ϵ_{nt} *can be upper bounded as,*

$$\frac{l(\mu - 1 - \log(\mu))}{\gamma} + \log\log(1/\epsilon_{\text{nt}}) \approx \frac{l(\mu - 1 - \log(\mu))}{\gamma} + 6.$$

with γ *depending only on the backtracking parameters.*

The proof is based on Theorem 4.3 on the convergence of the damped Newton's method with self-concordance assumption.

We see immediately that the number of iterations are linear in l. Each iteration requires a Newton's step computation, which is in the worst case a matrix inversion, which is of the order of $O((n+m)^3)$, so also polynomial. We also see that the bound does not depend on the penalisation term t.

4.4.3 Computational complexity analysis: complete

Now we can put things together

Theorem 4.5 *Assume (PI) is convex and enjoys a self-concordant barrier function and each of the (PIP) is solvable. Then employing an interior-point method with parameters (t_0, μ, γ) delivers a solution with accuracy ϵ in*

$$\left\lceil \frac{\log(l/\epsilon t_0)}{\log \mu} \right\rceil \times \left\lceil \frac{l(\mu - 1 - \log(\mu))}{\gamma} + 6 \right\rceil = o(l^2)$$

iterations. Each of them requiring the solution of a matrix inversion with complexity $< O((n + m)^3)$.

As such the interior-point method with self-concordant barrier is a polynomial-time method, with overall complexity of $< O(l^2(n + m)^3)$.

Theorem 4.5 delivers the convergence certificate guarantees we need: it tells us how many iterations we need to reach a certain accuracy, in terms of complementary slackness. It also tells us that each iteration can be carried out in polynomial time, making our life easy.

We have therefore developed a polynomial-time method for generic convex optimisation problems. This is a huge feat: we can actually solve some of the problems (using a second-order oracle) with very high accuracy!

I want to recall that first-order methods can offer better computational complexity *per iteration*, but they may take many many iteration (not polynomially bounded) to converge. Sometimes however, that's the only thing you have.

The next natural question is: which optimisation problems can be solve then in polynomial time? All convex ones?

4.5 Classes of easy problems

We now look at the classes of easy problems, problems that can be solved in polynomial time by using the interior-point method that we have developed. There are four of such classes, which can model a widespread array of applications. This is what makes (a sub-set of convex) optimisation a useful technology. In particular, this sub-set of convex optimisation is a "complete" domain:

- We have an amazing theory and we understand it inside out;
- We have efficient algorithms and we can get their complexity;
- The problems we can solve efficiently *matter* for society!

Frankly, you will not see much of these domains in your professional life. For example, machine learning matter for society but the theory behind is still not well understood. Binary optimisation matter for society, but we do not have efficient algorithms.

Our four musketeers are:

- Linear programs (LPs);
- Quadratic programs (QPs);
- Second-order conic programs (SOCPs);

- Semidefinite programs (SDPs).

And for all the above you have a self-concordant barrier, so that you can apply Theorem 4.5.

4.5.1 Linear programs

Linear programs (or LPs) are convex optimisation problems with linear cost and linear constraints. Linear programs in continuous optimisation do not have to be mixed with linear programs in integer programming, where the decision variables are integer. The latter are not linear at all!

A typical linear program is the following,

$$\begin{aligned} \underset{\boldsymbol{x}\in\mathbf{R}^n}{\text{minimise}} \quad & c^\mathsf{T}\boldsymbol{x}, && (4.19)\\ \text{subject to} \quad & A\boldsymbol{x} = b,\ D\boldsymbol{x} \leqslant e. && (4.20) \end{aligned}$$

For example:

$$\begin{aligned} \underset{x_1\in\mathbf{R},x_2\in\mathbf{R}}{\text{minimise}} \quad & 2x_1 + x_2 \\ \text{subject to} \quad & x_1 \geqslant 0, x_2 \geqslant 0 \\ & x_1 + x_2 \leqslant 0. \end{aligned}$$

The solution is on the boundary (vertex), from which the simplex method can be developed (which is a combinatorial method with worst-case exponential-time complexity). The interior-point method was applied to LPs successfully in 1984 with a complexity of $O(n^2m), m \geqslant n$ for n variables and m constraints. Typically one can solve instances with 10k variables and constraints. Furthermore, LPs can be used to approximate more complex optimisation problems by linearisation.

4.5.2 Quadratic Programs: QPs

Quadratic Programs or QPs are convex optimisation problems with convex quadratic cost and linear constraints,

$$\begin{aligned} \underset{\boldsymbol{x}\in\mathbf{R}^n}{\text{minimise}} \quad & \tfrac{1}{2}\boldsymbol{x}^\mathsf{T}Q\boldsymbol{x} + c^\mathsf{T}\boldsymbol{x}, && (4.21)\\ \text{subject to} \quad & A\boldsymbol{x} = b,\ D\boldsymbol{x} \leqslant e, && (4.22) \end{aligned}$$

with $Q \succeq 0$.

QPs have better properties than LP and their solution is not on the boundary. The level sets are ellipsoids, and in general, they are not more complicated to solve than LPs. In fact, writing the Lagrangian and the KKT conditions (here the constraint qualifications are satisfied)

$$S : \begin{bmatrix} Q\boldsymbol{x} + c + A^\top\lambda + D^\top\nu = 0 \\ A\boldsymbol{x} - b = 0 \\ D\boldsymbol{x} - e \leqslant 0 \\ \nu \geqslant 0 \\ (D\boldsymbol{x} - e)_i\nu_i = 0 \end{bmatrix} \rightarrow \begin{array}{ll} \text{find} & \boldsymbol{x}, \lambda, \nu \\ \text{s.t.} & \boldsymbol{x}, \lambda, \nu \in S \end{array} = \mathsf{LP} + \mathsf{sth.\ else}$$

we see that QPs are similar to LPs.

Here the computational complexity with interior-point is limited by matrix inversions. We can cite a few notable examples of QPs, such as least-squares problems, regression problems, optimal control, and so forth. QPs have been, and still are, the cornerstone of nonconvex optimisation with sequential quadratic programming (SQP).

4.5.3 Second-Order Conic Programs: SOCPs

Second-Order Conic Programs or SOCPs are convex optimisation problems with quadratic cost and quadratic constraints and a bit more. Compactly, they can be written as,

$$\begin{aligned} \underset{\boldsymbol{x}\in\mathbf{R}^n}{\text{minimise}} \quad & c^\mathsf{T}\boldsymbol{x} \\ \text{subject to} \quad & \|D_ix + e_i\|_2 \leqslant h_i^\mathsf{T}x + d_i, \quad i = 1, \dots, m \\ & A\boldsymbol{x} = b \end{aligned} \tag{4.23}$$

4.5.4 Semi-Definite Programs: SDPs

Semi-Definite Programs or SDPs are convex optimisation problems with linear cost and with semidefinite constraints.

Let's us define a Linear matrix $F : \mathbf{R}^n \to \mathbf{S}^n$:

$$F(x) = x_1F_1 + x_2F_2 + \ldots + x_nF_n + G \qquad \boldsymbol{x} \in \mathbf{R}^n, F_1, \ldots, F_n, G \in \mathbf{S}^n.$$

and, the SDP:

$$\begin{array}{ll} \underset{\boldsymbol{x}}{\text{minimise}} & c^\top \boldsymbol{x} \\ \text{subject to} & F(\boldsymbol{x}) \succeq 0 \quad \text{(SDP constraint, LMI..)} \\ & A\boldsymbol{x} = b. \end{array}$$

4.6 Summary

In these four classes we have developed a theory of algorithms for optimisation. Without it, we wouldn't be able to solve anything. Without it, we wouldn't be able to know how many iterations we need to run to obtain a result with a certain accuracy. Without it, we could wait forever to get nothing in return.

When talking about algorithms, we care about complexity, and which type of oracles we can afford to use: zero-order, first-order, second-order, and so forth.

And, we also care about the properties (convexity, smoothness) of the problems, and we use them to decide which algorithm to employ.

And, finally, by using the algorithmic lens, we can decide how to model our problem, so that we are able to solve it (we are guided in terms of assumptions). To further hammer this last point, consider the following last example.

Example 4.1 *Consider the problem,*

$$\textit{(Pex)} \qquad \underset{\boldsymbol{x}\in\mathbf{R}^n}{\text{minimise}} \ \frac{1}{2}\boldsymbol{x}^\mathsf{T}\mathbf{Q}\boldsymbol{x} + c^\mathsf{T}\boldsymbol{x} + \lambda\|\boldsymbol{x}\|_\infty,$$

$\mathbf{Q} \succeq 0$. *How would you solve the problem?*

At face value: since the cost is not differentiable, we could employ the sub-gradient method. This would converge very slowly.

However, isn't the term $\|\cdot\|_\infty$ prox-friendly? Then the proximal method can be employed instead. Here we could even use its accelerated variant.

Is the dimension not-too big? Say $n \sim 1000$? Then, transform the problem into a QP and solve it via interior-point methods (which are much faster), how? By using the epigraph form:

$$\textit{(Pex)} \iff \begin{array}{ll} \text{minimise}_{\boldsymbol{x}\in\mathbf{R}^n, t>0} & \frac{1}{2}\boldsymbol{x}^\mathsf{T}\mathbf{Q}\boldsymbol{x} + c^\mathsf{T}\boldsymbol{x} + \lambda t, \\ \textit{subject to} & \|\boldsymbol{x}\|_\infty \leqslant t \iff |[\boldsymbol{x}]_i| \leqslant t \, \forall i \\ & \iff [\boldsymbol{x}]_i \leqslant t, [\boldsymbol{x}]_i \geqslant -t, \forall i \end{array}$$

Optimisation problems can be transformed into other ones.

4.7 References

- BV: Chapters 10, 11.

◆ ◇ ◇

4.8 Exercises

Exercise 4.1 *Consider the problem,*

$$\min_{x_1 \in \mathbf{R}, x_2 \in \mathbf{R}} \frac{1}{2}(x_1^2 + x_2^2), \quad \textit{subject to } x_2 \geqslant 1 - x_1.$$

1. *Introduce a logarithmic barrier for the constraint, write the penalised problem, and find the analytical centre.*
2. *Plot in a 2D sketch the qualitative level curves of the penalised problem for $t = 1$ and $t = 10$.*
3. *Argue that the problem is equivalent to the one dimensional problem,*

$$\min_{y \in \mathbf{R}} \frac{1}{2} y^2, \quad \textit{subject to } y \geqslant 1/2,$$

   in the sense that $x_1^* = x_2^* = y^*$.
4. *Write the penalised problem for y and solve it analytically for $t = 1, 4, 10$.*
5. *What is the value of y as $t \to \infty$? Could you have guessed that ?*

Exercise 4.2 (Exam 2023) *Consider the problem,*

$$\min_{x \in \mathbf{R}^n} \quad \frac{1}{2} x^\top Q x + c^\top x + \|x\|_1,$$

with $Q \succeq 0$.

1. *Prove that the problem above can be formulated as a quadratic program as,*

$$\begin{array}{ll} \min\limits_{x \in \mathbf{R}^n, y \in \mathbf{R}^n} & \frac{1}{2} x^\top Q x + c^\top x + \mathbf{1}^\top y, \\ \textit{subject to} & x_i - y_i \leqslant 0, \quad x_i + y_i \geqslant 0, \quad y_i \geqslant 0, \quad \forall i, \end{array}$$

   where $\bullet_i$ represents the i-th component of a given vector $\bullet$.
2. *Apply the interior-point method to the quadratic program above using a logarithmic barrier, in particular write:*
   (a) *The modified cost function with the logarithmic barrier;*
   (b) *The Newton's step for the inner problems, that is: the linear system;*
   (c) *A damped strategy to solve each inner problems;*
   (d) *How many Newton iterations do you need overall? What is the total computational complexity (i.e., number of iterations times the complexity per iterations)?*
   (e) *Finally, focus on the constraint $y_i \geqslant 0$. Could one use $\log^2(y_i)$ as a barrier?*

Exercise 4.3 (Resit 2023) *Consider the problem*

$$\min_{x \in \mathbf{R}^n} x^\mathsf{T} W x, \quad \textit{subject to } x_i^2 = 1,\ i = 1, \ldots, n.$$

Assume that W is symmetric and its diagonal is 0.

1. *Is this problem convex?*
2. *Prove that the dual of the above problem is,*

$$\min_{\nu \in \mathbf{R}^n} \mathbf{1}^\top \nu, \quad \textit{subject to } W + \mathbf{diag}(\nu) \succeq 0,$$

   where $\mathbf{diag}(\nu)$ is a $n \times n$ matrix, with diagonal equal to ν. Prove that the dual problem is convex and it is in fact an SDP.
3. *We are going to solve this SDP via an interior-point method.*

(a) Write the modified cost function with the barrier function $\phi(X) : \mathbb{S}_+^n \to \mathbf{R}$*, defined as* $\phi(X) = \log\det(X^{-1}) = -\log\det(X)$ *for the set* $X \succeq 0$*. Recall that the space* $\mathbb{S}_+^n$ *is the space of symmetric matrices in* n *dimension.*

(b) Write the Newton's step, recalling that

$$\nabla_\nu \log\det(W + \mathbf{diag}(\nu)) = \mathbf{Diag}[(W + \mathbf{diag}(\nu))^{-1}],$$

where $\mathbf{Diag}[(\cdot)]$ *is a vector, whose entries are the diagonal elements of* $(\cdot)$*. And,*

$$\nabla_{\nu\nu} \log\det(W + \mathbf{diag}(\nu)) = -(W + \mathbf{diag}(\nu))^{-2}.$$

Appendix A

Recap from OPT201

A.1 Optimality conditions: sets

Consider the convex problem

$$\underset{\boldsymbol{x}\in X\subseteq\mathbf{R}^n}{\text{minimise}}\, f(x),$$

with $f : \mathbf{R}^n \to \mathbf{R}$ convex, and X convex. Then, necessary and sufficient conditions for optimality are,

$$\partial f(\boldsymbol{x}^*) + N_X(\boldsymbol{x}^*) \ni \mathbf{0},$$

where ∂f is the sub-differential of f and $N_X(\boldsymbol{x})$ is the normal cone operator of X at $\boldsymbol{x}$.

If f is differentiable then $\partial f \equiv \nabla f$ and we can write,

$$\nabla f(\boldsymbol{x}^*) + N_X(\boldsymbol{x}^*) \ni \mathbf{0}.$$

The above is equivalent (in the convex case, in which we are) to

$$\nabla f(\boldsymbol{x}^*)^\top(\boldsymbol{x} - \boldsymbol{x}^*) \geqslant 0, \forall \boldsymbol{x} \in X$$

All of these are set conditions, and it is hard to compute $\boldsymbol{x}^*$ from them.

A.2 Optimality conditions: equality constraints

Consider now the case $X = \{\boldsymbol{x} | A\boldsymbol{x} = b\}$, $A \in \mathbf{R}^{q\times n}$, and for simplicity the case in which $f \in \mathcal{C}(\mathbf{R}^n)$. The problem is still convex. We have the following implication.

Theorem A.1 *Consider the problem,* $\min_{\boldsymbol{x}\in\mathbf{R}^n} f(x)$, subject to $A\boldsymbol{x} = b$, *if the constraints are qualified, then the optimality conditions* $\nabla f(\boldsymbol{x}^*) + N_X(\boldsymbol{x}^*) \ni \mathbf{0}$ *can be equivalently written as*

$$\begin{cases} \nabla f(\boldsymbol{x}^*) + A^\top\lambda^* = 0 \\ A\boldsymbol{x}^* = b \end{cases}$$

- The variables $\lambda \in \mathbf{R}^q$ are the Lagrangian multipliers;
- We can write the optimality conditions as the saddle-points of the Lagrangian function,

$$\mathcal{L}(\boldsymbol{x}, \lambda) = f(\boldsymbol{x}) + \lambda^\top(A\boldsymbol{x} - b).$$

- These are non-linear equations: easier to solve them with first-order methods or Newton's.
- In the convex case as we are, the constraint $A\boldsymbol{x} = b$ is always qualified, provided that a solution exist! (I.e., b is in the image of A).

A.3 Optimality conditions: complete case

Consider now the case $X = \{\boldsymbol{x} | A\boldsymbol{x} = b, g(\boldsymbol{x}) \leqslant 0\}$, $A \in \mathbf{R}^{q\times n}$, $g : \mathbf{R}^n \to \mathbf{R}^p$, convex and differentiable, and for simplicity the case in which $f \in \mathcal{C}(\mathbf{R}^n)$. The problem is still convex. We have the following implication.

Theorem A.2 *Consider the problem,* $\min_{\boldsymbol{x}\in\mathbf{R}^n} f(x)$, subject to $A\boldsymbol{x} = b, g(\boldsymbol{x}) \leqslant 0$ *if the constraints are qualified, then the optimality conditions* $\nabla f(\boldsymbol{x}^*) + N_X(\boldsymbol{x}^*) \ni \mathbf{0}$ *can be equivalently written as*

$$\begin{cases} \nabla f(\boldsymbol{x}^*) + A^\top \lambda^* + \nabla^\top g(\boldsymbol{x}^*)\nu^* = 0 \\ A\boldsymbol{x}^* = b, \quad g(\boldsymbol{x}^*) \leqslant 0 \\ \nu^* \geqslant 0, \quad [\nu^*]_i [g(\boldsymbol{x}^*)]_i = 0, \forall i \in \{1, p\} \end{cases}$$

- The variables $\lambda \in \mathbf{R}^q, \nu^* \in \mathbf{R}^p$ are the Lagrangian multipliers or dual variables;
- We can write the optimality conditions as the saddle-points of the Lagrangian function,

  $$\mathcal{L}(\boldsymbol{x}, \lambda, \nu) = f(\boldsymbol{x}) + \lambda^\top (A\boldsymbol{x} - b) + \nu^\top g(\boldsymbol{x}),$$

  with the restriction that $\nu \geqslant 0$.
- These are non-linear inequalities: not so easy to solve.
- In the convex case as we are, the constraint $g(\boldsymbol{x}) \leqslant 0$ is qualified for example if there exists at least a feasible point $\boldsymbol{x}'$ for which $A\boldsymbol{x}' = b$ and $g(\boldsymbol{x}') < 0$. We call this condition: Slater's constraint qualification.

A.4 Extension to matrix decisions

It is often useful to look at problems whose decision variables are symmetric matrices. We can extend most of the results of these classes to this setting (and beyond to infinite dimensional Hilbert spaces), by considering the proper inner product. For squared symmetric matrices the inner product is the trace, as such,

$$\underset{X\in\mathbb{S}^n}{\text{minimise}}\, f(X), \qquad \text{subject to } X \succeq 0$$

we can form the Lagrangian with dual matrix variable Z and derive the optimality conditions,

$$\mathcal{L}(X, Z) = f(X) - \mathbf{trace}(ZX) \quad [Z \succeq 0] \qquad \begin{cases} \nabla_X \mathcal{L}(X, Z) = 0 \to \nabla f(X) - Z = 0 \\ X \succeq 0,\, Z \succeq 0,\, \mathbf{trace}(ZX) = 0. \end{cases}$$

A.5 Useful reformulations: epigraph and Schur's complement

Optimisation problems can be transformed into equivalent ones. By equivalent we mean different problems from which we can derive the solution of each other, for example:

$$\min_{x\in\mathbf{R}} f(x), \text{ subject to } |x| \leqslant 1 \quad \longleftrightarrow \quad \min_{x\in\mathbf{R}} f(x), \text{ subject to } x \leqslant 1, x \geqslant -1.$$

We are going to recap a few useful rules.

(A) Epigraph. (i.e., all the opt. problems can have a linear cost).

Consider the optimisation problem,

$$\min_{x\in\mathbf{R}^n} f(x), \text{ subject to } x \in X.$$

This problem is equivalent to the optimisation problem in $n + 1$ variables, with linear cost,

$$\min_{x\in\mathbf{R}^n, t\in\mathbf{R}} t, \text{ subject to } f(x) \leqslant t,\, x \in X.$$

(B) Substitutions.

$$\min_{x\in\mathbf{R}^n, y\in\mathbf{R}^m} f(x)+g(y), \text{ subject to } y=Ax+b \quad\longleftrightarrow\quad \min_{x\in\mathbf{R}^n} f(x)+g(Ax+b).$$

(C) Partial Min.

$$\min_{x\in X, y\in Y} f(x,y), \quad\longleftrightarrow\quad \min_{y\in Y} \underbrace{\left(\min_{x\in X} f(x,y)\right)}_{=\tilde{f}(y)}.$$

(D) Monotone transfer. Ψ_0 is a monotone increasing function, $\min f(x) \to \min \Psi_0 \circ f(x)$, ex:

$$\underbrace{\min_{x\in\mathbf{R}^n} \|Ax-b\|_2}_{\text{Non-diff, hard to solve}}, \quad\longleftrightarrow\quad \underbrace{\min_{x\in\mathbf{R}^n} \|Ax-b\|_2^2}_{\text{Super-easy to solve}}$$

(E) Schur's complement (very useful in SDPs)

Consider the squared symmetric matrix X, and partition it as follows,

$$X = \begin{bmatrix} A & B \\ B^\top & C \end{bmatrix}.$$

Then, the convex semidefinite constraint $X \succeq 0$ can be equivalently formulated in two ways,

$$\text{First possibility: } \{A \succ 0, \qquad S_1 = C - B^\top A^{-1} B \succeq 0\};$$

$$\text{Second possibility: } \{C \succ 0, \qquad S_2 = A - BC^{-1}B^\top \succeq 0\};$$

and conversely.

A.6 (Lagrangian) Duality

Consider the problem,

$$\underset{\boldsymbol{x}\in X}{\text{minimise}}\, f(x), \quad \text{subject to } A\boldsymbol{x}=b, g(\boldsymbol{x}) \leqslant 0$$

Define the Lagrangian,

$$\mathcal{L}(\boldsymbol{x},\lambda,\nu) = f(\boldsymbol{x}) + \lambda^\top (A\boldsymbol{x}-b) + \nu^\top g(\boldsymbol{x}),$$

We call primal problem:

$$(P) \quad \inf_{\boldsymbol{x}\in X}\left(\sup_{\lambda,\nu\geqslant 0} \mathcal{L}(\boldsymbol{x},\lambda,\nu)\right) = \begin{cases} \inf_{\boldsymbol{x}\in X} f(\boldsymbol{x}) & \text{if } A\boldsymbol{x}=b, g(\boldsymbol{x})\leqslant 0, \\ +\infty & \text{otherwise.} \end{cases}$$

We define the dual function:

$$q(\lambda,\nu) := \inf_{\boldsymbol{x}\in X} \mathcal{L}(\boldsymbol{x},\lambda,\nu),$$

and call the dual problem:

$$(D) \quad \sup_{\lambda,\nu\geqslant 0}\left(\inf_{\boldsymbol{x}\in X} \mathcal{L}(\boldsymbol{x},\lambda,\nu)\right) = \sup_{\lambda,\nu\geqslant 0} q(\lambda,\nu).$$

The dual function $q(\lambda,\nu)$ is always concave, so D is always convex! The proof of it is that q is the point-wise infimum of affine functions. Rem: $\max_i\{a_i^\top \boldsymbol{x} + b_i\}$ is convex.

Furthermore, $d^* = \text{val}(D) \leqslant \text{val}(P) = p^*$ or otherwise stated

$$\sup_{\lambda,\nu\geqslant 0}\inf_{\boldsymbol{x}\in X} \mathcal{L}(\boldsymbol{x},\lambda,\nu) \leqslant \inf_{\boldsymbol{x}\in X}\sup_{\lambda,\nu\geqslant 0} \mathcal{L}(\boldsymbol{x},\lambda,\nu).$$

For convex problems that have a solution, and under constraint qualification (e.g., under Slater's condition), then the KKT point is a saddle-point and we don't have duality gap (i.e., $d^* = p^*$). So for most convex problems solving the dual problem or the primal is the same.

A.7 Determining the dual problem: examples

Example A.1 (Dual of an LP) *Let's look at an LP in the (standard) form:*

$$\underset{x\in\mathbf{R}^n}{\text{minimise}}\ c^\top x, \qquad \textit{subject to}\ Ax = b,\ x \geqslant 0.$$

for matrix $A \in \mathbf{R}^{m\times n}$, and $b \in \mathbf{R}^m$. Introduce the Lagrangian multipliers $\lambda \in \mathbf{R}^m$ for $Ax - b = 0$ and $\nu \in \mathbf{R}^n$ for $-x \leqslant 0$, and write the Lagrangian function:

$$\mathcal{L}(x, \lambda, \nu) = c^\top x + \lambda^\top (Ax - b) - \nu^\top x.$$

The dual function is given by

$$q(\lambda, \nu) = \inf_x \mathcal{L}(x, \lambda, \nu) = -\lambda^\top b + \inf_x \left(c^\top x + \lambda^\top Ax - \nu^\top x\right) = -b^\top \lambda + \inf_x \left(x^\top (A^\top \lambda - \nu + c)\right),$$

which yields,

$$q(\lambda, \nu) = \begin{cases} -b^\top \lambda & \textit{if } A^\top \lambda - \nu + c = 0 \\ -\infty & \textit{otherwise.} \end{cases}$$

The dual problem is then given as,

$$\sup_{\lambda,\nu\geqslant 0} q(\lambda, \nu),$$

so,

$$\underset{\lambda,\nu\geqslant 0}{\text{minimise}}\ b^\top \lambda \qquad \textit{subject to: } A^\top \lambda - \nu + c = 0,\ \nu \geqslant 0.$$

In fact $\nu \geqslant 0$ is redundant and can be eliminated as

$$\underset{\lambda\in\mathbf{R}^m}{\text{minimise}}\ b^\top \lambda \qquad \textit{subject to: } A^\top \lambda + c \geqslant 0.$$

Note the duality between primal LP and dual LP, which is also a consequence of Farkas' lemma.

Example A.2 (Dual of a QCQP) *Consider the convex QCQP*

$$\begin{array}{ll} \underset{x\in\mathbf{R}^n}{\text{minimise}} & (1/2)x^\top P_0 x + c_0^\top x + r_0 \\ \textit{subject to} & (1/2)x^\top P_i x + c_i^\top x + r_i \leqslant 0, \qquad i = 1, \ldots, m, \end{array}$$

with P_0 positive definite and P_i positive semi-definite.

The Lagrangian is

$$\mathcal{L}(x, \nu) = (1/2)x^\top P(\nu)x + s(\nu)^\top x + r(\nu),$$

with

$$P(\nu) = P_0 + \sum_{i=1}^m \nu_i P_i, \qquad s(\nu) = s_0 + \sum_{i=1}^m \nu_i s_i, \qquad r(\nu) = r_0 + \sum_{i=1}^m \nu_i r_i.$$

This gives a dual function of the form,

$$q(\nu) = \inf_x \left((1/2)x^\top P(\nu)x + s(\nu)^\top x + r(\nu)\right)$$

which can be computed analytically for $\nu \geqslant 0$, since $P(\nu)$ is positive definite in this case. By optimality conditions,

$$x^*(\nu) = -P(\nu)^{-1} s(\nu),$$

and therefore

$$q(\nu) = -(1/2)s(\nu)^\top P(\nu)^{-1} s(\nu) + r(\nu).$$

The dual problem is therefore,

$$\begin{array}{ll} \underset{\nu\in\mathbf{R}^m}{\text{minimise}} & (1/2)s(\nu)^\top P(\nu)^{-1} s(\nu) - r(\nu) \\ \textit{subject to} & \nu \geqslant 0. \end{array}$$

Strong duality holds under Slater's condition, i.e., if there exists a x such that $(1/2)x^\top P_i x + c_i^\top x + r_i < 0$, $i = 1, \ldots, m$.

For $P_i = 0$ (the QP case), then $P(\nu)^{-1} = P_0^{-1}$ and

$$s(\nu)^\top P(\nu)^{-1} s(\nu) = (s_0 + S^\top \nu)^\top P_0^{-1}(s_0 + S^\top \nu) = \nu^\top S P_0^{-1} S^\top \nu + 2 s_0^\top P_0^{-1} \nu + s_0^\top P_0^{-1} s_0$$

where $S \in \mathbf{R}^{n \times m}$ is the matrix such that $S^\top = [s_1 | s_2 | \ldots | s_m]$. As such, with some re-labelling, the dual is

$$\begin{array}{ll} \underset{\nu \in \mathbf{R}^m}{\text{minimise}} & (1/2)\nu^\top Q \nu + w^\top \nu + z \\ \textit{subject to} & \nu \succcurlyeq 0, \end{array}$$

which is a convex QP.

Let's get back to the general case. The dual problem is

$$\begin{array}{ll} \underset{\nu \in \mathbf{R}^m}{\text{minimise}} & (1/2)s(\nu)^\top P(\nu)^{-1} s(\nu) - r(\nu) \\ \textit{subject to} & \nu \succcurlyeq 0. \end{array}$$

Use the epigraph form,

$$\begin{array}{ll} \underset{t, \nu \in \mathbf{R}^m}{\text{minimise}} & t - r(\nu) \\ \textit{subject to} & (1/2)s(\nu)^\top P(\nu)^{-1} s(\nu) \leqslant t \\ & \nu \succcurlyeq 0, t \geqslant 0, \end{array}$$

and then Schur's complement (since $P(\nu) \succ 0$),

$$\begin{array}{ll} \underset{t, \nu \in \mathbf{R}^m}{\text{minimise}} & t - r(\nu) \\ \textit{subject to} & \begin{bmatrix} t & s(\nu)^\top \\ s(\nu) & 2P(\nu) \end{bmatrix} \succeq 0 \\ & \nu \succcurlyeq 0, t \geqslant 0, \end{array}$$

Since $r(\nu), s(\nu)$ and $P(\nu)$ are affine functions of ν, then the above is a SDP.

A.8 Conjugate function

Consider a function $f : \mathbf{R}^n \to \mathbf{R}$. We define its conjugate function as,

$$f_\star(\boldsymbol{y}) := \sup_{\boldsymbol{x} \in \mathbf{R}^n} (\boldsymbol{y}^\top \boldsymbol{x} - f(\boldsymbol{x}))$$

When f is closed convex and proper, then

$$(\partial f)^{-1} = \partial f_\star$$

where ∂f is the subdifferential set.