

COMPUTATIONAL MATHEMATICS OPTIMIZATION

Based on prof. Antonio Frangioni's lectures

Gemma Martini

August 9, 2020

A sincere thank you to
Alessandro Cudazzo, Donato Meoli,
Giulia Volpi and all those who helped me
improving these notes in style and contents.

Contents

1 19th of September 2018	7
1.1 Introduction to machine learning problems	7
1.2 Optimization	9
1.2.1 Linear estimation	9
1.2.2 Low-rank approximation	10
1.2.3 Support vector machines	11
2 21st of September 2018	13
2.1 Mathematical background for optimization problems	13
2.1.1 Multi-objective Optimization	14
2.2 Infima, suprema and extended reals	16
2.3 (Monotone) Sequences in \mathbb{R} and optimization	17
2.4 Vector spaces and topology	17
2.5 Limit of a sequence in \mathbb{R}^n	21
3 27th of September 2018	23
3.1 Continuity	26
3.2 Derivatives	26
3.2.1 Multivariate differentiability	27
4 3rd of October 2018	31
4.1 Simple functions	35
4.1.1 Linear functions	35
4.1.2 Quadratic functions	35
5 5th of October 2018	37
5.1 Unconstrained optimization	37
5.1.1 First order model	38
5.1.2 Second order model	39
5.2 Convexity	40
5.2.1 Convex sets	40
5.2.2 Convex functions	43
6 11th of October 2018	47
6.0.1 Non-differentiable functions	50

7 17th of October 2018	53
7.1 Optimization algorithms	53
7.1.1 Gradient method for quadratic functions	54
8 19th of October 2018	59
8.0.1 Gradient method for quadratic functions (cont.)	59
8.0.2 Quadratic gradient descent in Matlab	62
9 24th of October 2018	65
9.1 Gradient method for non quadratic functions	65
9.1.1 Finding the best step size	66
10 25th of October 2018	71
11 8th of November 2018	83
11.1 Good practices for the project's design	83
12 14th of November 2018	85
12.1 General descent methods	85
12.2 Newton's method	86
12.2.1 Convex case	86
12.2.2 Interpretation of Newton's method	89
12.2.3 Non convex case	90
13 16th of November 2018	93
13.1 Quasi-newton methods	94
13.1.1 Davidson-Fletcher-Powell	94
13.1.2 Broyden-Fletcher-Goldfarb-Shanno	95
13.1.3 Poorman's approach	95
13.2 Conjugate gradient method	96
14 22nd of November 2018	99
14.1 Deflected gradient methods	99
14.1.1 Heavy ball gradient method	99
14.1.2 Accelerated gradient	100
14.2 Incremental gradient methods	102
14.3 Subgradient methods	103
15 28th of November 2018	105
15.1 Subgradient methods	105
15.1.1 Target level stepsize	107
15.2 Deflected subgradient	108
15.3 Smoothed gradient methods	109
15.4 Cutting-plane algorithm	111
15.5 Bundle methods	113

16 30th of November 2018	115
16.1 Constrained optimization	115
16.1.1 Linear equality constraints	116
16.1.2 Background for linear inequality constraints	118
17 6th of December 2018	125
17.1 Duality	125
17.2 Lagrangian duality	126
17.3 Specialized dual	128
17.3.1 Linear programs	128
17.3.2 Quadratic programs	128
17.3.3 Conic program	129
17.4 Fenchel's duality	130
18 12th of December 2018	131
18.1 Quadratic problem with linear equality constraints	131
18.2 Inequality constrained problems	133
18.2.1 Projected gradient method	133
18.2.2 Projected gradient method with box constraints	136
18.2.3 Active-set method for quadratic programs	137
18.2.4 Frank-Wolfe method	138
19 14th of December 2018	141
19.1 Dual methods for linear constrained optimization	141
19.1.1 Separable problems and partial dual	142
19.2 Primal/dual methods or barrier methods	142
19.2.1 Barrier function and central path	142
19.3 Primal-dual interior point method	145

Lecture 1

19th of September 2018

A. Frangioni

This course will deal with the optimization and numerical analysis of machine learning problems. We are not going to discuss difficult problems (e.g. NP-hard problems), besides we will present an efficient solution for simple ones (often **convex** ones), since we are dealing with huge amount of data.

Let us start with a warm up on machine learning problems.

1.1 Introduction to machine learning problems

Machine learning techniques are not as “young” as it might seem, the intuition has been there for ages, but we did not have enough calculus power. Machine learning algorithms have started working well recently, thanks to the many improvements in computer performances; for this reason, it is becoming a more and more popular subject to study.

The main idea behind machine learning is to take a huge amount of data (e.g. frames of a video for object-recognition) and squeeze them, in order to process them. This intuitive concept is translated into mathematical terms as “building a **model**” that fits our data. As in practical engineering problems, people want to construct a model (a small sized representation of the large thing we want to produce in the end) and try to understand its behaviour, before actually build such an object. Take as an example the problem of designing a jet. It is not clever to start building the plane before designing a cheap prototype to better study its behaviour in the atmosphere.

The kind of models we want to build are cheap to construct and as close as possible to the real problem we are studying. In physics, people try to find the best mathematical model to describe a real world phenomenon. The main issue is computation, since the more accurate the model, the more costly the prediction phase. Hence, a good model is a trade-off between accuracy and simplicity, namely it provides good prediction without incurring in slow computations.

The model, though, has to be parametric: we do not have only one model, we have a “shape” of a model, which is fit to our problem through the tuning of some parameters.

Example 1.1.1. As an example, we are given three couples: $f(x_1) = y_1$, $f(x_2) = y_2$, $f(x_3) = y_3$, as shown in Figure 1.1.

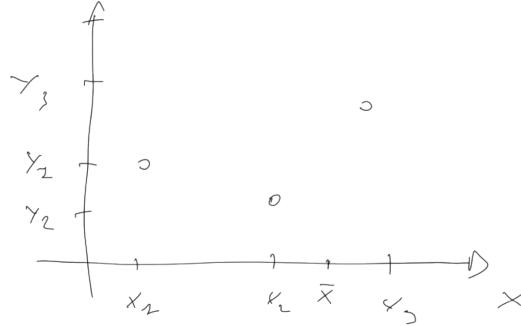


FIGURE 1.1: Geometric representation of the input. We are interested in finding a model that fits the input data and allows to predict \bar{y} out of \bar{x} .

We need to make some choices: first, we need to decide the kind of model we believe is a good approximation of the objective function, say a linear model $f(x) = ax + b$. After doing that, we are left with choosing its parameters (in order to pick a line among the whole family of linear functions), namely a and b .

The aim is to build a model that fits the data we are given and then tune the parameters in order to achieve a good accuracy for a given application (recall that the model is parametric, hence the right values for the parameters have to be learned).

Another important characteristic of a good model is that it should not take too long to be built.

In this course we do not concentrate on the problem of finding the model that best fits our data, but we are already given a problem and a model and we only study its behaviour through its parameters. In other words, within the family of models with a given outline, we want to find the one that better represents the phenomena observed. This is called **fitting** and it is clearly some sort of optimization problem, where the fitting task is typically the computational bottleneck.

However, machine learning is more than simply fitting: fitting minimizes training error (or empirical risk), but ML aims at minimizing test error (i.e. generalization error).

A machine learning solution first builds a model that fits the observed data and then performs an hyper-parameter tuning in order to achieve a good “predicting power” on unseen inputs. In machine learning, a model that fits the

data while achieving good performances on new examples is said to be “non **overfitting**”.

For machine learning purposes, a mathematical model should be:

- *accurate* (describing well the process under consideration)
- *computationally inexpensive* (providing answers rapidly)
- *general* (it can be applied to many different processes)

It goes without saying that achieving all these goals is practically impossible.

1.2 Optimization

In the rest of this lecture we are going to better understand what an optimization problem is, through some intuitive real world examples.

◆ Terminology

In this course we will refer to vectors using the bold notation $\mathbf{v} \in \mathbb{R}^n$. The i -th entry of vector \mathbf{v} is denoted as $v_i \in \mathbb{R}$, while the j -th pattern in the training set is a couple of vectors $(\mathbf{x}^j, \mathbf{y}^j)$.

For more details on vectors and vector spaces see Section 2.4.

1.2.1 Linear estimation

Let us consider a phenomenon measured by one number $y \in \mathbb{R}$ that is believed to depend on a vector $\mathbf{x} = [x_1, \dots, x_n]$. We are provided a set of observations: $\{(y^1, \mathbf{x}^1), \dots, (y^m, \mathbf{x}^m)\}$.

Definition 1.2.1 (Linear model). *Let $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ be the objective function. We call $\tilde{f}(\mathbf{x}) = \sum_{i=1}^n w_i x_i + w_0 = \mathbf{w}_+^T \mathbf{x} + w_0$ the **linear model** of f for a given set of parameters, which is a vector $\mathbf{w} = (w_0, \mathbf{w}_+) = (w_0, w_1, \dots, w_n) \in \mathbb{R}^{n+1}$.*

How can we evaluate the “similarity” between our model and the objective function? Through computing the “error” or difference between the objective function value and the model prediction on each input. Under this assumption, the error function may be used to find the best parameters for our model, through solving a minimum problem:

Definition 1.2.2 (Least squares problem). *Let $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ be the objective function, such that $f(\mathbf{x}) = \mathbf{y}$ and let $X\mathbf{w}$ be our linear model. Then we can find the best values for vector $\mathbf{w} \in \mathbb{R}^{n+1}$ by solving*

$$\min_{\mathbf{w}} \|\mathbf{y} - X\mathbf{w}\|$$

where $Y = \begin{pmatrix} \mathbf{y}^1 \\ \vdots \\ \mathbf{y}^m \end{pmatrix}$ and $X = \begin{pmatrix} 1 & \mathbf{x}^1 \\ \vdots & \vdots \\ 1 & \mathbf{x}^m \end{pmatrix}$.

If the matrix X is invertible then the simple solution is $\mathbf{w} = X^{-1}\mathbf{y}$. The point is that this operation is very costly when dealing with a huge number of entries (in the next paragraph we will see a way to manage it).

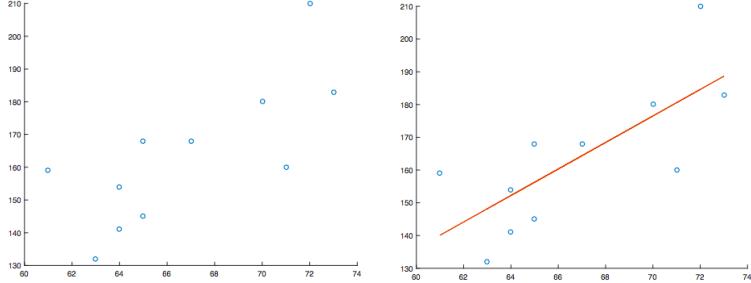


FIGURE 1.2: A linear estimation fitting example

1.2.2 Low-rank approximation

A (large, sparse) matrix $M \in M(n, m, \mathbb{R})$ describes a phenomenon depending on pairs (e.g., objects bought by customers) and we may want to approximate that matrix as the product between two smaller matrices (find a few features that describe most of users' choices): a “tall thin” matrix $A \in M(n, k, \mathbb{R})$ and a “fat large” $B \in M(k, m, \mathbb{R})$ (where $k \ll n, m$).

$$\boxed{M} \approx \boxed{A} \cdot \boxed{B}$$

This problem can be translated into a numerical analysis problem of the following shape

$$\min_{A, B} \|M - AB\|$$

The matrices A and B can be obtained from eigenvectors of $M^T M$ and MM^T , but that's a huge, possibly dense matrix. So a more efficient way should be used, which also avoids the explicit formation of $M^T M$ and MM^T because that would need a lot of memory.

Efficiently solving this problem requires:

- low-complexity computation
- avoiding the explicit forming of $M^T M$ and MM^T
- exploiting the structure of M (sparsity, similar columns, ...)
- ensuring that the solution is *numerically stable*

1.2.3 Support vector machines

Let us consider the so-called “decision problem”: given a set of values of many parameters (variables) “label” a person i as ill or healthy, $y^i \in \{1, -1\}$.

The geometric intuition in two dimensions is given by Figure 1.3. We would like to find the line that better splits the plane into two regions, because this could help to diagnose the next patient. The rationale here is to maximize the space between the line and the nearest points (called **margin**), in order to have a better accuracy.

The distance between the two hyperplanes (w_0, \mathbf{w}_+) and (w'_0, \mathbf{w}_+) in Figure 1.3 is computed as $\frac{2}{\|\mathbf{w}_+\|}$, so in order to maximize the distance between the planes we aim at minimizing $\|\mathbf{w}_+\|$. For each hyperplane that lies between (w_0, \mathbf{w}_+) and (w'_0, \mathbf{w}_+) the following holds

$$\begin{cases} \mathbf{w}_+ \mathbf{x}^i + w_0 \geq 1 & \text{if } y^i = 1 \\ \mathbf{w}_+ \mathbf{x}^i + w_0 \leq -1 & \text{if } y^i = -1 \end{cases}$$

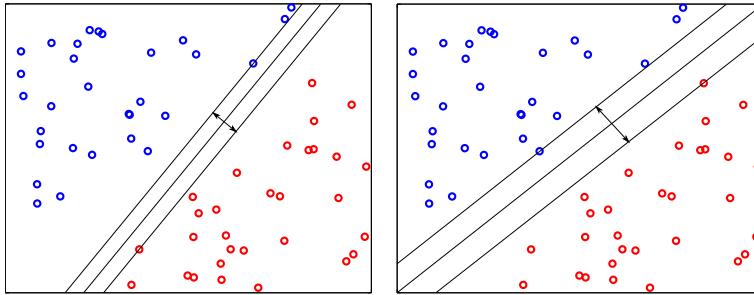


FIGURE 1.3: There are many possible boundaries that can be chosen as a model using many angular coefficients. Our best guess is the one that maximizes the distance between the line and the nearest points.

Under the hypotheses of this optimization problem, the maximum-margin separating hyperplane (assuming that any exists) is the solution of

$$\min_{\mathbf{w}} \{\|\mathbf{w}_+\|^2 : y^i(\mathbf{w}_+ \mathbf{x}^i + w_0) \geq 1, i = 1, \dots, m\}$$

In practice, most of the times there is no such line. To overcome this issue we introduce the concept of “penalty” that accounts for the number of points that are misclassified, through the following

Definition 1.2.3 (SVM Primal problem). *We term Support Vector Machine primal problem (SVM-P) a convex constrained problem with complex constraints that is formalized as*

$$\begin{cases} \min_{\mathbf{w}, \xi} \|\mathbf{w}_+\|^2 + C \sum_{i=1}^m \xi_i \\ y^i(\mathbf{w}_+ \mathbf{x}^i + w_0) \geq 1 - \xi_i, \xi_i \geq 0, \forall i = 1, \dots, m \end{cases} \quad (\text{SVM-P})$$

Where C is an **hyper-parameter** that weights the violation of the separating margin.

This definition formalizes the intuition that the approximated function may have a greater norm and lead to a very small misclassification error, or it could be the other way round. Both these solutions are acceptable and their performances depend only on the problem.

Whenever we are able to solve a multi-objective optimization problem we are also able to solve what is called the **dual problem**, which in our case has the following shape.

Definition 1.2.4 (SVM Dual problem). *We call Support Vector Machine dual problem (SVM-D) a convex constrained quadratic problem defined as*

$$\begin{cases} \max_{\alpha} \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i < \mathbf{x}^i, \mathbf{x}^j > \alpha_j \\ \sum_{i=1}^m y^i \alpha_i = 0 \\ 0 \leq \alpha_i \leq C, \forall i = 1, \dots, m \end{cases} \quad (\text{SVM-D})$$

The idea behind the theory of duality is to obtain the solution of a problem by solving an (apparently) different one.

Lemma 1.2.1. *Given an optimal solution α^* for the dual problem (SVM-D), then $\mathbf{w}^* = \sum_{i=1}^m \alpha_i^* y^i \mathbf{x}^i$ is optimal for the primal problem (SVM-P).*

A reader who has a deeper background in the field of machine learning knows that the scalar product in the dual form allows the usage of the so-called “kernel trick”. This mathematical transformation allows the mapping of the input space into a larger feature space where the points are more likely to be linearly separable.

Theoretically, the feature space can be infinite-dimensional, provided that the scalar product can be (efficiently) computed.

This whole course has the aim of presenting some techniques for solving efficiently **convex quadratic problems**, as the ones presented above.

Lecture 2

21st of September 2018

A. Frangioni

2.1 Mathematical background for optimization problems

Definition 2.1.1 (Minimum problem). *Let S be a set, called **feasible region** and let $f : S \rightarrow \mathbb{R}$ be any function, called **objective function**. We term **minimum problem** the problem of finding the minimum value of such f . Formally,*

$$f_* = \min_x \{f(x) : x \in S\} \quad (\text{P})$$

Definition 2.1.2 (Feasible solution). *Let $x \in F$ be a solution of the minimum problem in which the domain is a superset of $S \subset F$. We say that x is a **feasible solution** if $x \in S$. Conversely, x is **unfeasible** if $x \in F \setminus S$.*

Definition 2.1.3 (Optimal solution). *Under the same hypothesis of the above definition, we call x_* that realizes $f(x_*) = f_*$ an **optimal solution**, where $f_* \leq f(x) \forall x \in S, \forall v > f_* \exists x \in S$ s.t. $f(x) < v$.*

It is possible to find problems where there is no optimal solution at all.

Observation 2.1.1. *There are two cases in which it is not possible to find an optimal solution:*

1. *The domain is empty, which may be not trivial to prove, since it is an NP-hard problem sometimes;*
2. *We want to find the minimum of the objective function but the function is unbounded below ($\forall M \exists x_M \in S$ s.t. $f(x_M) \leq M$). Conversely, we aim to maximize a function, which is unbounded above.*

Example 2.1.1 (Bad optimization problems). *The following practical cases are examples of problems that do not admit an optimal solution:*

- *empty case ($S = \emptyset$):* $\min\{f(x) = x : x \in \mathbb{R} \wedge x \leq -1 \wedge x \geq 1\}$
- *unbounded (below):* $\min\{f(x) = x : x \in \mathbb{R} \wedge x \leq 0\}$
- *bad f and S:* $\min\{f(x) = x : x \in \mathbb{R} \wedge x > 0\}$
- *bad f:* let us consider an iterative algorithm that moves towards the optimum. It may happen that the function decreases and increases along a certain direction, such function is not continuous so it is impossible to reach the optimum:

$$\min \left\{ f(x) = \begin{cases} x & \text{if } x > 0 \\ 1 & \text{if } x = 0 \end{cases} \quad x \in [0, 1] \right\}$$

Solving an optimization problem can be one of the following procedures:

1. Finding x_* and proving it is optimal
2. Proving $S = \emptyset$
3. Constructively proving that the function is unbounded ($\forall M \exists x_M \in S$ s.t. $f(x_M) \leq M$).

Typically “ $x \in \mathbb{R}$ ” actually means “ $x \in \mathbb{Q}$ ” with up to k digits precision and most of the times we consider optimal a solution which is close to the true optimal value, modulo some error (we call \bar{x} , the approximated optimal).

Definition 2.1.4 (Absolute error). We call **absolute error** the gap between the real value and the one we obtained. Formally,

$$f(\bar{x}) - f_* \leq \varepsilon$$

Definition 2.1.5 (Relative error). We term **relative error** the absolute error, normalized by the true value of the function

$$(f(\bar{x}) - f_*) / |f_*| \leq \varepsilon$$

2.1.1 Multi-objective Optimization

It may happen that there are more than one function that need to be minimized (maximized) and they could be contrasting or have incomparable units (apples vs oranges).

In multi-objective optimization, typically there is not a feasible solution that minimizes all objective functions simultaneously. It is in this context that *Pareto optimal solutions* are introduced: such solutions cannot be improved in any of the objectives without degrading at least one of the other objectives.

Definition 2.1.6 (Pareto frontier). Let (P) be the minimum multi-objective optimization problem formalized as

$$\min_x \{[f_1(x), \dots, f_k(x)] : x \in S\}$$

A feasible solution $x^1 \in S$ is said to (**Pareto**) dominate another solution $x^2 \in S$, if

$$\begin{cases} f_i(x^1) \leq f_i(x^2) & \forall i \in \{1, 2, \dots, k\} \\ f_j(x^1) \leq f_j(x^2) & \forall j \in \{1, 2, \dots, k\} \end{cases}$$

A solution $x^* \in S$ (and the corresponding outcome $f(x^*)$) is called **Pareto optimal**, if there does not exist another solution that dominates it. The set of Pareto optimal outcomes is often called the **Pareto frontier**.

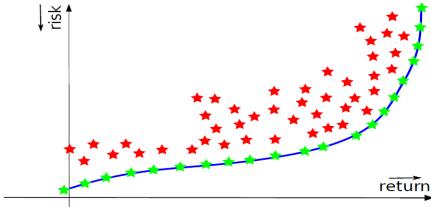


FIGURE 2.1: An example of Pareto frontier

Example 2.1.2. Let us take two functions f_1 and f_2 . We want to solve the minimization problem

$$\min_x \{[f_1(x), f_2(x)] : x \in S\} \quad (P)$$

and to do so we present two different approaches:

SCALARIZATION. Using a linear combination of the two functions, formally, $f(x) = \alpha f_1(x) + \beta f_2(x)$. An example, where $\alpha = 1$ is shown in Figure 2.2.

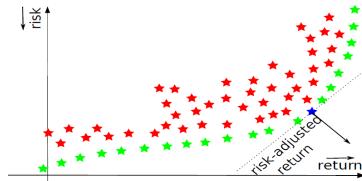


FIGURE 2.2: Maximize risk-adjusted return

BUDGETING. Intuitively corresponds to taking into account only one objective function and add the others as constraints, provided that the values of the other functions are not too high. In formal terms, $f(x) = f_1(x)$, where $S := S \cup \{x \in S : f_2(x) \leq b\}$. As an example see Figure 2.3.

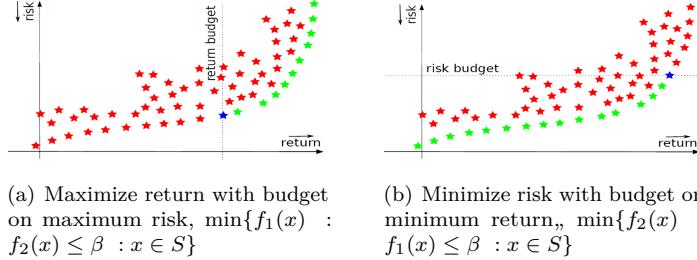


FIGURE 2.3: Budgeting

2.2 Infima, suprema and extended reals

Let us introduce some mathematical background on minimization (maximization).

Definition 2.2.1 (Totally ordered set). *We say that a set S is **totally ordered** if $\forall x, y \in S$, either $f(x) \leq f(y)$ or $f(y) \leq f(x)$.*

Definition 2.2.2 (Infima and suprema). *Let R be a totally ordered set and let s be one of its subsets ($S \subseteq R$):*

\underline{s} is the **infimum** of S ($\underline{s} = \inf S$) iff $\underline{s} \leq s \quad \forall s \in S \quad \wedge \quad \forall t > \underline{s} \exists s \in S$ s.t. $s \leq t$

\bar{s} is the **supremum** of S ($\bar{s} = \sup S$) iff $\bar{s} \geq s \quad \forall s \in S \quad \wedge \quad \forall t < \bar{s} \exists s \in S$ s.t. $s \geq t$

An attentive reader may notice that it is possible that infima (suprema) do not exist in \mathbb{R} .

Definition 2.2.3 (Extended real). *In the case of unbounded functions the value of infima or suprema are ∞ , and we call **extended reals** $\overline{\mathbb{R}} = -\infty \cup \mathbb{R} \cup +\infty$.*

Property 2.2.1. *The following holds for the extended reals:*

- For all $S \subseteq \mathbb{R}$, $\sup/\inf S \in \overline{\mathbb{R}}$
- $\inf S = -\infty$ indicates that there is no (finite) infimum
- $\inf \emptyset = \infty$, $\sup \emptyset = -\infty$

2.3 (Monotone) Sequences in \mathbb{R} and optimization

We are interested in studying sequences, because iterative methods start from a certain point and move towards the optimal.

❖ Terminology

We denote a sequence of iterates as $\{x_i\} \subset S$ and we denote the plugging a function f into such a sequence as $v_i = f(x_i)$.

Definition 2.3.1 (Limit). *Given a sequence $\{x_i\}$ the **limit** for $i \rightarrow \infty$ is defined as*

$$\lim_{i \rightarrow \infty} v_i = v \iff \forall \varepsilon > 0 \exists h \text{ s.t. } |v_i - v| \leq \varepsilon \forall i \geq h$$

It may happen that a sequence has or does not have a limit. For example $\{\frac{1}{n}\}$ has limit 0 for $n \rightarrow +\infty$, while $\{(-1)^n\}$ does not have any.

Fact 2.3.1. *Let us be given a monotone sequence, then the sequence **does** have a limit.*

Notice that a sequence either is monotone or it can be “split” into two monotone sequences.

Example 2.3.1. *Let us consider the sequence $\{(-1)^n\}$. It does not converge to any value, but it can be split into $\{(-1)^{2n}\}$ and $\{(-1)^{2n+1}\}$ and these two subsequences are both monotone.*

Forcing monotonicity on sequences in \mathbb{R} can be performed splitting the sequence into a non-decreasing sequence and a non-increasing sequence:

$$\underline{v}_i^* = \inf\{v_h : h \leq i\} \text{ and } \bar{v}_i^* = \sup\{v_h : h \leq i\}$$

This way we get $\underline{v}_1 \leq \underline{v}_2 \leq \dots$ and $\bar{v}_1 \geq \bar{v}_2 \geq \dots$ and they have a limit: $\liminf_{i \rightarrow \infty} \underline{v}_i \lim_{i \rightarrow \infty} \underline{v}_i^* = \underline{v}_\infty^* = f_*$ and $\limsup_{i \rightarrow \infty} \bar{v}_i \lim_{i \rightarrow \infty} \bar{v}_i^* = \bar{v}_\infty^* = f_*$

Fact 2.3.2. *The following holds:*

- $\bar{v}_i \geq \underline{v}_i \Rightarrow \limsup_{i \rightarrow \infty} v_i \geq \liminf_{i \rightarrow \infty} v_i$
- $\lim_{i \rightarrow \infty} v_i = v \iff \limsup_{i \rightarrow \infty} v_i = v = \liminf_{i \rightarrow \infty} v_i$

2.4 Vector spaces and topology

For the problems we are going to study, real numbers are used to describe each feature of a dataset and this has the mathematical equivalent in the term “vector”.

Definition 2.4.1 (Euclidean vector space). *We call **Euclidean space***

$$\mathbb{R}^n := \left\{ \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} : x_i \in \mathbb{R}, i = 1, \dots, n \right\}$$

Equivalently, we can characterize the Euclidean space as Cartesian product of \mathbb{R} n times: $\mathbb{R}^n = \mathbb{R} \times \mathbb{R} \times \dots \mathbb{R}$. Every vector space and in particular Euclidean spaces are closed under sum and scalar multiplication.

The main operations on elements of the Euclidean space (vectors) are:

$$\text{SUM: } \mathbf{x} + \mathbf{y} := \begin{pmatrix} x_1 + y_1 \\ x_2 + y_2 \\ \vdots \\ x_n + y_n \end{pmatrix}$$

$$\text{SCALAR MULTIPLICATION: } \alpha \mathbf{x} = \begin{pmatrix} \alpha x_1 \\ \alpha x_2 \\ \vdots \\ \alpha x_n \end{pmatrix}$$

In linear algebra, $\mathbf{x} \in \mathbb{R}^n$ is a column vector $\in \mathbb{R}^n$, whenever a row vector is needed for dimensionality issues it is denoted as x^T .

Definition 2.4.2 (Basis). *Any vector $\mathbf{x} \in \mathbb{R}^n$ can be obtained from a linear combination of a set of vectors that is called “basis”. In Euclidean spaces there exists a special basis that is called “canonical”, where each vector \mathbf{u}^i has a 1 in position i and a 0 elsewhere.*

Definition 2.4.3 (Finite vector space). *Let $V(K, n, m)$ be a vector space. It is said to be **finite** if its basis have a finite cardinality.*

At this point, it is crucial to observe that not all vector spaces are finite nor they are a totally ordered set.

In this course we will deal with the concept of limit over a vector space very often and for this purpose we are going to introduce a topology on vector spaces (i.e. norm, scalar product, distance).

Definition 2.4.4 (Euclidean norm). *Let $\mathbf{x} \in \mathbb{R}^n$, we define **euclidean norm** of \mathbf{x} :*

$$\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^n x_i^2} = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$$

Fact 2.4.1. *Any norm on a vector space has the following properties:*

1. $\|\mathbf{x}\| \geq 0$ and $\forall \mathbf{x} \in \mathbb{R}^n$, $\|\mathbf{x}\| = 0 \iff \mathbf{x} = \mathbf{0}$;
2. $\|\alpha \mathbf{x}\| = |\alpha| \|\mathbf{x}\|$, $\forall \mathbf{x} \in \mathbb{R}^n$, $\alpha \in \mathbb{R}$;
3. $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$, $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ (triangle inequality).

Fact 2.4.2 (Cauchy-Schwartz inequality). Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$. The following holds:

$$\langle \mathbf{x}, \mathbf{y} \rangle^2 \leq \langle \mathbf{x}, \mathbf{x} \rangle \langle \mathbf{y}, \mathbf{y} \rangle \equiv |\langle \mathbf{x}, \mathbf{y} \rangle| \leq \|\mathbf{x}\| \|\mathbf{y}\|, \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n$$

Fact 2.4.3 (Parallelogram Law).

$$2\|\mathbf{x}\|^2 + 2\|\mathbf{y}\|^2 = \|\mathbf{x} + \mathbf{y}\|^2 + \|\mathbf{x} - \mathbf{y}\|^2$$

Fact 2.4.4.

$$\|\mathbf{x} + \mathbf{y}\|^2 = \|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 + 2\langle \mathbf{x}, \mathbf{y} \rangle$$

The 2-norm is a special case of the more general p -norm.

Definition 2.4.5 (p -norm). Let $p \geq 1$ be a real number, the p -norm of a vector $\mathbf{x} = (x_1, \dots, x_n)$ is defined as follow:

$$\|\mathbf{x}\|_p := \left(\sum_{i=1}^n |x_i|^p \right)^{1/p}$$

We require $p \geq 1$ for the general definition of the p -norm because the triangle inequality fails to hold if $p < 1$. The p -norm is convex for $p \geq 1$ and non convex for $p < 1$.

The most important p -norms are the following:

- $\|\mathbf{x}\|_1 := \sum_{i=1}^n |x_i|$
- $\|\mathbf{x}\|_\infty := \max\{|x_i| : i = 1, \dots, n\}$
- $\|\mathbf{x}\|_i := |\{i : |x_i| > 0\}|$

Fact 2.4.5. For any given finite-dimensional vector space $V(K, n)$ (e.g. \mathbb{R}^n), all norms on V are equivalent (therefore, convergence in one norm implies convergence in any other norm). Formally, $\forall \|\cdot\|_A, \|\cdot\|_B$:

$$\exists 0 < \alpha < \beta \text{ s.t. } \alpha \|\mathbf{x}\|_A \leq \|\mathbf{x}\|_B \leq \beta \|\mathbf{x}\|_A \quad \forall \mathbf{x} \in V$$

This rule may not apply in infinite-dimensional vector spaces such as function spaces. For such cases we define the

Fact 2.4.6 (Holder's inequality). Let $V(K, n)$ be a (possibly infinite) vector space, for any two norms $\forall \|\cdot\|_A, \|\cdot\|_B$ the **Holder's inequality** holds

$$\langle \mathbf{x}, \mathbf{y} \rangle^2 \leq \|\mathbf{x}\|_A \|\mathbf{y}\|_B \quad 1/A + 1/B = 1$$

Definition 2.4.6 (Ball). Let $\bar{\mathbf{x}} \in \mathbb{R}^n$. We term **ball** centered in $\bar{\mathbf{x}}$ and having ε as radius as the set of points that are close enough to \mathbf{x} : $B(\bar{\mathbf{x}}, \varepsilon) = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x} - \bar{\mathbf{x}}\| \leq \varepsilon\}$.

Let us take a unit ball, if the center of the unit-ball is in the origin $(0, 0)$, then each point on the unit-ball will have the same p -norm (i.e. 1). The unit ball therefore describes all points that have “distance” 1 from the origin, where “distance” is measured by the p -norm. In Figure 2.4 we may observe the different shapes of the same ball varying the value of p in the p -norm.

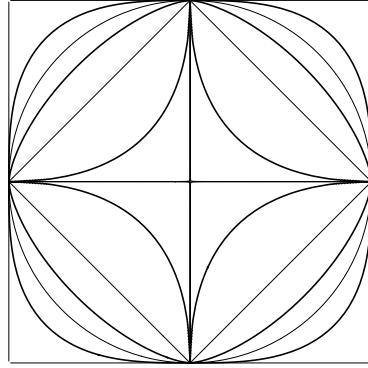


FIGURE 2.4: The shapes of balls centered in the origin of radius 1 varying the value of p -norm.

Definition 2.4.7 (Standard scalar product). Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ we define the **standard scalar product** between these two vectors as

$$\langle \mathbf{x}, \mathbf{y} \rangle := \mathbf{y}^T \mathbf{x} = \sum_{i=1}^n x_i y_i = x_1 y_1 + \cdots + x_n y_n$$

Fact 2.4.7. A scalar product has the following properties:

1. $\langle \mathbf{x}, \mathbf{y} \rangle = \langle \mathbf{y}, \mathbf{x} \rangle \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ (symmetry)
2. $\langle \mathbf{x}, \mathbf{x} \rangle \geq 0, \quad \forall \mathbf{x} \in \mathbb{R}^n, \quad \langle \mathbf{x}, \mathbf{x} \rangle = 0 \iff \mathbf{x} = \mathbf{0};$
3. $\langle \alpha \mathbf{x}, \mathbf{y} \rangle = \alpha \langle \mathbf{x}, \mathbf{y} \rangle, \quad \forall \mathbf{x} \in \mathbb{R}^n, \alpha \in \mathbb{R};$
4. $\langle \mathbf{x} + \mathbf{y}, \mathbf{z} \rangle = \langle \mathbf{x}, \mathbf{z} \rangle + \langle \mathbf{y}, \mathbf{z} \rangle, \quad \forall \mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{R}^n.$

An important geometric characterization of the scalar product is the one that uses angles:

$$\langle \mathbf{x}, \mathbf{y} \rangle = \|\mathbf{x}\| \|\mathbf{y}\| \cos \theta$$

According to this new definition, we can observe that \mathbf{x} and \mathbf{y} are orthogonal ($\mathbf{x} \perp \mathbf{y}$) iff $\langle \mathbf{x}, \mathbf{y} \rangle = 0$ and in all the other cases they somehow point in the same direction.

We could rewrite the scalar product between \mathbf{x} and \mathbf{y} as $\mathbf{y}^T I \mathbf{x}$, where $I \in D(\mathbb{R}, n)$ is the identity matrix. Thanks to this observation. we can state the more general form of the scalar product:

Definition 2.4.8 (Scalar product). *Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and let $M \in M(\mathbb{R}, n)$ be a positive definite matrix (for further details see). We call **scalar product** the following*

$$\langle \mathbf{x}, \mathbf{y} \rangle_M := \mathbf{y}^T M \mathbf{x}$$

Aggiungere reference al punto in cui si definisce una matrice definita positiva

Definition 2.4.9 (Euclidean distance). *The **Euclidean distance** between points \mathbf{x} and \mathbf{y} in \mathbb{R}^n is the length of the line segment connecting them. In Cartesian coordinates, if $\mathbf{x} = (x_1, x_2, \dots, x_n)$ and $\mathbf{y} = (y_1, y_2, \dots, y_n)$ are two points in the n -dimensional Euclidean space, then the distance (d) from \mathbf{x} to \mathbf{y} , or from \mathbf{y} to \mathbf{x} is given by*

$$d(\mathbf{x}, \mathbf{y}) := \|\mathbf{x} - \mathbf{y}\| = \sqrt{(x_1 - y_1)^2 + \dots + (x_n - y_n)^2}$$

Fact 2.4.8. *The Euclidean distance has the following properties:*

1. $d(\mathbf{x}, \mathbf{y}) \geq 0 \quad \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n, \quad d(\mathbf{x}, \mathbf{y}) = 0 \iff \mathbf{x} = \mathbf{y}$
2. $d(\alpha \mathbf{x}, 0) = |\alpha| d(\mathbf{x}, 0) \quad \forall \mathbf{x} \in \mathbb{R}^n, \alpha \in \mathbb{R}$
3. $d(\mathbf{x}, \mathbf{y}) \leq d(\mathbf{x}, \mathbf{z}) + d(\mathbf{z}, \mathbf{y}) \quad \forall \mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{R}^n$ (triangle inequality)

2.5 Limit of a sequence in \mathbb{R}^n

We have now all the tools to define the notion of limit of a sequence in \mathbb{R}^n .

Definition 2.5.1 (Limit of a sequence in the Euclidean space). *Let $\{\mathbf{x}_i\} \subset \mathbb{R}^n$ be a sequence in \mathbb{R}^n . We say that $\{\mathbf{x}_i\}$ converges to a **limit** \mathbf{x} for $i \rightarrow +\infty$ and denote $\lim_{i \rightarrow \infty} \mathbf{x}_i = \mathbf{x}$ or $\{\mathbf{x}_i\} \rightarrow \mathbf{x}$ if $\forall \varepsilon > 0 \exists h$ s.t. $d(\mathbf{x}_i, \mathbf{x}) \leq \varepsilon \quad \forall i \geq h$ or, equivalently, $\forall \varepsilon > 0 \exists h$ s.t. $\mathbf{x}_i \in \mathcal{B}(\mathbf{x}, \varepsilon) \quad \forall i \geq h$ or, even $\lim_{i \rightarrow \infty} d(\mathbf{x}_i, \mathbf{x}) = 0$.*

In the definition of limit there is the idea that the points of $\{\mathbf{x}_i\}$ eventually all come arbitrarily close to \mathbf{x} . Moreover, since \mathbb{R}^n is not totally ordered, there is no obvious \liminf nor \limsup .

Lecture 3

27th of September 2018

A. Frangioni

Definition 3.0.1 (Minimizing sequence). *Let $\{x_i\} \in S$ be a sequence and let $f : S \rightarrow \mathbb{R}^n$. We say that $\{x_i\}$ is a **minimizing sequence** if the sequence of function values $\{f(x_i)\}$ tends to the greatest lower bound of f ($f_* = \min\{f(x) : x \in S\}$).*

Example 3.0.1. *As an example, consider the following minimum problems and sequences:*

$$\min\{f(x) = x : x \in \mathbb{R} \wedge x > 0\} \text{ and the sequence } \{x_i = 1/i\}$$

$$\min\{1/x : x \in \mathbb{R} \wedge x > 0\} \text{ and the sequence } \{x_i = i\}$$

In both cases, $\{f(x_i)\} \rightarrow 0$, but it is not an optimum.

We want conditions that ensure $\{f(x_i)\} \rightarrow f_* \Rightarrow \{x_i\} \rightarrow x_* \in S$ optimal solution.

Definition 3.0.2 (Interior and border of a set). *Given $S \subseteq \mathbb{R}^n$, we say that $\mathbf{x} \in \text{int}(S)$ (\mathbf{x} is an **interior point**) if it lies inside a ball contained in S . Formally, $\exists r > 0$ s.t. $\mathcal{B}(\mathbf{x}, r) \subseteq S$.*

*Moreover, we term **border points** those $\mathbf{x} \in \partial(S)$ such that all the points in the ball centered in x lie for a half inside the set S and for the other half outside. Formally, $\forall r > 0 \exists \mathbf{y}, \mathbf{z} \in \mathcal{B}(\mathbf{x}, r)$, where $\mathbf{y} \in S \wedge \mathbf{z} \notin S$. Intuitively, a point x lies on the border if, .*

Notice that a point on the boundary is not necessarily inside the ball, in that case we talk about **open set** (a set which is identical to its interior: $S = \text{int}(S)$).

Definition 3.0.3 (Closure of a set). *Let $S \subseteq \mathbb{R}^n$ we term **closure** of S the set $\text{cl}(S) = \text{int}(S) \cup \partial S$.*

Definition 3.0.4 (Closed set). We say that a set $S \subseteq \mathbb{R}^n$ is **closed** if it coincides with its closure: $S = \text{cl}(S)$.

Equivalently, a set S is termed **closed** if its complementary $(\mathbb{R}^n \setminus S)$ is open.

It is interesting to notice that all the functions that lead to minimizing sequences and do not converge to an optimum are all defined in open sets.

Notice that there are sets that are both open and closed, for example \mathbb{R}^n .

Definition 3.0.5 (Bounded set). Let $S \subseteq \mathbb{R}^n$. We say that S is **bounded** if $\exists r > 0$ such that $S \subseteq \mathcal{B}(0, r)$.

Intuitively, a bounded set does not go to ∞ .

Definition 3.0.6 (Compact set). Let $S \subseteq \mathbb{R}^n$. We term S **compact** if it is both closed and bounded.

Definition 3.0.7 (Accumulation point). Let $\{x_i\} \subseteq S$ be a sequence. We say that x is an **accumulation point** if $\exists \{x_{n_i}\}$ subsequence of $\{x_i\}$ converging to x . Formally, $\{x_{n_i}\} \rightarrow x$ or $\liminf_{i \rightarrow \infty} d(x_i, x) = 0$.

Theorem 3.0.1 (Bolzano-Weierstrass). Let $\{x_i\} \subseteq S$ be a bounded, real sequence. Then it has a converging subsequence.

Fact 3.0.2. Let S be a compact set and let $\{x_i\} \subseteq S$ be a minimizing sequence for the objective function f . The limit of the sequence is a feasible solution.

Proof. According to Bolzano-Weierstrass theorem $\{x_i\}$ has an accumulation point $x \in S$. Since it is minimizing $f(x)$ is feasible. \square

Why did we say *feasible* but not *optimal*? If the function is not continuous (cfr. Figure 3.1) it may happen that the sequence is minimizing, but the limit is not the optimum.

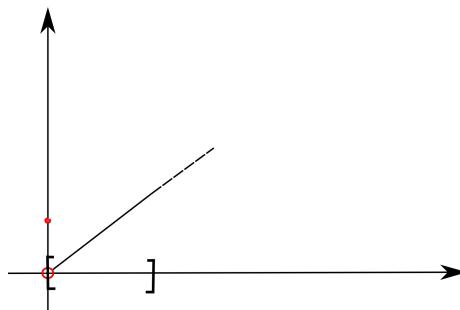


FIGURE 3.1: Case of non-continuity of the objective function in the border point $(0, 0)$.

Definition 3.0.8 (Domain). Let $f : D \rightarrow \mathbb{R}$. We term D **domain** of f and denote $D = \text{dom}(f)$.

In this course we will not take into account the domain of functions, because we can force all functions to be defined in the whole space as follows:

$$f : \mathbb{R}^n \rightarrow \bar{\mathbb{R}}, \text{ where } f(x) = \infty \text{ for } x \notin D.$$

Definition 3.0.9 (Graph and epigraph). Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$. We term **graph** $gr(f) = \{(f(\mathbf{x}), \mathbf{x}) : \mathbf{x} \in \text{dom}(f)\}$.

Conversely, we term **epigraph** $\text{epi}(f) = \{(v, \mathbf{x}) : \mathbf{x} \in \text{dom}(f) \wedge v \geq f(\mathbf{x})\}$, see ??.

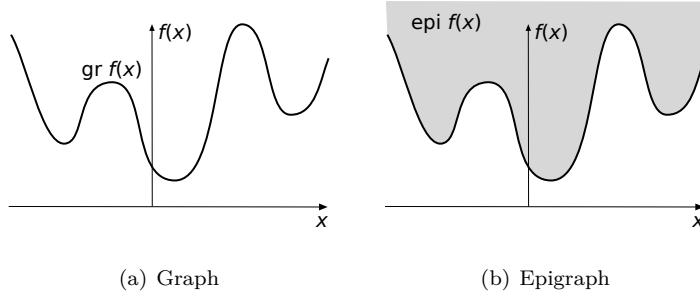


FIGURE 3.2: Graph and epigraph

When dealing with multidimensional inputs spaces it is crucial to have mathematical tools that make possible to somehow “see” the function’s behaviour in a lower-dimensional space.

Definition 3.0.10 (Level and sublevel set). Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$. We term **level set** the set of all the inputs that have the same output value. Formally, $L(f, v) = \{\mathbf{x} \in \text{dom}(f) : f(\mathbf{x}) = v\}$.

Conversely, we term **sublevel set** the set of all the inputs which image is smaller than a fixed value v : $S(f, v) = \{\mathbf{x} \in \text{dom}(f) : f(\mathbf{x}) \leq v\}$, see Figure 3.3.

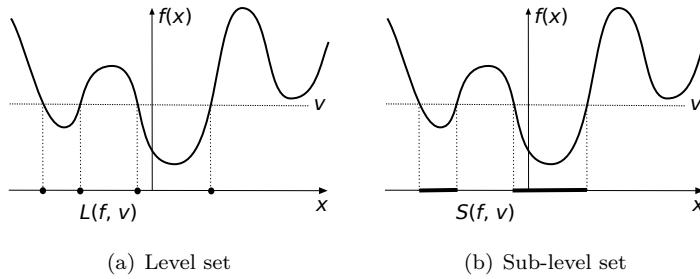


FIGURE 3.3: Level and sub-level sets

3.1 Continuity

Definition 3.1.1 (Continuity). Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$. We say that f is **continuous** in \mathbf{x} if $\forall \varepsilon > 0 \exists \delta > 0$ such that $\forall y \in \mathcal{B}(x, \delta) |f(y) - f(x)| < \varepsilon$.

Property 3.1.1. Let $f, g : \mathbb{R}^n \rightarrow \mathbb{R}$. The following holds:

1. $f + g, f \cdot g$ continuous at \mathbf{x}
2. $\max\{f, g\}$ and $\min\{f, g\}$ continuous at \mathbf{x}
3. $f \circ g \equiv f(g(\cdot))$ continuous at \mathbf{x}

Theorem 3.1.2 (Intermediate value). Let $f : \mathbb{R} \rightarrow \mathbb{R}$. f is continuous on $[a, b]$ if $\forall v \in \mathbb{R}$ s.t. $\min\{f(a), f(b)\} \leq v \leq \max\{f(a), f(b)\} \exists c \in [a, b] : f(c) = v$.

Theorem 3.1.3 (Weierstrass extreme value theorem). Let $S \subseteq \mathbb{R}^n$ be a compact set and let f be continuous on S . Then (P) has an optimal solution.

Equivalently, let $S \subseteq \mathbb{R}^n$ compact and let f continuous on S . Then all accumulation points of any minimizing sequence are optima and there is at least one.

Definition 3.1.2 (Lipshitz continuity). Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$. We term f **Lipshitz continuous** (L.c.) on $S \subseteq \mathbb{R}^n$ if $\exists L > 0$ such that

$$|f(\mathbf{x}) - f(\mathbf{y})| \leq L \|\mathbf{x} - \mathbf{y}\| \quad \forall \mathbf{x}, \mathbf{y} \in S$$

More generally, f is **globally Lipshitz continuous** if $S = \mathbb{R}^n$ and it is **locally Lipshitz continuous** at \mathbf{x} if $\exists \varepsilon > 0 \exists S = \mathcal{B}(\mathbf{x}, \varepsilon)$.

Notice that Lipshitz continuity offers a stronger form of continuity and that the L constant value depends on S . The wider the set the smaller L .

Fact 3.1.4. Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$. On a compact set $S \subseteq \mathbb{R}^n$ if f is continuous then it is Lipshitz continuous.

Definition 3.1.3 (Lower(upper) semi-continuity). Let $\{\mathbf{x}_i\} \subseteq \mathbb{R}^n$ be a sequence with accumulation point in \mathbf{x} and let $f : \mathbb{R}^n \rightarrow \mathbb{R}$. f is **lower (upper) semi-continuous** (l.(u.)s.c.) at \mathbf{x} if $f(\mathbf{x}) \leq \liminf_{i \rightarrow \infty} f(\mathbf{x}_i)$ ($f(\mathbf{x}) \geq \limsup_{i \rightarrow \infty} f(\mathbf{x}_i)$).

Equivalently, $\liminf_{\mathbf{y} \rightarrow \mathbf{x}} f(\mathbf{y}) \geq f(\mathbf{x})$, $\limsup_{\mathbf{y} \rightarrow \mathbf{x}} f(\mathbf{y}) \leq f(\mathbf{x})$.

3.2 Derivatives

In this section we address the problem of inferring information on a complicated function around a certain value \bar{x} using very simple functions, that are able to provide reliable information around that \bar{x} . Those functions are called “derivatives”.

Let us start with a brief recap on single-dimension differentiability.

Definition 3.2.1 (Left and right derivative). Let $f : \mathbb{R} \rightarrow \mathbb{R}$. We term **left derivative** $f'_-(x) = \lim_{t \rightarrow 0^-} [f(x+t) - f(x)]/t$.

Conversely, **right derivative** $f'_+(x) = \lim_{t \rightarrow 0^+} [f(x+t) - f(x)]/t$.

Definition 3.2.2 (Differentiable). Let $f : \mathbb{R} \rightarrow \mathbb{R}$. We say that f is **differentiable** at $x \in \text{dom}(f)$ if both left and right derivatives exist and coincide. Formally, $\exists f'_-, f'_+$ and $f'_-(x) = f'_+(x)$.

Fact 3.2.1. Let $f : \mathbb{R} \rightarrow \mathbb{R}$ and differentiable in $x \in \text{dom}(f)$ then f is continuous in x .

Theorem 3.2.2 (Mean value theorem). Let $f : \mathbb{R} \rightarrow \mathbb{R}$ continuous on $[a, b]$ and differentiable on (a, b) then $\exists c \in (a, b)$ s.t. $f'(c) = (f(b) - f(a))/(b - a)$.

Theorem 3.2.3 (Rolle's theorem). Let $f : \mathbb{R} \rightarrow \mathbb{R}$. If $f(a) = f(b)$ then $\exists c \in (a, b)$ s.t. $f'(c) = 0$

Corollary 3.2.4. In the same hypothesis of Rolle's theorem, let a and b consecutive roots of f . Then $f'(a)$ and $f'(b)$ have opposite sign.

3.2.1 Multivariate differentiability

Definition 3.2.3 (Partial derivative). Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$. We term **partial derivative** of f w.r.t. x_i at $\mathbf{x} \in \mathbb{R}^n$ as

$$\frac{\partial f}{\partial x_i}(\mathbf{x}) = \lim_{t \rightarrow 0} \frac{f(x_1, \dots, x_{i-1}, x_i + t, x_{i+1}, \dots, x_n) - f(\mathbf{x})}{t}$$

In other words, it is just $f'(x_1, \dots, x_{i-1}, x, x_{i+1}, \dots, x_n)$, considering each component x_j where $j \neq i$ as constant.

Definition 3.2.4 (Gradient). Given $f : \mathbb{R}^n \rightarrow \mathbb{R}$. We term **gradient** of f the vector of all the partial derivatives and we denote $\mathbb{R}^n \ni \nabla f = (\frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_n})^T$.

The gradient is the direction on which the function increases more rapidly, while the opposite of the gradient suggests the direction on which the function decreases most.

Definition 3.2.5 (Directional derivative). Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$. We term **directional derivative** of f in point $\mathbf{x} \in \text{dom}(f)$ along direction $\mathbf{d} \in \mathbb{R}^n$

$$\frac{\partial f}{\partial \mathbf{d}}(\mathbf{x}) := \lim_{t \rightarrow 0} \frac{f(\mathbf{x} + t\mathbf{d}) - f(\mathbf{x})}{t}$$

Notice that in a multivariate space it is not possible to assure differentiability checking if the directional derivatives are equal, because there is an infinite number of different directions.

Let us now introduce the notion of multivariate differentiability.

Definition 3.2.6 (Differentiable). Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$. We say that f is **differentiable** at $\mathbf{x} \in \mathbb{R}^n$ if \exists a linear function $\phi(h) = \langle c, h \rangle + f(\mathbf{x})$ s.t.

$$\lim_{\|h\| \rightarrow 0} \frac{\|f(\mathbf{x} + h) - \phi(h)\|}{\|h\|} = 0$$

The intuition here is that the linear function should approximate f pretty well around x .

Fact 3.2.5. Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ differentiable at \mathbf{x} . Then f is locally Lipschitz continuous at \mathbf{x} .

Corollary 3.2.6. Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ differentiable at \mathbf{x} . Then f is continuous at \mathbf{x} .

Notice that the converse does not hold.

Fact 3.2.7. Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$, let $\mathbf{x} \in \mathbb{R}^n$ and let us assume $\exists \delta > 0$ s.t. $\forall i \frac{\partial f}{\partial x_i}(\mathbf{y})$ is continuous $\forall \mathbf{y} \in \mathcal{B}(\mathbf{x}, \delta)$.

Then f differentiable at point \mathbf{x} .

Notice that the converse of Proposition 3.2.7 does not hold either.

We are now ready to introduce a class of functions that allows good results for optimization: \mathcal{C}^1 class.

Definition 3.2.7 (\mathcal{C}^1 class). We call **\mathcal{C}^1 -class** the class of functions with continuous gradient.

Fact 3.2.8. Let $f \in \mathcal{C}^1$, then f is differentiable everywhere and also continuous everywhere.

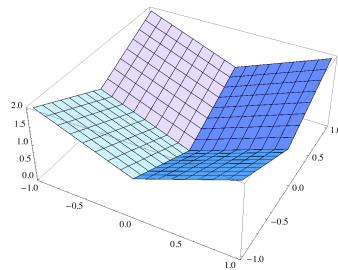
Definition 3.2.8 (First order model). Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and let $\mathbf{x} \in \text{dom}(f)$ We term **first order model** of f at point x

$$L_x(y) = \nabla f(\mathbf{x})(y - \mathbf{x}) + f(\mathbf{x})$$

Fact 3.2.9. Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and let $x \in \text{dom}(f)$ such that f is differentiable at \mathbf{x} . Then $(L_x, f(\mathbf{x})) \perp S(f, f(\mathbf{x})) \perp \nabla f(\mathbf{x})$.

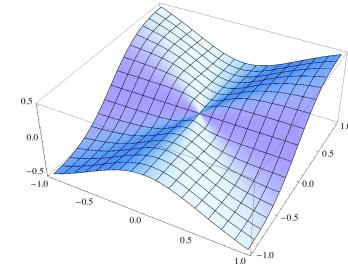
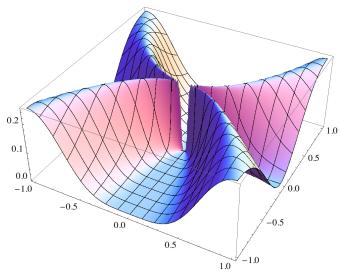
Geometrically speaking, we can observe that a function is smooth whenever its levelsets are smooth.

Example 3.2.1. Let us consider some functions that are not differentiable in the minimum.



The function $f(x_1, x_2) = \| [x_1, x_2] \| = |x_1| + |x_2|$ has some kinks.

The function $f(x_1, x_2) = \frac{x_1^2 x_2}{x_1^2 + x_2^2}$ may be put to 0 in $(0, 0)$ for continuity, but it is still not differentiable in $(0, 0)$ although the function admits directional derivatives in $(0, 0)$.



The function $f(x_1, x_2) = \left(\frac{x_1^2 x_2}{x_1^2 + x_2^2}\right)^2$ is not continuous in 0. There are some directions that lead to the limit 0, while there are some other directions (the parabolas above) that do not lead to value 0.

Lecture 4

3rd of October 2018

A. Frangioni

Example 4.0.1 (On derivatives). Let $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ such that $f(x, y) = x^2 e^y$. Compute the partial derivatives and the directional derivatives in the two directions $\mathbf{d}_1 = (0, 1)^T$ and $\mathbf{d}_2 = (1, 0)^T$.

- $\frac{\partial f}{\partial x} = 2xe^y$
- $\frac{\partial f}{\partial y} = x^2 e^y$
- $\frac{\partial f}{\partial \mathbf{d}_1} = \lim_{t \rightarrow 0} \frac{f(x+t \cdot 0, y+t \cdot 1)}{t} = \lim_{t \rightarrow 0} \frac{f(x, y+t)}{t}$. An attentive reader may notice that the directional derivative in direction \mathbf{d}_1 is equivalent to the derivative on the second component.
- $\frac{\partial f}{\partial \mathbf{d}_2} = \lim_{t \rightarrow 0} \frac{f(x+t \cdot 1, y+t \cdot 0)}{t} = \lim_{t \rightarrow 0} \frac{f(x+t, y)}{t}$. Conversely, with respect to what stated before, the directional derivative of f along the direction \mathbf{d}_2 is equivalent to the partial derivative w.r.t the first component.

Let us compute the scalar product between the gradient and the direction \mathbf{d}_1 :

$$\frac{\partial f}{\partial \mathbf{d}_1} = \left\langle \begin{pmatrix} 2xe^y \\ x^2 e^y \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\rangle = \begin{pmatrix} 2xe^y \cdot 0 \\ x^2 e^y \cdot 1 \end{pmatrix} = \begin{pmatrix} 0 \\ x^2 e^y \end{pmatrix}$$

The intuition of this example is formalized in the following

Fact 4.0.1. Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$. The directional derivative along a certain direction $\mathbf{d} \in \mathbb{R}^n$ can be computed as the scalar product between the gradient of the function and the direction, formally

$$\frac{\partial f}{\partial \mathbf{d}} = \langle \nabla f, \mathbf{d} \rangle$$

Definition 4.0.1 (Vector-valued function). A function which codomain is multi-dimensional is called **vector-valued function**. Formally, $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$, where

$$f(\mathbf{x}) = \begin{pmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \vdots \\ f_m(\mathbf{x}) \end{pmatrix} \in \mathbb{R}^m.$$

For such functions the computation of the derivative requires to specify not only the component with respect to the one the derivation should be performed, but also the index of the function.

Definition 4.0.2 (Partial derivative for vector-valued functions). Let $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$, the partial derivative of the j -th function with respect to the i -th component is

$$\frac{\partial f_j}{\partial x_i}(\mathbf{x}) = \lim_{t \rightarrow 0} \frac{f_j(x_1, x_2, \dots, x_{i-1}, \dots, x_i + t, x_{i+1}, \dots, x_n) - f_j(\mathbf{x})}{t}$$

where $t \in \mathbb{R}$.

Definition 4.0.3 (Jacobian). Let $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ be a vector-valued function. We call **Jacobian** the matrix of all its first-order partial derivatives.

$$Jf(\mathbf{x}) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \dots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \dots & \frac{\partial f_m}{\partial x_n} \end{pmatrix} = \begin{pmatrix} \nabla f_1(\mathbf{x}) \\ \nabla f_2(\mathbf{x}) \\ \vdots \\ \nabla f_m(\mathbf{x}) \end{pmatrix}$$

Example 4.0.2. Let $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ such that $f(x, y) = x^2 e^y$ as before, where the gradient was

$$\nabla f(\mathbf{x}) = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix} = \begin{pmatrix} 2xe^y \\ x^2 e^y \end{pmatrix}$$

Let us compute the second derivative of this function:

$$\nabla^2 f(\mathbf{x}) = \begin{pmatrix} \frac{\partial^2 f}{\partial x \partial x} & \frac{\partial^2 f}{\partial y \partial x} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y \partial y} \end{pmatrix} = \begin{pmatrix} 2e^y & 2xe^y \\ 2xe^y & x^2 e^y \end{pmatrix}$$

The second order derivative of a vector function is a matrix, defined as

Definition 4.0.4 (Hessian). Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$, we call **Hessian** of f

$$\nabla^2 f(\mathbf{x}) := J \nabla f(\mathbf{x}) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2}(\mathbf{x}) & \frac{\partial^2 f}{\partial x_2 \partial x_1}(\mathbf{x}) & \dots & \frac{\partial^2 f}{\partial x_n \partial x_1}(\mathbf{x}) \\ \frac{\partial^2 f}{\partial x_1 \partial x_2}(\mathbf{x}) & \frac{\partial^2 f}{\partial x_2^2}(\mathbf{x}) & \dots & \frac{\partial^2 f}{\partial x_n \partial x_2}(\mathbf{x}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_n}(\mathbf{x}) & \frac{\partial^2 f}{\partial x_2 \partial x_n}(\mathbf{x}) & \dots & \frac{\partial^2 f}{\partial x_n^2}(\mathbf{x}) \end{pmatrix}$$

The size of the matrix grows very rapidly with the number of derivatives that we make.

In optimization, handling Hessians is a crucial task, because such matrices are very large hence unfeasible most of the times.

In most cases, the Hessian is symmetric matrix, as stated in the following

Theorem 4.0.2 (Schwartz's theorem). *Let $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ such that $\nabla^2 f(\mathbf{x})$ exists and it is continuous and let us take $\mathbf{x} \in \mathbb{R}^n$. $\exists \delta > 0$ s.t. $\forall \mathbf{x}' \in \mathcal{B}(\mathbf{x}, \delta)$ and $\forall i, j \in \{1, 2, \dots, n\}$ $\frac{\partial^2 f}{\partial x_j \partial x_i}(\mathbf{x}')$ and $\frac{\partial^2 f}{\partial x_i \partial x_j}(\mathbf{x}')$ exist and they are equal:*

$$\frac{\partial^2 f}{\partial x_j \partial x_i}(\mathbf{x}') = \frac{\partial^2 f}{\partial x_i \partial x_j}(\mathbf{x}')$$

Definition 4.0.5 (\mathcal{C}^2 functions). *Let $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$. We say that f belongs to \mathcal{C}^2 class iff $\nabla^2 f(x)$ is continuous.*

Corollary 4.0.3. *Let $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ such that $f \in \mathcal{C}^2$, then the Hessian is symmetric and the gradient is continuous.*

We are now interested in providing some approximations to a generic function f . Such approximations are useful tools to have a hint on where the function moves along a decreasing direction, but the information is accurate only locally.

Definition 4.0.6 (First order Taylor model). *Let $f : \mathbb{R}^n \rightarrow \mathbb{R} \in \mathcal{C}^1$, we define the **first-order Taylor's formula** as a linear approximation of the function f in a neighbourhood of \mathbf{x} .*

Formally, for a given ball $\mathcal{B}(\mathbf{x}, \delta)$, $\forall \mathbf{x}' \in \mathcal{B}(\mathbf{x}, \delta) \exists \alpha \in (0, 1)$ s.t.

$$f(\mathbf{x}') \approx L_{\mathbf{x}}(\mathbf{x}') = \langle \nabla f(\alpha \mathbf{x} + (1 - \alpha)\mathbf{x}'), \mathbf{x}' - \mathbf{x} \rangle + f(\mathbf{x})$$

*Equivalently, we can write the so-called **remainder version of first-order Taylor formula** as*

$$f(\mathbf{x}') \approx L_{\mathbf{x}}(\mathbf{x}') = \langle \nabla f(\mathbf{x}), \mathbf{x}' - \mathbf{x} \rangle + f(\mathbf{x}) + R_2(\mathbf{x}') \quad (4.0.1)$$

*where R_2 is called **quadratic remainder** and is such that $\lim_{h \rightarrow 0} \frac{R(h)}{\|h\|} = 0$.*

Notice the general Lagrangian form of the remainder:

$$R_k(\mathbf{x}') = \frac{f^{(k+1)}(\beta)}{(k+1)!} \cdot (\mathbf{x}' - \mathbf{x})^2$$

where $\beta \in [\mathbf{x}, \mathbf{x}']$ and f is $k+1$ times differentiable.

Notice that the Taylor's approximation works *only locally*: the furthest we get from \mathbf{x} the more distant the function and the model are.

Definition 4.0.7 (Second order Taylor model). *Let $f : \mathbb{R}^n \rightarrow \mathbb{R} \in \mathcal{C}^1$, we define the **second-order Taylor's formula** as a quadratic approximation of the function f in a neighbourhood of \mathbf{x} . Formally, given a ball $\mathcal{B}(\mathbf{x}, \delta)$, $\forall \mathbf{x}' \in \mathcal{B}(\mathbf{x}, \delta)$*

$$f(\mathbf{x}') \approx Q_{\mathbf{x}}(\mathbf{x}') = L_{\mathbf{x}}(\mathbf{x}') + \frac{1}{2}(\mathbf{x}' - \mathbf{x})^T \nabla^2 f(\mathbf{x})(\mathbf{x}' - \mathbf{x}) + R_3(\mathbf{x}' - \mathbf{x}) \quad (4.0.2)$$

with $\lim_{h \rightarrow 0} R(h) \|h\|^2 = 0$ or, equivalently, the remainder vanishes at least cubically, or the error is $O(\|\mathbf{x}' - \mathbf{x}\|^3)$.

Example 4.0.3. Let $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ such that $f(\mathbf{x}) = f\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = x_1^2 e^{x_2}$, as above and $\nabla f(\mathbf{x}) = \begin{pmatrix} 2x_1 e^{x_2} \\ x_1^2 e^{x_2} \end{pmatrix}$ and $\nabla^2 f(\mathbf{x}) = \begin{pmatrix} 2e^{x_2} & 2x_1 e^{x_2} \\ 2x_1 e^{x_2} & x_1^2 e^{x_2} \end{pmatrix}$. In $\mathbf{x} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \in \mathbb{R}^2$ we have $f(\mathbf{x}) = 1$ and $\nabla f(\mathbf{x}) = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$.

Compute the first and second order Taylor models for f with $\alpha = 1$.

The linear model has the following shape

$$L_{(1,0)}(x_1, x_2) = \langle \begin{pmatrix} 2 \\ 1 \end{pmatrix}, \begin{pmatrix} x_1 - 1 \\ x_2 - 0 \end{pmatrix} \rangle + 1 = 2x_1 - 2 + x_2 + 1 = 2x_1 + x_2 - 1$$

And the quadratic model is

$$\begin{aligned} f(\mathbf{x}') &= \langle \nabla f\left(\alpha \cdot \begin{pmatrix} 2 \\ 1 \end{pmatrix} + (1-\alpha)\mathbf{x}'\right), \mathbf{x}' - \begin{pmatrix} 2 \\ 1 \end{pmatrix} \rangle + f\left(\begin{pmatrix} 2 \\ 1 \end{pmatrix}\right) + \frac{1}{2} \cdot \nabla^2 f\left(\begin{pmatrix} 2 \\ 1 \end{pmatrix}\right) \cdot \left(\mathbf{x}' - \begin{pmatrix} 2 \\ 1 \end{pmatrix}\right) \\ Q_{[1,0]}(x_1, x_2) &= 2x_1 + x_2 - 1 + \frac{1}{2} \cdot (x_1 - 1, x_2 - 0) \cdot \begin{pmatrix} 2 & 2 \\ 2 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 - 1 \\ x_2 - 0 \end{pmatrix} \\ &= 2x_1 + x_2 - 1 + \frac{1}{2} \cdot (2x_1 - 2 + 2x_2, 2x_1 - 2 + x_2) \cdot \begin{pmatrix} x_1 - 1 \\ x_2 - 0 \end{pmatrix} \\ &= 2x_1 + x_2 - 1 + \frac{1}{2} \cdot ((2x_1 - 2 + 2x_2) \cdot (x_1 - 1) + (2x_1 - 2 + x_2) \cdot x_2) \\ &= 2x_1 + x_2 - 1 + \frac{1}{2} \cdot (2x_1^2 - 2x_1 + 2x_1x_2 - 2x_1 + 2 - 2x_2 + 2x_1x_2 - 2x_2 + x_2^2) \\ &= 2x_1 + x_2 - 1 + \frac{1}{2} \cdot (2x_1^2 - 4x_1 + 4x_1x_2 + 2 - 4x_2 + x_2^2) \\ &= 2x_1 + x_2 - 1 + x_1^2 - 2x_1 + 2x_1x_2 + 1 - 2x_2 + \frac{1}{2}x_2^2 \\ &= x_1^2 + 2x_1x_2 - x_2 + \frac{1}{2}x_2^2 \end{aligned} \tag{4.0.3}$$

The Taylor model can be extended to the more general k -th order. Such an expansion requires the computation of the k -th order derivatives, but $\nabla^k f(\mathbf{x})$ is a tensor of order $k \equiv n^k$ numbers. For $k > 2$ this approach is practically unfeasible.

As it happens often in computer science, it is important to find a good trade-off between the complexity of the model and the accuracy of the approximation: if the model is too simple (but efficient) the approximation is not very good; conversely, complex and accurate approximations are computationally heavy.

Fact 4.0.4. Let $f : \mathbb{R}^n \rightarrow \mathbb{R} \in \mathcal{C}^1$. If f is Lipschitz continuous on $S \in \mathbb{R}^n$, then in such set the norm of the gradient is bounded by the Lipschitz constant.

Formally, $\sup\{\|\nabla f(\mathbf{x})\| : \mathbf{x} \in S\} \leq L$. If S convex $\sup\{\|\nabla f(\mathbf{x})\| : \mathbf{x} \in S\} = L$.

Moreover, we can prove

Fact 4.0.5. Let $f : \mathbb{R}^n \rightarrow \mathbb{R} \in \mathcal{C}^1$. f is Lipschitz continuous on $S \in \mathbb{R}^n$ iff its Hessian is bounded on S . Formally, $\sup\{\|\nabla^2 f(\mathbf{x})\| : \mathbf{x} \in S\} \leq L$.

Fact 4.0.6. Let $f : \mathbb{R}^n \rightarrow \mathbb{R} \in \mathcal{C}^1$. If the gradient of f is Lipschitz continuous in all points $\mathbf{x}' \in \mathbb{R}^n$ that are close to $\mathbf{x} \in \mathbb{R}^n$, then $f(\mathbf{x}') \leq L_x(\mathbf{x}') + \frac{L}{2} \|\mathbf{x}' - \mathbf{x}\|^2$.

4.1 Simple functions

In the rest of the course we will deal mostly with some linear and quadratic approximations of the objective function. In this section, we introduce a couple of examples and considerations on such functions.

4.1.1 Linear functions

In this scenario, $f : \mathbb{R}^n \rightarrow \mathbb{R}$ has the following shape: $f(\mathbf{x}) = \mathbf{c}^T \mathbf{x}$, for a fixed $\mathbf{c} \in \mathbb{R}^n$.

It holds that $\nabla f(\mathbf{x}) = \mathbf{c}$, $\nabla^2 f(\mathbf{x}) = 0$ and that level sets are parallel hyperplanes orthogonal to \mathbf{c} .

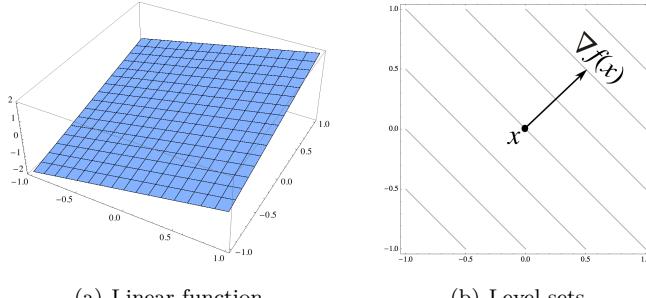


FIGURE 4.1: Graphical example of linear function.

4.1.2 Quadratic functions

A quadratic function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is formalized as $f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T Q \mathbf{x} + \mathbf{q}^T \mathbf{x}$, where $Q \in \mathcal{M}(n, \mathbb{R})$, $\mathbf{q}^T \in \mathbb{R}^n$.

In Figure 4.2 we can see the plot of the quadratic function where

$$Q = \begin{bmatrix} 6 & -2 \\ -2 & 6 \end{bmatrix}, \mathbf{q} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

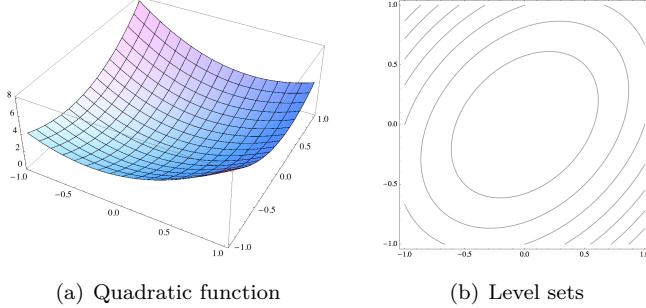


FIGURE 4.2: Graphical example of a quadratic function.

In quadratic functions the gradient is a linear function on \mathbf{x} $\nabla f(\mathbf{x}) = Q\mathbf{x} + \mathbf{q}$, while the Hessian is just Q and the level sets are ellipsoids. Sometimes such ellipses can degenerate to lines, in the case that one of the axis has become $+\infty$.

Fact 4.1.1. *Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be a quadratic function with parameters $Q \in \mathcal{M}(n, \mathbb{R})$ and $\mathbf{q}^T \in \mathbb{R}^n$. If Q is symmetric, then it has spectral decomposition.*

$$\text{Proof. } \mathbf{x}^T Q \mathbf{x} = \frac{[(\mathbf{x}^T Q \mathbf{x}) + (\mathbf{x}^T Q \mathbf{x})^T]}{2} = \mathbf{x}^T \left[\frac{(Q+Q^T)}{2} \right] \mathbf{x} = \mathbf{x}^T H \Lambda H^T$$

In the rest of the course we will assume for simplicity that Q is symmetric. Another “lucky case” is when Q is non singular (i.e. all its eigenvalues are not 0): in this case, $\bar{\mathbf{x}} = -Q^{-1}\mathbf{q}$, where $\bar{\mathbf{x}}$ is called the center of the ellipsoid. Let us assume that we moved the origin in such $\bar{\mathbf{x}}$, so $\mathbf{x}' = \mathbf{x} - \bar{\mathbf{x}}$ and $f_{\bar{\mathbf{x}}}(\mathbf{x}') = \frac{1}{2}\mathbf{x}'^T Q \mathbf{x}'$.

Fact 4.1.2. *Along H_i : $f(\alpha) = f_{\bar{\mathbf{x}}}(\alpha H_i) = \alpha^2 \lambda_i$*

Moreover, the size of the axes of the level curves is proportional to $\sqrt{1/\lambda_i}$, where λ_i are the eigenvalues. If the eigenvalues are close to 0, then the axes get very long, while when they are negative, we no longer have axes.

This is formalized in

Fact 4.1.3. *Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ such that $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q \mathbf{x} + \mathbf{q}^T \mathbf{x}$. If Q is positive definite $\bar{\mathbf{x}} = -Q^{-1}\mathbf{q}$ is the minimum of f , while if Q is indefinite (i.e. $\exists \lambda_i < 0$) f is unbounded below.*

Lecture 5

5th of October 2018

A. Frangioni

5.1 Unconstrained optimization

Until now we stated that the best conditions for finding the optimum are encountered when the domain is a compact set and we have many derivatives.

Now we need to consider when we can stop our algorithm.

Definition 5.1.1 (Unconstrained optimization problem). *We term **unconstrained optimization problem** the following*

$$(P) f_* = \min\{f(\mathbf{x}) : \mathbf{x} \in \mathbb{R}^n\}$$

In unconstrained optimization, we deal with an unbounded set (\mathbb{R}^n), hence Weierstrass theorem does not apply. Because of this reason, we have no guarantee that a minimum \mathbf{x}_* exists; moreover, provided its existence, finding it is an NP-hard problem. In order to make things easier, in practice we use a weaker condition: **local minimality**.

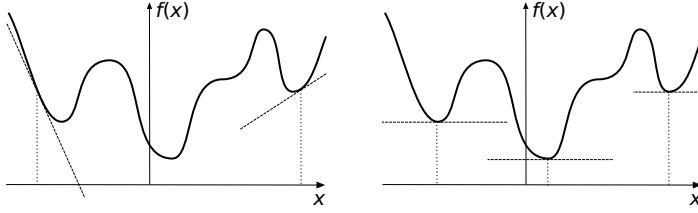
Definition 5.1.2 (Local minimum). *Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$. \mathbf{x}_* is a **local minimum** if it is a global minimum in a ball around \mathbf{x}^* . Formally,*

$$\min\{f(\mathbf{x}) : \mathbf{x} \in \mathcal{B}(\mathbf{x}_*, \varepsilon)\}$$

for some $\varepsilon > 0$.

Also, \mathbf{x}^* is a **strict local minimum** if $f(\mathbf{x}) < f(\mathbf{x}') \forall \mathbf{x}' \in \mathcal{B}(\mathbf{x}_*, \varepsilon)$.

If $f'(\mathbf{x}) < 0$ or $f'(\mathbf{x}) > 0$, \mathbf{x} clearly cannot be a local minimum, as shown in Figure 5.1.



(a) Points where the derivative is not 0.

(b) Points where the derivative is 0.

FIGURE 5.1: Minima and not minima.

5.1.1 First order model

💡 Do you recall?

The first order model of f is $L_{\mathbf{x}}(\mathbf{x}') = f(\mathbf{x}) + \nabla f(\mathbf{x})(\mathbf{x}' - \mathbf{x})$, such that $\forall \mathbf{x}' \in \mathbb{R}^n$ close to \mathbf{x} $f(\mathbf{x}') = L_{\mathbf{x}}(\mathbf{x}') + R(\mathbf{x}' - \mathbf{x})$, where $R(\cdot)$ is called **residual** and it has the property of quadratic convergence: $\lim_{\|h\| \rightarrow 0} \frac{R(h)}{\|h\|} = 0$.

Fact 5.1.1. Let f be differentiable, if \mathbf{x} is a local minimum, then $\nabla f(\mathbf{x}) = 0$.

In optimization, we are interested in moving towards a (local) minimum \mathbf{x}_* as fast as possible.

Fact 5.1.2. Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be the objective function of the optimization problem (P) and let $\mathbf{x} \in \mathbb{R}^n$ be the current point at a generic iteration. In order to get closer to the optimum, we need to take a step along the anti-gradient direction. Formally, $\mathbf{x}(\alpha) = \mathbf{x} - \alpha \nabla f(\mathbf{x})$, where $\alpha \in \mathbb{R}$ is called step size.

Proof by contradiction. Let us assume that \mathbf{x} is a local minimum but $\nabla f(\mathbf{x}) \neq 0$.

In our case, $\mathbf{x}' = \mathbf{x} - \alpha \nabla f(\mathbf{x})$, so let us plug it into the *remainder version* of the Tailor's first-order model:

$$\begin{aligned} f(\mathbf{x}') &= \langle \nabla f(\mathbf{x}), \mathbf{x}' - \mathbf{x} \rangle + f(\mathbf{x}) + R(\mathbf{x}' - \mathbf{x}) \\ &= \langle \nabla f(\mathbf{x}), \mathbf{x} - \alpha \nabla f(\mathbf{x}) - \mathbf{x} \rangle + f(\mathbf{x}) + R(\mathbf{x} - \alpha \nabla f(\mathbf{x}) - \mathbf{x}) \\ &= \langle \nabla f(\mathbf{x}), -\alpha \nabla f(\mathbf{x}) \rangle + f(\mathbf{x}) + R(-\alpha \nabla f(\mathbf{x})) \\ &= f(\mathbf{x}) - \alpha \|\nabla f(\mathbf{x})\|^2 + R(-\alpha \nabla f(\mathbf{x})) \end{aligned}$$

Once we fixed the moving direction, we can choose the step size α , so it can be proved that $\lim_{\alpha \rightarrow 0} \frac{R(-\alpha \nabla f(\mathbf{x}))}{\|\alpha \nabla f(\mathbf{x})\|} = 0$, that is equivalent by definition to $\forall \varepsilon > 0 \exists \bar{\alpha} > 0$ s.t. $\frac{R(-\alpha \nabla f(\mathbf{x}))}{\alpha \|\nabla f(\mathbf{x})\|} \leq \varepsilon \forall 0 \leq \alpha < \bar{\alpha}$.

If we take $\varepsilon < \|\nabla f(\mathbf{x})\|$, we get $R(-\alpha \nabla f(\mathbf{x})) < \alpha \|\nabla f(\mathbf{x})\|^2$, then

$$f(\mathbf{x}(\alpha)) = f(\mathbf{x}) - \alpha \|\nabla f(\mathbf{x})\|^2 + R(-\alpha \nabla f(\mathbf{x})) < f(\mathbf{x})$$

$\forall \alpha < \bar{\alpha}$ \mathbf{x} cannot be a local minimum. \square

Proposition 5.1.2 states that the first order model allows to find the decreasing direction, but if the gradient is 0 we do not know if we are in presence of a minimum, maximum or a saddle point. To discriminate among those, we exploit the information provided by the second derivative.

5.1.2 Second order model

Fact 5.1.3. *Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ such that $f \in C^2$. If \mathbf{x} is a local minimum then the gradient is positive semidefinite. Formally, $\nabla^2 f(\mathbf{x}) \succeq 0$.*

Proof by contradiction. Our contradictory hypothesis is that we are in a local minimum, but the Hessian is not positive semidefinite, there there is a direction $\mathbf{d} \in \mathbb{R}^n$ of negative curvature. Formally, $\exists \mathbf{d}$ s.t. $\mathbf{d}^T \nabla^2 f(\mathbf{x}) \mathbf{d} < 0$ or equivalently, the Hessian has a negative eigenvalue $\lambda_i < 0$.

Let us consider that \mathbf{d} has norm 1 without loss of generality and let us be given a step $\mathbf{x}(\alpha) = \mathbf{x} + \alpha \mathbf{d}$ and then take the second-order Taylor formula as follows (where there is no linear term involved because $\nabla f(\mathbf{x}) = 0$):

$$\begin{aligned} f(\mathbf{x}(\alpha)) &= f(\mathbf{x}) + \frac{1}{2} (\alpha \mathbf{d})^T \nabla^2 f(\mathbf{x}) (\alpha \mathbf{d}) + R(\mathbf{x}(\alpha) - \mathbf{x}) \\ f(\mathbf{x}(\alpha)) &= f(\mathbf{x}) + \frac{1}{2} \alpha^2 \mathbf{d}^T \nabla^2 f(\mathbf{x}) \mathbf{d} + R(\alpha \mathbf{d}) \end{aligned}$$

with $\lim_{\|h\| \rightarrow 0} \frac{R(h)}{\|h\|^2} = \lim_{\alpha \rightarrow 0} \frac{R(\alpha \mathbf{d})}{\alpha^2} = 0$ or equivalently $\forall \varepsilon > 0 \exists \bar{\alpha} > 0$ s.t. $R(\alpha \mathbf{d}) \leq \varepsilon \alpha^2 \forall 0 \leq \alpha < \bar{\alpha}$.

At this point, since this condition holds for each ε we are allowed to take the most convenient: $\varepsilon < -\frac{1}{2} \mathbf{d}^T \nabla^2 f(\mathbf{x}) \mathbf{d}$, so that we obtain this condition on the residual $R(\alpha \mathbf{d}) < -\frac{1}{2} \alpha^2 \mathbf{d}^T \nabla^2 f(\mathbf{x}) \mathbf{d}$, hence

$$f(\mathbf{x}(\alpha)) = f(\mathbf{x}) + \frac{1}{2} \alpha^2 \mathbf{d}^T \nabla^2 f(\mathbf{x}) \mathbf{d} + R(\alpha \mathbf{d}) < f(\mathbf{x}) \forall 0 \leq \alpha < \bar{\alpha}$$

The contradiction lies in the fact that $\mathbf{x}(\alpha)$ leads to a value of f smaller than the one in \mathbf{x} . \square

Proposition 5.1.3 states that a positive semidefinite Hessian is a necessary condition for a local minimum. Follows a proposition that gives a sufficient condition.

Fact 5.1.4. *Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ s.t. $f \in C^2$ and let the Hessian be symmetric (hence real eigenvalues). If $\nabla f(\mathbf{x}) = 0$ and the Hessian is strictly positive definite ($\nabla^2 f(\mathbf{x}) \succ 0$) then \mathbf{x} is a local minimum.*

Proof. $\mathbf{x} \in \mathbb{R}^n$ is a point where the gradient of the function is 0 we get the following second order Taylor approximation for $\mathbf{x}_{\text{new}} = \mathbf{x} + \mathbf{d}$ (unitary step size for simplicity):

$$f(\mathbf{x} + \mathbf{d}) = f(\mathbf{x}) + \frac{1}{2} \mathbf{d}^T \nabla^2 f(\mathbf{x}) \mathbf{d} + R(\mathbf{d}) \text{ with } \lim_{h \rightarrow 0} \frac{R(\mathbf{d})}{\|\mathbf{d}\|^2} = 0$$

Hence, by definition of limit $\forall \varepsilon > 0 \exists \delta > 0$ s.t. $R(\mathbf{d}) \leq \varepsilon \|\mathbf{d}\|^2 \forall \mathbf{d} \in \mathbb{R}^n$ s.t. $\|\mathbf{d}\| < \delta$.

Since the Hessian is strictly positive definite, the smallest eigenvalue λ_{\min} is strictly greater than 0, hence the variational characterization of eigenvalues $\mathbf{d}^T \nabla^2 f(\mathbf{x}) \mathbf{d} \geq \lambda_{\min} \|\mathbf{d}\|^2$.

Aggiungere la ref alla variational characterization nella parte di Numerical Methods

We are now ready to pick the ε we prefer ($\varepsilon < \lambda_{\min}$) to get $\forall \mathbf{d}$ s.t. $\|\mathbf{d}\| < \delta$

$$f(\mathbf{x} + \mathbf{d}) = f(\mathbf{x}) + \frac{1}{2} \mathbf{d}^T \nabla^2 f(\mathbf{x}) \mathbf{d} + R(\mathbf{d}) \geq f(\mathbf{x}) + (\lambda_{\min} - \varepsilon) \|\mathbf{d}\|^2 > f(\mathbf{x})$$

The term $\lambda_{\min} - \varepsilon$ is strictly positive. □

In the rest of this lecture, we will provide conditions that ensure that once a local minimum is found, it is also a global minimum.

So far, we said that the local minima are those points where the gradient is 0 and the Hessian is positive semidefinite. An easy way to ensure that the Hessian is positive semidefinite in a ball around \mathbf{x} is to have that the Hessian is positive semidefinite everywhere ($\forall \mathbf{x} \in \mathbb{R}^n$) and this is guaranteed to be true whenever f is a convex function.

5.2 Convexity

Let us introduce convexity as far as both sets and functions are concerned.

5.2.1 Convex sets

Definition 5.2.1 (Convex hull). *Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ we term **convex hull** and denote $\text{conv}(\mathbf{x}, \mathbf{y}) = \{\mathbf{z} = \alpha \mathbf{x} + (1 - \alpha) \mathbf{y} : \alpha \in [0, 1]\}$ the segment joining \mathbf{x} and \mathbf{y} .*

Definition 5.2.2 (Convex set). *Intuitively, we say that C is a **convex set** if for each couple in the set, the line joining such points belongs to the set. Formally, $C \subset \mathbb{R}^n$ is a **convex set** if $\forall \mathbf{x}, \mathbf{y} \in C \text{ } \text{conv}(\mathbf{x}, \mathbf{y}) \subseteq C$.*

Notice that “disconnected sets” cannot be convex sets.

Definition 5.2.3 (Convex hull of a set). *Given a set $S \subseteq \mathbb{R}^n$, we can “complete” it to a convex set $\text{conv}(S)$, called **convex hull of S** :*

$$\begin{aligned}\text{conv}(S) &= \bigcup\{\text{conv}(\mathbf{x}, \mathbf{y}) : \mathbf{x}, \mathbf{y} \in S\} \\ &= \bigcap\{C : C \text{ is convex } \wedge C \supseteq S\}\end{aligned}$$

Equivalently, the convex hull of S can be defined iteratively as the convex hull of all the couples of points in S .

Equivalently, we can define the convex hull of a set S as the smallest convex set containing S .

Note

A more general definition of a convex hull is the following:

$$\text{conv}(\{\mathbf{x}_1, \dots, \mathbf{x}_k\}) = \left\{ \mathbf{x} = \sum_{i=1}^k \alpha_i \mathbf{x}_i : \sum_{i=1}^k \alpha_i = 1, \alpha_i \geq 0 \forall i \right\}$$

Definition 5.2.4 (Unitary simplex). *We term **unitary simplex** the set of k non-negative scalars summing to 1, formally*

$$\Theta^k = \left\{ \sum_{i=1}^k \alpha_i \mathbf{x}_i \in \mathbb{R}^k : \sum_{i=1}^k \alpha_i = 1, \alpha_i \geq 0 \forall i \right\}$$

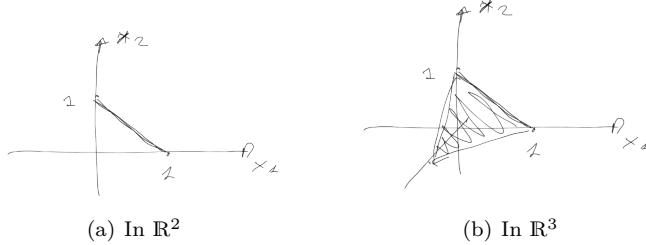


FIGURE 5.2: Example of unitary simplexes.

Our goal is to find the simplest possible convex set that approximates our set.

Fact 5.2.1. *A convex set is equal to its convex hull. Formally,*

$$C \text{ is convex} \iff C = \text{conv}(C)$$

We are interested in sufficient conditions for convexity.

Definition 5.2.5 (Cone). *We term **cone** the set $\mathcal{C} = \{\mathbf{x} \in \mathbb{R}^n : \alpha \mathbf{x} \in \mathcal{C} \forall \alpha \geq 0\}$.*

An attentive reader may notice that the definition of cone is a relaxation of the unitary simplex, where we do not require the unitary sum.

Example 5.2.1. *The following sets are convex:*

- Convex polytope $\text{conv}(\{\mathbf{x}_1, \dots, \mathbf{x}_k\})$, unitary simplex Θ
- Affine hyperplane: $\mathcal{H} := \{\mathbf{x} \in \mathbb{R}^n : \mathbf{vect}^T \mathbf{x} = b\}$
- Affine close (open) subspace: $\mathcal{S} := \{\mathbf{x} \in \mathbb{R}^n : \mathbf{a}^T \mathbf{vect} \mathbf{x} \leq (<)b\}$
- Close (open) ball of radius r in p -norm, $p \geq 1$: $\mathcal{B}_p(\mathbf{x}, r) = \{\mathbf{y} \in \mathbb{R}^n : \|\mathbf{y} - \mathbf{x}\|_p \leq (<)r\}$
- Close (open) ellipsoid: $\mathcal{E}(Q, \mathbf{x}, r) := \{y \in \mathbb{R}^n : (y - \mathbf{x})^T Q (y - \mathbf{x}) \leq (<)r\}$ with $Q \succeq 0$. Notice that ellipsoids are levelsets of quadratic functions.
- Cones
- Conical hull of a finite set of directions:

$$\text{cone}(\{\mathbf{d}_1, \dots, \mathbf{d}_k\}) = \left\{ \mathbf{d} = \sum_{i=1}^k \mu_i \mathbf{d}_i : \mu_i \geq 0 \forall i \right\}$$

- Lorentz's cone (ice-cream cone): $\mathbb{L} = \left\{ \mathbf{x} \in \mathbb{R}^n : \mathbf{x}_n \geq \sqrt{\sum_{i=1}^{n-1} \mathbf{x}_i^2} \right\}$
- Cone of positive semidefinite matrices: $\mathbb{S}_+ = \{A \in (M)(n, \mathbb{R}^n) : A \succeq 0\}$

Fact 5.2.2. *The following operations preserve convexity.*

1. Given a possibly infinite family of convex sets $(\{C_i\}_{i \in I})$, the intersection $(\bigcap_{i \in I} C_i)$ is convex;
2. If we have convex sets in different subspaces, their Cartesian product is a convex set (C_1, \dots, C_k convex $\iff C_1 \times \dots \times C_k$ convex);
3. Given a convex set, its image under a linear mapping (i.e. scaling, translation, rotation) is a convex set. Formally, let C be a convex set. Then $A(C) := \{\mathbf{x} = A\mathbf{y} + \mathbf{b} : \mathbf{y} \in C\}$ for a given $A \in \mathcal{M}(n, \mathbb{R})$ and $\mathbf{b} \in \mathbb{R}^n$ is convex;
4. If C is a convex set its inverse image under a linear mapping is convex as well. Formally, $A^{-1}(C) := \{x : Ax + b \in C\}$ is convex;
5. Let C_1 and C_2 convex and let $\alpha_1, \alpha_2 \in \mathbb{R}$. Then $\alpha_1 C_1 + \alpha_2 C_2 := \{\mathbf{x} = \alpha_1 \mathbf{x}_1 + \alpha_2 \mathbf{x}_2 : \mathbf{x}_1 \in C_1, \mathbf{x}_2 \in C_2\}$ is convex;
6. Let $C \subseteq \mathbb{R}^{n_1} \times \mathbb{R}^{n_2}$ be a convex set. Slicing and projection lead to convex sets as well, as shown in Figure 5.3:

SLICE: $C_s(\mathbf{y}) := \{\mathbf{x} \in \mathbb{R}^{n_1} : (\mathbf{x}, \mathbf{y}) \in C\}$ is convex

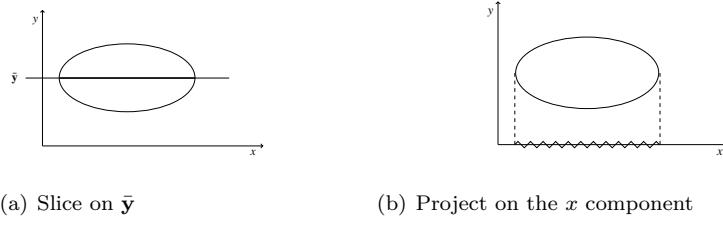


FIGURE 5.3: Pictorial examples of slicing and projecting.

PROJECTION: $C_p := \{\mathbf{x} \in \mathbb{R}^{n_2} : \exists \mathbf{y} \text{ s.t. } (\mathbf{x}, \mathbf{y}) \in C\}$ is convex

7. Given a convex set C , the interior of such set and its closure are convex.

Theorem 5.2.3. \mathcal{P} is a polyhedron iff any of its points can be written as a combination of a point in a convex set and one in a cone. Formally, iff $\exists \{\mathbf{x}_1, \dots, \mathbf{x}_k\} \subseteq \mathbb{R}^n$ and $\{\mathbf{d}_1, \dots, \mathbf{d}_h\} \subseteq \mathbb{R}^n$ s.t. $\mathcal{P} = \text{conv}(\{\mathbf{x}_1, \dots, \mathbf{x}_k\}) + \text{cone}(\{\mathbf{d}_1, \dots, \mathbf{d}_h\})$.

Notice that whenever we are interested in proving that a set with a certain shape is convex, we could try to derive it from an object that we know is convex through the operations we enumerated above.

5.2.2 Convex functions

In the rest of this lecture we will introduce the class of convex functions, that have the characteristic of having local minima coinciding with global minima.

Definition 5.2.6 (Graph). Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$, we term **graph** of f the set of ordered pairs (\mathbf{x}, y) such that $f(\mathbf{x}) = y$.

Definition 5.2.7 (Epigraph). Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$, we term **epigraph** or **super-graph** of f the set of points lying on or above its graph. Formally,

$$\text{epi}(f) = \{(\mathbf{x}, \mu) : \mathbf{x} \in \mathbb{R}^n, \mu \in \mathbb{R}, \mu \geq f(\mathbf{x})\} \subseteq \mathbb{R}^{n+1}$$

Definition 5.2.8 (Convex function). Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be a function. We say that f is **convex** if $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, the segment that joins $f(\mathbf{x})$ and $f(\mathbf{y})$ lies above the function.

In other words, f is **convex** iff $\text{epi}(f)$ is convex.

Equivalently, we say that f is **convex** if $\forall \mathbf{x}, \mathbf{y} \in \text{dom}(f)$ for any $\alpha \in [0, 1]$, $\alpha f(\mathbf{x}) + (1 - \alpha)f(\mathbf{y}) \geq f(\alpha\mathbf{x} + (1 - \alpha)\mathbf{y})$.

Equivalently, $\forall \mathbf{x}^1, \dots, \mathbf{x}^k, \alpha \in \Theta^k$

$$f \left(\sum_{i=1}^k \alpha_i \mathbf{x}^i \right) \leq \sum_{i=1}^k \alpha_i f(\mathbf{x}^i)$$

Definition 5.2.9 (Strict convexity). Let $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$. We term f **strictly convex** iff $\alpha f(\mathbf{x}) + (1 - \alpha)f(\mathbf{y}) > f(\alpha\mathbf{x} + (1 - \alpha)\mathbf{y})$.

Definition 5.2.10 (Concave function). Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be a function. We say that f is **concave** if $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, the segment that joins $f(\mathbf{x})$ and $f(\mathbf{y})$ lies below the function. Formally, $\forall \mathbf{x}, \mathbf{y} \in \text{dom}(f)$ for any $\alpha \in [0, 1]$, $\alpha f(\mathbf{x}) + (1 - \alpha)f(\mathbf{y}) \leq f(\alpha\mathbf{x} + (1 - \alpha)\mathbf{y})$.

Definition 5.2.11 (Sublevel graph). Let $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ be a function. We term **sublevel graph** of f on a scalar $v \in \mathbb{R}$ and denote $S(f, v)$ the projection on the \mathbf{x} axis of the portions of the graph of f which lie below the constant value v .

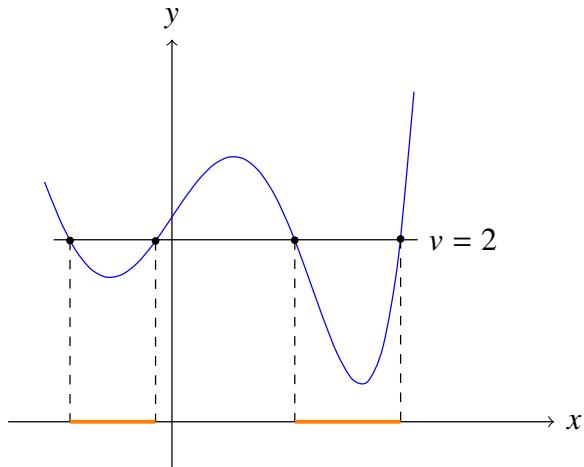


FIGURE 5.4: $f(x) = x^5 - 8x^3 + 10x + 15$ in blue and its sublevel graph in orange.

Fact 5.2.4. The following holds:

- Let f convex. Then $S(f, v)$ is convex $\forall v \in \mathbb{R}$;
- f is concave if $-f$ is convex.

★ Mantra

Convex analysis is a one-sided world.

The second statement of Proposition 5.2.4 is useful to make a comparison between minimizing and maximizing. In particular, if our aim is to maximize the function, we can be sure to have found a global maximum if the function is concave.

Definition 5.2.12 (Strong convexity). Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$. We term f **strongly convex modulus** a scalar $\tau > 0$ iff $f(\mathbf{x}) - \frac{\tau}{2} \|\mathbf{x}\|^2$ is convex.

Formally,

$$\alpha f(\mathbf{x}) + (1 - \alpha)f(\mathbf{y}) \geq f(\alpha\mathbf{x} + (1 - \alpha)\mathbf{y}) + \frac{\tau}{2}\alpha(1 - \alpha)\|(\mathbf{y} - \mathbf{x})\|^2$$

In the next lecture we will provide techniques that allow to check if a function is convex in practice.

Lecture 6

11th of October 2018

A. Frangioni

In the previous lecture we introduced many different definitions about convexity, while in this chapter we will deal with the task of checking when a function is convex, hence a local minimum is also a global minimum.

As we already stated for convex sets, we can prove that a function is convex if we are able to derive it from convex functions, through “convex friendly” operations.

Note

A useful tool to have insights about convexity is the software called CVX, designed to model convex objects. A pretty easy way to check if an object is convex is to try to write it in CVX. If such an operation is possible, then the object is convex.

Let us enumerate some convex functions:

Linear functions: $f(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$ (they are both convex and concave);

Quadratic functions: $f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T Q \mathbf{x} + \mathbf{q}^T \mathbf{x}$ is convex iff $Q \succeq 0$;

Exponential function: $f(\mathbf{x}) = e^{a\mathbf{x}}$ for any $a \in \mathbb{R}$ and $\mathbf{x} \in \mathbb{R}^n$

Anti-logarithmic function: $f(x) = -\log(x)$ for $x > 0$

Monomial: $f(x) = x^a$ for $a \geq 1$ or $a \leq 0$ on $x \geq 0$;

p -norm: $f(\mathbf{x}) = \|\mathbf{x}\|_p$ for $p \geq 1$;

Maximum: $f(x) = \max\{x_1, \dots, x_n\}$;

Indicator function: for any convex set C , its indicator function $\mathbb{1}_C(x)$ is convex:

$$\mathbb{1}_C(x) = \begin{cases} 0 & \text{if } x \in C \\ +\infty & \text{if } x \notin C \end{cases}$$

Sum of m largest eigenvalues: Let $A \in S(n, \mathbb{R})$ such that its eigenvalues (sorted in increasing order) are $\lambda_1 \geq \lambda_2 \geq \dots \lambda_n$. $f_m(A) = \sum_{i=1}^m \lambda_i$ is soncex.

Fact 6.0.1. *The following operations preserve convexity:*

Linear non-negative combination: let $f, g : \mathbb{R}^n \rightarrow \mathbb{R}$ be convex functions and let $\alpha, \beta \in \mathbb{R}_+$, then $\alpha f + \beta g$ is convex ;

Supremum: let $\{f_i\}_{i \in I}$ be a set of infinitely many convex functions, then $\sup_{i \in I} f_i(\mathbf{x})$ is convex, see Figure 6.1(a);

Pre-composition with linear function: let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be a convex function, then $f(A\mathbf{x} + \mathbf{b})$ is convex;

Post-composition with increasing convex function: let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be a convex function and let $g : \mathbb{R} \rightarrow \mathbb{R}$ be a convex increasing function. Then $g \circ f = g(f(\mathbf{x}))$ is convex;

Infimal convolution: let $f_1, f_2 : \mathbb{R}^n \rightarrow \mathbb{R}$ be convex functions. Then $f(\mathbf{x}) = \inf\{f_1(\mathbf{x}_1) + f_2(\mathbf{x}_2) : \mathbf{x}_1 + \mathbf{x}_2 = \mathbf{x}\}$ is convex;

Image under linear mapping: let $g : \mathbb{R}^n \rightarrow \mathbb{R}$ be a convex function. Then $f(\mathbf{x}) = \inf\{g(\mathbf{y}) : A\mathbf{y} = \mathbf{x}\}$ is convex;

Partial minimization: let $g \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} : \mathbb{R}^{n+m} \rightarrow \mathbb{R}$ be a convex function. Then $f(\mathbf{x}) = \inf \{g \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix} : \mathbf{y} \in \mathbb{R}^m\}$ is convex;

Perspective or dilation: let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be a convex function. Then

$$\tilde{f}(\mathbf{x}, u) = \begin{cases} u \cdot f(\mathbf{x}/u) & \text{if } u > 0 \\ \infty & \text{otherwise} \end{cases}$$

is convex, see Figure 6.1(b).

Fact 6.0.2. Let $f : \mathbb{R}^n \rightarrow \mathbb{R} \cup \infty$ be a convex function. If $\exists \bar{\mathbf{x}} \in \text{dom}(f)$ such that $f(\bar{\mathbf{x}}) = -\infty$, then $f \equiv -\infty$.

From now on we will solve the issue of functions with non a convex domain, saying that in those points where the function is not defined, we value it $+\infty$.

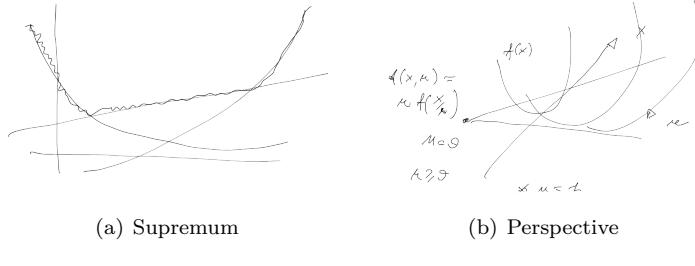


FIGURE 6.1: Examples of convexity-preserving operations.

Fact 6.0.3. Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be a convex function. Then f is Lipschitz continuous on each bounded convex set $S \subseteq \text{int}(\text{dom}(f))$, but we have no information on the behaviour on the border of the domain.

Moreover, a function f , which is continuous but not Lipschitz continuous is not convex.

A couple of examples of Proposition 6.0.3 can be found in Figure 6.2.

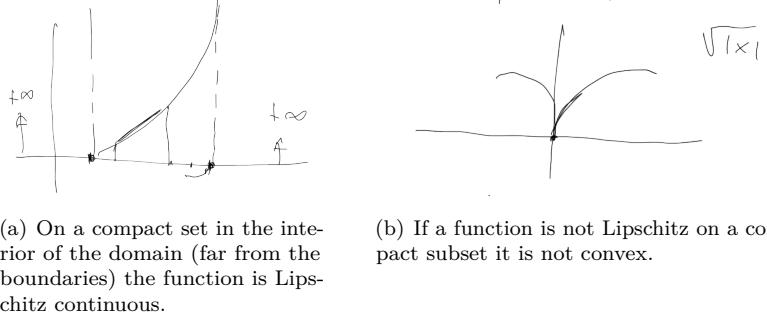


FIGURE 6.2

Fact 6.0.4. Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be a convex function. Then it is Lipschitz continuous on any bounded set and continuous everywhere.

It happens often that the set of points in which a function is non differentiable has measure 0.

Theorem 6.0.5 (Convexity characterization). Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be in \mathcal{C}^1 . It is convex on a convex set $C \subseteq \mathbb{R}^n$ iff

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})(\mathbf{y} - \mathbf{x}) \quad \forall \mathbf{x}, \mathbf{y} \in C$$

Riscrivere proof

Proof. \Rightarrow $\alpha(f(y) - f(x)) \geq f(\alpha(y - x) + x) - f(x)$, send $\alpha \rightarrow 0$

\Leftrightarrow) TODO

□

Theorem 6.0.6. Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ in C^1 be a convex function. $\mathbf{x} \in \mathbb{R}^n$ is a stationary point for f iff \mathbf{x} is a global minimum.

Fact 6.0.7. Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be twice differentiable ($f \in C^2$). f is convex on an open set $S \subseteq \mathbb{R}^n$ iff the Hessian is positive semidefinite.

This proposition gives us an algorithm to check if a function is convex or not: we only need to compute the eigenvalues of the Hessian and check if they are positive.

There are some functions which do not have differentiability property.

A way to work with functions which are not defined on all \mathbb{R}^n is to solve the following problem:

$$(P) \equiv \inf\{f_X(\mathbf{x}) = f(\mathbf{x}) + \mathbf{1}_X(\mathbf{x}) : \mathbf{x} \in \mathbb{R}^n\}$$

thanks to the following

Theorem 6.0.8 (Essential objective). \mathbf{x}_* optimal for $(P) \iff \mathbf{x}_*$ local minimum of f_X .

6.0.1 Non-differentiable functions

It is often the case that the objective function under consideration is not differentiable and in such situation we need to use different tools to be sure to move towards the optimum. It is in this context that we introduce *subgradients* and *subdifferentials*.

Definition 6.0.1 (Subgradient). Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$. We say that $\mathbf{s} \in \mathbb{R}^n$ is a *subgradient* of f at point $\mathbf{x} \in \mathbb{R}^n$ if $\forall \mathbf{y} \in \mathbb{R}^n$ the following holds:

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \mathbf{s}(\mathbf{y} - \mathbf{x})$$

Let us assume that the minimum of the non differentiable function resides in one of its kinky points, then for $\mathbf{s} = \mathbf{0}$ we have a subgradient which is flat and this is a sufficient condition for minimality, for a pictorial example see Figure 6.3.

The issue here is that it is unfeasible to check if $\mathbf{s} = \mathbf{0}$ is a subgradient for f , since we should check all possible $\mathbf{y} \in \mathbb{R}^n$.

Definition 6.0.2 (Subdifferential). Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and let $\mathbf{x} \in \mathbb{R}^n$. We call *subdifferential* the set of all possible subgradients at $\mathbf{x} \in \mathbb{R}^n$.

Formally,

$$\partial f(\mathbf{x}) := \{\mathbf{s} \in \mathbb{R}^n : \mathbf{s} \text{ is a subgradient at } \mathbf{x}\}$$

Definition 6.0.3 (Descent direction). Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and let $\mathbf{x} \in \mathbb{R}^n$. We say that $\mathbf{d} \in \mathbb{R}^n$ is a *descent direction* if $\langle \mathbf{s}, \mathbf{d} \rangle < 0 \ \forall \mathbf{s} \in \partial f(\mathbf{x})$.

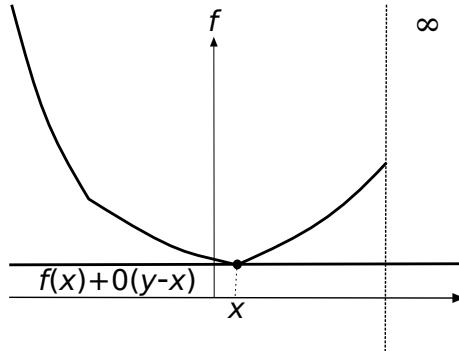


FIGURE 6.3: Pictorial example of subgradients of a non differentiable function.

Theorem 6.0.9. Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$. \mathbf{x} global minimum $\iff 0 \in \partial f(\mathbf{x})$.

Notice that in general, when we are not in proximity of a border (where f is unbounded above) we get that the subdifferential is a compact interval.

Formally, $\partial f(\mathbf{x})$ closed and convex, compact $\forall \mathbf{x} \in \text{int dom}(f)$.

Moreover, we can prove the following

Fact 6.0.10. Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$.

$$\partial f(\mathbf{x}) = \{\nabla f(\mathbf{x})\} \iff f \text{ differentiable at } \mathbf{x}$$

💡 Do you recall?

In Chapter 4, in Proposition 4.0.1 we derived the directional derivative from the gradient as $\frac{\partial f}{\partial \mathbf{d}}(\mathbf{x}) = \langle \nabla f(\mathbf{x}), \mathbf{d} \rangle$.

The following fact tries to provide a similar characterization of the subdifferential.

Fact 6.0.11. Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and let $\mathbf{x} \in \mathbb{R}^n$. If $\frac{\partial f}{\partial \mathbf{d}}(\mathbf{x}) = \sup\{\langle \mathbf{s}, \mathbf{d} \rangle : \mathbf{s} \in \partial f(\mathbf{x})\}$ then \mathbf{d} is a descent direction $\iff \langle \mathbf{s}, \mathbf{d} \rangle < 0 \ \forall \mathbf{s} \in \partial f(\mathbf{x})$.

As in the differentiable case, we are interested in moving in the steepest descent direction, formally $\mathbf{s}_* = -\text{argmin}\{\|\mathbf{s}\| : \mathbf{s} \in \partial f(\mathbf{x})\}$.

Example 6.0.1. Let us assume that we know the minimum point $\mathbf{x}_* \in \mathbb{R}^n$ for $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and we are in \mathbf{x} at the current iteration. We would like to move towards \mathbf{x}_* using only the information provided by the subdifferential. $(-\partial f(\mathbf{x}))$ is convex and compact and all its elements $(-\mathbf{g}) \in \partial f(\mathbf{x})$ (subgradients) “point towards \mathbf{x}_* ”. Formally, $\langle \mathbf{g}, \mathbf{x}_* - \mathbf{x} \rangle < 0$.

Fact 6.0.12 (Linearity of subdifferential). Let $f, g : \mathbb{R}^n \rightarrow \mathbb{R}$ and take $\alpha, \beta \in \mathbb{R}_+$, then $\partial[\alpha f + \beta g](\mathbf{x}) = \alpha \partial f(\mathbf{x}) + \beta \partial g(\mathbf{x})$.

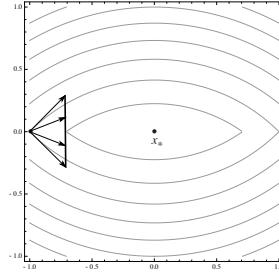


FIGURE 6.4: There are many different subgradients in \mathbf{x} . We pick the one with minimum norm among the ones which have a negative scalar product with $\mathbf{x} - \mathbf{x}_*$.

Fact 6.0.13 (Chain rule). *Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$.*

- *Let $A \in M(n, \mathbb{R})$ and $\mathbf{b} \in \mathbb{R}^n$ then $\partial[f(A\mathbf{x} + \mathbf{b})] = A^T[\partial f](A\mathbf{x} + \mathbf{b})$;*
- *Let $g : \mathbb{R} \rightarrow \mathbb{R}$ be an increasing function. Then $\partial[g(f(\mathbf{x}))] = [\partial g](f(\mathbf{x}))[\partial f](\mathbf{x})$.*

Definition 6.0.4 (ε -subgradient). *Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$. We say that $\mathbf{s} \in \mathbb{R}^n$ is ε -subgradient at $\mathbf{x} \in \mathbb{R}^n$ if it defines a support hyperplane passing ε below $\text{epi}(f)$. Formally,*

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \mathbf{s}(\mathbf{y} - \mathbf{x}) - \varepsilon \quad \forall \mathbf{y} \in \mathbb{R}^n$$

Fact 6.0.14. *Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and let $\mathbf{x} \in \mathbb{R}^n$. Saying that $\mathbf{0} \in \mathbb{R}^n$ is a ε -subgradient for f in \mathbf{x} means that the function value $f(\mathbf{x})$ cannot be at distance grater than ε from the minimum $f(\mathbf{x}_*)$.*

Formally, $0 \in \partial_\varepsilon f(\mathbf{x}) \iff \mathbf{x}$ is ε -optimal.

We are now allowed to compute $\mathbf{s}_* = \operatorname{argmin}\{\|\mathbf{s}\| : \mathbf{s} \in \partial_\varepsilon f(\mathbf{x})\}$.

If $\mathbf{s}_* = \mathbf{0}$ then \mathbf{x} is ε optimal. Otherwise, $\exists \alpha > 0$ s.t. $f(\mathbf{x} - \alpha \mathbf{s}_*) \leq f(\mathbf{x}) - \varepsilon$ ($-\mathbf{s}_*$ is of ε -descent).

The ε -subgradient is very powerful, but the issue is that is even more expensive to compute than the subgradient.

Lecture 7

17th of October 2018

A. Frangioni

7.1 Optimization algorithms

💡 Do you recall?

We are interested in finding the minimum of a function through an iterative procedure, such that we start from an initial guess \mathbf{x}_0 and go on ($\mathbf{x}^i \rightsquigarrow \mathbf{x}^{i+1}$). We want to move towards the optimum.

Notice that with $\mathbf{x}^i \in \mathbb{R}^n$ we refer to the point \mathbf{x} at i -th iteration.

How to be sure we are in an optimum?

- (strong) $\{\mathbf{x}^i\} \rightarrow \mathbf{x}_*$: the whole sequence converges to an optimal solution;
- (weaker) all accumulation points of $\{\mathbf{x}^i\}$ are optimal solutions;
- (weakest) at least one accumulation point of $\{\mathbf{x}^i\}$ is optimal.

Such iterative process, can be held in two different ways:

LINE SEARCH: first choose $\mathbf{d}^i \in \mathbb{R}^n$ (direction), then choose $\alpha^i \in \mathbb{R}$ (that we term stepsize or equivalently “learning rate”) s.t. $\mathbf{x}^{i+1} \leftarrow \mathbf{x}^i + \alpha^i \mathbf{d}^i$;

TRUST REGION: first choose α^i (that we call *trust radius* and somehow indicates the ball in which we think that the information that we have is accurate), then choose \mathbf{d}^i .

In both these alternatives, it is crucial to choose a proper model to approximate function the objective function f .

7.1.1 Gradient method for quadratic functions

The simplest model we can build is a linear one, namely $L^i(\mathbf{x}) = L_{\mathbf{x}^i}(\mathbf{x}) = f(\mathbf{x}^i) + \nabla f(\mathbf{x}^i)(\mathbf{x} - \mathbf{x}^i)$ and find the descending direction of the model.

The first order information is accurate only around \mathbf{x}^i and this is why we should not move too far from \mathbf{x}^i , so we want $\alpha_i \rightarrow 0$ and then we pick the steepest descent direction, computed as $\mathbf{d}_i = \arg \min_d \left\{ \lim_{t \rightarrow 0} \frac{f(\mathbf{x} + t\mathbf{d})}{t} \right\} = -\nabla f(\mathbf{x}^i)$.

Notice that a too short step is bad either, because the gain in the value of the function is very little.

Alternatively, we can choose to pick a different step size at each iteration $\alpha^i = \arg \min_{\alpha} f(\mathbf{x}^i + \alpha \mathbf{d}^i)$ s.t. $\alpha \geq 0$.

The main drawback of choosing linear functions to approximate the objective function is that linear functions are unbounded below. The easiest family of functions which are still simple, yet more complex than linear ones are quadratic functions:

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T Q \mathbf{x} + \mathbf{q}^T \mathbf{x}$$

where $Q \succeq 0$, otherwise f would be unbounded below.

The minimum is the point in which the gradient ($\nabla f(\mathbf{x}) = Q\mathbf{x} + \mathbf{q}$) is 0.

A formalization of the gradient descent algorithm for quadratic functions can be found in Algorithm 7.1.1.

ALGORITHM 7.1.1 Pseudocode for quadratic functions local minimum detection.

```

1: procedure SDQ( $f, \mathbf{x}, \varepsilon$ )
2:   while ( $\|\nabla f(\mathbf{x})\| > \varepsilon$ ) do
3:      $d \leftarrow -\nabla f(\mathbf{x});$ 
4:      $\alpha \leftarrow \frac{\|d\|^2}{(d^T Q d)};$ 
5:      $x \leftarrow \mathbf{x} + \alpha d;$ 
6:   end while
7: end procedure
```

As usual, we need to choose an error threshold ε under the one we feel allowed to think that the optimum has been reached and we set it as the norm of the gradient.

As far as the step size is concerned, let us find the minimum of $f(\mathbf{x}^i + \alpha \mathbf{d}^i)$ such

that $\alpha \geq 0$:

$$\begin{aligned}
\min_{\alpha} f(\mathbf{x}^i + \alpha \mathbf{d}^i) &= \min_{\alpha} \frac{1}{2} (\mathbf{x}^i + \alpha \mathbf{d}^i)^T Q (\mathbf{x}^i + \alpha \mathbf{d}^i) + \mathbf{q}^T (\mathbf{x}^i + \alpha \mathbf{d}^i) \\
&= \min_{\alpha} \frac{1}{2} \mathbf{x}^{i^T} Q \mathbf{x}^i + \frac{1}{2} \mathbf{x}^{i^T} Q \alpha \mathbf{d}^i + \frac{1}{2} \alpha \mathbf{d}^{i^T} Q \mathbf{x}^i + \frac{1}{2} \alpha \mathbf{d}^{i^T} Q \alpha \mathbf{d}^i + \mathbf{q}^T \mathbf{x}^i + \alpha \mathbf{q}^T \mathbf{d}^i \\
&\stackrel{(1)}{=} \min_{\alpha} \frac{1}{2} \mathbf{x}^{i^T} Q \mathbf{x}^i + \alpha \mathbf{x}^{i^T} Q \mathbf{d}^i + \frac{1}{2} \alpha^2 \mathbf{d}^{i^T} Q \mathbf{d}^i + \mathbf{q}^T \mathbf{x}^i + \alpha \mathbf{q}^T \mathbf{d}^i \\
&\stackrel{(2)}{=} \min_{\alpha} \alpha \mathbf{x}^{i^T} Q \mathbf{d}^i + \frac{1}{2} \alpha^2 \mathbf{d}^{i^T} Q \mathbf{d}^i + \alpha \mathbf{q}^T \mathbf{d}^i \\
&= \min_{\alpha} \left[\frac{1}{2} (\mathbf{d}^i)^T Q \mathbf{d}^i \right] \alpha^2 + \alpha [(\mathbf{x}^{i^T} Q + \mathbf{q}^T) \mathbf{d}^i] \\
&= \min_{\alpha} \left(\left(\frac{1}{2} \mathbf{d}^{i^T} Q \mathbf{d}^i \right) \alpha^2 + \alpha \cdot \left(\mathbf{x}^{i^T} Q + \mathbf{q}^T \right) \mathbf{d}^i \right)
\end{aligned}$$

where $\stackrel{(1)}{=}$ is due to the fact that Q is symmetric and $\stackrel{(2)}{=}$ follows from the fact that $\frac{1}{2} \mathbf{x}^{i^T} Q \mathbf{x}^i$ and $\mathbf{q}^T \mathbf{x}^i$ are constant with respect to α .

The minimum of such f can be found deriving and imposing the gradient to 0.

$$\begin{aligned}
\frac{\partial f(\mathbf{x}^i + \alpha \mathbf{d}^i)}{\partial \alpha} &= 0 \\
\frac{\partial \left(\left(\frac{1}{2} \mathbf{d}^{i^T} Q \mathbf{d}^i \right) \alpha^2 + \alpha \cdot \left(\mathbf{x}^{i^T} Q + \mathbf{q}^T \right) \mathbf{d}^i \right)}{\partial \alpha} &= 0 \\
\cancel{\alpha} \cdot \frac{1}{2} \mathbf{d}^{i^T} Q \mathbf{d}^i \cdot \alpha + (\mathbf{x}^{i^T} Q + \mathbf{q}^T) \mathbf{d}^i &= 0 \\
\mathbf{d}^{i^T} Q \mathbf{d}^i \cdot \alpha &= -(\mathbf{x}^{i^T} Q + \mathbf{q}^T) \mathbf{d}^i \\
\alpha &= -\frac{(\mathbf{x}^{i^T} Q + \mathbf{q}^T) \mathbf{d}^i}{\mathbf{d}^{i^T} Q \mathbf{d}^i}
\end{aligned}$$

Per me da qui non ha senso...

Hence, the minimum value of f is computed as

$$\begin{aligned}
f(\mathbf{x}^i + \alpha \mathbf{d}^i) &= \frac{1}{2} (\mathbf{x}^i + \alpha \mathbf{d}^i)^T Q (\mathbf{x}^i + \alpha \mathbf{d}^i) + \mathbf{q}^T (\mathbf{x}^i + \alpha \mathbf{d}^i) \\
&= \cancel{\frac{1}{2} \mathbf{x}^{i^T} Q \mathbf{x}^i} + (\mathbf{x}^i)^T Q (\alpha \mathbf{d}^i) + \frac{1}{2} \alpha^2 (\mathbf{d}^i)^T Q \mathbf{d}^i + \cancel{\mathbf{q}^T \mathbf{x}^i} + \alpha (\mathbf{q}^T \mathbf{d}^i) \\
&\stackrel{(1)}{=} \left(\frac{1}{2} (\mathbf{d}^i)^T Q \mathbf{d}^i \right) \alpha^2 + \alpha \left((\mathbf{x}^i Q + \mathbf{q}^T) \mathbf{d}^i \right) \\
&\stackrel{(2)}{=} \left(\frac{1}{2} (\mathbf{d}^i)^T Q \mathbf{d}^i \right) \alpha - \|\mathbf{d}^i\|
\end{aligned} \tag{7.1.1}$$

What if $(\mathbf{d}^i)^T Q \mathbf{d}^i = 0$? If Q is strictly positive definite this cannot happen, so the algorithm never breaks down.

... Fino a qui

At this point we are left with a few issues:

- Pick a proper $\varepsilon > 0$ (e.g. $\varepsilon = 0$ does not lead to convergence with machine precision)
- Check if $\{\mathbf{x}^i\}$ converges. This can be easily proved by construction:

$$\lim_{i \rightarrow \infty} \langle \nabla f(\mathbf{x}^i), \nabla f(\mathbf{x}^{i+1}) \rangle \stackrel{(*)}{=} \langle \nabla f(\mathbf{x}_*), \nabla f(\mathbf{x}_*) \rangle = 0$$

because at each step the gradients are orthogonal and $\stackrel{(*)}{=}$ follows from the fact that the gradient is continuous.

- Assess the convergence speed.

In general, showing how fast $\|\mathbf{x}^i - \mathbf{x}_*\|$ decreases is more involved than showing how fast $f(\mathbf{x}^i) - f_*$ decreases, but we do not know f_* . We concentrate on computing $\lim_{i \rightarrow \infty} \frac{f(\mathbf{x}^{i+1}) - f_*}{f(\mathbf{x}^i) - f_*} = R$. According to the values of p and R we get the following alternatives:

SUBLINEAR: $p = 1, R = 1$;

LINEAR: $p = 1, R < 1$;

SUPERLINEAR: $p = 1, R = 0$;

QUADRATIC: $p = 2, R > 0$.

Since the optimum is in $\mathbf{x}_* = -Q^{-1}\mathbf{q}$, we get that

$$\begin{aligned} f(\mathbf{x}_*) &= \frac{1}{2}(-Q^{-1}\mathbf{q})^T Q(-Q^{-1}\mathbf{q}) - \mathbf{q}^T Q^{-1}\mathbf{q} \\ &= \frac{1}{2}\mathbf{q}^T Q^{-1} Q Q^{-1}\mathbf{q} - \mathbf{q}^T Q^{-1}\mathbf{q} \\ &\stackrel{(*)}{=} \frac{1}{2}\mathbf{q}^T Q^{-1}\mathbf{q} - \mathbf{q}^T Q^{-1}\mathbf{q} \\ &= -\frac{1}{2}\mathbf{q}^T Q^{-1}\mathbf{q} \end{aligned}$$

Algebraically speaking, the minimum \mathbf{x}_* of a quadratic function is a stationary point i.e. $\nabla f(\mathbf{x}) = Q\mathbf{x} + \mathbf{q} = 0$ iff $\mathbf{x}_* = -Q^{-1}\mathbf{q}$ when Q is invertible. At this point, we need to assess the convergence speed of the algorithm, thus we need to define an error function \bar{f} , such that “the error at \mathbf{x} is the distance between \mathbf{x} and \mathbf{x}_* in $\|\cdot\|_Q$ ”:

$$\begin{aligned}
\bar{f}(\mathbf{x}) &= \frac{1}{2}(\mathbf{x} - \mathbf{x}_*)^T Q(\mathbf{x} - \mathbf{x}_*) \\
&= \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \frac{1}{2}\mathbf{x}_*^T Q\mathbf{x}_* - \mathbf{x}^T(Q\mathbf{x}_*) \\
&\stackrel{(2)}{=} \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \frac{1}{2}(Q^{-1}\mathbf{q})^T Q(Q^{-1}\mathbf{q}) - \mathbf{x}^T \cancel{Q}(-Q^{-1}\mathbf{q}) \\
&= \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \frac{1}{2}Q^{-1}\mathbf{q}^T Q(Q^{-1}\mathbf{q}) + \mathbf{q}^T \mathbf{x} \\
&= \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \frac{1}{2}\mathbf{q}^T \cancel{Q} \cancel{Q}^{-1}\mathbf{q} + \mathbf{q}^T \mathbf{x} \\
&= \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \frac{1}{2}\mathbf{q}^T Q^{-1}\mathbf{q} + \mathbf{q}^T \mathbf{x} \\
&= \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{q}^T \mathbf{x} + \frac{1}{2}\mathbf{q}^T Q^{-1}\mathbf{q} \\
&= f(\mathbf{x}) - f(\mathbf{x}_*)
\end{aligned} \tag{7.1.2}$$

where $\stackrel{(2)}{=}$ follows from the equality $\mathbf{x}_* = -Q^{-1}\mathbf{q}$.

Rispetto alle slide, qui denoto con $\bar{f}(\cdot)$ la funzione che quantifica l'errore, anzichè chiamarla $f_*(\cdot)$, perché questa seconda notazione si confonde con f_* (senza argomento), che è il valore ottimo della funzione obiettivo.

Lecture 8

19th of October 2018

A. Frangioni

8.0.1 Gradient method for quadratic functions (cont.)

💡 Do you recall?

The family of quadratic functions is the simplest possible family of functions where a minimum exists. A quadratic function is defined as: $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{q}^T \mathbf{x}$, so its gradient is computed as $\nabla f(\mathbf{x}) = Q\mathbf{x} + \mathbf{q}$. We are interested in finding local minima of the function f .

In last lecture we were left with analyzing the convergence of the algorithm

Fact 8.0.1. *A quadratic function $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$, s.t. $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} + \mathbf{q}^T \mathbf{x}$ admits a minimum iff $Q \succeq 0$ (Q is positive semidefinite).*

The gradient method generates points that move along orthogonal directions. More formally, $\mathbf{x}^{i+1} = \mathbf{x}^i + \alpha^i \mathbf{d}^i$, where $\mathbf{d}^i = -\nabla f(\mathbf{x}^i) = -Q\mathbf{x}^i - \mathbf{q}$

Fact 8.0.2. *Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be a quadratic function and let us consider the gradient descent algorithm for minimizing f , where at iteration i the moving direction $\mathbf{d}^i \in \mathbb{R}^n$ and the step size $\alpha^i \in \mathbb{R}$ are computed. The following holds: $\langle \mathbf{d}^i, \mathbf{d}^{i+1} \rangle = 0$.*

Proof. Let us rewrite \mathbf{d}^{i+1} in terms of \mathbf{d}^i :

$$\begin{aligned}\mathbf{d}^{i+1} &= -Q\mathbf{x}^{i+1} - \mathbf{q} \\ &= -Q(\mathbf{x}^i + \alpha^i \mathbf{d}^i) - \mathbf{q} \\ &= -Q\mathbf{x}^i - Q\alpha^i \mathbf{d}^i - \mathbf{q} \\ &= \mathbf{d}^i - Q\alpha^i \mathbf{d}^i\end{aligned}$$

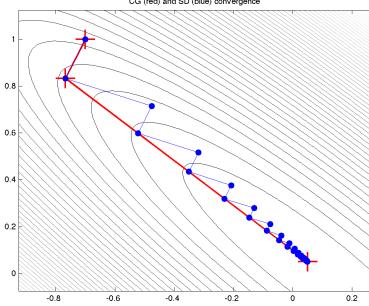


FIGURE 8.1: Some iterations of the gradient method

If we plug this into the scalar product $\langle \mathbf{d}^i, \mathbf{d}^{i+1} \rangle$ we get

$$\begin{aligned} \langle \mathbf{d}^i, \mathbf{d}^{i+1} \rangle &= \langle \mathbf{d}^i, \mathbf{d}^i - Q\alpha^i \mathbf{d}^i \rangle \\ &= \|\mathbf{d}^i\|^2 - \mathbf{d}^i{}^T Q\alpha^i \mathbf{d}^i \\ &= \|\mathbf{d}^i\|^2 - \alpha^i \mathbf{d}^i{}^T Q \mathbf{d}^i \\ &\stackrel{*}{=} \|\mathbf{d}^i\|^2 - \frac{\|\mathbf{d}^i\|^2}{\mathbf{d}^i{}^T Q \mathbf{d}^i} \cdot \cancel{\mathbf{d}^i{}^T Q \mathbf{d}^i} \end{aligned}$$

where $\stackrel{*}{=}$ follows from $\alpha^i := \frac{\|\mathbf{d}^i\|^2}{\mathbf{d}^i{}^T Q \mathbf{d}^i}$. \square

Let us go back to the error function

$$\bar{f}(\mathbf{x}^{i+1}) = f(\mathbf{x}^{i+1}) - f_* = \frac{1}{2} (\mathbf{x}^{i+1} - \mathbf{x}_*)^T Q (\mathbf{x}^{i+1} - \mathbf{x}_*)$$

The $(i + 1)$ th iteration is computed as

$$\begin{aligned} \mathbf{x}^{i+1} &= \mathbf{x}^i + \alpha^i \mathbf{d}^i \\ &\stackrel{(a)}{=} \mathbf{x}^i + \frac{\|\mathbf{d}^i\|^2}{\mathbf{d}^i{}^T Q \mathbf{d}^i} \mathbf{d}^i \\ &\stackrel{(b)}{=} \mathbf{x}^{i+1} = \mathbf{x}^i + \frac{\|-\mathbf{Q}\mathbf{x}^i - \mathbf{q}\|^2}{(-\mathbf{Q}\mathbf{x}^i - \mathbf{q})^T Q (-\mathbf{Q}\mathbf{x}^i - \mathbf{q})} \end{aligned}$$

where $\stackrel{(a)}{=}$ follows from the definition $\alpha^i := \frac{\|\mathbf{d}^i\|^2}{\mathbf{d}^i{}^T Q \mathbf{d}^i}$ and $\stackrel{(b)}{=}$ from the fact that the moving direction is the opposite of the gradient (formally, $\mathbf{d}^i = -\mathbf{Q}\mathbf{x}^i - \mathbf{q}$).

This means not only **linear convergence**, but a bit more, because linear convergence only takes into consideration steps in proximity to the limit, while this formula holds at the beginning as well.

Fact 8.0.3. Let $A \in M(n, \mathbb{R})$ be a positive semidefinite matrix. $\forall \mathbf{v} \in \mathbb{R}^n$ $\mathbf{v}^T A \mathbf{v} = \|\mathbf{v}\|_A^2$.

Fact 8.0.4. If Q is positive definite we can say that the error goes like $1 - \left(\frac{\|\mathbf{d}_i\|^4}{(\mathbf{d}_i^T Q \mathbf{d}_i)(\mathbf{d}_i^T Q^{-1} \mathbf{d}_i)} \right)$ or, equivalently, $1 - \frac{\|\mathbf{d}_i\|_I^2}{\|\mathbf{d}_i\|_Q^2} \cdot \frac{\|\mathbf{d}_i\|_I^2}{\|\mathbf{d}_i\|_{Q^{-1}}^2}$.

Fare proof, $\bar{f}(\mathbf{x}^{i+1}) = 1 - \left(\frac{\|\mathbf{d}_i\|^4}{(\mathbf{d}_i^T Q \mathbf{d}_i)(\mathbf{d}_i^T Q^{-1} \mathbf{d}_i)} \right) \bar{f}(\mathbf{x}^i)$ (hint: for $y^i = x^i - x_*$, $d^i = Qy^i$)

We are measuring a vector with three different norms. We would like to estimate $\frac{\langle \mathbf{d}_i, \mathbf{d}_i \rangle}{\mathbf{d}_i^T Q \mathbf{d}_i}$.

Fact 8.0.5. Let $A \in M(n, \mathbb{R})$ be a positive semidefinite matrix. Given λ_i the eigenvalues of A , $\frac{1}{\lambda_i}$ are the eigenvalues of the matrix A^{-1} .

We can say that $\lambda_n \|\mathbf{x}\|^2 \leq \mathbf{x}^T Q \mathbf{x} \leq \lambda_1 \|\mathbf{x}\|^2$, where λ_n is the smallest eigenvalue, while λ_1 is the biggest.

We are looking for a close formula for calculating the convergence rate, since it depends recursively by the steps done. So we want to do a worst case analysis in order to find a faster way to calculate convergence rate.

We want to prove that R is smaller than 1 so we are looking for an upper-bound.

A coarse upperbound is $(1 - \frac{\lambda_n}{\lambda_1})$, but we can prove more:

Fact 8.0.6. Let $A \in M(n, \mathbb{R})$ be a positive definite matrix. Given λ_i the eigenvalues of A ,

$$\forall \mathbf{v} \in \mathbb{R}^n \quad \frac{\|\mathbf{v}\|^4}{(\mathbf{v}^T A \mathbf{v})(\mathbf{v}^T A^{-1} \mathbf{v})} \geq \frac{4\lambda^1 \lambda^n}{(\lambda^1 + \lambda^n)^2}$$

We won't see the proof of this fact.

R close to 0 means that the algorithm is converging fast, so when the larger eigenvalue (λ_1) and the smallest eigenvalue (λ_n) are very close to each-other the algorithm is converging fast. We can say that, since the eigenvalues represent the length of the axis of the ellipsoids (level sets), the algorithm is converging fast when the ellipsoid has a round shape.

We can provide an even better estimate, which is $(\frac{\lambda_1 - \lambda_n}{\lambda_1 + \lambda_n})^2$. At this point, provided that we can estimate the number of iterations needed to reach convergence, how can we say that such a number is a good result? As usual, it depends on how that number is obtained. If it is dimensional independent it is very good, because it scales well (when the size of the space - number of variables - increases). It only depends on the conditioning of the matrix Q .

Of course as n grows Q changes, so in practice it may happen that the conditioning of the problem is worsening as n grows.

If the balls are very rounded the zig zags needed to start converging are very few.

Note

So far, we provided a bound for the convergence speed of the algorithm when Q is positive definite. What can we say when Q is positive semidefinite? The algorithm works, but we can't provide an upper-bound for the convergence rate. We are even more restrictive when dealing with machine precision, since if there is an eigenvalue which is bigger than zero, but very close to zero, it turns out to be 0 on the machine, so we cannot provide an upper-bound. We will see how to deal with this case.

TODO: mettere reference

8.0.2 Quadratic gradient descent in Matlab

When we are trying to move from theory to practice we need to take into account some practical issue, such as picking a good value for ε . A good idea would be the norm of the gradient, but in order not to end up in a loop we need to provide also some performances bound, e.g. the maximum number of iterations or the maximum amount of time.

Let us suppose that we implemented the quadratic gradient descent algorithm in Matlab, with the following signature:

```
1 function [x, status] = SDQ(Q, q, x, fStar, eps, MaxIter)
```

where `fStar` is the optimal value, which is used to provide an estimate of the convergence.

Note

When implementing functions for optimization is important to:

- check the coherence of input values (namely, if the user passed allowed parameters);
- give all possible information about your result (for example if the algorithm stopped because the maximum number of iterations was reached or because the epsilon value was reached);
- check at any iteration the values of variables during computations (e.g. before dividing by a quantity that may be smaller than the precision check it before computing the ratio);
- design a good log, in order to understand what is going on at each step.

Example 8.0.1. In the implementation of `SDQ`* provided by professor Frangioni, the execution keeps track of the actual ratio between the error at one step and

*<https://elearning.di.unipi.it/mod/resource/view.php?id=5033>

the error at the next step. This information provides insights about how many orders of magnitude the error decreases. We can find starting points where the ratio is exactly R . We can observe that when the conditioning is quite good the error decrease faster than the R limitation, but as soon as we change the values for Q and q things may change completely.

If the reader wants to run some examples, he should notice that if the conditioning grows, the number of iterations needed to find the minimum increases as well.

Trying the algorithm on some examples shows that the theoretical results are reflected well in the practical case.

Error

As mentioned above, if we picked as starting point for the algorithm a value that does not lead to a minimizing sequence, we need to stop iterating after a while, possibly when we are close enough to the solution, i.e. the absolute error ε_A is small enough: $\varepsilon_A = |f(x^i) - f_*| \leq \varepsilon$. It is in this context that we introduce the concept of **relative error**, that compares the error with the value of the function.

Definition 8.0.1 (Relative error). Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be a quadratic function and let us consider the gradient descent algorithm for minimizing f , where at iteration i the moving direction $\mathbf{d}^i \in \mathbb{R}^n$ and the step size $\alpha^i \in \mathbb{R}$ are computed. We term **relative error** at the i -th iteration with respect to the optimal value and denote ε_R

$$\varepsilon_R = |f(x^i) - f_*| / |f_*| = \varepsilon_A / |f_*| \leq \varepsilon$$

This relative error is invariant for scaling transformations. Notice that if f^* might be zero the formula should be changed.

In practice, computing such error is unfeasible because we do not know f^* . In this very common case we can substitute f^* with a good lower bound $\underline{f} \leq f^*$ for f^* , but in this course we will not focus on finding \underline{f} .

Alternative stopping conditions are bounds on the norm of the gradient:

- $\|\nabla f(x^i)\| \leq \varepsilon$ (“absolute version”)
- $\|\nabla f(x^i)\| / \|\nabla f(x^0)\| \leq \varepsilon$ (“relative version”)

The second stopping condition is expressed in relation to the value of the norm of the gradient at the starting point.

We usually choose the norm of the gradient as a threshold for precision, but we do not know how this quantity relates to ε_A or ε_B .

Example 8.0.2. Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ a convex function and let $X = \mathcal{B}(\mathbf{0}, r)$. Estimate ε_A when $\|\nabla f(\mathbf{x}^i)\| \leq \varepsilon$. Provided that we are studying a convex function we can easily find the minimum in a ball. We can then minimize the linear function in the range of the ball and that minimum is surely a lower bound.

Lecture 9

24th of October 2018

A. Frangioni

As we said so far, dealing with quadratic (approximations of) functions is pretty straightforward. In this chapter and a few subsequent ones, we will explore functions that have different shapes.

9.1 Gradient method for non quadratic functions

💡 Do you recall?

Let us consider that we are sitting in a point \mathbf{x}^i at the i -th iteration. In order to compute the next step, in the case of quadratic functions, we decided to put $\alpha^i = \frac{\|\mathbf{d}^i\|^2}{\mathbf{d}^{iT} Q \mathbf{d}^i}$ and $\mathbf{d}^i = -\nabla f(\mathbf{x}^i) = -Q\mathbf{x}^i - \mathbf{q}$. We may recall from last lecture that the proof of the orthogonality of the gradient (Proposition 8.0.2) does not depend on the quadratic nature of our functions, so it could work in the non-quadratic case as well.

We would like to find a more general form for the step size, which does not depend on the fact that the function is quadratic.

The algorithm for finding local minima of non quadratic functions has the same structure of the one used for quadratic ones, i.e. first compute the direction of the step and then compute its size and it is formalized in Algorithm 9.1.1.

ALGORITHM 9.1.1 Pseudocode for non quadratic functions local minimum detection.

```

1: procedure SDQ( $f, \mathbf{x}, \varepsilon$ )
2:   while ( $\|\nabla f(\mathbf{x})\| > \varepsilon$ ) do
3:      $\mathbf{d} \leftarrow -\nabla f(\mathbf{x});$ 
4:      $\alpha \leftarrow \frac{\|\mathbf{d}\|^2}{(\mathbf{d}^T Q \mathbf{d})};$ 
5:      $\mathbf{x} \leftarrow \mathbf{x} + \alpha \mathbf{d};$ 
6:   end while
7: end procedure

```

We will see that, differently from the quadratic case (where the gradient was $\nabla f(\mathbf{x}) = Q\mathbf{x} + \mathbf{q}$) computing the gradient in this more general case is not trivial. We are now interested in assessing the converge speed of Algorithm 9.1.1.

Theorem 9.1.1 (Convergence speed). *Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be a function in C^2 and let \mathbf{x}_* be a local minimum for f such that the Hessian of f is strictly positive definite ($\nabla^2 f(\mathbf{x}_*) \succ 0$). If $\{\mathbf{x}^i\}$ converges it is a minimizing sequence. Formally,*

$$\{\mathbf{x}^i\} \rightarrow \mathbf{x}_* \implies \{f(\mathbf{x}^i)\} \rightarrow f(\mathbf{x}_*)$$

linearly, with same R as the quadratic case, depending on λ_1 and λ_n , respectively the biggest and smallest eigenvalues of the Hessian.

This theorem means that if the function is differentiable and the Hessian is strictly positive definite then when getting closer and closer to the minimum, the function is more and more similar to a quadratic function.

This similarity is a good new, since we can use the same methods of the quadratic case, but, as usual, we must pay attention to **conditioning**.

9.1.1 Finding the best step size

At this point we are left with the task of computing a step length, say α^i such that $f(\mathbf{x}^i + \alpha^i \mathbf{d}^i) < f(\mathbf{x}^k)$. This may necessitate a one-dimensional (or line) search, i.e. finding the local minimum of the one dimensional function φ^i , s.t.

$$\varphi^i(\alpha) = f(\mathbf{x}^i + \alpha \mathbf{d}^i),$$

where $\mathbf{d}^i = -\nabla f(\mathbf{x}^i)$.

Let us omit the index i , since we are concentrating on a single iteration. We are interested in finding $\alpha^* \in \mathbb{R}$ such that $\varphi'(\alpha^*) = 0$

Example 9.1.1. *Let us take $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ s.t. $f(x_1, x_2) = x_1^2 e^{x_2}$. We can differentiate f and obtain $\nabla f(x_1, x_2) = (2x_1 e^{x_2}, x_1^2 e^{x_2})^T$. Let us suppose that at the i -th iteration $\mathbf{x} = (0, 1)^T$, so $\nabla f(1, 0) = (2, 1)$. Now $\mathbf{x}(\alpha) = \mathbf{x} - \alpha \nabla f(\mathbf{x}) = (1, 0)^T - \alpha(2, 1)^T = (1 - 2\alpha, 0 - \alpha)^T$. At this point we obtain $\varphi(\alpha) = f(\mathbf{x}(\alpha)) = (1 - 2\alpha)^2 e^{-\alpha}$.*

In general, we are not interested in finding those points where $\varphi'(\alpha) = 0$, instead we want to force the directional derivative to be small (by picking a threshold ε' s.t. $|\varphi'(\alpha)| \leq \varepsilon'$).

Fact 9.1.2. Let $\varphi : \mathbb{R} \rightarrow \mathbb{R}$, such that $\varphi(\alpha) = f(\mathbf{x}^i + \alpha \mathbf{d}^i)$, $\varphi'(\alpha) = \langle \nabla f(\mathbf{x}^i + \alpha \mathbf{d}^i), \mathbf{d}^i \rangle$.

Proof.

$$\frac{\partial f(\mathbf{x}^i + \alpha \mathbf{d}^i)}{\partial \alpha} = \frac{\partial f}{\partial \mathbf{x}^i + \alpha \mathbf{d}^i} \cdot \frac{\partial \mathbf{x}^i + \alpha \mathbf{d}^i}{\partial \alpha} = \nabla f(\mathbf{x}^i + \alpha \mathbf{d}^i) \cdot \mathbf{d}^i$$

□

Fact 9.1.3. We claim that $\varepsilon' = \varepsilon$.

Proof. We want to find the relationship between the two parameters ε and ε' . Assuming that we've got a black box that finds α , given ε' , we are interested in computing ε' .

Key idea: Normalization of the direction.

We may normalize the direction of movement \mathbf{d}^i without perturbing the behaviour of the algorithm: $\mathbf{d}^i = -\frac{\nabla f(\mathbf{x}^i)}{\|\nabla f(\mathbf{x}^i)\|}$. Note that we're not worried of dividing by the norm of the gradient, since if it gets 0 we have already stopped the procedure.

In this new context $\|\mathbf{d}^i\| = 1$ and

$$\begin{aligned} \varphi'(0) &= \frac{\partial f(\mathbf{x})}{\partial \mathbf{d}} \\ (\text{From Proposition 9.1.2}) &= \langle \nabla f(\mathbf{x}), \mathbf{x} \rangle \\ &= \langle \nabla f(\mathbf{x}), \frac{-\nabla f(\mathbf{x})}{\|\nabla f(\mathbf{x})\|} \rangle \\ &= -\frac{\langle \nabla f(\mathbf{x}), \nabla f(\mathbf{x}) \rangle}{\|\nabla f(\mathbf{x})\|} \\ &= -\frac{\|\nabla f(\mathbf{x})\|^2}{\|\nabla f(\mathbf{x})\|} \\ &= -\|\nabla f(\mathbf{x})\| \end{aligned}$$

$$|\varphi'(\alpha^i)| = |\langle \nabla f(\mathbf{x}^i + \alpha \mathbf{d}^i), \mathbf{d}^i \rangle| = |\langle \nabla f(\mathbf{x}^{i+1}), \mathbf{d}^i \rangle| = \left| \langle \nabla f(\mathbf{x}^{i+1}), -\frac{\nabla f(\mathbf{x}^i)}{\|\nabla f(\mathbf{x}^i)\|} \rangle \right|$$

If the iterations converge to the optimum point (formally, $\{\mathbf{x}^i\} \rightarrow \mathbf{x}$)

$$\begin{aligned} \lim_{i \rightarrow \infty} |\varphi'(\alpha^i)| &= \lim_{i \rightarrow \infty} \left| \langle \nabla f(\mathbf{x}^{i+1}), \frac{\nabla f(\mathbf{x}^i)}{\|\nabla f(\mathbf{x}^i)\|} \rangle \right| \\ &= \left| \langle \nabla f(\mathbf{x}), \frac{\nabla f(\mathbf{x})}{\|\nabla f(\mathbf{x})\|} \rangle \right| \\ &= \left| \frac{\langle \nabla f(\mathbf{x}), \nabla f(\mathbf{x}) \rangle}{\|\nabla f(\mathbf{x})\|} \right| \\ &= \|\nabla f(\mathbf{x})\| \leq \varepsilon' \end{aligned}$$

At each iteration of the gradient descent algorithm we check the condition $\|\nabla f(\mathbf{x}^i)\| > \varepsilon$, hence we have the thesis. \square

Per each phase the new epsilon is obtained $\varepsilon \leftarrow \varepsilon \|\nabla f(\mathbf{x}^i)\|$.

If we can prove that the algorithm is converging we know when to stop.

This convergence is not the perfect mathematical convergence, since $\varepsilon \neq 0$, because the line search will never terminate.

In the rest of this subsection we will propose three different kinds of algorithms for performing the line search on φ :

- First order algorithms;
- Second order algorithms;
- Zero order algorithms;
- Inexact line search;

where the first three approaches provide an exact solution, while the fourth one is less accurate but faster.

First order algorithms

This class of algorithms makes use of the first derivative of the function φ for finding the exact minimum of φ , namely we want to find the minimum points of φ , which corresponds to points where the first order derivative is zero and it goes from negative to positive.

We will present three different algorithms for this class, two of them are iterative, while the last one is a direct method. For the first two methods, we would like to reduce the range in which performing the search, at each step.

Doubling: How can we be sure that in a given range there is a point where the derivative is 0? Rolle's theorem, as shown in Figure 9.1.

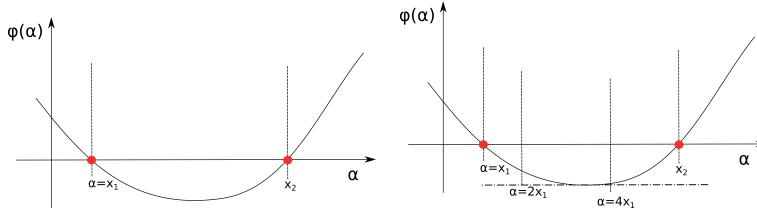


FIGURE 9.1: First, we restrict from \mathbb{R} to $[x_1, x_2]$, then double α until the derivative is greater than 0

Since the gradient is continuous the directional derivative is continuous, so φ is continuous (the scalar product is continuous).

Actually, we only need to find where the derivative is positive, because the 0 of the derivative is between the previous value and this point. A formalization of the algorithm can be found in Algorithm 9.1.2.

ALGORITHM 9.1.2 First algorithm for exact line search

```
1:  $\bar{\alpha} \leftarrow x_1$ ; #or whatever value > 0
2: while ( $\varphi'(\bar{\alpha}) < 0$ ) do
3:    $\bar{\alpha} \leftarrow 2\bar{\alpha}$ ; #or whatever factor > 1
4: end while
```

Since we are dealing with machine precision we stop when $\alpha < -10^{308}$, which is the smallest value for a double.

Works if φ **coercive**: $\lim_{\alpha \rightarrow \infty} \varphi(\alpha) = \infty$ (ex. f strongly convex)

Exercise 9.1.1. Build an example where $\bar{\alpha}$ exists but it is not found by this algorithm.

Solution: The function changes its derivative in a range between α and 2α .

Bisection: we pick the middle point of the interval $[\alpha_-, \alpha_+]$, as shown in Algorithm 9.1.3.

ALGORITHM 9.1.3 Bisection algorithm

```
1: procedure LSBM( $(\varphi', \bar{\alpha}, \varepsilon)$ )
2:    $\alpha_- \leftarrow 0$ ;
3:    $\alpha_+ \leftarrow \bar{\alpha}$ ;
4:    $\alpha \leftarrow \alpha_+$ ;
5:   while ( $|\varphi'(\alpha)| > \varepsilon$ ) do
6:      $\alpha \leftarrow (\alpha_+ + \alpha_-)/2$ ;
7:     if ( $\varphi'(\alpha) < 0$ ) then
8:        $\alpha_- \leftarrow \alpha$ ;
9:     else
10:     $\alpha_+ \leftarrow \alpha$ ;
11:   end if
12:   end while
13: end procedure
```

Quadratic approximation: we use the first order information to build a quadratic model m that approximates φ in a ball centered in α .

We may use the information we have about the function, since we know $\varphi(\alpha^-)$, $\varphi'(\alpha^-)$, $\varphi(\alpha^+)$ and $\varphi'(\alpha^+)$.

At this point we can write a model ($m(\alpha) = a\alpha^2 + b\alpha + c$) and specialize it with the information we have, via computing a linear system:

$$\begin{cases} a(\alpha^-)^2 + b(\alpha^-) + c = \varphi(\alpha^-) \\ a(\alpha^+)^2 + b(\alpha^+) + c = \varphi(\alpha^+) \\ 2a\alpha^- + b = \varphi'(\alpha^-) \\ 2a\alpha^+ + b = \varphi'(\alpha^+) \end{cases}$$

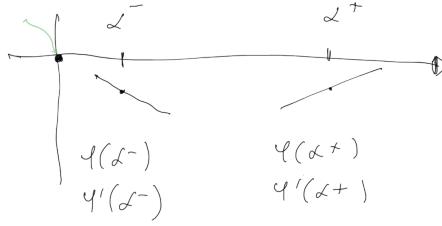


FIGURE 9.2: The information we have about function φ

Then we look for the stationary point of this function and that's the ball where I'm likely to find the root of the derivative.

Fact 9.1.4. *Let $\varphi : \mathbb{R} \rightarrow \mathbb{R}$ such that $\varphi \in \mathcal{C}^3$, then quadratic interpolation has convergence of order $1 < p < 2$ (superlinear).*

In Figure 9.3 we can observe a situation in which the hypotheses of Proposition 9.1.4 are not satisfied.

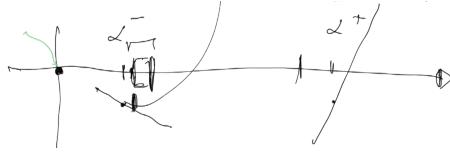


FIGURE 9.3: If the function is not \mathcal{C}^3 and one derivative is very big, then the range does not shrink much.

We would like to modify the formula to have at least linear convergence.

We can ensure to move not too close to one of the extremes, for example more than 10%.

It is possible to build a more complex model, i.e. a cubic function. Although this is pretty involved, it allows the convergence to get quadratic, which is better than linear.

Lecture 10

25th of October 2018

A. Frangioni

💡 Do you recall?

During last lecture we started dealing with algorithms for performing a line search for the minimum of φ , in particular we introduced first order algorithms.

Second order algorithms

Theorem 10.0.1. Given $f : \mathbb{R}^n \rightarrow \mathbb{R}$ such that $f \in \mathcal{C}^2$ and let $\varphi(\alpha) = \mathbf{x} + \alpha\mathbf{d}$, $\exists \varphi''(\alpha) = \mathbf{d}^T \nabla^2 f(\mathbf{x} + \alpha\mathbf{d}) \mathbf{d}$ and it is continuous.

Proof.

$$\begin{aligned}\frac{\partial \langle \nabla f(\mathbf{x} + \alpha\mathbf{d}), \mathbf{d} \rangle}{\partial \alpha} &= \frac{\partial \langle \mathbf{d}, \nabla f(\mathbf{x} + \alpha\mathbf{d}) \rangle}{\partial \alpha} \\ &= \frac{\partial (\nabla f(\mathbf{x} + \alpha\mathbf{d}) \cdot \mathbf{d})}{\partial \alpha} \\ &= \mathbf{d} \cdot \left(\frac{\partial \nabla f(\mathbf{x} + \alpha\mathbf{d})}{\partial \alpha} \cdot \frac{\partial (\mathbf{x} + \alpha\mathbf{d})}{\partial \alpha} \right) \\ &= \mathbf{d}^T \nabla^2 f(\mathbf{x} + \alpha\mathbf{d}) \cdot \mathbf{d}\end{aligned}$$

□

Since we are looking for a point where the derivative $\varphi'(\alpha) = 0$, we may use the second derivative to write a model and, assuming to trust the model, it can be studied.

Definition 10.0.1 (Model–Newton tangent method). *Our model, in this case is*

$$\varphi'(\alpha) \approx \varphi'(\alpha^k) + \varphi''(\alpha^k)(\alpha - \alpha^k) = 0 \text{ iff } \alpha = \alpha^k - \varphi'(\alpha^k)/\varphi''(\alpha^k)$$

In this context, solving $\varphi'(\alpha) = 0$ implies finding those α such that $\alpha = \alpha^k - \varphi'(\alpha^k)\varphi''(\alpha^k)$

Spiegazione: sono nel punto α^k e stimo $\varphi'(\alpha)$ come $\varphi'(\alpha^k) + \varphi''(\alpha^k)[\text{coefficiente angolare}] \cdot (\alpha - \alpha^k)$

ALGORITHM 10.0.1 Pseudocode for Newton method.

```

1: procedure LSNM( $\varphi'$ ,  $\varphi''$ ,  $\alpha$ ,  $\varepsilon$ )
2:   while ( $|\varphi'(\alpha)| > \varepsilon$ ) do
3:      $\alpha \leftarrow \alpha - \frac{\varphi'(\alpha)}{\varphi''(\alpha)}$ ;
4:   end while
5: end procedure

```

We need to understand when and why $\varphi''(\alpha) \neq 0$ and when and why this method converges. The following theorem formalizes the fact that if we start from a point α^0 which is close to the optimum we reach the optimum with quadratic speed.

Theorem 10.0.2. *Let $\varphi : \mathbb{R}^n \rightarrow \mathbb{R}$ and let $\varphi \in C^3$ and take $\alpha_* \in \mathbb{R}$ such that $\varphi'(\alpha_*) = 0$ and $\varphi''(\alpha_*) \neq 0$. $\exists \delta > 0$ s.t. if $\alpha^0 \in [\alpha_* - \delta, \alpha_* + \delta]$ then $\{\alpha^k\} \rightarrow \alpha_*$, with $p = 2$.*

Proof. We are in the hypothesis that the function φ is three times differentiable and we would like to prove that $\alpha^{k+1} - \alpha_* \rightarrow 0$

We want to compute how much the error is, if compared to the error at the previous iteration. Since (1) $\alpha^{k+1} := \alpha^k - \frac{\varphi'(\alpha^k)}{\varphi''(\alpha^k)}$ and (2) $\varphi'(\alpha_*) = 0$ and we get

$$\begin{aligned}
\alpha^{k+1} - \alpha_* &\stackrel{(1)}{=} \alpha^k - \frac{(\varphi'(\alpha^k))}{\varphi''(\alpha^k)} - \alpha_* \\
&= \alpha^k - \alpha_* - \frac{(\varphi'(\alpha^k))}{\varphi''(\alpha^k)} \\
&\stackrel{(2)}{=} \alpha^k - \alpha_* - \frac{(\varphi'(\alpha^k) - \varphi'(\alpha_*))}{\varphi''(\alpha^k)} \\
&= \frac{-\varphi''(\alpha^k) \cdot (\alpha_* - \alpha_k) - \varphi'(\alpha^k) + \varphi'(\alpha_*)}{\varphi''(\alpha^k)}
\end{aligned}$$

Let us write the second form of the first order Taylor's model centered in α^k (Definition 4.0.6, Equation (4.0.1))

$$\exists \beta \in [\alpha^k, \alpha_*] \text{ s.t. } \varphi'(\alpha_*) = \varphi'(\alpha^k) + \varphi''(\alpha^k)(\alpha_* - \alpha^k) + \varphi'''(\beta) \frac{(\alpha_* - \alpha^k)^2}{2}$$

and plug it into the above equation:

$$\begin{aligned}
\alpha^{k+1} - \alpha_* &= \frac{-\varphi''(\alpha^k) \cdot (\alpha_* - \alpha_k) - \varphi'(\alpha^k) + \varphi'(\alpha_*)}{\varphi''(\alpha^k)} \\
&= \frac{-\cancel{\varphi''(\alpha^k)} \cdot \cancel{(\alpha_* - \alpha_k)} - \cancel{\varphi'(\alpha^k)} + \cancel{\varphi'(\alpha^k)} + \cancel{\varphi''(\alpha^k)} \cdot \cancel{(\alpha_* - \alpha_k)} + \frac{1}{2}\varphi'''(\beta) \cdot (\alpha_* - \alpha^k)}{\varphi''(\alpha^k)} \\
&= \frac{\frac{1}{2}\varphi'''(\beta) \cdot (\alpha_* - \alpha^k)}{\varphi''(\alpha^k)}
\end{aligned}$$

We can say that the quantity $2\varphi''(\alpha^k)$ does not become too small and that the numerator $\varphi'''(\beta)$ does not become too big. This is proved since $\exists \delta > 0$ s.t. $\varphi''(\alpha) \geq k_2 > 0$ and also $|\varphi'''(\beta)| \leq k_1 < \infty$. We can go on bounding the difference between α^{k+1} and α_* as follows: for $\alpha, \beta \in [\alpha_* - \delta, \alpha_* + \delta]$

$$\begin{aligned}
|\alpha^{k+1} - \alpha_*| &\stackrel{(6)}{\leq} \frac{-\varphi'''(\beta)}{2\varphi''(\alpha^k)} \cdot (\alpha^k - \alpha_*)^2 \\
&\leq \underbrace{\left[\frac{k_1}{2k_2}\right]}_{\leq 1} (\alpha^k - \alpha_*)^2
\end{aligned} \tag{10.0.1}$$

In general, $\left[\frac{k_1}{2k_2}\right]$ may be very large, but it is multiplied by $(\alpha^k - \alpha_*)^2$, which means that if we start close enough to α^* the upper-bound still makes sense.

$$|\alpha^{k+1} - \alpha_*| \leq \underbrace{\left[\frac{k_1}{2k_2}\right]}_{\leq 1} (\alpha^k - \alpha_*) \cdot (\alpha^k - \alpha_*)$$

hence $|\alpha^{k+1} - \alpha_*| < |\alpha^k - \alpha_*|$ and this proves the theorem. \square

This approach is pretty accurate, but it is usually undesirable to devote substantial resources to finding a value of α to precisely minimize f . This is because the computing resources needed to find a more precise minimum along one particular direction could instead be employed to identify a better search direction.

The following class of algorithms circumvent the problem of the existence of derivatives, but in general allow to find a solution without computing derivatives at all.

★ Mantra

The more derivatives we have, the smallest number of points we need (second derivative \rightarrow two points, third derivative \rightarrow zero points). The opposite holds as well.

Zero order algorithms

Definition 10.0.2 (Unimodal function). *Let $f : \mathbb{R} \rightarrow \mathbb{R}$. We say that f is unimodal if for some value $m \in \mathbb{R}$ it is monotonically increasing for each $x \leq m$ and monotonically decreasing for any $x \geq m$ or vice-versa. Notice that any unimodal function admits only one maximum (minimum) value $f(m)$.*

Let us be given φ , we are looking for α_* within an error tolerance ε . Notice that if the function φ is not unimodal we have no guarantee that the interval that we are discarding does not contain the “deepest” minimum. We know only the function values in given points and we would like to shrink a candidate interval in the best way possible.

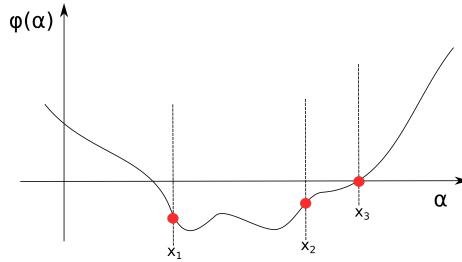


FIGURE 10.1: The candidate interval is $[x_1, x_2]$, since $\varphi(x_2) > \varphi(x_1)$ and we are allowed to exclude the interval $[x_3, +\infty)$ since the value in x_3 is bigger than $\varphi(x_2)$.

We propose an elegant solution via golden ratio.

Definition 10.0.3. *In mathematics, two quantities are in the **golden ratio** if their ratio is the same as the ratio of their sum to the larger of the two quantities. Formally, $\forall a, b \in \mathbb{R}$ such that $a > b > 0$,*

$$a + b : a = a : b$$

and we denote r_g such ratio.

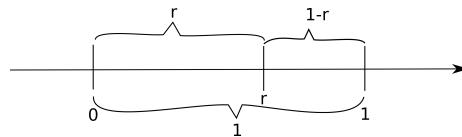


FIGURE 10.2: The relationship between r and $1 - r$ is $r : 1 = (1 - r) : r$.

Let us start from an interval $[\alpha_-, \alpha_+] \subseteq \mathbb{R}$, and let us take r as the inverse of the golden ratio $r = 1/\sqrt{5}-1/2 \approx 0.618$.

ALGORITHM 10.0.2 Pseudocode for zero-th order approach for local minimum detection.

```

1: procedure LSGRM( $\varphi, \alpha, \varepsilon$ )
2:    $\alpha_- \leftarrow 0;$ 
3:    $\alpha_+ \leftarrow \alpha;$ 
4:    $\alpha'_- \leftarrow (1 - r)\alpha;$ 
5:    $\alpha'_+ \leftarrow r\alpha;$ 
6:   while ( $\alpha_+ - \alpha_- > \varepsilon$ ) do ▷ not the same  $\varepsilon$ 
7:     if  $\varphi(v'_l) > \varphi(v'_r)$  then
8:        $\alpha_- \leftarrow \alpha'_-;$ 
9:        $\alpha'_- \leftarrow \alpha \leftarrow \alpha'_+;$ 
10:       $\alpha'_+ \leftarrow \alpha_+(1 - r)(\alpha_+ - \alpha_-);$ 
11:    else
12:       $\alpha_+ \leftarrow \alpha'_+;$ 
13:       $\alpha'_+ \leftarrow \alpha \leftarrow \alpha'_-;$ 
14:       $\alpha'_- \leftarrow \alpha_- + r(\alpha_+ - \alpha_-);$ 
15:    end if
16:   end while
17: end procedure

```

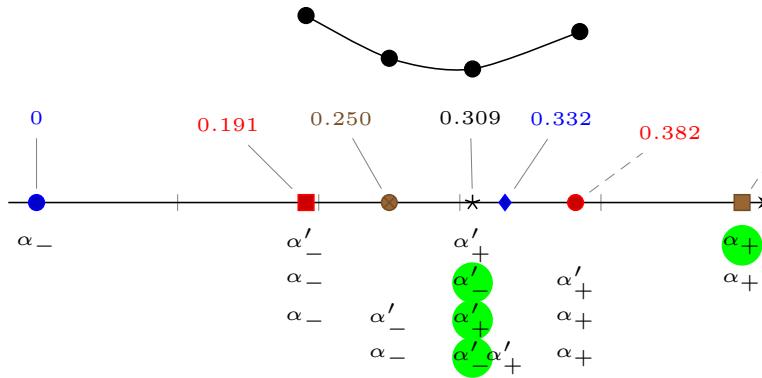


FIGURE 10.3: Plot of iterates of zero order algorithm for line search. The lines below the x axis represent the different values of α_- , α_+ , α'_- , α'_+ across different iterations, while the greed dot marks how the value α changes.

Example 10.0.1. Let us try Algorithm 10.0.2 on a toy example, displayed in Figure 10.3. Let $\alpha = 0$ and let $\varepsilon = 1$.

$$\begin{aligned}
\alpha_- &= 0 \\
\alpha_+ &= 0.5 \\
\alpha'_- &= r\alpha = (1 - 0.618) \cdot 0.5 = 0.191 \\
\alpha'_+ &= r \cdot \alpha = 0.618 \cdot 0.5 = 0.309
\end{aligned}$$

Since $0.5 - 0 < 1$ we enter the loop. According to the values of function φ in α_- and α_+ we enter the if branch:

$$\begin{aligned}
\alpha_- &= \alpha'_- = 0.191 \\
\alpha'_- &= \alpha'_+ = 0.309 \\
\alpha &= \alpha'_+ = 0.309 \\
\alpha'_+ &= \alpha_- + r \cdot (\alpha_+ - \alpha_-) = 0.191 + 0.618 \cdot (0.5 - 0.191) = 0.382
\end{aligned}$$

Provided that $0.5 - 0.191 < 1$ we enter the loop. According to the values of function φ in α_- and α_+ we enter the else branch:

$$\begin{aligned}
\alpha_+ &= \alpha'_+ = 0.382 \\
\alpha'_+ &= \alpha'_- = 0.309 \\
\alpha &= \alpha'_- = 0.309 \\
\alpha'_- &= \alpha_- + (1 - r) \cdot (\alpha_+ - \alpha_-) = 0.191 + 0.309 \cdot (0.382 - 0.191) = 0.25
\end{aligned}$$

Provided that $0.382 - 0.191 < 1$ we enter the loop. According to the values of function φ in α_- and α_+ we enter the if branch:

$$\begin{aligned}
\alpha_- &= \alpha'_- = 0.25 \\
\alpha'_- &= \alpha'_+ = 0.309 \\
\alpha &= \alpha'_+ = 0.309 \\
\alpha'_+ &= \alpha_- + r \cdot (\alpha_+ - \alpha_-) = 0.25 + 0.618 \cdot (0.382 - 0.25) = 0.332
\end{aligned}$$

Inexact line search

In the rest of this lecture we will present a line search method to determine the maximum amount to move along a given search direction, starting with a relatively large estimate of the step size for movement along the search direction, and iteratively shrinking the step size (“backtracking”) until a decrease of the objective function is observed that adequately corresponds to the decrease that is expected, based on the local gradient of the objective function.

Definition 10.0.4 (Armijo’s condition). *Let $\varphi : \mathbb{R}^+ \cup \{0\} \rightarrow \mathbb{R}$ such that $\varphi(\alpha) = f(\mathbf{x} + \alpha \mathbf{d})$. The **Armijo’s condition** selects those real values α such that the function value in those points ($\varphi(\alpha)$) is smaller than the value of a line which passes through 0 and is less steep than the tangent to the curve in such point.*

Formally, $\alpha \in \mathbb{R}$ satisfies **Armijo's condition** if for some control parameter $0 < m_1 < (\ll)1$ the following holds

$$\varphi(\alpha) \leq y_a(\alpha) = \varphi(0) + m_1 \alpha \varphi'(0) \quad (A)$$

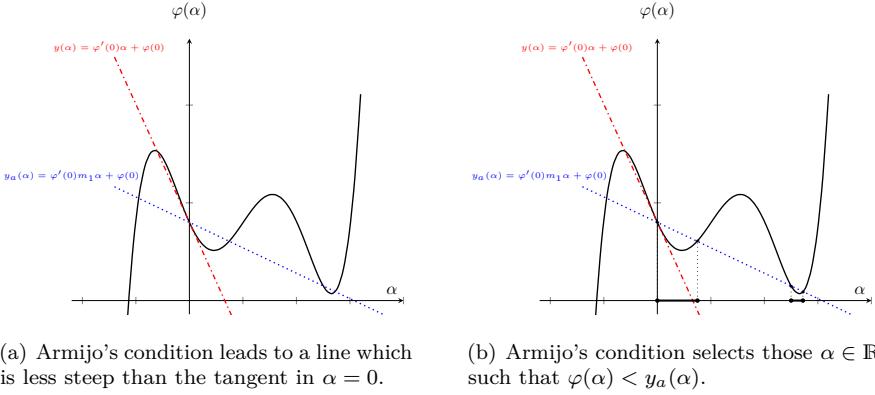


FIGURE 10.4: Graphical hint of the functioning of Armijo's condition.

Intuitively, provided that we are minimizing φ , $\varphi'(0) < 0$. This means that the slope of the line y_a ($m_1 \varphi'(0)$) is smaller in absolute value (hence bigger) than the slope of the tangent to the curve φ in 0, therefore the line y_a is less steep than y .

An attentive reader may notice that $\alpha \approx 0$ lead to values of $\varphi(\alpha) < y(\alpha)$, hence smaller than $y_a(\alpha)$, but this would lead to a very slow convergence. In order to overcome this issue one can decide to start from a large α and reduce it until it satisfies Armijo's condition (as shown in Algorithm 10.0.3) or introduce another algebraic constraint:

ALGORITHM 10.0.3 Pseudocode for backtracking line search.

```

1: procedure BLS( $\varphi, \varphi', \alpha, m_1, \tau$ )
2:   while ( $\varphi(\alpha) > \varphi(0) + m_1 \alpha \varphi'(0)$ ) do
3:      $\alpha \leftarrow \tau \alpha$ ; ▷ With  $\tau < 1$ 
4:   end while
5: end procedure

```

Definition 10.0.5 (Goldstein's condition). Let $\varphi : \mathbb{R}^+ \cup \{0\} \rightarrow \mathbb{R}$ such that $\varphi(\alpha) = f(\mathbf{x} + \alpha \mathbf{d})$ and let $m_1 \in \mathbb{R}$ be the Armijo's constant. The **Goldstein's condition** selects those real values α that satisfy Armijo's condition but also such that the function value in those points ($\varphi(\alpha)$) is bigger than the value of a line which passes through 0 and is less steep than the tangent to the curve in such point.

Formally, $\alpha \in \mathbb{R}$ satisfies the **Goldstein's condition** if for some control parameter $m_1 < m_2 < 1$ the following holds

$$\varphi(\alpha) \geq y_g(\alpha) = \varphi(0) + m_2\alpha\varphi'(0) \quad (G)$$

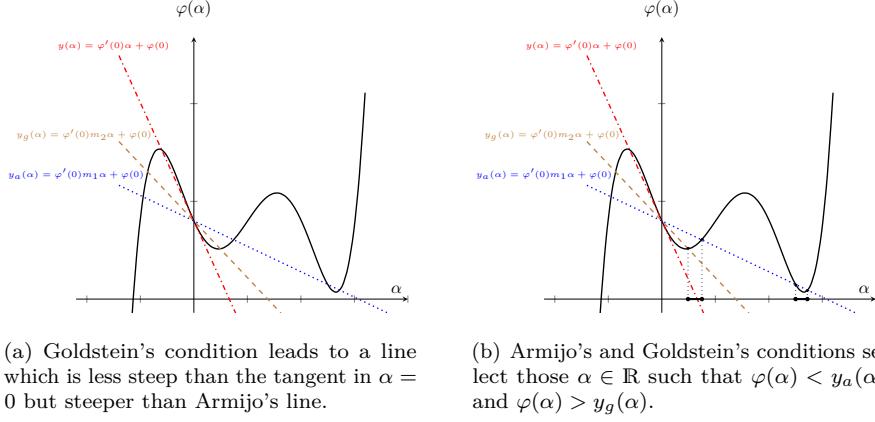


FIGURE 10.5: Conjunction of Armijo's and Goldstein's conditions.

It may happen that the points that satisfy both Goldstein's and Armijo's conditions does not contain a local minimum.

To circumvent this problem another condition comes to help us.

Definition 10.0.6 (Wolfe's condition). *Let $\varphi : \mathbb{R}^+ \cup \{0\} \rightarrow \mathbb{R}$ such that $\varphi(\alpha) = f(\mathbf{x} + \alpha\mathbf{d})$ and let $m_1 \in \mathbb{R}$ be the Armijo's constant. The **Wolfe's condition** selects those real values α that allow for a decrease in the slope (curvature condition), following the reasoning that close to the optimum the tangent line is almost horizontal.*

Formally, let $m_1 < m_3 < 1$

$$\varphi'(\alpha) \geq m_3\varphi'(0) \quad (W)$$

Figure 10.6(b) shows the intervals selected by Wolfe's rule. Provided that we are moving along a direction of negative curvature for f , we have that $\varphi'(\alpha) < 0$, therefore Wolfe's condition is satisfied also by those points α where the derivative is positive.

Sometimes, the curvature condition is modified to force the step length to stay in at least a broad neighborhood of a local minimizer or stationary point of the univariate function φ .

Provided that $\varphi'(0) < 0$, $|\varphi'(0)| = -\varphi'(0)$; follows a stronger variant of Wolfe's condition.

Definition 10.0.7 (Strong Wolfe's condition). *Let $\varphi : \mathbb{R}^+ \cup \{0\} \rightarrow \mathbb{R}$ such that $\varphi(\alpha) = f(\mathbf{x} + \alpha\mathbf{d})$ and let $m_1 \in \mathbb{R}$ be the Armijo's constant. The **strong***

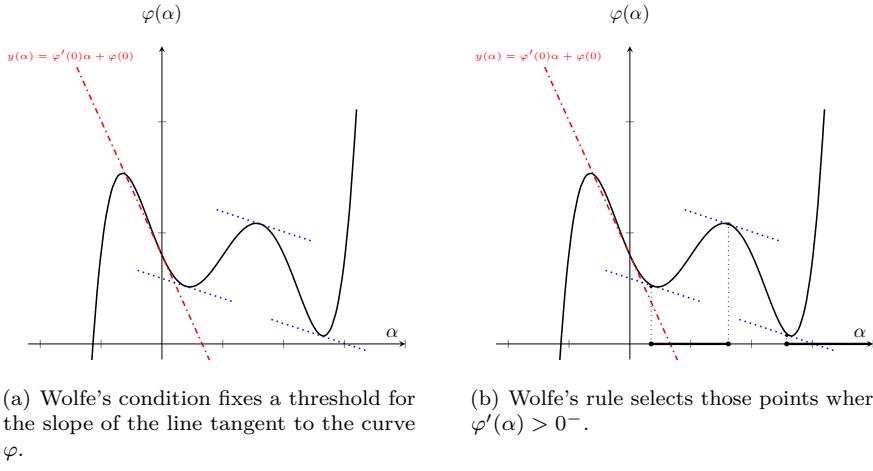


FIGURE 10.6: Wolfe's condition.

Wolfe's condition selects those real values α that allow for a decrease in the slope but prevent it to become big in absolute value. Formally, let $m_1 < m_3 < 1$

$$|\varphi'(\alpha)| \leq m_3 |\varphi'(0)| = -m_3 \varphi'(0) \quad (W')$$

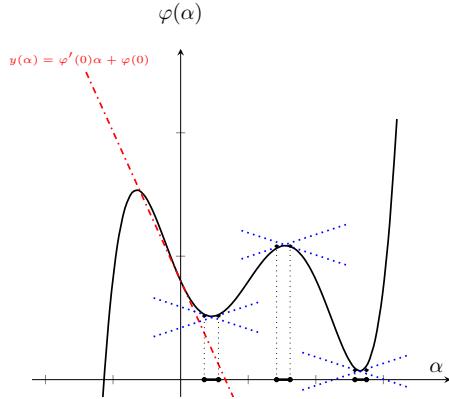


FIGURE 10.7: Strong Wolfe's condition.

Fact 10.0.3. Let $\varphi : \mathbb{R}^+ \cup \{0\} \rightarrow \mathbb{R}$ such that $\varphi(\alpha) = f(\mathbf{x} + \alpha \mathbf{d})$. If $\varphi'(\alpha) \geq 0$ and both Armijo's and Wolfe's (or Armijo's and strong Wolfe's) conditions hold then all local minima (maxima) are captured unless m_1 is too close to 1.

For a graphical hint of Proposition 10.0.3, see Figure 10.8, where we can

observe that if $m_1 \approx 1$, the line is pretty steep, therefore the intersection between Armijo's and Wolfe's regions is very small. In practice it is common to choose $m_1 \approx 0.0001$.

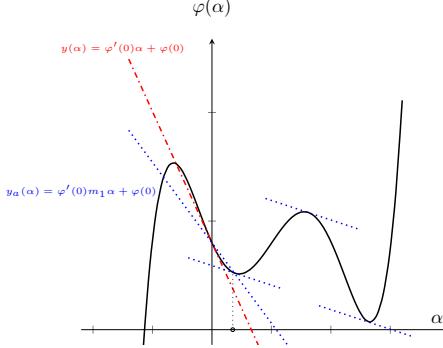


FIGURE 10.8: $m_1 \approx 1$ and the intersection between Armijo's and Wolfe's ranges.

The m_i are like the hyperparameters of machine learning. Less formally, if we choose an m_1 far enough from 1 everything works fine.

Theorem 10.0.4. Let $\varphi \in \mathcal{C}^1$ and $\varphi(\alpha)$ bounded below for $\alpha \geq 0$ then $\exists \alpha$ s.t. (A) \cap (W') holds.

Proof. Let us write Armijo's line: $y_a(\alpha) = \varphi(0) + m_1\varphi'(0)\alpha$. Let us define $d(\alpha) = y_a(\alpha) - \varphi(\alpha) = \varphi(0) + m_1\varphi'(0)\alpha - \varphi(\alpha)$, such that $d'(\alpha) = y'_a(\alpha) - \varphi'(\alpha) = m_1\varphi'(\alpha) - \varphi'(\alpha) = (m_1 - 1)\varphi'(\alpha)$. It goes without saying that $d(0) = 0$ and $d'(0) = \underbrace{(m_1 - 1)}_{<1} \underbrace{\varphi'(0)}_{<1} > 0$.

In $\alpha = 0$ the derivative of φ is negative, hence the function φ is decreasing. From the hypothesis we know that φ is bounded below, hence at a certain point it will start increasing its value until it will intersect with Armijo's line eventually. Let us call $\bar{\alpha} > 0$ the smallest value such that $d(\bar{\alpha}) = 0$. All the points $\alpha \in]0, \bar{\alpha}[$ satisfy Armijo's condition.

We are left with the task of proving that also strong Wolfe's condition holds for those $\alpha \in]0, \bar{\alpha}[$. 0 and $\bar{\alpha}$ are the two roots of the function d , so we can use Rolle's theorem, in order to prove that the function d has a stationary point in the interval $]0, \bar{\alpha}[$. Let us call the stationary point $\alpha^* \in]0, \bar{\alpha}[$, $d'(\alpha^*) = 0$ iff $\varphi'(\alpha^*) = -m_1\varphi'(0)$, therefore $|\varphi'(\alpha^*)| = m_1 \cdot |\varphi'(0)|$ and this proves Wolfe's rule, because $m_1 < m_3 < 1$, hence $|\varphi'(\alpha^*)| \leq m_3|\varphi'(0)|$. \square

How can we find such a point?

Fact 10.0.5. Let $\varphi : \mathbb{R}^+ \cup \{0\} \rightarrow \mathbb{R}$ such that $\varphi(\alpha) = f(\mathbf{x} + \alpha\mathbf{d})$ and let ∇f be L -Lipschitz. Then φ' is L -Lipschitz as well and L does not depend on x^i .

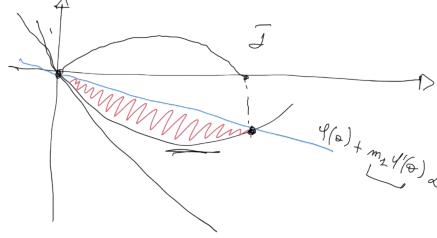


FIGURE 10.9: If φ is not going to $-\infty$ the Armijo's line (blue) and the function will meet in $\bar{\alpha} > 0$.

Proof.

$$\begin{aligned}
 |\varphi'(\alpha_1) - \varphi'(\alpha_2)| &= |\mathbf{d}^T \nabla f(\mathbf{x} + \alpha_1 \mathbf{d}) - \mathbf{d}^T \nabla f(\mathbf{x} + \alpha_2 \mathbf{d})| \\
 &= \mathbf{d}^T \underbrace{|\nabla f(\mathbf{x} + \alpha_1 \mathbf{d}) - \nabla f(\mathbf{x} + \alpha_2 \mathbf{d})|}_{\leq L \cdot |\mathbf{x} + \alpha_1 \mathbf{d} - (\mathbf{x} + \alpha_2 \mathbf{d})|} \\
 &\leq \mathbf{d}^T L \cdot |\mathbf{x} + \alpha_1 \mathbf{d} - (\mathbf{x} + \alpha_2 \mathbf{d})| \\
 &= \mathbf{d}^T L |\alpha_1 - \alpha_2| \mathbf{d} \\
 &\stackrel{*}{=} L \cdot |\alpha_1 - \alpha_2|
 \end{aligned}$$

where $\stackrel{*}{=}$ follows from the fact that \mathbf{d} has norm 1. \square

Fact 10.0.6. Let $\varphi : \mathbb{R}^+ \cup \{0\} \rightarrow \mathbb{R}$ such that $\varphi(\alpha) = f(\mathbf{x} + \alpha \mathbf{d})$, φ bounded below and ∇f L -Lipschitz. $\bar{\alpha} \in \mathbb{R}$ has the same behaviour of $\|\nabla f(\mathbf{x}^i)\|$. Formally,

$$\bar{\alpha} > (1 - m_1) \frac{\|\nabla f(\mathbf{x}^i)\|}{L}$$

Proof. According to Proposition 10.0.5, φ' is L -Lipschitz and

$$L(\bar{\alpha} - 0) \geq \varphi'(\bar{\alpha}) - \varphi'(0) > (1 - m_1)(-\varphi'(0)) \stackrel{*}{=} (1 - m_1)(\|\nabla f(\mathbf{x}^i)\|)$$

where $\stackrel{*}{=}$ follows from

$$\varphi'(\alpha) = \langle \nabla f(\mathbf{x} + \alpha \mathbf{d}), \mathbf{d} \rangle = \frac{-\nabla f(\mathbf{x} + \alpha \mathbf{d}) / \|\nabla f(\mathbf{x} + \alpha \mathbf{d})\|}{\|\nabla f(\mathbf{x} + \alpha \mathbf{d})\|^2}$$

$$\text{therefore } \bar{\alpha} > (1 - m_1) \frac{\|\nabla f(\mathbf{x}^i)\|}{L}.$$

\square

Now we can prove the following

Theorem 10.0.7. If $(A) \cap (W)$ holds $\forall i$ then either $\{f(\mathbf{x}^i)\} \rightarrow -\infty$ or $\{\|\nabla f(\mathbf{x}^i)\|\} \rightarrow 0$.

Proof. By contradiction, we assume that $\|\nabla f(\mathbf{x}^i)\|$ diverges. Formally, $-\varphi'(0) = \|\nabla f(\mathbf{x}^i)\| \geq \varepsilon > 0 \forall i$. Then

1. $(W) \implies \alpha^i \geq \bar{\alpha} > (1 - m_1) \frac{\|\nabla f(\mathbf{x}^i)\|}{L}$. Let $\delta := \frac{(1-m_1) \cdot \varepsilon}{L}$, then $\alpha^i \geq \delta > 0$;
2. $(A) \implies f(\mathbf{x}^{i+1}) \leq f(\mathbf{x}^i) - m_1 \alpha^i \|\nabla f(\mathbf{x}^i)\| \leq f(\mathbf{x}^i) - m_1 \delta \varepsilon$;
3. So $\{f(\mathbf{x}^i)\} \rightarrow -\infty$ (or $\{\|\nabla f(\mathbf{x}^i)\|\} \rightarrow 0$).

□

Backtracking is similar: for simplicity, $\alpha = 1$ (input)
 $\|\nabla f(\mathbf{x}^i)\| > \varepsilon \forall i \implies \bar{\alpha} > \delta > 0 \forall i$
 $h = \min\{k : \tau^{-k} \leq \delta\} \implies \alpha^i \geq \tau^{-h} > 0 \forall i \implies f(\mathbf{x}^{i+1}) \leq f(\mathbf{x}^i) - m_1 \tau^{-h} \varepsilon \implies \{f(\mathbf{x}^i)\} \rightarrow -\infty$ or ↘.

Lecture 11

8th of November 2018

A. Frangioni

11.1 Good practices for the project's design

In this lecture some good suggestions for implementing the gradient method in Matlab:

- When we have to implement a function the first question we need to ask ourselves is: “how many parameters should the function take?”;
- The interface needs to be designed, i.e. in the case of the gradient method we would like the user to be able to run it although he might not know what a good starting point could be;
- There might be some parameters with the same semantic of hyper-parameters, so they need to be adjusted in order to specify the algorithm;
- Another thing that should be taken into consideration is that sometimes, when measuring the number of times the code of the function has been executed we need to count also the calls to functions inside a single step;
- It is important to check the conditions that should be satisfied by the input data for the theoretical coherence. In case of not valid input an error message should be returned;
- Another important thing to be done is building a set of test functions, possibly limit case for the problem, to see how the algorithm works;
- `diff(f, x)` calculates the gradient of the function f in the variable x . If we want to get a simplified version, we may run `simplify(diff(f,x))`;
- A very nice tool to compute derivatives for arbitrary functions is **Adigator***

*<https://sourceforge.net/projects/adigator>

Lecture 12

14th of November 2018

A. Frangioni

★ Mantra

If you want better convergence, use a better model.

12.1 General descent methods

💡 Do you recall?

So far, we chose as direction for the step the direction where the decrease of the function is maximum ($\mathbf{d}^i = -\nabla f(\mathbf{x}^i)$) and for that descent direction we applied a line search for finding the optimal step size α . This was possible thanks to the **convergence argument** which says that in proximity of the stationary point of the linear model the norm of the gradient goes to 0. Formally, $\|\nabla f(\mathbf{x})\| \rightarrow 0$ when $\varphi'(0) \rightarrow 0$, where $\varphi'(\alpha) = <\nabla f(\mathbf{x}^i + \alpha^i \mathbf{d}^i), \mathbf{d}^i>$.

In this lecture we will propose a different choice for the moving direction \mathbf{d}^i , that keeps satisfying the convergence argument.

For example, let us take as direction a rotation of the opposite of the gradient, the value of φ' is then the cosine of the angle of the rotation times the opposite of the norm of the gradient. We have an infinite number of angles that we can choose, so we have a lot of flexibility.

Let us define the angle θ^i between the search direction \mathbf{d}^i and the steepest descent direction $-\nabla f(\mathbf{x}^i)$ as

$$\cos \theta^i = \frac{\langle \nabla f(\mathbf{x}), \mathbf{d}^i \rangle}{\|\nabla f(\mathbf{x}^i)\| \cdot \|\mathbf{d}^i\|}$$

Note that the angle between \mathbf{d}^i and the gradient should not be too close to 90° , otherwise the cosine would get approximately 0..

Theorem 12.1.1 (Zoutendijk's theorem). *Let us consider minimum problem (P) where the objective function $f \in \mathcal{C}^1$, ∇f is L-Lipschitz and f bounded below. Let us consider an iteration $\mathbf{x}^{i+1} = \mathbf{x}^i + \alpha^i \mathbf{d}^i$, where \mathbf{d}^i is a descent direction and $\alpha^i \in \mathbb{R}$ satisfies both Armijo's and Wolfe's conditions. Then*

$$\sum_{i=1}^{\infty} \cos^2(\theta^i) \|\nabla f(\mathbf{x}^i)\|^2 < \infty$$

 Do you recall?

A positive infinite sequence, whose corresponding series converges to a number, converges to 0 reasonably fast.

Therefore, if we choose an angle θ^i which is bounded below then the norm of the gradient goes to 0 quickly. More formally,

Fact 12.1.2. *Under the same hypotheses of Theorem 12.1.1, $\cos(\theta^i) \geq \varepsilon > 0 \implies \|\nabla f(\mathbf{x}^i)\| \rightarrow 0$.*

Proof. As stated in the recap, the following holds:

$$\cos^2(\theta^i) \|\nabla f(\mathbf{x}^i)\|^2 \rightarrow 0$$

therefore, one between $\cos(\theta^i)$ and $\|\nabla f(\mathbf{x}^i)\|^2$ should converge to zero, but it is no the cosine, because it is bounded below by $\varepsilon > 0$. \square

12.2 Newton's method

Newton's method attempts to solve the minimum problem (P) by constructing a sequence $\{\mathbf{x}^i\}$ from an initial guess (starting point) $\mathbf{x}^0 \in \mathbb{R}^n$ that converges towards a minimizer \mathbf{x}_* of f by using a sequence of second-order Taylor approximations of f around the iterates.

12.2.1 Convex case

In the next line we will explore Newton's method in the simplest case, when the function is perfectly convex (i.e. the Hessian is strictly positive definite).

Theorem 12.2.1. Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be a function such that $\nabla^2 f(\mathbf{x}^i) \succ 0$. Then the second order model in proximity of point \mathbf{x} ($Q_{\mathbf{x}^i}(\mathbf{y})$ for $\mathbf{y} \in \mathcal{B}(\mathbf{x}, \delta)$) admits a minimum.

Proof. Let us write \mathbf{x} instead of \mathbf{x}^i to ease notation. The second order Taylor's expansion of f around the point \mathbf{x} (recall Definition 4.0.7) is:

$$\begin{aligned} Q_{\mathbf{x}}(\mathbf{y}) &= f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} - \mathbf{x} \rangle + \frac{1}{2} (\mathbf{y} - \mathbf{x})^T \nabla^2 f(\mathbf{x}) (\mathbf{y} - \mathbf{x}) \\ &= f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} \rangle - \langle \nabla f(\mathbf{x}), \mathbf{x} \rangle + \\ &\quad + \frac{1}{2} \left(\mathbf{y}^T \nabla^2 f(\mathbf{x}) \mathbf{y} - 2\mathbf{x}^T \nabla^2 f(\mathbf{x}) \mathbf{y} + \mathbf{x}^T \nabla^2 f(\mathbf{x}) \mathbf{x} \right) \\ &= f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \mathbf{y} \rangle - \langle \nabla f(\mathbf{x}), \mathbf{x} \rangle + \\ &\quad + \frac{1}{2} \mathbf{y}^T \nabla^2 f(\mathbf{x}) \mathbf{y} - \mathbf{x}^T \nabla^2 f(\mathbf{x}) \mathbf{y} + \frac{1}{2} \mathbf{x}^T \nabla^2 f(\mathbf{x}) \mathbf{x} \end{aligned}$$

We are looking for the minimum of $Q_{\mathbf{x}}$, that is a stationary point. Let us differentiate $Q_{\mathbf{x}}$:

$$\begin{aligned} \frac{\partial Q_{\mathbf{x}}(\mathbf{y})}{\partial \mathbf{y}} &\stackrel{*}{=} \frac{\partial (\langle \nabla f(\mathbf{x}), \mathbf{y} \rangle + \frac{1}{2} \mathbf{y}^T \nabla^2 f(\mathbf{x}) \mathbf{y} - \mathbf{x}^T \nabla^2 f(\mathbf{x}) \mathbf{y})}{\partial \mathbf{y}} \\ &= \frac{\partial (\langle \nabla f(\mathbf{x}) + \frac{1}{2} \mathbf{y}^T \nabla^2 f(\mathbf{x}) - \mathbf{x}^T \nabla^2 f(\mathbf{x}), \mathbf{y} \rangle)}{\partial \mathbf{y}} \\ &= \nabla f(\mathbf{x}) - \mathbf{x}^T \nabla^2 f(\mathbf{x}) + \frac{1}{2} \nabla^2 f(\mathbf{x}) \mathbf{y} \end{aligned}$$

where $\stackrel{*}{=}$ is given by the fact that $f(\mathbf{x})$, $-\langle \nabla f(\mathbf{x}), \mathbf{x} \rangle$ and $\frac{1}{2} \mathbf{x}^T \nabla^2 f(\mathbf{x}) \mathbf{x}$ are constant terms with respect to \mathbf{y} . The equality obtained above allows to say that \mathbf{y} is a stationary point iff

$$\begin{aligned} \frac{1}{2} \nabla^2 f(\mathbf{x}) \mathbf{y} &= \nabla f^2(\mathbf{x}) \mathbf{x} - \nabla f(\mathbf{x}) \\ \mathbf{y} &= 2 \cdot \left(\mathbf{x} - [\nabla^2 f(\mathbf{x})]^{-1} \nabla f(\mathbf{x}) \right) \end{aligned}$$

□

The moving direction of the gradient descent algorithm using Newton's method is obtained taking the opposite of the inverse of the Hessian times the gradient of the function. Formally, $\mathbf{d}^i = -[\nabla^2 f(\mathbf{x}^i)]^{-1} \nabla f(\mathbf{x}^i)$.

An attentive reader may notice that Theorem 12.2.1 holds only if the Hessian is invertible.

Moreover, even if the Hessian is invertible, the computation of the descent direction requires to solve a linear system. A way to circumvent this problem is putting the gradient to 0, so we can write the Taylor form of the gradient and solve a linear equation which is $\nabla f(\mathbf{x}) \approx \nabla f(\mathbf{x}^i) + \nabla^2 f(\mathbf{x}^i)(\mathbf{x} - \mathbf{x}^i)$.

Theorem 12.2.2. Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ s.t. $f \in \mathcal{C}^3$ and let \mathbf{x}_* be a stationary point such that $\nabla f(\mathbf{x}_*) = 0$ and $\nabla^2 f(\mathbf{x}_*) \succ 0$. Then $\exists \mathcal{B}(\mathbf{x}_*, r)$ s.t. $\mathbf{x}^1 \in \mathcal{B}$ and $\{\mathbf{x}^i\} \rightarrow \mathbf{x}_*$ quadratically.

Notice that Theorem 12.2.2 states that the descent direction computed through Newton's method is correct both close and far from the minimum point.

An attentive reader may notice that the scalar product between the function value at the i -th iterate $f(\mathbf{x}^i)$ and the descent direction \mathbf{d}^i should be negative but also not too close to 0. Such condition is ensured when the function f is such that $uI \preceq \nabla^2 f \preceq LI$, which implies that the function is strongly convex, in other words that the eigenvalues of the Hessian do not get too close to zero (in terms of machine precision).

Theorem 12.2.3 (Global convergence of Newton's method). Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ s.t. $f \in \mathcal{C}^3$ and such that satisfies $uI \preceq \nabla^2 f \preceq LI$ and $\cos \theta^i = \frac{\langle \nabla f(\mathbf{x}^i), \mathbf{d}^i \rangle}{\|\nabla f(\mathbf{x}^i)\| \cdot \|\mathbf{d}^i\|}$. Then Newton's method converges globally.

Proof.

Aggiungere la ref alla variational characterization nella parte di Numerical Methods

The variational characterization of the eigenvalues and the definition of \mathbf{d}^i ($\mathbf{d}^i := -[\nabla^2 f(\mathbf{x}^i)]^{-1} \nabla f(\mathbf{x}^i) \iff \nabla^2 f(\mathbf{x}^i) \mathbf{d}^i = -\nabla f(\mathbf{x}^i)$) leads to the following upper-bounds:

1.

$$\mathbf{d}^{i^T} \nabla f(\mathbf{x}^i) = -(\mathbf{d}^i)^T \nabla^2 f(\mathbf{x}^i) \mathbf{d}^i \leq -\lambda_{\min} \|\mathbf{d}^i\|^2$$

2.

$$\|\nabla f(\mathbf{x}^i)\| = \|\nabla^2 f(\mathbf{x}^i) \mathbf{d}^i\| \leq \|\nabla^2 f(\mathbf{x}^i)\| \cdot \|\mathbf{d}^i\| = \lambda_{\max} \|\mathbf{d}^i\|$$

From the definition of $\cos \theta^i = \frac{\langle \nabla f(\mathbf{x}^i), \mathbf{d}^i \rangle}{\|\nabla f(\mathbf{x}^i)\| \cdot \|\mathbf{d}^i\|}$ we get

$$\cos(\theta^i) \leq -\lambda_{\min}/\lambda_{\max} \leq -u/L$$

□

Theorem 12.2.4. Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ s.t. $f \in \mathcal{C}^3$ and such that satisfies $uI \preceq \nabla^2 f \preceq LI$ and $\cos(\theta^i) \leq -\lambda_{\min}/\lambda_{\max} \leq -u/L$. Then the method not only converges, but we also have that from some iteration onward, $\alpha^i = 1$ always satisfies Armijo's condition.

Proof.

$$\begin{aligned}
f(\mathbf{x}^i + \mathbf{d}^i) &= f(\mathbf{x}^i) + \langle \nabla f(\mathbf{x}^i), \mathbf{d}^i \rangle + \frac{1}{2} \cdot (\mathbf{d}^i)^T [\nabla^2 f(\mathbf{x}^i)] \mathbf{d}^i + R_3(\|\mathbf{d}^i\|) \\
&= f(\mathbf{x}^i) + (\mathbf{d}^i)^T \nabla f(\mathbf{x}^i) + \frac{1}{2} \cdot (\mathbf{d}^i)^T [\nabla^2 f(\mathbf{x}^i)] \mathbf{d}^i + R_3(\|\mathbf{d}^i\|) \\
&= f(\mathbf{x}^i) - \nabla f(\mathbf{x}^i)^T \left([\nabla^2 f(\mathbf{x}^i)]^{-1} \right)^T \nabla f(\mathbf{x}^i) + \\
&\quad + \frac{1}{2} \nabla f(\mathbf{x}^i)^T \left([\nabla^2 f(\mathbf{x}^i)]^{-1} \right)^T \cancel{\nabla^2 f(\mathbf{x}^i)} \cancel{[\nabla^2 f(\mathbf{x}^i)]^{-1}} \nabla f(\mathbf{x}^i) + R_3(\|\mathbf{d}^i\|) \\
&\stackrel{*}{=} f(\mathbf{x}^i) - \nabla f(\mathbf{x}^i)^T [\nabla^2 f(\mathbf{x}^i)]^{-1} \nabla f(\mathbf{x}^i) + \\
&\quad + \frac{1}{2} \nabla f(\mathbf{x}^i)^T [\nabla^2 f(\mathbf{x}^i)]^{-1} \nabla f(\mathbf{x}^i) + R_3(\|\mathbf{d}^i\|) \\
&= f(\mathbf{x}^i) - \frac{1}{2} \cdot \nabla f(\mathbf{x}^i)^T \cdot \underbrace{[\nabla^2 f(\mathbf{x}^i)]^{-1} \nabla f(\mathbf{x}^i)}_{\mathbf{d}^i} + R_3(\|\mathbf{d}^i\|) \\
&= f(\mathbf{x}^i) - \frac{1}{2} \langle \nabla f(\mathbf{x}^i), \mathbf{d}^i \rangle + R_3(\|\mathbf{d}^i\|)
\end{aligned} \tag{12.2.1}$$

where $\stackrel{*}{=}$ follows from Corollary 4.0.3 ($f \in \mathcal{C}^2 \Rightarrow \nabla^2 f \in S(n, \mathbb{R})$) and the fact that the inverse of a symmetric matrix (if it exists) is a symmetric matrix as well. \square

It can be proved that the convergence is superlinear.

If we start with a step size of 1, we end up in a situation in which the line search isn't computed when we are close to the minimum.

This works under the assumption that the eigenvalues are bounded both above and below (deriving from the bounds on the Hessian).

12.2.2 Interpretation of Newton's method

Riscrivere meglio

Let us consider Newton's method from a different point of view. We can construct a different space where the gradient method coincides with Newton's method. Given R which is not singular, we make a variable change ($\mathbf{y} = R\mathbf{x}$) — which is possible given that R is non singular — and we get $f_{\mathbf{y}}(\mathbf{y}) = \frac{1}{2}\mathbf{y}^T I\mathbf{y} + qR^{-1}\mathbf{y}$, which has as an Hessian the identity matrix, which is the optimal matrix for convergence in Newton's method.

Formally, the descending direction $\mathbf{d}_{\mathbf{y}}$ is computed as follows: $\mathbf{d}_{\mathbf{y}} = -\nabla f_{\mathbf{y}}(\mathbf{y}) = -\mathbf{y} - R^{-1}q$. Since we chose 1 as step size we obtain that $\nabla f_{\mathbf{y}}(\mathbf{y} + \mathbf{d}_{\mathbf{y}}) = \nabla f_{\mathbf{y}}(\mathbf{y} - \mathbf{y} - R^{-1}\mathbf{q}) = \nabla f_{\mathbf{y}}(-R^{-1}\mathbf{q}) = 0$.

It takes only one iteration, because all the eigenvalues are 1 so the the ratio between the greatest and the smallest is 1 and the subtraction is 0.

If we do the inverse operation ($\mathbf{x} = R^{-1}\mathbf{y}$) to the direction we get the direction in \mathbf{x} variable.

Problem:

We made a lot of assumptions on the Hessian, without the ones there's no guarantee that we are moving on a descending direction. How can we relax these constraints?

12.2.3 Non convex case

Saying that the function f is not convex is equivalent to say that the Hessian is indefinite. In such case we propose to compute the descent direction (previously defined as $\mathbf{d}^i = -[\nabla^2 f(\mathbf{x}^i)]^{-1} \nabla f(\mathbf{x}^i)$) substituting the inverse of the Hessian with a matrix H' which is strictly positive definite ad has more or less the same properties of the Hessian.

For example, we can build H^i summing to the Hessian a multiple of the identity matrix: $H^i = \nabla^2 f(\mathbf{x}^i) + \varepsilon^i I \succ 0$. The sequence $\{H^i\}$ is supposed to converge to a matrix H' that is strictly positive definite.

A proper choice for ε should ensure that all the eigenvalues are at least δ , for an “appropriately chosen smallish δ ”. Formally,

$$\varepsilon = \max\{0, \delta - \lambda_{\min}\}$$

The real value δ should not be too small for both numerical (any double $\leq 1e-16$ is treated as 0 by computers) and algorithmic (the smaller $\lambda_{\min}(\nabla^2 f(\mathbf{x}^i) + \varepsilon I)$ the more elongated the axes of $S(Q_{\mathbf{x}^i}, \cdot)$) reasons.

It can be proved that the ε we chose is the solution to an optimization problem: $\min\{\|H - \nabla^2 f(\mathbf{x}^i)\| \mid H \succeq \delta I\}$. The choice for δ in the Matlab code is 10^{-6} .

Observation 12.2.1. Note that these constraints are important and we will get back to them later on in the course.

Could we use a different norm instead of the 2-norm? Yes, for example we can use Frobenius's norm, changing a bit the algorithm, but we still get convergence. We would need to solve $\min\{\|H - \nabla^2 f(\mathbf{x}^i)\|_F \mid H \succeq \delta I\}$, which is performed in two steps:

1. compute spectral decomposition $\nabla^2 f(\mathbf{x}^i) = H \Lambda H^T$
2. $H^i = H \bar{\Lambda} H^T$ with $\bar{\gamma}^i = \max\{\lambda^i, \delta\}$

In both cases, $\{\mathbf{x}^i\} \rightarrow \mathbf{x}_*$ with $\nabla^2 f(\mathbf{x}_*) \succeq \delta I$, which implies $\varepsilon^i = 0$ and $H^i = \nabla^2 f(\mathbf{x}^i)$ eventually. It holds also that we have superlinear convergence if “ H^i looks like $\nabla^2 f(\mathbf{x}^i)$ along \mathbf{d}^i ”, formally $\lim_{i \rightarrow \infty} \|(H^i - \nabla^2 f(\mathbf{x}^i))\mathbf{d}^i\| \|\mathbf{d}^i\| = 0$.

Computational complexity

We still need to compute eigenvalues, which takes $O(n^3)$, which is too much if we are in multidimensional spaces.

As a closing observation we want to stress that Newton's method is very fast to converge to a local minimum, but this behaviour may represent a problem, because the algorithm may miss global minima.

Lecture 13

16th of November 2018

A. Frangioni

What happens when the Hessian isn't strictly positive definite? If there are some negative eigenvalues, I can desume that there are some directions in which the function goes to $-\infty$. The model has no minimum, unless we restrict to a **compact set**.

In particular, we may decide to restrict to a part of the space where we can trust our model, which is called **trust region**.

Finding such a region is a NP-hard problem, if we don't restrict to a ball.

Definition 13.0.1 (Karush-Khun-Tucker conditions). *Any optimal solution of the problem \mathbf{x}^{i+1} must satisfy that $\exists \lambda \geq 0$ s.t.*

KARUSH: $[H^i + \lambda I]\mathbf{x}^{i+1} = -\nabla f(\mathbf{x}^i)$ [linear];

KUHN: $H^i + \lambda I \succeq 0$ [semidefinite];

TUCKER: $\lambda(r - \|\mathbf{x}^{i+1}\|) = 0$ [nonlinear].

Where the last condition means that λ has to be 0, unless the solution we get is exactly on the border of the ball.

What's the difference between this approach and what we used to do before?
We have two different cases:

- $\|\mathbf{x}^{i+1}\| < r \implies \lambda = 0 \implies$ normal Newton step (\mathcal{T} has no effect);
- $\lambda > 0 (= \text{small radius } r) \implies$ like in line search with $\varepsilon^i = \lambda$.

The problem is computing these systems of equations taking less than $O(n^3)$.

Key idea

We don't need to compute the Hessian, we can use the first order information to infer things on the second order matrix, although we don't really need to compute the Hessian.

13.1 Quasi-newton methods

At a step we have: $m^i(x) = \nabla f(\mathbf{x}^i)(x - \mathbf{x}^i) + \frac{1}{2}(x - \mathbf{x}^i)^T H^i(x - \mathbf{x}^i)$, $\mathbf{x}^{i+1} = \mathbf{x}^i + \alpha^i d^i$

At the next step we recompute the gradient in \mathbf{x}^{i+1} and the matrix H^{i+1} :
 $m^{i+1}(x) = \nabla f(\mathbf{x}^{i+1})(x - \mathbf{x}^{i+1}) + \frac{1}{2}(x - \mathbf{x}^{i+1})^T H^{i+1}(x - \mathbf{x}^{i+1})$

How should H^i be chosen? It should satisfy the following:

1. positive definite ($H^{i+1} \succ 0$);
2. we know the gradient in the previous point, we know the gradient in the current point, we construct H^{i+1} such that the model works: $\nabla m^{i+1}(\mathbf{x}^i) = \nabla f(\mathbf{x}^i)$;
3. $\|H^{i+1} - H^i\|$ is “small”.

This new model isn't too different from the previous one, because of the third property. Or equivalently $H^{i+1}(\mathbf{x}^{i+1} - \mathbf{x}^i) = \nabla f(\mathbf{x}^{i+1}) - \nabla f(\mathbf{x}^i)$, which we call **secant equation** and we denote (S).

To ease notation we define s^i such that: $s^i = \mathbf{x}^{i+1} - \mathbf{x}^i = \alpha^i d^i$ and $y^i = \nabla f(\mathbf{x}^{i+1}) - \nabla f(\mathbf{x}^i)$.

s^i is chosen, while y^i is decided by the function.

In order to have a matrix H^i that satisfies the first two condition we could check that $H^{i+1}s^i = y^i$, because this implies $s^i y^i = (s^i)^T H^{i+1} s^i$ and this implies 1. and 2., hence we obtain the **curvature condition** $s^i y^i > 0$.

Theorem 13.1.1. *Wolf condition implies $s^i y^i > 0$, using the notation we introduced: (W) \implies (C).*

Proof.

$$\begin{aligned} \varphi'(\alpha^i) &= \nabla f(\mathbf{x}^{i+1})d^i \geq m_3 \varphi'(0) = m_3 \nabla f(\mathbf{x}^i)d^i \\ &\Downarrow \\ (\nabla f(\mathbf{x}^{i+1}) - \nabla f(\mathbf{x}^i))d^i &\geq (m_3 - 1)\varphi'(0) > 0 \end{aligned}$$

□

Observation 13.1.1. *We may observe that this theorem implies that if we perform Armijo Wolf exact line search condition (C) can always be satisfied.*

13.1.1 Davidson-Fletcher-Powell

How can we choose a H^i that satisfies the three conditions enumerated above? Taking $H^{i+1} = \text{argmin}\{\|H - H^i\| : H \in (S) \text{ and } H \succeq 0\}$ is a good idea and for this minimum problem holds the following:

Theorem 13.1.2 (Davidson-Feltcher-Powell). *The new matrix is obtained at each step constructing a rank two matrix, obtained from H^i as a rank two correction, as follows: $H^{i+1} = (I - \rho^i y^i (s^i)^T) H^i (I - \rho^i s^i (y^i)^T) + \rho^i y^i (y^i)^T$*

Let us denote $B^i = H^{i-1}$. At any step we need to compute $B^{i+1} = (H^{i+1})^{-1}$, because we need to solve the system. We have some formulas that give us a way to compute $(H^{i+1})^{-1}$ from H^{i-1} .

Theorem 13.1.3 (Sherman-Morrison-Woodbury). *The inverse of a matrix of the form $A + ab^T$ has the following shape: $[A + ab^T]^{-1} = \frac{A^{-1} - A^{-1}ab^TA^{-1}}{1 - b^TA^{-1}a}$.*

Observation 13.1.2. *From Theorem 13.1.3 we can conclude that $B^{i+1} = \frac{B^i + \rho^i s^i (s^i)^T - B^i y^i (y^i)^T B^i}{(y^i)^T B^i y^i}$.*

It's important to notice that this operation has a cost of $O(n^2)$.

We can do better, in terms of computational complexity.

13.1.2 Broyden-Fletcher-Goldfarb-Shanno

We can use directly B^i , the inverse of H^i . Write (S) for B^{i+1} : $s^i = B^{i+1}y^i \implies B^{i+1} = \arg\min \{\|B - B^i\| : \dots\}$.

$$B^{i+1} = B^i + \rho^i [(1 + \rho^i (y^i)^T B^i y^i) s^i (s^i)^T - (B^i y^i (s^i)^T + s^i (y^i)^T B^i)]$$

This formula proves to be more stable than the other one.

This method takes $O(n^2)$.

The two B^i 's, obtained from DFP and BFGS, are different although both sensible, but we can use a convex combination of the two.

Observation 13.1.3. *How can we choose H^1 ? The value we choose will make a difference in the results, at least for the first steps.*

Let us see a couple of choices for B^1 :

- Scalar multiples of identity, but how to choose the scalar?
- Compute the gradient in every direction and approximate H . This will cost $O(n^3)$, but it should be done only once.

Let us compute the space needed to store the B^i 's: order of n^2 is still a lot. What happens if we restrict to working with information of the last k operations?

13.1.3 Poorman's approach

At each step we only consider B^{i-k} and k rank one operations. This operations cost n each, and we have k lines. The problem is that B^{i-k} takes $O(n^2)$ space. We can optimize if we choose B^{i-k} to be simpler, say a multiple of the identity, or finite difference of the gradient. Then the space complexity is $O(kn)$.

I need to tune the algorithm to find the right k which gives me enough precision and also keeps the computational cost low.

Final observation of quasi Newton methods

We may notice that this variation of Newton method doesn't get trapped in local minima, as Newton method did. In the end, the fact that quasi Newton isn't that precise at the beginning may be a good feature.

13.2 Conjugate gradient method

💡 Do you recall?

In the gradient method, the angle between two consecutive directions is exactly 90° , as can be seen in Figure 13.1.

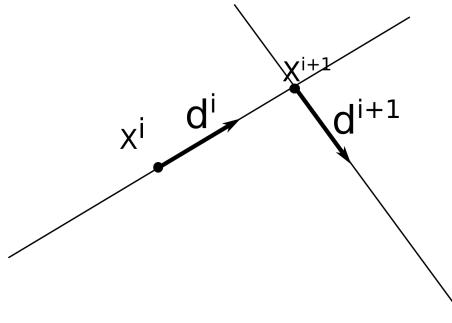


FIGURE 13.1: Geometric idea on how the new direction is chosen.

We would like to take into account not only the subspace spanned by d^{i+1} but we would like to optimize over larger and larger subspaces (spanned by d^i and d^{i+1}).

Definition 13.2.1 (Q-conjugate). Let v and w be vectors in \mathbb{R} . We say that v and w are ***Q-conjugate*** if $(v)^T Q w = 0$.

We would like pick a direction to be Q -conjugate with all the previous iterations. The point is that we can't take into account all the previous directions, but we will see that we only need the previous direction to obtain all the information we need.

ALGORITHM 13.2.1 Pseudocode for conjugate gradient method for quadratic functions.

```

1: procedure CGQ( $Q, q, x, \varepsilon$ )
2:    $d^- \leftarrow 0;$ 
3:   while ( $\|\nabla f(x)\| > \varepsilon$ ) do
4:     if ( $d^- = 0$ ) then
5:        $d \leftarrow -\nabla f(x);$ 
6:     else
7:        $\beta = (\nabla f(x)^T Q d^-) / ((d^-)^T Q d^-);$ 
8:        $d \leftarrow -\nabla f(x) + \beta d^-;$ 
9:     end if
10:     $\alpha \leftarrow (\nabla f(x)^T d) / (d^T Q d);$ 
11:     $x \leftarrow x + \alpha d;$ 
12:     $d^- \leftarrow d;$ 
13:   end while
14: end procedure

```

The number of iterations needed to converge is proportional to the clusterization of the eigenvalues of the matrix Q .

The algorithm that was presented is for quadratic functions, but the same algorithm works for non quadratic function as well, as long as we change the formula for β .

The pseudocode of Algorithm 13.2.2 is referred to Fletcher-Reeves definition of β^i $\beta^i = \|\nabla f(\mathbf{x}^i)\| / \|\nabla f(\mathbf{x}^{i-1})\|^2$.

This algorithm converges in at most n iterations.

ALGORITHM 13.2.2 Pseudocode for conjugate gradient method for arbitrary functions

```

1: procedure CGA( $Q, q, x, \varepsilon$ )
2:    $\nabla f^- = 0;$ 
3:   while ( $\|\nabla f(x)\| > \varepsilon$ ) do
4:     if ( $\nabla f^- = 0$ ) then
5:        $d \leftarrow -\nabla f(x);$ 
6:     else
7:        $\beta = \|\nabla f(\mathbf{x}^i)\|^2 / \|\nabla f^-\|^2;$ 
8:        $d \leftarrow -\nabla f(x) + \beta d^-;$ 
9:     end if
10:     $\alpha \leftarrow \text{AWLS}(f(x + \alpha d));$ 
11:     $x \leftarrow x + \alpha d;$ 
12:     $d^- \leftarrow d;$ 
13:     $\nabla f^- \leftarrow \nabla f(x);$ 
14:   end while
15: end procedure

```

We have three different formulas for β_i , which coincide in the quadratic case.

1. Polak-Ribière: $\beta^i = \frac{[\nabla f(\mathbf{x}^i)^T (\nabla f(\mathbf{x}^i) - \nabla f(\mathbf{x}^{i-1}))]}{\|\nabla f(\mathbf{x}^{i-1})\|^2}$
2. Hestenes-Stiefel: $\beta^i = \frac{\nabla f(\mathbf{x}^i)^T (\nabla f(\mathbf{x}^i) - \nabla f(\mathbf{x}^{i-1}))}{(\nabla f(\mathbf{x}^i) - \nabla f(\mathbf{x}^{i-1}))^T d^{i-1}}$
3. Dai-Yuan: $\beta^i = \frac{\|\nabla f(\mathbf{x}^i)\|^2}{(\nabla f(\mathbf{x}^i) - \nabla f(\mathbf{x}^{i-1}))^T d^{i-1}}$

Some of these algorithms require some hypothesis on the function in order for the conjugate method to converge.

1. Fletcher-Reeves requires $m_1 < m_2 < \frac{1}{2}$ for $(A) \cap (W')$ to work;
2. $(A) \cap (W') \not\Rightarrow d^i$ of Polak-Ribière is of descent, unless $\beta_{PR}^i = \max\{\beta^i, 0\}$.

Sometimes it's important to restart from scratches if the algorithm isn't converging, because many bad choices may lead to a bad result.

The idea of taking the gradient and modify it instead of multiplying by a factor, adding the previous direction.

It's possibile to design hybrids between quasi-Newton and conjugate method.

Lecture 14

22nd of November 2018

A. Frangioni

14.1 Deflected gradient methods

The idea behind this family of algorithms, is to determine the next position x^{i+1} using the gradient and summing to it something else that gives us more information.

This kind of algorithms work also in cases in which the gradient isn't continuous.

This methods use the information about the previous iterations without exploiting properties about the second order derivative.

14.1.1 Heavy ball gradient method

The intuition behind this algorithm may be expressed through the following metaphor: an object is moving in the space and it's subject to a force. We can observe that the heavier the object, the stronger should be the force imposed in order to make it describe a certain trajectory.

In this interpretation, we may define the $(i + 1)$ -th iteration as

$$x^{i+1} \leftarrow x^i - \alpha^i \nabla f(x^i) + \beta^i (x^i - x^{i-1}),$$

where β^i is called **momentum**, x^i **heavy** and $\nabla f(x^i)$ **force**.

This isn't a descent algorithm: we are not choosing a direction and doing line search. We have no guarantee that the value of the function after one iteration will be smaller than the previous one.

First thing to do is to choose α^i and β^i properly.

We can prove for some cases that these methods are better than gradient method, although they aren't as good as Newton or quasi Newton. Their strength resides in their simplicity though.

Notice that if the smaller eigenvalue isn't zero (i.e. quadratic case) we have a close formula to choose α and β independently from the iteration:

$$\alpha = \frac{4}{(\sqrt{\lambda^1} + \sqrt{\lambda^n})^2}, \quad \beta = \max \left\{ \left| 1 - \sqrt{\alpha \lambda^n} \right|, \left| 1 - \sqrt{\alpha \lambda^1} \right| \right\}^2$$

We may observe that the step we take is something that goes like $\frac{1}{L}$, where L is the Lipschitz constant, since λ_n is very small. With these choices the rate is the following. We observe that in the gradient the rate is the same, although there aren't the square roots

$$\|x^{i+1} - x_*\| \leq \left(\frac{\sqrt{\lambda^1} - \sqrt{\lambda^n}}{\sqrt{\lambda^1} + \sqrt{\lambda^n}} \right) \|x^i - x_*\|$$

An alternative idea could be choosing β^i and finding α^i using line search. A possible issue is that we don't know if we are moving along a descending direction, but in this method it is perfectly acceptable not to make any movement at a single step (notice that in gradient method if one step has size 0 then we will not move anymore).

β^i is seen as an hyperparameter, hence its value is tuned rerunning the algorithm several times.

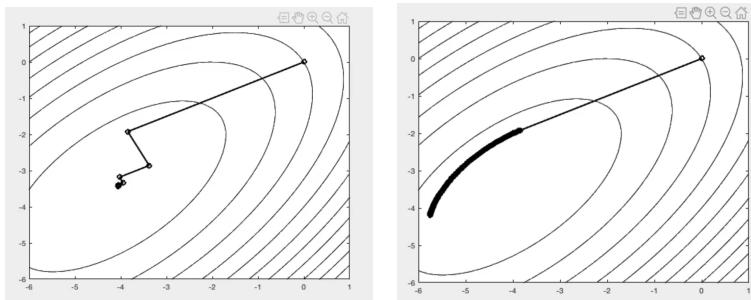


FIGURE 14.1: On the left side Newton method and on the right side the heavy ball method.

The plot of the convergence of the heavy ball method gives a graphical idea of the fact that the direction isn't orthogonal to the one at the previous iteration, since we have the gradient plus some quantity.

In particular, the bigger β^i the "less orthogonal" the steps are. This feature allows the algorithm to have good performances on elongated functions.

14.1.2 Accelerated gradient

This method works only on convex functions, it has some similarities with heavy ball, but it's slightly different.

ALGORITHM 14.1.1 Pseudocode for accelerated gradient method.

```

1: procedure ACCG( $f, \nabla f, x, \varepsilon$ )
2:    $x_- \leftarrow x;$ 
3:    $\gamma \leftarrow 1;$ 
4:   repeat
5:      $\gamma_- \leftarrow \gamma;$ 
6:      $\gamma \leftarrow (\sqrt{4\gamma^2 + \gamma^4} - \gamma^2)/2;$ 
7:      $\beta \leftarrow \gamma(1/\gamma_- - 1);$ 
8:      $y \leftarrow x + \beta(x - x_-);$ 
9:      $g \leftarrow \nabla f(y);$ 
10:     $x_- \leftarrow x;$ 
11:     $x \leftarrow y - (1/L)g;$ 
12:   until ( $\|g\| > \varepsilon$ )
13: end procedure

```

The rational behind this algorithm is the following: When we are in a certain point at a certain iteration, we go on a little bit β^i and we end up in a point y . The gradient is computed in that point and used to choose the next point.

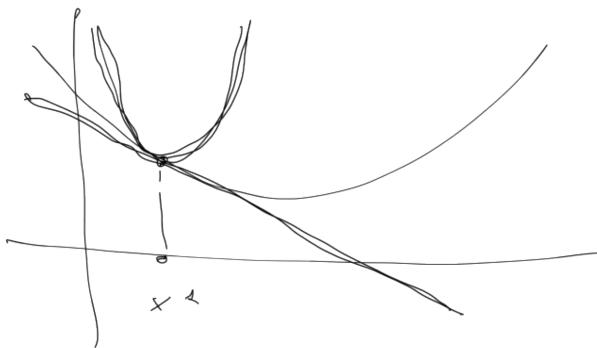


FIGURE 14.2: We build a linear model, which is a lowerbound for the function, then we build a quadratic model, which is above our function.

If we choose γ optimally then the quadratic approximation is very close to the function. We want to find the value for γ s that gives best results in the worst case.

We can prove that the error $\|f(x^i) - f_*\| \leq \sigma^i(f(x^i) - f_*) \searrow 0$ is multiplied by this factor δ_i , that goes like the inverse of i^2 .

Notice that if we choose α small, it will always be small.

The convergence is sublinear.

Theorem 14.1.1 (Optimality of accelerated gradient). *If the function isn't*

strongly convex no algorithm has better convergence than $|f(x^i) - f_| = 3L \frac{\|x^i - x_*\|^2}{32(i+1)^2}$.*

Observation 14.1.1. *This theorem tells us that this algorithm never gets worse than $|f(x^i) - f_*| = 3L \frac{\|x^i - x_*\|^2}{32(i+1)^2}$, but this doesn't imply that this method is fast on average. The state of the art provides a lot of different formulas for β , which of the ones leads to some theoretical results.*

From now on we will move towards a different family of functions, that aren't even differentiable, hence we can't compute the gradient.

14.2 Incremental gradient methods

This method has good performances in real world machine learning cases, where the function is differentiable but we do not want to compute the gradient.

Let $I = \{1, \dots, m\}$ be the set of observations, $X = [X^i \subset \mathcal{X}]_{i \in I}$ the set of inputs and $y = [y^i]_{i \in I}$ the set of outputs. Our goal is to explain y from X .

Since these vectors are uniformly distributed over the space (at least this is our hypothesis) when they get summed we expect some of them to cancel out.

The idea is to rewrite the problem as learning a linear function in the feature space, formally $\min \{ \sum_{i \in I} l(y^i, \langle \Phi(X^i), w \rangle) : w \in \mathbb{R}^n \}$, where $l(\cdot, \cdot)$ is called **loss function**.

We are now interested in computing the gradient, which is not hard to compute since the function is linear: $\nabla f(w) = \sum_{i \in I} \nabla f^i(w) = \sum_{i \in I} -A^i(y^i - A^i w)$.

The issue here is that computing the gradient, although it's the gradient of a very simple function, takes too long to be computed (since there are too many vectors in machine learning datasets).

To overcome this problem we choose to restrict to only a subset of observations. How can we choose such set? Randomly, of course.

At this point the algorithm isn't deterministic anymore, but it's completely **stochastic**.

The intuition behind this algorithm is to take only one observation, compute what is needed on this observation and make a small step.

An online application may be a sensor that produces hundreds of outputs per second and it's not possible to store each of them. They should be used to infer some information and then thrown away.

We study the converge of this kind of algorithms from a stochastic point of view.

In machine learning we always need some regularization, because the tuning of hyperparameters clearly takes into account only the error in the samples that have been seen. Let us regularize the model as follows

$$\min \left\{ \sum_{i \in I} l(y^i, \langle \Phi(X^i), w \rangle) + \mu \Omega(w) : w \in \mathbb{R}^n \right\}$$

The usage of regularization may be useful, since we want to keep close to the minimum, but not reach it. It's enough to change slightly the problem and then solve it.

The regularization hyperparameter $\Omega(w)$ may be chosen as follows:

1. Lasso regularizer (best known): $\Omega(w) = \|w\|_1$;
2. In order to increase sparsity: $\Omega(w) = \|w\|^2$;
3. Leading to feature selection: many $w_j = 0$ as possible.

Ω function is not differentiable, so the function gets non differentiable, as an example look at Figure 14.3, which represents the plot of $f(w_1, w_2) = (3w_1 + 2w_2 - 2)^2 + 10(|w_1| + |w_2|)$.

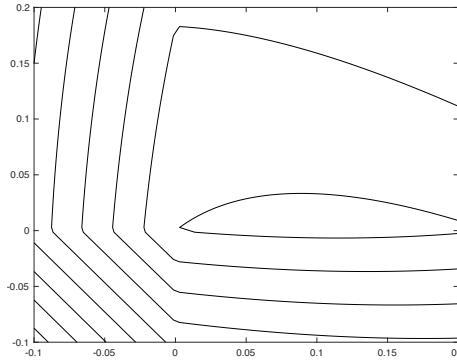


FIGURE 14.3: This function has a lot of kinky points.

14.3 Subgradient methods

These methods are thought for convex functions that are not differentiable.

Let us consider a kinky point: how can we choose between all the subgradients of that point? We assume to be able to compute some subgradients; since the function is convex we may recall

Property 14.3.1. *Let f be a convex function, $\forall g \in \partial f(x), \forall y \in \mathbb{R}^n g(y) \equiv f(y) \geq f(x) + g(y - x)$.*

Let x^* be the optimum, $\langle x^* - x^i, g \rangle$ is smaller than 0. This means that the angle between x and x^* is acute. If we knew the exact direction $x - x^*$ the line search would land on x^* . For a pictorial representation see Figure 14.4

The intuition behind this algorithm is to move in the direction of $-g$, but with a small α^i , because if the step is too large we may end up in a point which is actually further from x^* than the previous step. In that point we may

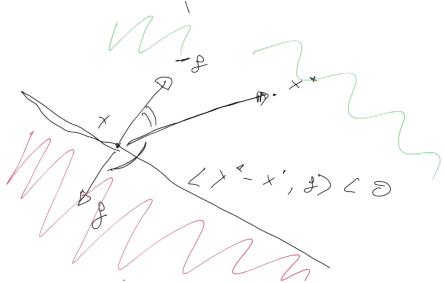


FIGURE 14.4: Since we know that $\langle g, x^* \rangle$ is negative we get that the angle between g and x^* is larger than 90° . Moreover, we know where the optimum is not (red region), hence we restrict to the green region. At this point we will perform a line search on g direction. Since the function is not smooth, the line search may not succeed since the value of the function along the half line from x in g direction may remain the same, but there may be some points there that are closer to x^* .

find a point where perform line search, because that point isn't kinky. In this context we won't try to minimize $\|f(x) - f(x^*)\|$, but we will minimize $\|x - x^*\|$, because the function may zigzag near that point. It goes without saying that choosing a too small value for α leads to a too slow convergence speed.

Lecture 15

28th of November 2018

A. Frangioni

15.1 Subgradient methods

We are still in the hypothesis of convex objective functions that are not differentiable in the whole domain.

 Do you recall?

Property 14.3.1 of last lecture: for f convex function, $\forall g \in \partial f(x), \forall y \in \mathbb{R}^n g(y) \equiv f(y) \geq f(x) + g(y - x)$. This is a characterization of the function with respect to the model.

Let us assume we know the minimum point x_* . We observe that the scalar product between the subgradient and the direction we should choose is negative. Formally, $f(x_*) \geq f(x) + \langle g, x_* - x \rangle$, hence $\langle g, x_* - x \rangle \leq f(x_*) - f(x) \leq 0$.

We want to bound the distance between the point at the “next step” and the optimum:

$$\begin{aligned} \|x^{i+1} - x_*\|^2 &\stackrel{(1)}{=} \|x^i - \alpha^i d^i - x_*\|^2 \\ &= \|x^i - x_*\|^2 + 2\alpha^i g^i \frac{(x_* - x^i)}{\|g^i\|} + (\alpha^i)^2 \\ &\stackrel{(2)}{\leq} \|x^i - x_*\|^2 - 2\alpha^i \frac{(f(x^i) - f(x_*))}{\|g^i\|} + (\alpha^i)^2 \end{aligned} \quad (15.1.1)$$

Where $\stackrel{(1)}{=}$ follows from the definition of a normalized step: $x^{i+1} = \frac{x^i - \alpha^i g^i}{\|g^i\|}$ and $\stackrel{(2)}{\leq}$ follows from the inequality we stated above.

Observation 15.1.1. *The distance from the optimum is bounded by a square function in α_i (the step size), where the linear part is negative and the quadratic part is positive.*

Hence, if the steps are short enough we get close to the optimum fast enough, because the linear part dominates the quadratic one.

Formally, $-2 \frac{(f(x^i) - f(x_*))}{\|g^i\|} < 0 + \alpha^i \searrow 0_+$, hence $(\alpha^i)^2 \searrow 0_+$.

An attentive reader may notice that from Equation (15.1.1) follows $\|x^{i+1} - x_\|^2 < \|x^i - x_*\|^2 + (\alpha_i)^2$ and $\|x^{i+1} - x_*\|^2 \leq \|x^1 - x_*\|^2 + \sum_{k=1}^i (\alpha^k)^2$.*

The point here is that we cannot choose a too small α , recall Armijo conditions.

If the series of the squares of step sizes does not diverge, then the sequence does not diverge as well, hence it converges somewhere, say \bar{x} .

The convergence of the series of the squares may be obtained using the following

Definition 15.1.1 (Diminishing-Square Summable). *We term a series that diverges, while the series of the squares of the terms diverges, as **diminishing-square summable**.*

Formally,

$$(DSS) \sum_{i=1}^{\infty} \alpha^i = \infty \wedge \sum_{i=1}^{\infty} (\alpha^i)^2 < \infty.$$

Assumptions: function convex, definite in all its domain, hence the norm of the gradient will never become very large.

Fact 15.1.1. *We claim that the sequence of the stepsizes is a diminishing-square sequence.*

Proof by contraddiction. Let us assume $f(x^i) - f_* \geq \varepsilon > 0$, $\forall i$. Then,

$$\|x^{i+1} - x_*\|^2 \leq \|x^1 - x_*\|^2 - \delta \cdot \sum_{k=1}^i \alpha^k + \sum_{k=1}^i (\alpha^k)^2$$

The contraddiction is due to the fact that as $i \rightarrow \infty$ the right-hand side goes to $-\infty$. \square

This family of algorithms is clearly incredibly robust in theory, but in practice it does not work very well.

Let us make an experiment: let us suppose we know the minimum of the function f .

Definition 15.1.2 (Plyak stepsize). *Let f be our convex objective function and let x_* be the optimum for f . We term **Polyak stepsize**:*

$$(PSS) \alpha^i = \beta^i \frac{(f(x^i) - f(x_*))}{\|g^i\|}$$

where $\beta \in (0, 2)$.

If we pick a Polyak stepsize, then $\|x^{i+1} - x_*\|^2 < \|x^i - x_*\|^2$, so $\{x^i\} \rightarrow x_*$. The best value for β_i is 1. In this case, we obtain what follows by substituting $\beta_i = 1$ in Equation (15.1.1)

$$\frac{(f(x^i) - f(x_*))^2}{\|g^i\|^2} \leq \|x^i - x_*\|^2 - \|x^{i+1} - x_*\|^2$$

The problem is that we don't know the optimum.

Let us assume that we know it and compute the efficiency:

Since we know that the sequence is bounded, we know that the objective function is globally Lipschitz (the norm of the gradient is bounded above).

The point is that the sequence $\{x_1\}$ is not necessarily monotone, so we pick the so-called record value of best value ($\underline{f}^i = \min\{f(x^h) : h = 1, \dots, i\}$)

$$\frac{(\underline{f}^i - f(x_*))^2}{L^2} \leq \frac{(f^i - f(x_*))^2}{\|g_i\|^2} \leq \|x^i - x_*\|^2 - \|x^{i+1} - x_*\|^2$$

Summing for $i = 1, \dots, k$ we obtain a telescopic series:

$$\|x^1 - x_*\| - \|\cancel{x}^2 - x_*\| + \|\cancel{x}^2 - x_*\| - \|\cancel{x}^3 - x_*\| + \dots + \|\cancel{x}^k - x_*\| - \|x^{k+1} - x_*\|$$

Hence resulting in

$$k \cdot \frac{(\underline{f}^k - f(x_*))^2}{L^2} \leq \|x^1 - x_*\|^2 - \|x^{k+1} - x_*\|^2 \leq \|x^1 - x_*\|^2 = R$$

which is equivalent to $(\underline{f}^k - f(x^*))^2 \leq \frac{R^2 L^2}{k}$, which is again equivalent to $\underline{f}^k - f(x^*) \leq \sqrt{\frac{RL}{k}}$, where L is the Lipschitz constant.

The issue here is that the convergence is sublinear: $k \geq \frac{1}{\varepsilon^2}$.

Theorem 15.1.2. *Take an algorithm that uses only the subgradient. It's possible to construct a function that makes the algorithm converge with sublinear speed. Hence, we cannot do better.*

It comes without saying that although this algorithm is not very good it is the "less bad" it can be.

There are some lucky cases in which we do know the optimal value, but this is not the case usually.

15.1.1 Target level stepsize

Let us assume $f(x_*)$ is unknown. The only information available is that this value is below any value in any iteration.

The rationale behind this algorithm is to assume to know the optimal value and as soon as we realize it is not correct we change it.

Let us first give an informal description of the algorithm:

- δ is the displacement: how much below the function is with respect to the best value obtained so far;
- reference value $f_{rec} = f$. At the beginning is the value at the first iterate and then we define the target value as the difference between the reference value and some δ (at the beginning δ_0).

ALGORITHM 15.1.1 Pseudocode for target level stepsize.

```

1: procedure SGPTL( $f, g, x, i_{max}, \beta, \delta_0, R, \rho$ )
2:    $r \leftarrow 0$ ;
3:    $\delta \leftarrow \delta_0$ ;
4:    $f_{ref} \leftarrow f_{rec} \leftarrow f(x)$ ;
5:    $i \leftarrow 1$ ;
6:   while ( $i < i_{max}$ ) do
7:      $g = g(x)$ ;
8:      $\alpha = \beta(f(x) - (f_{ref} - \delta)) / \|g\|^2$ ;
9:      $x \leftarrow x - \alpha g$ ;
10:    if ( $f(x) \leq f_{ref} - \delta/2$ ) then
11:       $f_{ref} \leftarrow f_{rec}$ ;
12:       $r \leftarrow 0$ ;
13:    else
14:      if ( $r > R$ ) then
15:         $\delta \leftarrow \delta\rho$ ;
16:         $r \leftarrow 0$ ;
17:      else
18:         $r \leftarrow r + \alpha \|g\|$ ;
19:      end if
20:    end if
21:   end while
22:    $f_{rec} \leftarrow \min\{f_{rec}, f(x)\}$ ;
23:    $i \leftarrow i + 1$ ;
24: end procedure

```

At this point we defined the algorithm and we are ready to implement it, except for the fact that we need to choose of a lot of parameters. A way to choose them is to use the ML approach: try many possible values.

Two big issues of this algorithm are that it does not provide a good stopping criterion and it is very sensitive to many parameters.

15.2 Deflected subgradient

The idea is to use the same trick of ball-step (also called primal-dual).

Let us assume that our function was differentiable. The subgradient method collapses to the gradient method and we know that the gradient method does not provide a good convergence. Yet, deflection is possible: $d^i = \gamma^i g^i + (1 -$

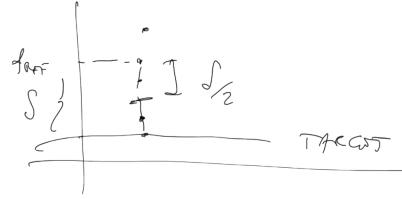


FIGURE 15.1: There are two cases: either the function value is significantly below the reference (for example $\frac{\delta}{2}$ below the reference) or it's not. If we are in the “happy case” (we just move the reference to the best value). For what concerns the “unhappy case”, if we are unhappy for 1 iteration, no problem. Two iterations? no problem. Many iterations? Problem: after some iterations in which we are not improving it means that we have to decrease the reference values. How? r is updated at each bad step and reset when a good step occurs. When r gets too large we decrease δ and reset everything.

$\gamma^i) d^{i-1}$, $x^{i+1} = x^i - \alpha^i d^i$. We can prove that d^i approximates the subgradient. We can also prove that the algorithm converges in the end. The parameters of this algorithm are two: β (stepsize) and γ (deflection). In order to choose them we have two different approaches:

STEP SIZE-RESTRICTED \equiv deflection-first. We first choose β and when choosing α we need to take into account β : $\alpha^i = \frac{\beta^i(f(x^i) - f_*)}{\|d^i\|} \wedge \beta^i \leq \gamma^i$ “as deflection \nearrow , step size has to \searrow ”;

DEFLECTION-RESTRICTED \equiv stepsize-first. We first choose γ , then we pick a step size that depends on γ :

$$(DSS) \wedge \frac{\alpha^{i-1} \|d^{i-1}\|}{(f(x^i) - f_*) + \alpha^{i-1}} \|d^{i-1}\| \leq \gamma^i$$

“as $f(x^i) \rightarrow f_*$, deflection \searrow ”

This algorithm gets the optimal $O(1/\varepsilon^2)$ on average, sadly not worst case.

15.3 Smoothed gradient methods

Let us assume that the target function is a **Lagrangian function**.

Definition 15.3.1 (Lagrangian function). *Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be a function of the following shape:*

$$f(x) = \max\{x^T A z : z \in Z\}$$

where Z is convex and bounded.

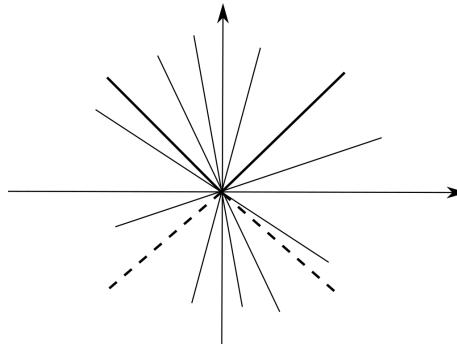


FIGURE 15.2: Let us take the absolute value function $f(x) = |x|$. This function would be differentiable, if it wasn't for some nasty points (in this case 0, were the optimization problem has many optimal solutions). In 0 there are many subgradients, so it has many optima.

Let us assume that Z is also compact.

A graphical example in the case of $f(x) = |x| = \max\{x, -x\} = \max\{zx : z \in [-1, 1]\}$ is shown in Figure 15.2.

In the case of the absolute value, the nasty trick is “to make it have only one optimal solution” in the point 0.

This result is obtained adding a small quadratic term: $f(x) = \max\{x^T Az - \mu \|z\|^2 : z \in G\}$ that is shown in Figure 15.3. At this point the new function f_μ is not the original function anymore, but it is very close to it whenever the μ is small.

Notice that this new function is smooth (differentiable).

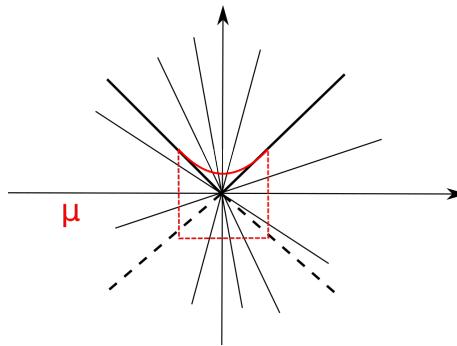


FIGURE 15.3: Geometric intuition of the usage of variable μ .

It might be not very easy to compute once chosen the value for μ .

We are solving a problem which is different from the original one and would be exactly the same if $\mu = 0$.

On the other hand for $\mu = 0$ we have the problem which is not differentiable once again, so we need to keep close to 0 but not too close.

At this point we are in the situation of $f_\mu(x) \leq f(x) \leq f_\mu(x) + \mu R$, such that as $\mu \searrow 0$, “ $\operatorname{argmin} \{f_\mu(x)\} \rightarrow x_*$ ”.

The new function is not only convex, but it is also Lipschitz continuous: ∇f_μ Lipschitz with $L = \|A\|^2/\mu$, but it is “less and less Lipschitz” as $\mu \searrow 0$.

Fact 15.3.1. *If $f_* > -\infty$ and picking a very special value of μ ($\mu = \varepsilon/(2R)$), then an appropriate ACCG obtains $f(x^i) - f_* \leq \varepsilon$ for $i \geq 4 \cdot \|A\| \cdot \|x_*\| \cdot \sqrt{R}/\varepsilon$.*

We observe that the convergence is much better, because it depends on $O(1/\varepsilon)$ instead of $O(1/\varepsilon^2)$.



FIGURE 15.4: At the beginning we will make a lot of bad steps (the upper gray line). We can improve (pick the black line) changing the ε value and we obtain something that looks more stable. The more precision we want, the smaller the step we make. At the ending it pays, but at the beginning it is not so.

15.4 Cutting-plane algorithm

We cheated to get first order information, we want to do more. We want to cheat and have also the second order information.

We want to use the same idea of limited memory quasi Newton methods. Using some limited memory of the Hessian, in order to understand the curvature.

The point is that the directional derivatives are defined and (if computed massively) give me a hint of the curvature of the function (cfr. Figure 15.5). Notice that it is not possible to build a matrix, because it is not defined.

Let us say we have performed i iterates, then we have collected i subgradients ($\mathcal{B} = \{(x^i, f^i = f(x^i), g^i \in \partial f(x^i))\} \equiv$ bundle of first-order information) and function values.

We can now define a piece wise linear function defined as the maximum of the first order model:

$$f_{\mathcal{B}}(x) = \max\{f^i + g^i(x - x^i) : (x^i, f^i, g^i) \in \mathcal{B}\}$$

At this point we can apply Newton method: first we minimize the model and the use the minimum as next point. We collect information in that specific point and then we repeat.

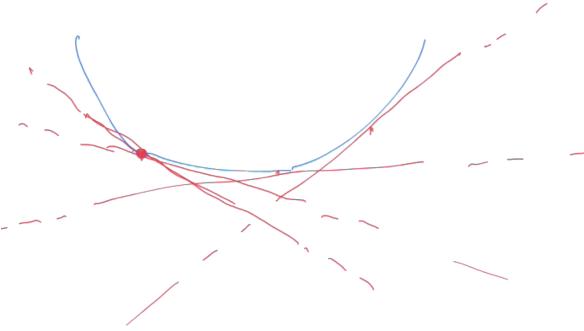


FIGURE 15.5: The idea is that we do not compute a subgradient and discard it. We store it, thus resulting in a model

Notice that the model is always below the objective function ($f_{\mathcal{B}}(x) \leq f(x) \forall x$), hence $\min\{f_{\mathcal{B}}(x)\} \leq f_*$, so $x_{\mathcal{B}}^* \in \operatorname{argmin}\{f_{\mathcal{B}}(x)\} \approx x_*$.

This function has many kinky points, so how to minimize it? Dirty trick from Ricerca Operativa:

$$\min\{f_{\mathcal{B}}(x)\} = \min\{v : v \geq f^i + g^i(x - x^i), \text{ where } (x^i, f^i, g^i) \in \mathcal{B}\}$$

And on this problem, we can use the simplex method.

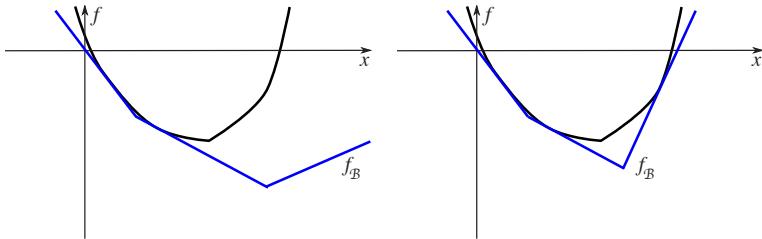


FIGURE 15.6: A geometric representation of how the model (blue line) changes after one iteration.

At this point we have obtained both an upperbound and a lower bound, which go one towards the other.

We may decide to stop iterating when they are “close enough”.

The problem is that at each step we need to solve a minimization problem and the convergence is very slow.

It's also possibile that the blue function (the model) does not have a minimum (unbounded below).

How to overcome this problem? Use so-called bundle methods.

15.5 Bundle methods

The intuition behind this is that we want to keep quite close to the point where the model corresponds to the function, because the further we get the more the difference from the function.

A way to express this is to add a quadratic quantity to the function that grows when we move outside the current point.

Definition 15.5.1 (Stabilized master problem). *We term **stabilized master problem** the following*

$$\min\{f_{\mathcal{B}}(x) + \mu\|x - \bar{x}\|^2/2\}$$

This improved model cannot be undounded below (quadratic function), but we need to choose μ and \bar{x} wisely.

A possible way is to move the **stability center** whenever the current value is better than the best encountered so far.

ALGORITHM 15.5.1 Pseudocode for boundle method.

```

1: procedure PBM( $f, g, \bar{x}, m_1, \varepsilon$ )
2:   choose  $\mu$ ;
3:    $\mathcal{B} \leftarrow \{(\bar{x}, f(\bar{x}), g(\bar{x}))\}$ ;
4:   while ( true ) do
5:      $x^* \leftarrow \operatorname{argmin} \{f_{\mathcal{B}}(x) + \mu\|x - \bar{x}\|^2/2\}$ ;
6:     if ( $\mu\|x^* - \bar{x}\|_2 \leq \varepsilon$ ) then
7:       break;
8:     end if
9:     if ( $f(x^*) \leq f(\bar{x}) + m_1(f_{\mathcal{B}}(x^*) - f(\bar{x}))$ ) then
10:       $\bar{x} \leftarrow x^*$ ;
11:      possibly decrease  $\mu$ ;
12:    else
13:      possibly increase  $\mu$ ;
14:    end if
15:     $\mathcal{B} \leftarrow \mathcal{B} \cup (x^*, f(x^*), g(x^*))$ ;
16:   end while
17: end procedure

```

This algorithm may never move (without cycling, luckily), but at least we gained some information.

The bundle method converges in few steps, although each step is quite costly.

We reached a point where to solve an unconstrained problem we need to solve a constrained one, so from next lecture we will start dealing with constrained optimization problems.

Lecture 16

30th of November 2018

A. Frangioni

16.1 Constrained optimization

In this lecture we address the problem of finding the **optimum** of a function in a subset of its domain, called X . The term optimum differs from the minimum, because the optimum in that subset may not be a minimum of the whole function.

$$f_* = \min\{f(x) : x \in X\}$$

Definition 16.1.1 (Local optimum). *Given a function f and a constraint set X , we denote **local optimum** the point where the function assumes the minimum value inside the set X . Formally, $\min\{f(x) : x \in \mathcal{B}(x_*, \varepsilon) \cap X\}$ for some $\varepsilon > 0$.*

Notice that the only points in which the constraint adds some informations are the ones on the boundary, as shown in Figure 16.1

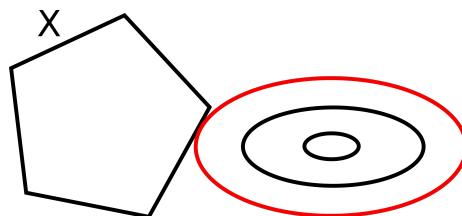


FIGURE 16.1: The red line is level set of the function corresponding to the smallest value that touches the set X . The point in the intersection is not a saddle point of the function f , although it is the minimum.

There are two kinds of constraints:

FAKE ONES: this first kind is such that the minimum of the function lies inside the set X , hence there is no need to use the constraints at all;

REAL ONES: when the optimal is on the boundary. This is the case of linear functions, because the gradient is constant $\nabla f(x) = c$.

At this point we want to decide if a point on the boundary is an optimum. In this context it is important how the boundary is defined.

16.1.1 Linear equality constraints

A constraint of this kind is very simple: it is a subspace, as shown in Figure 16.2.

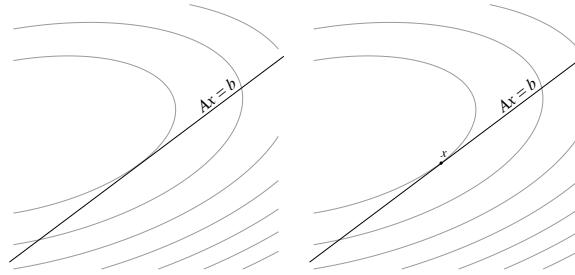


FIGURE 16.2: Linear constraint and a point on the boundary.

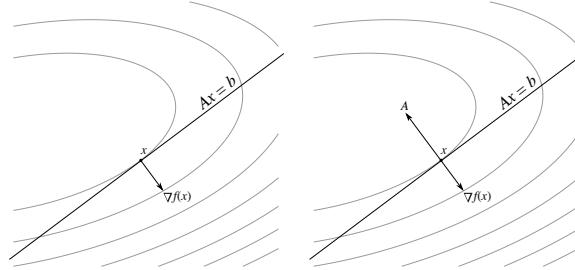


FIGURE 16.3: The gradient is orthogonal to the level set in that point, when the function is smooth. The same holds for matrix A .

$\min\{f(x) : Ax = b\}$, where the rank of A counts the number of linearly independent rows.

Let us assume that there are not linearly independent rows in A , then or this behaviour is reflected in B or the system does not have any solution.

In the case of presence of linearly dependent columns such columns may be eliminated to ease the computation without loss of information.

The intuition behind what follows is that each linear constraint kills one degree of freedom, formally $\det(A_B) \neq 0 \Rightarrow Ax = b \equiv x_B = A_B^{-1}(b - A_N x_N) \Rightarrow$

We want to extract a submatrix A_B from $A \in M(m, n, \mathbb{R})$, such that $A_B \in M(m, \mathbb{R})$ and then the system induces a partitioning in the variables as well.

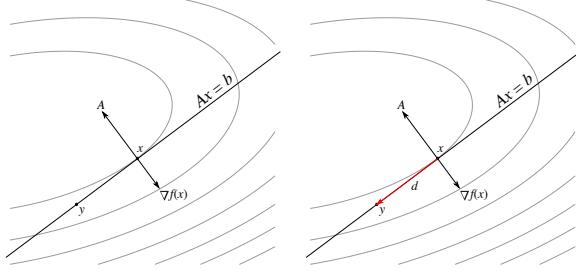


FIGURE 16.4: If we take any other point in the space it has to be orthogonal to A .

$A = [A_B, A_N]$, $x = [x_B, x_N]$, so the system becomes $A_B x_B + A_N x_N = b$ and (since A_B is non singular) $X_B + A_B^{-1} x_N = A_B^{-1} b$ in other words, given the independent variables we can compute the values of the dependent ones and this is a linear operation.

The original optimization problem becomes an optimization problem on a reduced space, formally $\min\{r(w) = f(Dw + d) : w \in \mathbb{R}^{n-m}\}$, where $D = \begin{bmatrix} -A_B^{-1} A_N \\ I \end{bmatrix}$ and $d = \begin{bmatrix} A_B^{-1} b \\ 0 \end{bmatrix}$. For each point in the smaller space we can compute the function in the larger space.

How can we compute the gradient of $r(w)$? The gradient is $\nabla r(w) = D^T \nabla f(Dw + d)$. The fact that w^* is an optimum implies that $\nabla r(w^*) = 0$.

$$D = \begin{bmatrix} -A_B^{-1} A_N \\ I \end{bmatrix} \text{ then } AD = [A_B, A_N] \cdot \begin{bmatrix} -A_B^{-1} A_N \\ I \end{bmatrix} = -A_B \cdot A_B^{-1} \cdot A_N + A_N = 0.$$

Now the point is taking a multiple of matrix A , finding a feasible x and corresponding w (because there is a bijection), then finding the value μ that allows the equality.

Theorem 16.1.1. Let $Ax = b$ be a linear system and w such that $x = [A_B^{-1}(b - A_N w), w]$. If $\exists \mu \in \mathbb{R}^m$ s.t. $\mu A = \nabla f(x)$ then $r(w) = D^T \nabla f(x) = 0$, see Figure 16.5. In other words, it is equivalent to find a stationary point x for the original problem (P) or finding the stationary point w for (R).

Definition 16.1.2 (Poorman's Karush Kuhn-Tucker conditions). A point is a good candidate for being a minimum of the constrained problem if and only if it satisfy **Poorman's KKT conditions**, namely the problem is feasible and that $\exists \mu \in \mathbb{R}^m$ s.t. $\mu A = \nabla f(x)$.

Theorem 16.1.2. Let f be a convex function, then KKT conditions are enough for optimality.

A very naive explanation of the theorem is that if the function is convex also the restriction is convex and a stationary point of a convex function is a minimum.

Our idea is to characterize the directions we can move along in order to find new points that satisfy the constraint. Formally, $Ax = b$ is our constraint

and we want to move towards $x + d$ and stay in the feasible region. How? $A(w + d) = b \Leftrightarrow Ax + Ad = b \Leftrightarrow Ad = 0$, since $Ax = b$. The only way to move along the constraint is choosing a direction which scalar product with A is 0, hence 0 scalar product with the gradient.

From now on we would like to study the behaviour on constrained problems where the constraints are equalities, but inequalities.

In order to do this we need some mathematical background.

16.1.2 Background for linear inequality constraints

Definition 16.1.3 (Tangent cone). We call **tangent cone** of X at x $T_X(x) = t$.

$$\{d \in \mathbb{R}^n : \exists \{z_i \in X\} \rightarrow x \wedge \{t_i \geq 0\} \rightarrow 0 \text{ s.t. } d = \lim_{i \rightarrow \infty} \frac{z_i - x}{t_i}\}$$

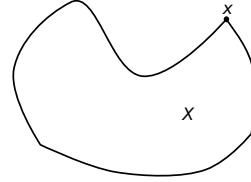


FIGURE 16.5: When the boundary has this shape we can move along directions that point inside the constraints. Tangent directions are not allowed.

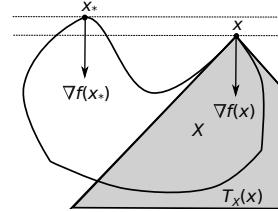


FIGURE 16.6: Geometric representation of tangent cone, which is the region of the space where we can pick the directions. The intuition is to zoom in x and the result of this zooming is a cone.

Theorem 16.1.3. Let \mathcal{C} be a cone, $\forall x \in \mathcal{C} \ \alpha x \in \mathcal{C}, \forall \alpha > 0$.

Theorem 16.1.4. Given a function f , where x is a **local optimum** $\langle \nabla f(x), d \rangle \geq 0 \ \forall d \in T_X(x)$.

Proof. **Proof by contraddiction:** Assume $\exists d \in T_X(x)$ such that $\langle \nabla f(x), d \rangle < 0$, but x is a local optimum.

By definition $\exists X \supset \{z_i\} \rightarrow x$ and $\{t_i\} \rightarrow 0$ such that $d = \lim_{i \rightarrow \infty}^* \frac{z_i - x}{t_i}$.
First order Taylor $f(z_i) - f(x) = \langle \nabla f(x), (z_i - x) \rangle + R(z_i - x)$.

$$\begin{aligned} \lim_{i \rightarrow \infty} \frac{f(z_i) - f(x)}{t_i} &= \lim_{i \rightarrow \infty} \left\langle \nabla f(x), \frac{z_i - x}{t_i} \right\rangle + \frac{R(z_i - x)}{t_i} \\ &= \langle \nabla f(x), d \rangle + \lim_{i \rightarrow \infty} \frac{R(z_i - x)}{t_i} \\ &\stackrel{(1)}{=} \langle \nabla f(x), d \rangle \\ &< 0 \end{aligned} \quad (16.1.1)$$

Where, $\stackrel{(1)}{=}$ follows from $\lim_{i \rightarrow \infty} \frac{R(z_i - x)}{t_i} = 0$ by Taylor. \square

Observation 16.1.1. *The optimum of Theorem 16.1.4 is global when the function is convex, because in that case $X \subseteq x + T_X(x)$. For a geometric idea see Figure 16.7.*

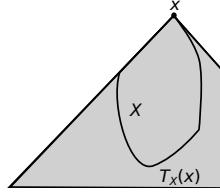


FIGURE 16.7: Convex function.

Observation 16.1.2. *Notice that the rule $\langle \nabla f(x), d \rangle \geq 0 \forall d \in T_X(x) \Rightarrow x$ local optimum does not hold. Let us see a counter example: $\min\{x_2 : x_2 \geq x_1^3\}$, displayed in Figure 16.9.*

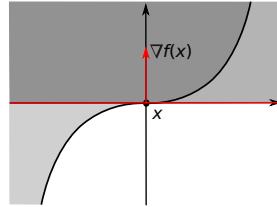


FIGURE 16.8: Let us suppose we pick the direction towards the left part of the function in the saddle point. That direction is promising and actually the value of the function decreases. In this case the problem is that the constraint is not convex.

Now we need a more manageable object for $T_X(x)$.

Definition 16.1.4 (Cone of feasible directions).

*Intuitively, we are in x and we want to find all directions **feasible cone** such that there exist small but not 0 steps such that all the points on this direction are feasible. Formally, a **feasible cone** is $F_X(x) = \{d \in \mathbb{R}^n : \exists \bar{\varepsilon} > 0 \text{ such that } x + \varepsilon d \in X, \forall \varepsilon \in [0, \bar{\varepsilon}]\}$.*

Fact 16.1.5. *The properties of such cone are:*

1. T_X closed, F_X in general not (hence the cone of feasible directions is the tangent cone minus the tangent directions);
2. $\text{cl}(F_X) \subseteq T_X$, where $\text{cl}(F_X)$ is the closure of the cone of feasible directions;
3. If X convex then the cones coincide: T_X and F_X convex and $\text{cl } F_X = T_X$.

We are now interested in finding a better characterization of the cone of feasible directions, hence we introduce a new characterization of the set of constraints X .

FIRST REPRESENTATION:

$$X = \{x \in \mathbb{R}^n : g_i(x) \leq 0 \ i \in \mathcal{I}, h_j(x) = 0 \ j \in \mathcal{J}\}$$

where \mathcal{I} is the set of inequality constraints and \mathcal{J} is the set of equality constraints;

SECOND REPRESENTATION:

$$X = \{x \in \mathbb{R}^n : G(x) \leq 0, H(x) = 0\}$$

where $G = [g_i(x)]_{i \in \mathcal{I}} : \mathbb{R}^n \rightarrow \mathbb{R}^{|\mathcal{I}|}$ and $H = [h_i(x)]_{i \in \mathcal{J}} : \mathbb{R}^n \rightarrow \mathbb{R}^{|\mathcal{J}|}$;

THIRD REPRESENTATION: (hiding equalities)

$$X = \{x \in \mathbb{R}^n : g_i(x) \leq 0 \ i \in \mathcal{I}, h_j(x) \leq 0 \ \wedge \ h_j(x) \geq 0 \ j \in \mathcal{J}\}$$

FOURTH REPRESENTATION: (hiding inequalities into a single function)

$$X = \{x \in \mathbb{R}^n : g(x) = \max\{g_i(x) : i \in \mathcal{I}\} \leq 0 \ i \in \mathcal{I}, h_j(x) = 0 \ j \in \mathcal{J}\}$$

Definition 16.1.5 (Active constraints). *We term **active constraints** at $x \in X$ the following set*

$$\mathcal{A}(x) = \{i \in \mathcal{I} : g_i(x) = 0\} \subseteq \mathcal{I}$$

Let us introduce some useful notation on the subject: let $\mathcal{B} \subseteq \mathcal{I}$ a subset of indices.

We denote $G_{\mathcal{B}} = [g_i(x)]_{i \in \mathcal{B}} : \mathbb{R}^n \rightarrow \mathbb{R}^{|\mathcal{B}|}$ the corresponding set of inequalities.

Definition 16.1.6 (First-order feasible direction cone). We term **First-order feasible direction cone at $x \in X$** :

$$D_X(x) = \{d \in \mathbb{R}^n : \langle \nabla g_i(x), d \rangle \leq 0 \text{ } i \in \mathcal{A}(x) \text{ } \langle \nabla h_j(x), d \rangle = 0 \text{ } j \in \mathcal{J}\} = \{d \in \mathbb{R}^n : (JG_{\mathcal{A}(x)}(x))d \leq 0, (JH(x))d = 0\}$$

Intuitively, the fact that we are looking at the active set means that we are zooming very close to 0.

In this cone we require that all the directions inside it have a negative scalar product with the gradient of the constraints.

A visual example of a first order feasible direction cone is displayed in Figure 16.9.

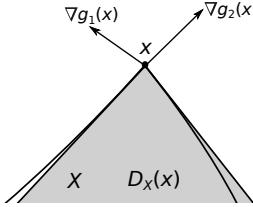


FIGURE 16.9: The first-order feasible direction cone is made of those directions that are orthogonal to the gradient of the constraints $g_1(x)$ and $g_2(x)$ in x .

Fact 16.1.6. The tangent cone is a subset of the first-order feasible direction cone. Formally, $T_X(x) \subseteq D_X(x)$.

We would like the first-order feasible direction cone to be exactly equal to the tangent cone and this is almost always true, except for some pathological cases.

Example 16.1.1. Let our minimum problem be $\min\{\dots : x_1^2 + (x_2 - 1)^2 - 1 \leq 0, x_2 \leq 0\}$.

The plot of such functions is shown in Figure 16.10.

Our claim is that the only feasible point is $X = \{x = [0, 0]\}$.

The only feasible direction is 0, hence cone of feasible direction and the tangent cone are the singleton $\{[0, 0]\}$.

On the other hand, the set of directions that have non-negative scalar product with both g_1 and g_2 are all the x axis.

We would like to ensure we are not in one of these pathological cases and to do this we introduce some conditions.

Fact 16.1.7. The following holds:

AFFINE CONSTRAINTS (AFFC): Let g_i and h_j be affine constraints. Then, $\forall i \in \mathcal{I}$ and $j \in \mathcal{J}$ $T_X(x) = D_X(x) \forall x \in X$.

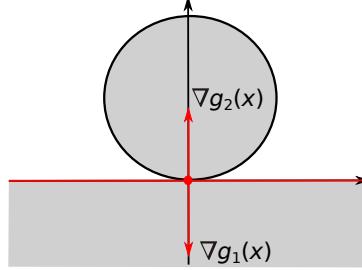


FIGURE 16.10: The circle represents the quadratic constraint, while the semi-plane represents the first degree constraint.

SLATER'S CONDITION (SLAC): Let g_i convex $\forall i \in \mathcal{I}$ and let h_j affine $\forall j \in \mathcal{J}$
 $\exists \bar{x} \in X$ s.t. $g_i(\bar{x}) < 0 \forall i \in \mathcal{I}$. Then $T_X(x) = D_X(x) \forall x \in X$;

LINEAR INDEPENDENCE (LINI): $\bar{x} \in X \wedge$ the vectors $\{\nabla g_i(\bar{x}) : i \in \mathcal{A}(\bar{x})\} \cup \{\nabla h_j(\bar{x}) : j \in \mathcal{J}\}$ linearly independent $\implies T_X(\bar{x}) = D_X(\bar{x})$. Among all these conditions this is the only local one.

It goes without saying that we cannot check all the directions in order to exclude the nasty pathological cases.

Definition 16.1.7 (Dual cone). Let D_X be a **polyhedral cone** $\mathcal{C} = \{d \in \mathbb{R}^n : Ad \leq 0\}$, for some $A \in \mathbb{R}^{k \times n}$.

We term **dual cone** $\mathcal{C}^* = \{c = \sum_{i=1}^k \lambda_i A_i : \lambda \geq 0\}$.

Lemma 16.1.8 (Farka's lemma). Intuitively, this lemma says that pick a vector: either it belongs to the dual cone or there exists a vector in the polyhedral cone which has a negative scalar product with it.

Equivalently, either $c \in \mathcal{C}^*$ or $c \notin \mathcal{C}^*$.

More formally, either $\exists \lambda \geq 0$ s.t. $c = \sum_{i=1}^k \lambda_i A_i$ or $\exists d$ s.t. $Ad \leq 0 \wedge c, d >> 0$.

Theorem 16.1.9 (Karush-Kuhn-Tucker conditions). Let us assume that we found an optimal solution x_* and the constraints qualification holds.

Then $\exists \lambda \in \mathbb{R}_+^{|\mathcal{I}|}$ and $\mu \in \mathbb{R}^{|\mathcal{J}|}$ such that

$$\nabla f(x) + \sum_{i \in \mathcal{A}(x)} \lambda_i \nabla g_i(x) + \sum_{j \in \mathcal{J}} \mu_j \nabla h_j(x) = 0$$

It is interesting to notice that we did not impose $\mu \geq 0$. Let us take an equality constraint $h_j(x) = 0$. This is equivalent to write $h_j(x) \leq 0 \wedge h_j(x) \geq 0$, thus leading to two different multipliers, say λ_j^+ and λ_j^- .

The term of the sum concerning h_j looks like this $\lambda_j^+ \nabla h_j(x_*) - \lambda_j^- \nabla h_j(x_*) = (\lambda_j^+ - \lambda_j^-) \cdot \nabla h_j(x_*)$, where both λ_j^+ and λ_j^- are ≥ 0 , hence their difference (denoted by μ_j) may be either positive or negative.

Fact 16.1.10. *The Karush-Kuhn-Tucker conditions are also written as:*

FEASIBILITY: $x \in X \equiv g_i(x) \leq 0 \quad i \in \mathcal{I}, \quad h_j(x) = 0 \quad j \in \mathcal{J}$

KKT-G: $\nabla f(x) + \sum_{i \in \mathcal{I}} \lambda_i \nabla g_i(x) + \sum_{j \in \mathcal{J}} \mu_j \nabla h_j(x) = 0$

COMPLEMENTARITY SLACKNESS: $\sum_{i \in \mathcal{I}} \lambda_i g_i(x) = 0$

where the second and the third equations formalize the definition of KKT, where the terms of the first sum should be summed only if their constraints are active.

Fact 16.1.11. *Let (P) be a convex problem. In this case, if the Karush-Kuhn-Tucker conditions hold then x is a global optimum.*

Lecture 17

6th of December 2018

A. Frangioni

17.1 Duality

💡 Do you recall?

The KKT-G condition:

$$\nabla f(x) + \sum_{i \in \mathcal{I}} \lambda_i \nabla g_i(x) + \sum_{j \in \mathcal{J}} \mu_j \nabla h_j(x) = 0$$

We would like to understand what is the function of which this is the gradient.

Definition 17.1.1 (Lagrangian function). *Let (P) be our minimum problem over $f : \mathbb{R}^n \rightarrow \mathbb{R}$. We term **Lagrangian function** the following:*

$$L(x; \lambda, \mu) = f(x) + \sum_{i \in \mathcal{I}} \lambda_i g_i(x) + \sum_{j \in \mathcal{J}} \mu_j h_j(x)$$

Fact 17.1.1. *A necessary condition for optimality is that the gradient of the Lagrangian function is 0. Formally, $\nabla L(x; \lambda, \mu) = 0$.*

A first approach to use the Lagrangian function would be taking λ and μ such that the Langrangian is positive semidefinite.

But this is a too strict requirement, in fact we impose:

Definition 17.1.2 (Critical cone). *Let us assume (x, λ, μ) satisfies (KKT). We define the **critical cone** as*

$$C(x, \lambda, \mu) = \left\{ d \in \mathbb{R}^n : \begin{array}{ll} < \nabla g_i(x), d > = 0 & i \in \mathcal{A}(x) \text{ s.t. } \lambda_i^* > 0 \\ < \nabla g_i(x), d > \leq 0 & i \in \mathcal{A}(x) \text{ s.t. } \lambda_i^* = 0 \\ < \nabla h_j(x), d > = 0 & i \in \mathcal{J} \end{array} \right\}$$

Theorem 17.1.2. Let us assume we have a point (x, λ, μ) that satisfies the Karush-Kuhn-Tucker conditions and satisfies the linear independence of the constraints. If x is local optimum then $d^T \nabla_{xx}^2 L(x, \lambda, \mu)d \geq 0 \forall d \in C(x, \lambda, \mu)$.

Informally, if the hypothesis holds, then the Hessian of the Lagrangian function is $\succeq 0$ on the critical cone.

Observation 17.1.1. (x, λ, μ) satisfies $(KKT) \wedge \nabla_{xx}^2 L(x, \lambda, \mu) \succ 0$ on $C(x, \lambda, \mu)$ then x local optimum.

We would like to say something more about λ and μ . Until now we considered the Lagrangian as a function of x , but what if we consider the Lagrangian in terms of λ and μ ?

17.2 Lagrangian duality

Definition 17.2.1 (Lagrangian relaxation). Let us consider the Lagrangian function. We term **Lagrangian relaxation** the function where we fixed λ and μ and minimize on x :

$$\psi(\lambda, \mu) = \min\{L(x, \lambda, \mu) : x \in \mathbb{R}^n\}$$

The relaxation leads to an unconstrained problem and we learnt how to solve one of those.

This Lagrangian relaxation leads to the definition of **lagrangian dual** ψ .

Property 17.2.1. The dual function has the following properties:

- ψ is concave, but $\psi(\lambda, \mu) = -\infty$;
- ψ is non differentiable, although f , g_i and h_j are;
- Let \bar{x} be optimal in $(R_{\lambda, \mu})$, then $[\sum_{i \in \mathcal{I}} \nabla g_i(\bar{x}) + \sum_{j \in \mathcal{J}} \nabla h_j(\bar{x})] \in \partial\psi(\lambda, \mu)$
- \forall fixed $\lambda \geq 0, \mu, \bar{x} \in X \quad \psi(\lambda, \mu) = \min_x L(x, \lambda, \mu) \leq L(\bar{x}, \lambda, \mu) \leq f(\bar{x})$

$$\psi(\lambda, \mu) = \begin{pmatrix} \lambda_1 g_1(\bar{x}) \\ \vdots \\ \lambda_k g_k(\bar{x}) \\ \lambda_1 \mu_1(\bar{x}) \\ \vdots \\ \lambda_p \mu_p(\bar{x}) \end{pmatrix} \text{ is such that } \begin{pmatrix} g_1(\bar{x}) \\ \vdots \\ g_k(\bar{x}) \\ \mu_1(\bar{x}) \\ \vdots \\ \mu_p(\bar{x}) \end{pmatrix} \text{ belongs to the supergradient.}$$

Theorem 17.2.2 (Weak duality). \forall fixed $\lambda \in \mathbb{R}^+ \cup \{0\}, \mu \in \mathbb{R}$ such that $\psi(\lambda, \mu) \leq v(P)$ and let us take any feasible $\bar{x} \in X$ and $g(\bar{x}) \leq 0, h(\bar{x}) = 0$.

It can be proved that $\psi(\lambda, \mu) = \min_x L(x, \lambda, \mu) \leq L(\bar{x}, \lambda, \mu) \leq f(\bar{x})$.

Proof. $L(x, \lambda, \mu) = f(x) + \sum_i \lambda_i g_i(x) + \sum_j \mu_j h_j(x)$. Let us assume we have some $\bar{\lambda}, \bar{\mu}, \bar{x}$, where \bar{x} is feasible (aka $g_i(\bar{x}) \leq 0$ and $h_i(\bar{x}) = 0$ and $\bar{\lambda} \geq 0$).

The value of the dual function is

$$\psi(\bar{\lambda}, \bar{\mu}) = \min_x \{f(x) + \sum_i \lambda_i g_i(x) + \sum_j \mu_j h_j(x)\} \leq f(\bar{x}) + \sum_i \bar{\lambda}_i g_i(\bar{x}) + \sum_j \bar{\mu}_j h_j(\bar{x}),$$

where the last term is cancelled, since $h_i(\bar{x}) = 0$.

On the other hand, $\sum_i \bar{\lambda}_i g_i(x) \leq 0$, which implies that $f(x) \leq 0$. \square

Observation 17.2.1. This theorem gives us the way to prove that a point is an optimum. Let us assume that we found $\lambda_* \leq 0$, x_* and μ_* such that an equality holds: $\psi(\lambda_*, \mu_*) = f(x_*)$. The value of ψ gives us the information about how far we are from the optimum. Let us assume we have $\bar{\lambda}, \bar{\mu}$ and \bar{x} then $\psi(\bar{\lambda}, \bar{\mu}) \leq f(\bar{x}) \leq \psi(\bar{\lambda}, \bar{\mu}) + \varepsilon$ and this tells that we are far from the optimum of a factor ε .

If everything in the original function was convex then the Lagrangian is convex and the computations are easy.

At this point we want to find the biggest lower bound and it may be computed since the function is concave.

$$\max\{ \psi(\lambda, \mu) : \lambda \in \mathbb{R}_+^{|\mathcal{I}|}, \mu \in \mathbb{R}^{|\mathcal{T}|} \}$$

Notice that the constraints on λ are very easy, so the problem may be considered somehow unconstrained.

We can use local methods to compute the maximum of this function.

If we are able to compute the gradient of ψ then we have the supergradients of the function f .

Is (P) equal to (D)? Yes, provided that everything is convex. In general the Lagrangian gives a lowerbound.

$$v(D) \leq v(P)$$

Example 17.2.1. Let us take a concave objective function $\min\{-x^2 : 0 \leq x \leq 1\}$, such that its Lagrangian function is $L(x, \lambda) = -x^2 + \lambda_1(x-1) - \lambda_2 x$. It goes without saying that the Lagrangian function is unbounded below (upside-down parabola).

Formally, $\psi(\lambda) = \min_{x \in \mathbb{R}} L(x, \lambda) = -\infty$, $\forall \lambda \in \mathbb{R}^2$, which means that the Lagrangian dual is $v(D) = -\infty < v(P) = -1$.

Theorem 17.2.3. Let us assume that f , g and h are convex, and constraints qualification holds. If the problem has an optimum then the value of the primal and the value of the dual are the same.

$$\text{Formally, } T_X(x_*) D_X(x_*) \Rightarrow v(D) = v(P).$$

Proof. We are assuming x_* is an optimum, so the necessary conditions hold, hence we have the Karush Khun Tucker conditions, then we can find (λ^*, μ^*) that satisfy the KKT with x .

Then our claim is that (λ^*, μ^*) is an optimal solution of the dual (D) and the value of the dual is equal to the value of the primal.

x_* is a stationary point for the Lagrangian function, since it satisfies the KKT conditions then x_* is exactly the minimum, since the Lagrangian function is convex.

Hence, the value of the dual $v(D) \geq \psi(\lambda^*, \mu^*) = L(x_*, \lambda^*, \mu^*) = f(x_*) = v(P) \geq v(D)$. \square

How many solution may $\psi(\lambda, \mu)$ have? In principle many, but what if f is strongly convex, than everything in the Lagrangian is strongly convex, then the solution of ψ is unique and it is also differentiable, but typically not two times differentiable. At this point the Lagrangian dual has a single solution and the optimum of the Lagrangian dual corresponds to an optimum for f .

We only translated a minimum problem on convex functions to another minimum problem on other convex functions. We will see soon that in some cases this approach is advantageous in others it is not.

17.3 Specialized dual

17.3.1 Linead programs

$$(P) \quad \min \{ cx : Ax \geq b \}$$

$$\text{Lagrangian function: } L(x, \lambda) = cx + \lambda(b - Ax) = \lambda b + (c - \lambda A)x$$

Dual function:

$$\psi(\lambda) = \min_{x \in \mathbb{R}^n} L(x, \lambda) = \begin{cases} -\infty & \text{if } c - \lambda A \neq 0 \\ \lambda b & \text{if } c - \lambda A = 0 \end{cases}$$

Since we can find a minimum only on a function of the second case we have that:

$$(D) \quad \max \{ \psi(\lambda) : \lambda \geq 0 \} \equiv \max \{ \lambda b : \lambda A = c, \lambda \geq 0 \}$$

Since the lagrangian is far from having an unique optimal solution the dual does not have a unique maximum so the do not match usually.

17.3.2 Quadratic programs

Notice that the quadratic case is simpler than the linear case.

$$(P) \quad \min \left\{ \frac{1}{2} \|x\|_2^2 : Ax = b \right\} \text{ (linear least-norm solution)}$$

$L(x, \mu) = \frac{1}{2} \|x\|_2^2 + \mu(Ax - b)$, $\nabla_x L = x + \mu A = 0 \iff x = -\mu A$, which is strictly convex since x^2 is strictly convex.

$$\psi(\mu) = \min_{x \in \mathbb{R}^n} L(x, \mu) = L(-\mu A, \mu) = -\frac{1}{2} \mu^T (AA^T) \mu - \mu b$$

Then the Lagrangian dual (D) $\max \left\{ -\frac{1}{2} \mu^T (AA^T) \mu - \mu b : \mu \in \mathbb{R}^m \right\}$

We can generalize with quadratic functions for general problems:

Strictly convex QP: (P)

$$\min \left\{ \frac{1}{2} x^T Q x + q x : Ax \geq b \right\}, Q \succ 0 \implies (D) \quad \max \left\{ \lambda b - \frac{1}{2} v^T Q^{-1} v : \lambda A - v = q, \lambda \geq 0 \right\} \text{ strong duality} \equiv v(P) = v(D) \text{ (almost) always holds}$$

17.3.3 Conic program

We can do duals of things that are not quadratic programs. Sometimes we want to have non linear things to be able to draw different shapes.

Definition 17.3.1 (Conic program). *We term **conic program** $\min\{cx : Ax \geq_K b\}$, where $x \geq_K y \equiv x - y \in K$, where K is a convex cone.*

Example 17.3.1. *It is easy to see that the \geq constraints we saw before are a special case of conic program, where the conic is \mathbb{R}_+^n .*

$$-x_1 - 2x_2 \leq 2$$

$$x_1 + x_2 \geq 1$$

$$2x_1 - x_2 \geq 0$$

Let us write $-x_1 - 2x_2 - 2 = s_0$, $x_1 + x_2 - 1 = s_1$ and $2x_1 - x_2 = s_2$, such that $s_0, s_1, s_2 \geq 0$.

We have a mapping from R^n (where the variables live) to \mathbb{R}^m (where the constraints live) and $Ax - b \in K$.

Rather than writing $s_0, s_1, s_2 \geq 0$ we could write $s_0 \geq \sqrt{s_1^2 + s_2^2}$ and this would still be a conic program.

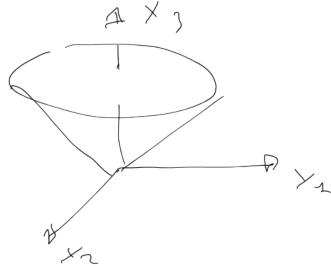


FIGURE 17.1: $x^3 \geq \sqrt{x_1^2 + x_2^2}$ is a convex cone.

The idea is to hide the non linear part in the cone, in particular in \geq_K . Let us see what happens to cones depending on the function.

There are three interesting cones:

- $K = \mathbb{R}_+^n \equiv$ sign constraints \equiv Linear Program;
- $K = \mathbb{L} = \{x \in \mathbb{R}^n : x_n \geq \sqrt{\sum_{i=1}^{n-1} x_i^2}\} \equiv$ Second-Order Cone (or Lorentz cone) Program, that generalizes linear programs;
- $K = \mathbb{S}_+ = \{A \succeq 0\} \equiv$ “ \succeq ” constraints \equiv SemiDefinite Program.

Given the problem that looks like linear except for the cone.

Definition 17.3.2 (Conic dual). *Conic dual: (D) $\max\{yb : yA = c, y \geq_{K^D} 0\}$, where $K^D = \{z : \langle z, x \rangle \geq 0 \ \forall x \in K\}$ is a dual cone.*

Sometimes constraints qualification does not hold, since the conic program is not linear sometimes.

Explicit form of second order cone program (SOCP) (“explicit data” D_i, d_i, p_i, q_i)
 $\min\{cx : \|D_i x - d_i\|_2 \leq p_i x - q_i, i = 1, \dots, m\}$.

It collapses to a linear program for $D_i = 0$ and $d_i = 0$.

It turns out that the dual has just this explicit form:

$$\max\left\{\sum_{i=1}^m \lambda_i d_i + \nu_i q_i : \sum_{i=1}^m \lambda_i D_i + \nu_i p_i = c, \|\lambda_i\|_2 \leq \nu_i, i = 1, \dots, m\right\}$$

where the ν_i are the dual variables of the rightmost part, while the λ_i are the dual variables of the leftmost part.

We can write the explicit form of semidefinit problems as $\min\{cx : \sum_{i=1}^n x_i A^i \succeq B\}$, where $A^i, B \in M(k, \mathbb{R})$.

There is an even more general form of duality, that we are going to introduce next.

17.4 Fenchel's duality

Definition 17.4.1 (Fenchel's conjugate). We denote **Fenchel's conjugate** of f $f^*(z) = \sup_x \{zx - f(x)\}$.

We can observe that f^* is always convex, even if f is not and it is closed if f is.

The following functions are such that f^* can be computed easily:

1. $f(x) = \frac{1}{2} \|x\|_2^2 \implies f^*(z) = \frac{1}{2} \|z\|_2^2$ (only function s.t. $f^* = f$);
2. $(\|\cdot\|_1)^*(z) = \beta_{B_\infty(0,1)}(z), (\|\cdot\|_\infty)^*(z) = \beta_{B_1(0,1)}(z)$;
3. $f(x) = \max\{g_i x - \alpha_i | i \in I\} \implies f^*(z) = \min \left\{ \sum_{i \in I} \alpha_i \theta_i : \sum_{i \in I} g_i \theta_i = z, \sum_{i \in I} \theta_i = 1, \theta_i \geq 0 | i \in I \right\}$.

Fact 17.4.1. If we are minimizing the sum of two convex functions (P) $\min\{f(x) + g(x)\}$, this problem is the same of maximizing the sum of “almost” the two conjugates: (D) $- \min\{f^*(z) + g^*(-z)\}$.

Lecture 18

12th of December 2018

A. Frangioni

18.1 Quadratic problem with linear equality constraints

Let A be a matrix in $M(m, n, \mathbb{R})$, where $m < n$ (otherwise the system is either fully determined or impossible) and $rk(A) = m$. The constrained quadratic problem may be written as

$$\min \left\{ \frac{1}{2}x^T Qx + qx : Ax = b \right\}$$

where $Q \succeq 0$.

A way to solve this problem is through Karusich Kuhn Tucker system:

$$\begin{array}{ll} (a) & \left[\begin{array}{cc} Q & A^T \\ A & 0 \end{array} \right] \left[\begin{array}{c} x \\ \mu \end{array} \right] = \left[\begin{array}{c} -q \\ b \end{array} \right] \\ (b) & \end{array} \quad (18.1.1)$$

where the first row (a) says that the gradient is a linear combination of the normals of the gradient of the constraints and the second one is just feasibility.

Everything is linear here. This system is symmetric, although indefinite, because it contains many 0s.

We are left with solving the KKT system:

REDUCED KKT: in this method we first add the hypothesis of non-singularity to the matrix Q so we get the following

$$\begin{cases} Qx + A^T \mu = -q & (a) \\ Ax = b & (b) \end{cases}$$

Multiply (a) by AQ^{-1}

$$\begin{cases} A\cancel{Q^{-1}}x + AQ^{-1}A^T\mu = -AQ^{-1}q & \text{(a)} \\ Ax = b & \text{(b)} \end{cases}$$

Multiply (b) by -1 and add to it (a)

$$\begin{cases} Ax + AQ^{-1}A^T\mu = -AQ^{-1}q & \text{(a)} \\ \cancel{Ax} + AQ^{-1}A^T\mu - \cancel{Ax} = -AQ^{-1}q - b & \text{(b)} \end{cases}$$

Multiply (a) by A^{-1}

$$\begin{cases} x + Q^{-1}A^T\mu = -Q^{-1}q & \text{(a)} \\ AQ^{-1}A^T\mu = -AQ^{-1}q - b & \text{(b)} \end{cases}$$

Isolate x

$$\begin{cases} x = -Q^{-1}(A^T\mu + q) & \text{(a)} \\ AQ^{-1}A^T\mu = -AQ^{-1}q - b & \text{(b)} \end{cases}$$

Notice that $0 \preceq AQ^{-1}A^T = M \in M(m, \mathbb{R})$ and may be much smaller than the original one, since its size depends on the number of constraints. The issue here is that the matrix M is less sparse than both A and Q .

NULL SPACE METHOD: In this method we need no assumption on Q .

First of all we rearrange the matrix A in order to have a small square matrix A_B and then the rest of the columns (A_N): $A = [A_B, A_N]$, $x = [x_B, x_N]$, $\det(A_B) \neq 0$.

Replacing A in (b) we get $A_Bx_B + A_Nx_N = b \iff x_B = A_B^{-1} \cdot (b - A_Nx_N)$, hence resulting in $x = Dx_N + d$, where

$$d = \begin{bmatrix} b \\ 0 \end{bmatrix}, D = \begin{bmatrix} -A_B^{-1}A_N \\ I \end{bmatrix} \in M(m, n-m)$$

Notice that D is a basis of the **null space** (or kernel) of A and it is not mandatory to build it like we did above, it is only important to obtain a basis of the kernel.

Let us multiply (a) by D^T and obtain

$$\begin{aligned} D^T(Qx + A^T\mu) &= -D^Tq \\ D^TQx + D^TA^T\mu &= -D^Tq \\ D^TQx + (\cancel{AD})^T\mu &= -D^Tq \\ D^TQ(Dx_N + d) &= -D^Tq \end{aligned} \tag{18.1.2}$$

Where in the last step we applied the definition $x = Dx_N + d$ and hence, $(D^T Q D)x_N = -D^T(Qd + q)$.

We term $H = D^T Q D \in M(n-m, \mathbb{R})$ the reduced Hessian of the problem and notice that whenever the number of constraints is close to the number of variables this matrix is very small.

It is important to note that in order to solve an equality constrained problem we can choose either the reduced KKT method or the null space method, depending on the structure of our problem.

18.2 Inequality constrained problems

The problem, in this case, is written as follows

$$(P) \min\{f(x) : Ax \leq b\}$$

18.2.1 Projected gradient method

The intuition behind this kind of algorithm is that we are interested in finding a direction for the line search not the opposite of the gradient (because then the step size should be put to 0 if we lie on the boundaries, see Figure 18.1), but it is enough to pick a direction which has a negative scalar product with the gradient.

The rationale is to pick the direction that minimizes the norm of the difference with the gradient. Formally

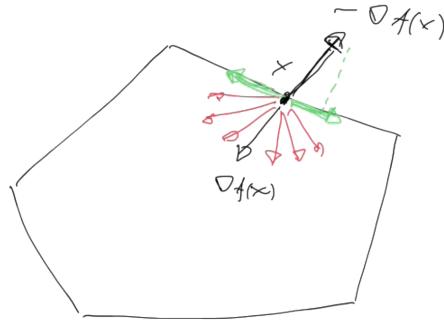


FIGURE 18.1: Geometric idea of how to choose the direction in case of lying on the boundary.

$$d = \operatorname{argmin}\{\|\nabla f(x) - d\|^2 : d \in \mathcal{D}_X(x)\}$$

where $\mathcal{D}_X(x)$ is the feasible cone. Notice that if the difference between the gradient and the direction is 0 it means that we can stop, since the descent direction would bring us outside the feasible set.

A more formal definition of the feasible cone $\mathcal{D}_X(x) = \{d \in \mathbb{R}^n : A_{\mathcal{A}(x)}d \leq 0\}$, where $\mathcal{A}(x)$ is the set of all the active constraints.

For sake of clarity we denote $\bar{A} = A_{\mathcal{A}(x)}$.

At this point we are ready to project the problem in such a way that inequality constraints become equality constraints:

$$\min \left\{ \frac{1}{2} \|\nabla f(x) + d\|^2 = \frac{1}{2} d^T Id + \nabla f(x)d : \bar{A}d = 0 \right\}$$

Note that in this case the Hessian is the identity matrix $I \succ 0$, hence the KKT conditions become simpler:

$$\begin{cases} Qx + A^T \mu = -q \iff d + A^T \mu = -\nabla f(x) \\ Ad = 0 \end{cases}$$

since $x = d$, $A = I$, $b = \nabla f(x)d$, so we get

$$\begin{aligned} d &= -\nabla f(x) - A^T \mu \\ Ad &= -A\nabla f(x) - AA^T \mu \\ AA^T \mu &= -A\nabla f(x) \end{aligned} \tag{18.2.1}$$

resulting in

$$\begin{cases} Qx + A^T \mu = -q \iff d + A^T \mu = -\nabla f(x) \\ Ad = 0 \end{cases}$$

Therefore, we need to solve a system in μ and then we have the direction. If the number of active constraints is small, solving the system is very fast.

Fact 18.2.1. *We can restrict to solve*

$$\begin{cases} \mu = -(\bar{A}\bar{A}^T)^{-1}\bar{A}\nabla f(x) \\ d = (I - \bar{A}^T(\bar{A}\bar{A}^T)^{-1}\bar{A})(-\nabla f(x)) \end{cases}$$

Proof. \bar{A} has full row rank, then is non singular and may be inverted.

$$\begin{cases} \mu = -(\bar{A}\bar{A}^T)^{-1}\bar{A}\nabla f(x) \\ d = -\nabla f(x) + \bar{A}^T(\bar{A}\bar{A}^T)^{-1}\bar{A}\nabla f(x) \end{cases} \iff \begin{cases} \mu = -(\bar{A}\bar{A}^T)^{-1}\bar{A}\nabla f(x) \\ d = (I - \bar{A}^T(\bar{A}\bar{A}^T)^{-1}\bar{A})(-\nabla f(x)) \end{cases}$$

□

If the set of active constraints contains some linear dependent ones, we take the maximal subset of our constraints such that the matrix \bar{A} has full rank.

This procedure is formalized in Algorithm 18.2.1, where at each stage we remove one of the constraints that lead to a negative μ , keeping a set of linearly independent constraints.

In the 14-th line of the algorithm we find that our step size is takes as the minimum step size among the ones that satisfy some subsets of the constraints.

ALGORITHM 18.2.1 Pseudocode for projected gradient method for quadratic functions.

```

1: procedure PGM( $f, A, b, x, \varepsilon$ )
2:   for ( $; ;$ ) do
3:      $B \leftarrow$  maximal  $\subseteq \mathcal{A}(x)$  s.t.  $\text{rank}(A_B) = |B|$ ;
4:     for ( $; ;$ ) do
5:        $d \leftarrow (I - A_B^T (A_B A_B^T)^{-1} A_B)(-\nabla f(x))$ ;
6:       if  $\langle \nabla f(x), d \rangle \leq \varepsilon$  then
7:          $\mu_B \leftarrow -(A_B A_B^T)^{-1} A_B \nabla f(x)$ ;
8:          $\mu_i \leftarrow 0 \forall i \notin B$ ;
9:         if  $\mu_B \geq 0$  then return
10:        end if
11:         $h \leftarrow \min\{i \in B : \mu_i < 0\}$ ;
12:      end if
13:       $\bar{\alpha} \leftarrow \min\{\alpha_i = (b_i - A_i x)/A_i d : A_i d > 0, i \notin B\}$ ;
14:       $x \leftarrow x + \alpha d$ ;
15:      if  $\bar{\alpha} > 0$  then
16:        break;
17:      end if
18:       $k \leftarrow \min\{i \notin B : A_i d > 0 : \alpha_i = 0\}$ ;
19:       $B \leftarrow B \cup \{k\}$ ;
20:    end for
21:     $\alpha \leftarrow \text{Line\_Search}(f, x, d, \bar{\alpha})$ ;
22:     $x \leftarrow x + \alpha d$ ;
23:  end for
24: end procedure

```

Fact 18.2.2. *The following holds:*

1. $d = 0 \wedge \mu \geq 0 \implies x$ optimal (from KKT);
2. $d = 0 \wedge \exists h \in B$ s.t. $\mu_h < 0 \implies \exists x' \in \{x \in \mathbb{R}^n : A_{B \setminus \{h\}} x = b_{B \setminus \{h\}}, A_h x \leq b_h\}$ s.t. $f(x') < f(x)$;
3. $d \neq 0$ descent direction: $H = I - A_B^T [A_B A_B^T]^{-1} A_B$ symmetric and idempotent $HH = H^T H = H \implies \langle d, \nabla f(x) \rangle < 0$.

Proof.

trivial;

- The intuition is that if we remove h from B next time we will have a descent direction.

$$\exists d \text{ s.t. } A_{B \setminus \{h\}}d = 0 \wedge A_h d < 0 \implies \langle \nabla f(x), d \rangle = \langle -\mu A_B, d \rangle = -\mu_h A_h d < 0;$$

- $\langle d, \nabla f(x) \rangle = \langle -H \nabla f(x), \nabla f(x) \rangle = -\nabla f(x)^T H \nabla f(x) = -(H \nabla f(x))^T H \nabla f(x)$.

□

Once we found the optimal face we move inside that face and we are dealing with a steepest descent, which is slow.

At a certain point of the execution the set B of the active constraints will stabilize and become the one of the optimal solution.

At this point the smart thing to do is to use the KKT conditions, since we know the set of active constraints. This idea is pursued in Section 18.2.3.

18.2.2 Projected gradient method with box constraints

Given the non linear minimum problem of the following shape

$$(P) \min \left\{ f(x) : l \leq x \leq u \right\}$$

a slightly different version of the projected gradient method forces the algorithm to put to 0 the component of the direction which would bring us to go outside the feasible region. This intuition is formalized in Algorithm 18.2.2.

ALGORITHM 18.2.2 Pseudocode for projected gradient method for quadratic functions in the case of box constraints.

```

1: procedure PGMBC( $f, l, u, x, \varepsilon$ )
2:    $d = -\nabla f(x)$ ;
3:    $\bar{\alpha} = \infty$ ;
4:   for ( $i = 1 \dots n$  s.t.  $d_i \neq 0$ ) do
5:     if ( $d_i < 0$ ) then
6:       if ( $x_i = l_i$ ) then
7:          $d_i = 0$ ;
8:       else
9:          $\bar{\alpha} \leftarrow \min\{\bar{\alpha}(x_i - l_i)/d_i\}$ ;
10:      end if
11:    else
12:      if ( $x_i = u_i$ ) then
13:         $d_i = 0$ ;
14:      else
15:         $\bar{\alpha} \leftarrow \min\{\bar{\alpha}(u_i - x_i)/d_i\}$ ;
16:      end if
17:    end if
18:    if ( $\langle \nabla f(x), d \rangle \leq \varepsilon$ ) then return
19:    end if
20:     $\alpha \leftarrow \text{Line\_Search}(f, x, d, \bar{\alpha})$ ;
21:     $x \leftarrow x + \alpha d$ ;
22:  end for
23: end procedure

```

Notice that we can assume that $l_i < u_i \forall i$, because otherwise that component would be fixed.

18.2.3 Active-set method for quadratic programs

Let us be given the following minimum problem

$$\min \left\{ \frac{1}{2} x^T Q x + q x : Ax \leq b \right\}$$

where an important hypothesis is that the problem is quadratic, once we know $A(x_*)$.

We start from a certain point, such that the constraints are satisfied as equalties. According to KKT conditions, a solution \bar{x} is optimal if \bar{x} is feasible and $\mu \geq 0$. Otherwise, if μ is not positive we eliminate something from the active set and start again. In the case of x unfeasible, we know the descent direction, we only need to revise the step size.

All this machinery is formalized in Algorithm 18.2.3.

ALGORITHM 18.2.3 Pseudocode for active set method for quadratic functions.

```

1: procedure ASMQP( $Q, q, A, b, x, \varepsilon$ )
2:   for ( $B \leftarrow \mathcal{A}(x); ;$ ) do
3:     solve ( $P_B$ )  $\min\{\frac{1}{2}x^T Qx + qx : A_Bx = b_B\}$  for
   ( $\bar{x}, \bar{\mu}_B$ );
4:     if ( $A_i\bar{x} \leq b_i \forall i \notin B$ ) then
5:       if ( $\mu_B \geq 0$ ) then return
6:       end if
7:        $h \leftarrow \min\{i \in B : \mu_i < 0\};$ 
8:        $B \leftarrow B \setminus \{h\};$ 
9:       continue;
10:      end if
11:       $d \leftarrow \bar{x} - x;$ 
12:       $\bar{\alpha} \leftarrow \min\{\alpha_i = (b_i - A_i x) / A_i d : A_i d > 0, i \notin B\};$ 
13:       $x \leftarrow x + \bar{\alpha}d;$ 
14:       $B \leftarrow \mathcal{A}(x);$ 
15:   end for
16: end procedure

```

Notice that this algorithm allows easy solving of the same problem, under box constraints, because we end up having two sets: L , indexes of variables fixed to the lower bound, and U , indexes of variables fixed to the upper bound.

$\{1, \dots, n\} \setminus (L \cup U)$ is the set of the indexes of the free variables.

Thanks to this consideration the problem to solve becomes

$$\min \left\{ \frac{1}{2}x_F^T Q_{FF}x_F + (q_F + u_U^T Q_{UF})x_F \right\} \left[+ \frac{1}{2}x_U^T Q_{UU}x_U + q_U u_U \right]$$

18.2.4 Frank-Wolfe method

Let us take a non-linear function and the following problem

Supposing f is convex we can make a first order model and minimize it over our constraints.

In this case we use the right constraints and an approximation of the function.

ALGORITHM 18.2.4 Pseudocode for Frank-Wolf method for non linear functions.

```
1: procedure FWM( $f, A, b, x, \varepsilon$ )
2:   while ( $\|\nabla f(x)\| > \varepsilon$ ) do
3:      $\bar{x} \leftarrow \operatorname{argmin} \{\langle \nabla f(x), y \rangle : Ay \leq b\};$ 
4:      $d \leftarrow \bar{x} - x;$ 
5:      $\alpha \leftarrow \text{Line\_Search}(f, x, d, 1);$ 
6:      $x \leftarrow x + \alpha d;$ 
7:   end while
8: end procedure
```

Provided that the computation at line 3 of Algorithm 18.2.4 is performed efficiently, then either $d = 0$ and we are in the optimum or $d > 0$ and we are moving toward the optimum.

The linear model is below the function in the quadratic case, hence we get a lower bound.

Lecture 19

14th of December 2018

A. Frangioni

In the previous lecture we addressed the problem of linear constrained optimization. Our first approach was to deal very little with constraints (projected gradient method), after a few improvements we took all of them and modify the function (Frank-Wolfe method).

19.1 Dual methods for linear constrained optimization

In this class of methods constraints are first class citizens.

Let us be given the following optimization problem:

$$\min \left\{ \frac{1}{2}x^T Qx + qx : Ax \leq b \right\}$$

We would like to work with dual feasibility.

\forall fixed $\lambda \geq 0$, this is the lagrangian problem $\psi(\lambda) = \min_x \{ \frac{1}{2}x^T Qx + qx + \lambda(b - Ax) \} \leq v$ and the Lagrangian dual is the following

$$(D) \max \{ \psi(\lambda) : \lambda \geq 0 \}$$

which is equivalent to the primal problem.

The solution of the Lagrangian problem is **unique** and this is due to the fact that ψ is concave and $Q \succ 0$. The optimal solution is then $x(\lambda) = Q^{-1}(\lambda A - q)$.

The function is differentiable in the solution, because we have only one sub (or super) gradient which is $\nabla \psi(\lambda) = b - Ax(\lambda)$.

At this point we only need to solve the dual problem, which has the shape of a box constrained optimization, where the box has only one boundary and this can be solved via quasi-Newton methods.

Notice that $\psi \notin C^2$ so the Hessian is not defined.

This dual approach is advantageous in the case of a small number of constraints, because the size of the problem decreases. For example, in the case of one constraint, the Lagrangian dual becomes a problem in one variable, hence solvable through a line search.

In this method the degenerate case (more than one constraint active at a time) is not an issue.

If the quadratic function is convex we may use quasi Newton method. Otherwise the global optimality is not guaranteed and may be used if we accept not to be able to solve the original problem, but only to find a lower bound.

19.1.1 Separable problems and partial dual

Let us assume that our constraints are separable, which means that it is not mandatory to work with all of them, but they can be splitted into constraints of groups of variables.

$$\min\{f(x) : Ax \leq b, Ex \leq d\}$$

We can decide to use a **partial dual**, writing the Lagrangian problem picking only some constraints that we chose:

$$\psi(\lambda) = \min_x \{f(x) + \lambda(b - Ax) : Ex \leq d\}$$

The Lagrangian dual method may be better than projected gradient or worse and it depends on the instance.

In the dual approaches we can't move inside the feasible solution. We find an optimum for the dual, which surely breaks feasibility. Then, if the variable is above the upperbound it gets decreased to the upper bound, otherwise if it is below the lower bound it takes the value of the lower bound.

This way we get an upperbound for the function to be minimized.

19.2 Primal/dual methods or barrier methods

This kind of methods are designed to overcome the cons of dual approaches, namely the fact that ψ does not have the Hessian (and this creates problems to quasi Newton method) and the fact that x is not feasible until the end.

At the same time this methods keep the unconstrained property of the Lagrangian dual.

19.2.1 Barrier function and central path

The rationale behind this algorithm is to minimize a function which penalizes the value of the original function when the solution is getting closer and closer to the boundaries of the feasible set.

$$(P_\mu) \min\{f_\mu(x) = f(x) - \mu \sum_{i=1}^m \log(b_i - A_i x)\}$$

The parameter μ is there to weight the proximity to the boundary.

Property 19.2.1. • If f is convex, f_μ is strictly convex;

- If $f \in \mathcal{C}^2$ then $f_\mu \in \mathcal{C}^2$, since $\log \in \mathcal{C}^\infty$;
- $\forall \mu \exists! x_\mu$ optimal of (P_μ) , since $\mu \sum_{i=1}^m \log(b_i - A_i x)$ is strictly convex;
- As $\mu \rightarrow 0$ x_μ converges to the analytic center of the optimal face. An example of this behaviour may be seen in Figure 19.1.

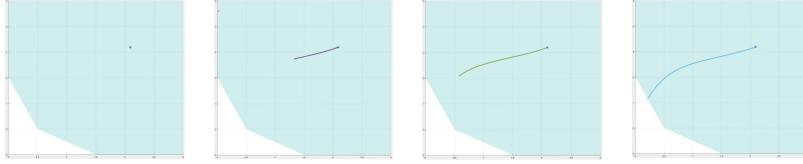


FIGURE 19.1: The trajectory converges to the optimal solution of the problem, when $\mu \rightarrow 0$.

Another interesting property is that, since the barrier function is **self concordant** x^i gets “close” to $x(\mu^i)$ in very few Newton’s steps.

In one step x^{i+1} is “much closer” to $x(\mu^i)$ than x^i was and x^{i+1} is “close” to $x(\mu^{i+1})$, with $\mu^{i+1} \ll \mu^i$ (more formally, $\mu^{i+1} = \tau \mu^i$, $\tau < 1$).

This behaviour may be observed in Figure 19.2

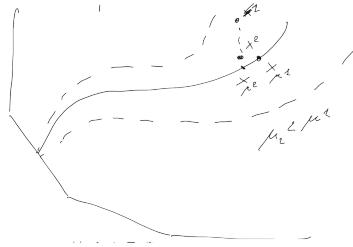


FIGURE 19.2: The dotted line represents a region where the Newton method is very efficient. We are starting from a point x_1 , which belongs to that region and we want to move towards x_{μ^1} . At next iterate x_2 is closer to x_{μ^2} than the current iterate.

The convergence is exponential if τ is very small, but these iterations are very costly, because the Hessian changes at each step, hence it needs to be recomputed.

Let us focus on computing the **Newton's step**.

First, we write the Karush-Kuhn-Tucker conditions:

PRIMAL FEASIBILITY: $Ax + s = b, s \geq 0;$

DUAL FEASIBILITY: $Qx + \lambda A = -q, \lambda \geq 0;$

COMPLEMENTARY SLACKNESS: $\lambda_i s_i = 0, i = 1, \dots, m.$

We can write a slackened version of KKT, imposing $\lambda_i s_i = \mu, i = 1, \dots, m,$ where $\mu \in \mathbb{R}^n$ and should be decreased over iterations until it gets closer enough to 0.

Let us construct $\Lambda, S \in \text{Diag}(m, \mathbb{R})$ such that the diagonal is made of λ_i and s_i respectively.

At this point, we rewrite the problem in terms of the displacement from the fixed current point we are in:

- $x \rightarrow x + \Delta x$
- $s \rightarrow s + \Delta s$
- $\lambda \rightarrow \lambda + \Delta \lambda$

The KKT system becomes:

PRIMAL FEASIBILITY: $Ax + A\Delta x + s + \Delta s = b, s \geq 0;$

DUAL FEASIBILITY: $Qx + Q\Delta x + \lambda A + \Delta \lambda A = -q, \lambda \geq 0;$

COMPLEMENTARY SLACKNESS: $\lambda_i s_i + \lambda_i \Delta s_i + s_i \Delta \lambda_i + \Delta \lambda_i \Delta s_i = \mu, i = 1, \dots, m.$

In this new system of coordinates the first two KKT remain linear, while the third one is no longer linear ($\Delta \lambda_i \Delta s_i$).

$$\begin{bmatrix} Q & A^T & 0 \\ A & 0 & I \\ 0 & S & \Lambda \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta s \end{bmatrix} \stackrel{(1)}{=} \begin{bmatrix} -(Qx + q) - \lambda A \\ b - Ax - s \\ \mu u - \Lambda S u - \Delta \Lambda \Delta S u \end{bmatrix} \approx \begin{bmatrix} 0 \\ 0 \\ \mu u - \Lambda S u \end{bmatrix} \quad (19.2.1)$$

Where (1) holds since $\Delta \lambda A = A^T \Delta \lambda^T$, although $\Delta \lambda$ is written without the “transpose” syntax to ease notation.

Notice that $u = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \in \mathbb{R}^m$ and has the purpose of adjusting dimension:

$$S \Delta \lambda = \begin{bmatrix} s_1 \Delta \lambda_1 \\ \vdots \\ s_m \Delta \lambda_m \end{bmatrix} \in \mathbb{R}^m; \Lambda \Delta s = \begin{bmatrix} \lambda_1 \Delta s_1 \\ \vdots \\ \lambda_m \Delta s_m \end{bmatrix} \in \mathbb{R}^m; \mu u = \begin{bmatrix} \mu \\ \vdots \\ \mu \end{bmatrix} \in \mathbb{R}^m;$$

$$\Lambda S u = \begin{bmatrix} -s_1 \lambda_1 \\ \vdots \\ -s_m \lambda_m \end{bmatrix} \in \mathbb{R}^m; \quad \Delta \Lambda \Delta s u = \begin{bmatrix} -\Delta \lambda_1 \Delta s_1 \\ \vdots \\ -\Delta \lambda_m \Delta s_m \end{bmatrix} \in \mathbb{R}^m;$$

The Newton method can be applied if we discard the non-linear part (written in red), pretending it does not exist. Notice that vector u is the vector of all 1s.

19.3 Primal-dual interior point method

This method is based on the observation that we can solve the dual problem

$$(D) \quad \max \left\{ -\lambda b - \frac{1}{2} x^T Q x : Qx + \lambda A = -q, \lambda \geq 0 \right\}$$

thus obtaining both a lower and upper bound for the solution x .

We term **complementarity gap** $(\frac{1}{2} x^T Q x + q x) - (-\lambda b - \frac{1}{2} x^T Q x) = \lambda s = \mu$.

Once we found a solution for Equation (19.2.1), we perform a step and compute a new couple of primal and dual solutions and reduce the gap μ .

We are left with solving Equation (19.2.1). The trick is to express one between Δs and $\Delta \lambda$ as a linear combination of the other.

For example, let us take the third line of Equation (19.2.1):

$$\begin{aligned} 0 \Delta x + S \Delta \lambda + \Lambda \Delta s &= \mu u - \Lambda S u \\ \Lambda \Delta s &= \mu u - \Lambda S u - S \Delta \lambda \\ \Delta s &= \Lambda^{-1} \mu u - \cancel{\Lambda^{-1} \Lambda S u} - \Lambda^{-1} S \Delta \lambda \\ \Delta s &= \Lambda^{-1} \mu u - S u - \Lambda^{-1} S \Delta \lambda \\ \Delta s &= \Lambda^{-1} \cdot (\mu u - S \Delta \lambda) - S u \\ \Delta s &= \Lambda^{-1} \cdot (\mu u - S \Delta \lambda) - s \end{aligned} \tag{19.3.1}$$

We obtain the modified normal equations (or KKT system)

$$\begin{bmatrix} Q & A^T \\ A & -\Lambda^{-1} S \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \end{bmatrix} = \begin{bmatrix} 0 \\ s - \mu \Lambda^{-1} u \end{bmatrix} \tag{19.3.2}$$

where the last row is derived working on the second row of Equation (19.2.1) ($A \Delta x + 0 \Delta \lambda + I \Delta s = 0 \Leftrightarrow A \Delta x = -\Delta s$), substituting Equation (19.3.1):

$$\begin{aligned} A \Delta x - [\Lambda^{-1} S] \Delta \lambda &= -\Delta s - [\Lambda^{-1} S] \Delta \lambda \\ &= -\Lambda^{-1} (\mu u - S \Delta \lambda) + s - \Lambda^{-1} S \Delta \lambda \\ &= -\Lambda^{-1} \mu u + \cancel{\Lambda^{-1} S \Delta \lambda} + s - \cancel{\Lambda^{-1} S \Delta \lambda} \\ &= s - \mu \Lambda^{-1} u \end{aligned} \tag{19.3.3}$$

With respect to normal equations of ??, we have in position (2, 2) a quantity $(-\Lambda^{-1}S)$, which is not 0, but it is the opposite of a strictly positive definite matrix.

A possible approach for solving this system is making some calculations and obtain something of the shape of reduced KKT (see Section 18.1).

$$\begin{cases} Q\Delta x + A^T\Delta\lambda = 0 \\ (Q + A^T\Lambda S^{-1}A)\Delta x = A^T(\lambda - \mu S^{-1}u) \end{cases} \quad (19.3.4)$$

where the first set of equations follows from the expansion of the first row of the KKT system (Equation (19.3.2)) and the second one is obtained taking the same row of the same system and substituting the value of $\Delta\lambda$ as follows:

$$\begin{aligned} A\Delta x - \Lambda^{-1}S\Delta\lambda &= s - \mu\Lambda^{-1}u \\ \Lambda^{-1}S\Delta\lambda &= A\Delta x - s + \mu\Lambda^{-1}u \\ \Delta\lambda &= (\Lambda^{-1}S)^{-1}A\Delta x - (\Lambda^{-1}S)^{-1}s + (\Lambda^{-1}S)^{-1}\mu\Lambda^{-1}u \\ \Delta\lambda &= S^{-1}\Lambda A\Delta x - S^{-1}\Lambda s + \mu S^{-1}\Lambda\Lambda^{-1}u \\ \Delta\lambda &= S^{-1}\Lambda A\Delta x - S^{-1}\Lambda s + \mu S^{-1}u \\ \Delta\lambda &= \mu S^{-1}u + \Lambda S^{-1}A\Delta x - \Lambda S^{-1}s \\ \Delta\lambda &= \mu S^{-1}u + \Lambda S^{-1}A\Delta x - \Lambda u \\ \Delta\lambda &= \mu S^{-1}u + \Lambda S^{-1}A\Delta x - \lambda \end{aligned} \quad (19.3.5)$$

Hence,

$$\begin{aligned} Q\Delta x + A^T\Delta\lambda &= 0 \\ Q\Delta x + A^T \cdot (\mu S^{-1}u + \Lambda S^{-1}A\Delta x - \lambda) &= 0 \\ Q\Delta x + A^T\mu S^{-1}u + A^T\Lambda S^{-1}A\Delta x - A^T\lambda &= 0 \\ Q\Delta x + A^T\Lambda S^{-1}A\Delta x &= -A^T\mu S^{-1}u + A^T\lambda \\ (Q + A^T\Lambda S^{-1}A)\Delta x &= A^T(\lambda - \mu S^{-1}u) \end{aligned} \quad (19.3.6)$$

We term $M = Q + A^T\Lambda S^{-1}A$ and the following holds.

Fact 19.3.1. *If A has full column rank (aka it is invertible), $M \succ 0$.*

At this point we need to factorize the matrix M , that changes at each iteration (since ΛS^{-1} does) and this is the bottleneck.

Cholesky factorization may be used, although its complexity is cube. Another downside of this approach is that the matrix M is much denser than A , Λ and S^{-1} .

An orthogonal approach to the reduced KKT is called **predictor-corrector** and it works computing a solution without taking into account the non linear term $\Delta\Lambda\Delta Su$, then computing it according to the approximated solution and repeat until convergence.

The bottleneck again is solving the system in Equation (19.2.1).

For what concerns implementation, we should start from a triplet (x, λ, s) , that could be not feasible and then compute the residuals and iterate until feasibility is reached.

$$r^D = -(Qx + q) - \lambda A, r^P = b - Ax - s.$$

When dealing with the step size we need to highlight the fact that $\lambda + \Delta\lambda \geq 0$, $s + \Delta s \geq 0$ should hold.

In order to achieve this we find the maximum α that satisfies the equality and then multiply it by a constant $\bar{\alpha} = 0.995$ (or 0.9995), in order to get closer.

Let us assume that we also have a bunch of box constraints, hence our problem becomes

$$(P) \quad \min \left\{ \frac{1}{2} x^T Q x + q x : Ax = b, 0 \leq x \leq u \right\}$$

In this special case, things simplify a lot.