Computational Mathematics Optimization

Based on prof. Antonio Frangioni's lectures

Gemma Martini

November 10, 2019

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 — A. Frangioni

This course will deal with the optimization and numerical analysis of machine learning problems. We are not going to solve difficult problems (e.g. NP-hard problems), besides we try to find 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 are starting working well nowadays, 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 in 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 the thing. 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 model is a good when it is a good tradeoff 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. *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.*

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 functions), namely* a *and* b.

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

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

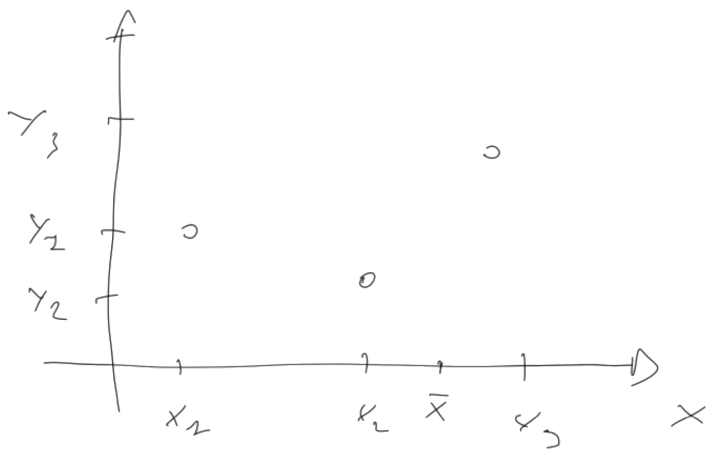


FIGURE 1.1: Geometric representation of the input. We are interested in finding a model that fits the input data and allows to predict $\bar{y}$ out of $\bar{x}$.

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 the given shape, we want to find the one that better represent our phenomenon. This is **fitting**, and it is clearly some sort of optimization problem and solving the fitting problem is typically the computational bottleneck.
However, ML >> fitting: fitting minimizes training error ≡ empirical risk, but ML aims at minimizing test error ≡ risk ≡ generalization error!

The aim of machine learning is to build a model that fits the data we are given and then to tune parameters in order to achieve a good "predicting power" on unseen inputs (in machine learning the technical term is "not **overfitting**").

So, a mathematical model should be:

- accurate (describes well the process at hand)
- computationally inexpensive (gives answers rapidly)
- general (can be applied to many different processes)

Typically impossible to have all three!

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.

1.2.1 Linear estimation

For example, a phenomenon measured by one number y is believed to depend on a vector $x = [x_1, \ldots, x_n]$ and a set of observations is available: $(y^1, x^1), \ldots, (y^m, x^m)$

Definition 1.1 (Linear model)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}^n$ be the objective function. We call $\widetilde{f}(x) = \sum_{i=1}^{n} w_i x_i + w_0 = wx + w_0$ the **linear model** of f for a given set of parameters, which is a vector $w = (w_0, w_1, \ldots, w_n) \in \mathbb{R}^{n+1}$.*

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

Definition 1.2 (Least squares problem)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}^n$ be the objective function, such that $f(x) = y$ and let Xw be our linear model. Then we can find the best values for vector $w \in \mathbb{R}^{n+1}$ by computing:*

$$y = \begin{pmatrix} y^1 \\ \vdots \\ y^m \end{pmatrix} \quad X = \begin{pmatrix} 1 & x^1 \\ \vdots & \vdots \\ 1 & x^m \end{pmatrix} \quad \min_w \|y - Xw\|$$

If the matrix X is invertible then the simple solution is $w = X^{-1}y$. 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).

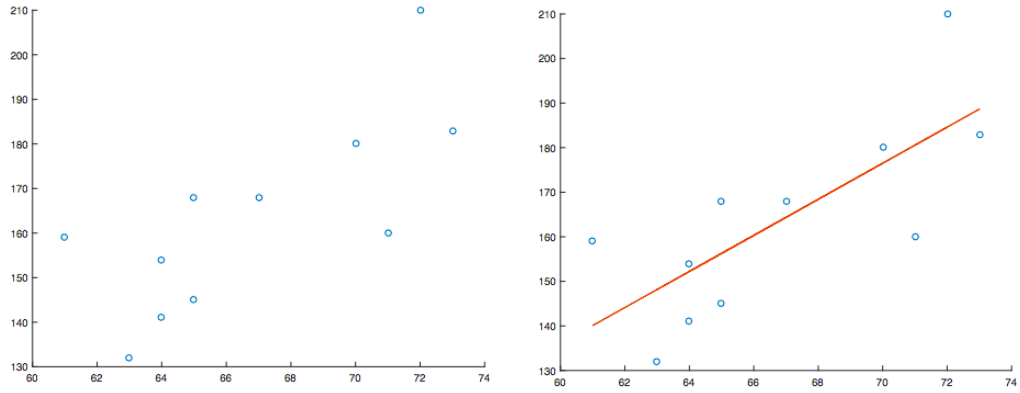


FIGURE 1.2: A linear estimation fitting example

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 chosen from 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): $A \in M(n, k, \mathbb{R})$ and a "fat and large" $B \in M(k, m, \mathbb{R})$ ($k \ll n, m$).

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

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

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

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 they need too much memory.

Efficiently solving this problem requires:

- low-complexity computation (of course)
- avoiding ever explicitly forming $M^T M$ and MM^T (too much memory)
- exploiting structure of M (sparsity, similar columns, . . .)
- ensuring the solution is numerically stable

1.2.3 Support vector machines

Let us take a decision problem: given a set of values of many parameters (aka variables) "label" a person 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 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_+, w_0) and (w_+, w_0') in Figure 1.3 is:

$$|w_0 - w_0'|/\|w_+\|$$

- Geometrically, the distance between these two hyperplanes is $\frac{2}{\|w_+\|}$, so to maximize the distance between the planes we want to minimize $\|w_+\|$
- We can always take the hyperplane in the middle:

$$\Rightarrow w_+ x^i + w_0 \geq 1 \text{ if } y^i = 1 \ , \ w_+ x^i + w_0 \leq -1 \text{ if } y^i = -1$$

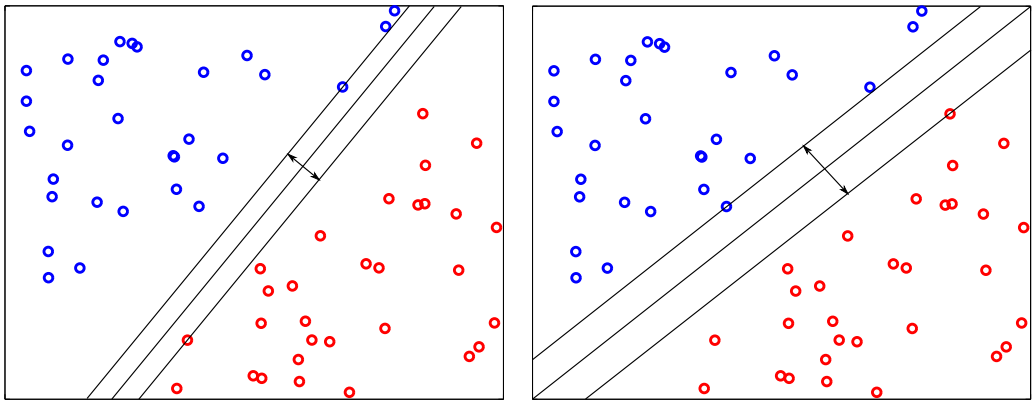


FIGURE 1.3: There are many possible boundaries that can be chosen as a model using many angular coefficients. Our best guess is the one that maximizes the distance between the line and the nearest points.

So we have this optimization problem and the maximum-margin separating hyperplane (assuming any exists) is the solution of:

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

In fact, what happens most of the times is that 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.

Definition 1.3 (SVM Primal problem)**.**

$$\begin{aligned} \min_{w,\xi} \quad & \|w_+\|^2 + C \sum_{i=1}^{m} \xi_i \\ s.t. \quad & y^i(w_+ x^i + w_0) \geq 1 - \xi_i \, , \, \xi_i \geq 0 \, , \, \forall i = 1, \ldots, m \end{aligned} \tag{SVM-P}$$

This is the Support Vector Machine primal problem (SVM-P) a convex constrained problem with complex constraints and it's a multi-objective optimization problem. Where C is called **hyper-parameter** and there are the weighs violation of separation against margin.

This formula 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.

This whole course has the aim of presenting some techniques for solving efficiently **convex quadratic problems**, as the ones presented above (SVM-P) or (SVM-D).

Whenever we are able to solve the multi-objective optimization problem we are also able to solve what is called the **dual problem**, which is formally defined as SPV-D and has the following shape in our case:

Definition 1.4 (SVM Dual problem)**.**

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i < x^i, x^j > \alpha_j \\ s.t. \quad & \sum_{i=1}^{m} y^i \alpha_i = 0 \\ & 0 \leq \alpha_i \leq C \, , \, \forall i = 1, \ldots, m \end{aligned} \tag{SVM-D}$$

a convex constrained quadratic program, but with simple constraints. The idea is solve one problem by solving an apparently different one:

$$\alpha^* \text{ optimal for (SVM-D)} \Rightarrow w_+^* = \sum_{i=1}^{m} \alpha_i^* y^i x^i \text{ optimal for (SVM-P)}$$

Dual formulation $\Rightarrow$ kernel trick: input space $\rightsquigarrow$ (larger) feature space where points are hopefully more linearly separable:

$$< x^i, x^j > \rightsquigarrow < \phi(x^i), \phi(x^j) >$$

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

2 21st of September 2018 — A. Frangioni

2.1 Mathematical background for optimization problems

Definition 2.1 (Minimum problem)**.** *Let X be a set, called **feasible region** and let $f : X \to \mathbb{R}$ be* any *function, called **objective function** we call **problem** the following:*

$$f_* = \min\{f(x) \ : x \in X\} \ (P)$$

Definition 2.2 (Feasible solution)**.** *Let $x \in F$ be a solution of the minimum problem in which the domain is a superset of $X \subset F$. We say that x is a **feasible solution** if $x \in X$. On the other hand, x is **unfeasible** if $x \in F \setminus X$.*

Definition 2.3 (Optimal solution)**.** *Under the same hypothesis of the above definition, we define x_* such that $f(x_*) = f_*$ an **optimal solution**, where $f_* \leq f(x) \, \forall x \in X$, $\forall v > f_* \, \exists \, x \in X$ s.t. $f(x) < v$.*

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

Example 2.1. *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 it is unbounded below ($\forall M \, \exists x_M \in X$ s.t. $f(x_M) \leq M$). On the other hand, we need to maximize the function, but it is unbounded above.*

We can now rewrite the problem of solving an optimization problem as:

1. finding x_* and proving it is optimal;
2. or proving $X = \emptyset$;
3. or constructively prove $\forall M \, \exists x_M \in X$ s.t. $f(x_M) \leq M$.

Typically $x \in \mathbb{R}$ actually mean $x \in \mathbb{Q}$ 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 ($\bar{x}$, the approximately optimal).

Definition 2.4 (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 2.5 (Relative error)**.** *We define as **relative error** the absolute error, normalized by the true value of the function:*

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

Multi-objective Optimization:
What happens if we have more than one objective function?
Often you need more than one, say:

$$\min\{[f_1(x), f_2(x)] \ : x \in X\} \ (P)$$

with f_1, f_2 contrasting and/or with incomparable units (apples vs. oranges)

In multi-objective optimization, there does not typically exist a feasible solution that minimizes all objective functions simultaneously. Therefore, attention is paid to *Pareto optimal solutions*; that is, solutions that cannot be improved in any of the objectives without degrading at least one of the other objectives.

Definition 2.6 (Pareto frontier)**.** *A feasible solution $x^1 \in X$ is said to (Pareto) dominate another solution $x^2 \in X$, if:*

$$f_i(x^1) \leq f_i(x^2) \quad for \quad all \quad indices \quad i \in \{1, 2, \ldots, k\}$$

$$f_j(x^1) \leq f_j(x^2) \quad for \quad all \quad indices \quad j \in \{1, 2, \ldots, k\}$$

A solution $x^* \in X$ (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****.*

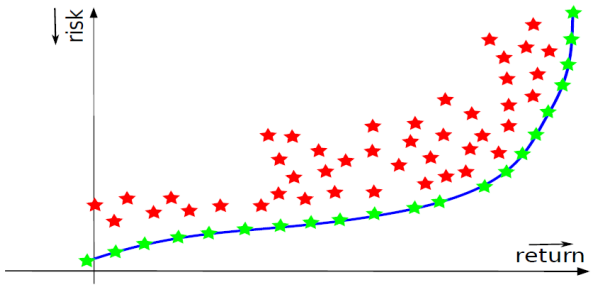


FIGURE 2.1: An example of Pareto frontier

We are provided with two practical solutions:

SCALARIZATION: using a linear combination of the two functions: $f(x) = \alpha f_1(x) + \beta f_2(x)$;

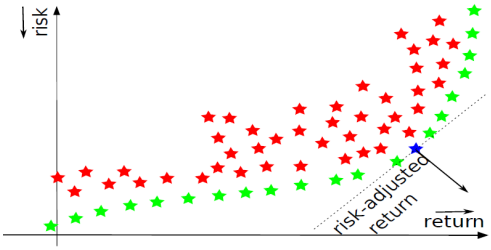


FIGURE 2.2: Maximize risk-adjusted return, $\min\{f_1(x) + \alpha f_2(x) \ : x \in X\}$

BUDGETING: $f(x) = f_1(x)$, $X := X \cup \{\, f_2(x) \leq b \,\}$, which 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.

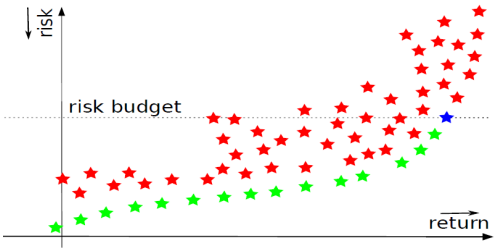


FIGURE 2.3: Maximize return with budget on maximum risk, $\min\{f_1(x) \ : f_2(x) \leq \beta \ : x \in X\}$

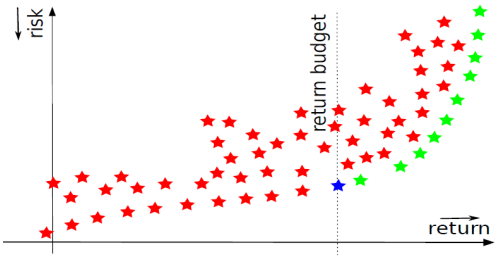


FIGURE 2.4: Minimize risk with budget on minimum return,, $\min\{f_2(x) \ : f_1(x) \leq \beta \ : x \in X\}$

2.1.1 Examples of Bad optimization problems

Here some problems that has no optimal solution:

- empty case ($X = \emptyset$): $\min\{x \ : \ x \in \mathbb{R} \wedge x \leq -1 \wedge x \geq 1\}$
- unbounded [below]: $\min\{x \ : \ x \in \mathbb{R} \wedge x \leq 0\}$;
- bad X: $\min\{x \ : \ x \in \mathbb{R} \wedge x > 0\}$;
- bad f and X: $\min\{x \ : \ x \in \mathbb{R} \wedge x > 0\}$;
- 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 but its non-continuity leads to the impossibility of reaching the optimum. As an example, let us take the following:

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

2.2 Infima, suprema and extended reals

Since we minimize/maximize stuff, infima/suprema are important:

Definition 2.7 (Totally ordered set)**.** *We say that set X is **totally ordered** if $\forall\, x, y \in X$, either $f(x) \leq f(y)$ or $f(y) \leq f(x)$.*

Definition 2.8 (Infima and suprema)**.** *Given a totally ordered set R and one of its subsets (say $S \subseteq R$):*

$$s \text{ is the } \textbf{infimum} \text{ of } S \Leftrightarrow \underline{s} = \inf S \quad \Leftrightarrow \quad \underline{s} \leq s \;\; \forall s \in S \;\; \wedge \;\; \forall t > \underline{s} \; \exists\, s \in S \text{ s.t. } s \leq t$$

$$s \text{ is the } \textbf{supremum} \text{ of } S \Leftrightarrow \bar{s} = \sup S \quad \Leftrightarrow \quad \bar{s} \geq s \;\; \forall s \in S \;\; \wedge \;\; \forall t < \bar{s} \; \exists\, s \in S \text{ s.t. } s \geq t$$

Issue: inf S/sup S may not exist in $\mathbb{R}$

Definition 2.9 (Extended real)**.** *In the case of unbounded functions the value of infima or suprema are ∞, and we call **extended reals** $\overline{\mathbb{R}} = -\infty \cup \mathbb{R} \cup +\infty$.*

- for all $S \subseteq \mathbb{R}, \sup/\inf S \in \overline{\mathbb{R}}$;
- $\inf S = -\infty$ just a convenient notation for 'there is no (finite) inf';
- $\inf \emptyset = \infty$, $\sup \emptyset = -\infty$.

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, hopefully.

Sequence of iterates $\{x_i\} \subset X$ and $v_i = f(x_i)$. Typically we cant get f in finite time ($\exists i v_i = f$), but we can get as close as we want: there in the limit.

Definition 2.10 (Limit)**.** *Given a sequence $\{x_i\}$ the **limit** for $i \to \infty$ is defined as*

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

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

Fact 2.1. *Let us be given a monotone sequence, then the sequence **does** have a limit.*

Notice that given a sequence either it is monotone or it can be "split" into two monotone sequences (for example $\{(-1)^n\}$ can be transformed into $\{(-1)^{2n}\}$ and $\{(-1)^{2n+1}\}$ and these two sequences are both monotone).

The obvious way to make $\{vi\}$ monotone: keep aside the best

$$v_i^* = \min\{v_h : h \leq i\} \qquad \text{(best value at interaction i)}$$

- $v_1^* \geq v_2^* \geq v_3^* \geq \ldots \Rightarrow v_\infty^* = \lim_{i \to \infty} v_i^* \geq f_*$ (asymptotic estimate);
- $\lim_{i \to \infty} v_i^* = v_\infty^* = f_* \Rightarrow \{v_i\}$ minimizing sequence (of values).

Forcing monotonicity on sequences in $\mathbb{R}$: the hard way

- Extract monotone sequences from $\{ v_i \}$ "the hard way":
  $\underline{v}_i = \inf\{ v_h : h \geq i \}$, $\bar{v}_i = \sup\{ v_h : h \geq i \}$
- $\underline{v}_1 \leq \underline{v}_2 \leq \underline{v}_3 \leq \ldots$, $\bar{v}_1 \geq \bar{v}_2 \geq \bar{v}_3 \geq \ldots \implies$ they still have a limit
- $\liminf_{i\to\infty} v_i := \lim_{i\to\infty} \underline{v}_i = \sup_i \underline{v}_i$
- $\limsup_{i\to\infty} v_i := \lim_{i\to\infty} \bar{v}_i = \inf_i \underline{v}_i$
- $\bar{v}_i \geq \underline{v}_i \implies \limsup_{i\to\infty} v_i \geq \liminf_{i\to\infty} v_i$
- $\lim_{i\to\infty} v_i = v \iff \limsup_{i\to\infty} v_i = v = \liminf_{i\to\infty} v_i$
- $\liminf_{i\to\infty} v_i = f_* \implies \{ v_i \}$ minimizing sequence (of values)
- A stronger definition: $\liminf_{i\to\infty} v_i = f_* \implies \lim_{i\to\infty} v_i^* = f_*$

FIGURE 2.5: Forcing monotonicity on sequences in $\mathbb{R}$: the hard way.

2.4 Vector spaces and topology

2.4.1 Euclidean space $\mathbb{R}^n$

Single numbers are not enough, except for objective function values.

Definition 2.11 (Euclidean vector space)**.** *We call* ***Euclidean space****:*

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

Equivalently, we can characterize the Euclidean space as Cartesian product of $\mathbb{R}$ *n times:* $\mathbb{R}^n = \mathbb{R} \times \mathbb{R} \times \ldots \mathbb{R}$*. Closed under sum and scalar multiplication.*

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

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

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

Usually $x \in \mathbb{R}^n$ usually considered column vector $\in \mathbb{R}^{n\times 1}$, otherwise a row vector is x^T.

Definition 2.12 (Finite vector space)**.** *each $x \in \mathbb{R}^n$ can be obtained from a finite basis (canonical base is u_i having 1 in position i and 0 elsewhere).*

Notes for vector space:

- not all vector spaces are finite;
- not a totally ordered set.

Concept of limit requires topology. So, in order to be able to compute limits in a vector space we need some topology definitions: norm, scalar product, distance.

2.4.2 (Euclidean) norm

Definition 2.13. *Let $x \in \mathbb{R}^n$ we define the **euclidean norm** of a vector:*

$$\|x\|_2 = \sqrt{\sum_{i=1}^{n} {x_i}^2} = \sqrt{< x, x >}$$

Fact 2.2. *The norms on a vector space have the following properties:*

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

Fact 2.3 (Cauchy-Schwartz inequality)**.** *Let $x, y \in \mathbb{R}^n$. The following holds:*

$$< x, y >^2 \leq < x, x >< y, y > \equiv |< x, y >| \leq \|x\| \|y\|, \ \forall x, y \in \mathbb{R}^n$$

Fact 2.4 (Parallelogram Law)**.**

$$2\|x\|^2 + 2\|y\|^2 = \|x + y\|^2 + \|x - y\|^2$$

Fact 2.5.

$$\|x + y\|^2 = \|x\|^2 + \|y\|^2 + 2 < x, y >$$

2.4.3 A useful norm generalization: p-norm

Many (but not all) derive from p-norm:

Definition 2.14 (p-norm)**.** *Let $p \geq 1$ be a real number, the p-norm of a vector $x = (x_1, \ldots, x_n)$ is defined as follow:*

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

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 $p \geq 1$, nonconvex for $p < 1$.

Here some norm derived from p-norm:

- $\|x\|_1 := \sum_{i=1}^{n} |x_i|$
- $\|x\|_\infty := \max\{|x_i| \ : \ i = 1, \ldots, n\}$
- $\|x\|_i := |\{i \ : \ |x_i| > 0\}|$
- Other ones (e.g. for matrices, etc.)

Fact 2.6. *For any given finite-dimensional vector space V (e.g. $\mathbb{R}^n$ is a finite vector space), all norms on V are equivalent in the sense that given two norms $\|\cdot\|_A$, $\|\cdot\|_B$:*

$$\exists\, 0 < \alpha < \beta \ \ s.t \ \ \alpha \|x\|_A \leq \|x\|_B \leq \beta \|x\|_A \qquad \forall x \in V$$

Therefore convergence in one norm implies convergence in any other norm. This rule may not apply in infinite-dimensional vector spaces such as function spaces, though:

Fact 2.7 (Holders inequality)**.**

$$< x, y >^2 \leq \|x\|_p \|y\|_q \quad 1/p + 1/q = 1$$

Definition 2.15 (Ball)**.** *We term **ball** centered in $\bar{x}$ and having ε as radius as the set of points that are close enough to $x \in \mathbb{R}^n$: $B(\bar{x}, \varepsilon) = \{x \in \mathbb{R}^n \ : \ \|x - \bar{x}\| \leq \varepsilon\}$.*

Let's 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 2.6 we may observe the different shapes of the same ball varying the value of p in the p-norm.

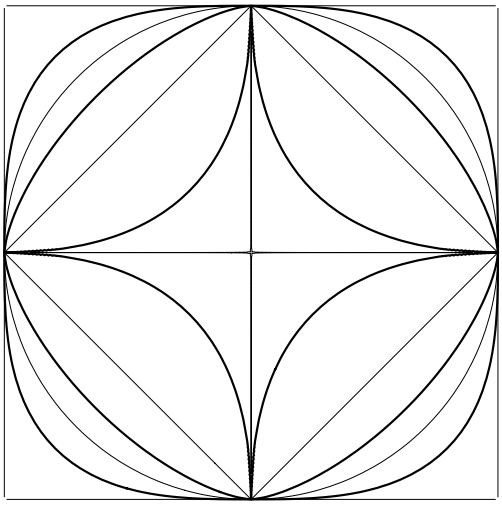


FIGURE 2.6: The shapes of balls centered in the origin of radius 1 varying the value of p-norm.

2.4.4 (Euclidean) Scalar Product

Definition 2.16 (Scalar product)**.** *Let $x, y \in \mathbb{R}^n$ we define the **scalar product** between these two vectors:*

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

Fact 2.8. *A scalar product has the following properties:*

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

Geometric interpretation of the scalar product, an important characterization of the scalar product is the one that uses angles:

$$< x, y >= \|x\| \, \|y\| \cos\theta$$

- $x \perp y \iff < x, y >= 0$ (orthogonality condition)
- $< x, y > > \ 0 \iff$ "x and y point in the same direction"

More General: $< x, y >_M := y^T M x$ with $M \succ 0$ ($x \longrightarrow M^{-1/2} x$)

2.4.5 (Euclidean) Distance

Definition 2.17 ((Euclidean) distance)**.** *The **Euclidean distance** between points x and y is the length of the line segment connecting them. In Cartesian coordinates, if $x = (x_1, x_2, ..., x_n)$ and $y = (y_1, y_2, ..., y_n)$ are two points in Euclidean n-space, then the distance (d) from x to y, or from y to x is given by:*

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

Fact 2.9. *The distance has the following properties:*

1. $d(x, y) \geq 0 \; \forall x, y \in \mathbb{R}^n \; , \; d(x, y) = 0 \iff x = y$
2. $d(\alpha x, 0) = |\alpha| d(x, 0) \;\; \forall x \in \mathbb{R}^n \, , \, \alpha \in \mathbb{R}$
3. $d(x, y) \leq d(x, z) + d(z, y) \;\; \forall x, y, z \in \mathbb{R}^n$ *(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 $\mathbb{R}^n$. **Limit of a sequence in R^n.**

Definition 2.18 (Limit of a sequence in the Euclidean space)**.** *Let $\{x_i\} \subset \mathbb{R}^n$ be a sequence in $\mathbb{R}^n$. The **limit** of $\{x_i\}$ for $i \to +\infty$ is the following:*

$$\lim_{i \to \infty} x_i = x \equiv \{x_i\} \to x$$

$$\Updownarrow$$

$$\forall \varepsilon > 0 \; \exists h \; s.t. \; d(x_i, x) \leq \varepsilon \; \forall i \geq h$$

$$\Updownarrow$$

$$\forall \varepsilon > 0 \; \exists h \; s.t. \; x_i \in \mathcal{B}(x, \varepsilon) \; \forall i \geq h$$

$$\Updownarrow$$

$$\lim_{i \to \infty} d(x_i, x) = 0$$

Note:

- points of x_i eventually all come arbitrarily close to x;
- no obvious lim inf / lim sup ($\mathbb{R}^n$ is not totally ordered).

3 27th of September 2018 — A. Frangioni

Definition 3.1 (Minimizing sequence)**.** *Let us consider the* ***minimum problem*** *defined in Definition 2.1:*

$$f_* = \min\{f(x) \ : x \in X\} \ (P)$$

we term ***minimizing sequence*** *a sequence that gets closer to the optimal value:* $\{x_i\}$ *s.t.* $\{f(x_i)\} \to f_*$.

It goes without saying that we would like to avoid minimizing sequences that do not lead to an optimum.

As an example, consider:
$\min\{x \ : \ x \in \mathbb{R} \wedge x > 0\}$ $\{x_i = 1/i\}$, where $\{f(x_i)\} \to 0$, but it is not an optimum or
$\min\{1/x \ : \ x \in \mathbb{R} \wedge x > 0\}$ $\{x_i = i\}$, where $\{f(x_i)\} \to 0$, but it is not an optimum.

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

Definition 3.2 (Interior and border of a set)**.** *Given* $S \subseteq \mathbb{R}^n$*, we say that* $x \in int(S)$ *(*x *is an* ***interior point****) if* $\exists r > 0$ *s.t.* $\mathcal{B}(x, \ r) \subseteq S$.

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

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: $S = int(S)$).

Definition 3.3 (Closure of a set)**.** *Let* $S \subset \mathbb{R}^n$ *we term* ***closure*** *of* S *the set* $cl(S) = int(S) \cup \partial S$.

Definition 3.4 (Closed set)**.** *We say that a set* $S \in \mathbb{R}^n$ *is* ***closed*** *if it coincides with his closure:* $S = cl(S)$.

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

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

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

Definition 3.5 (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 3.6 (Compact set)**.** *Let* $S \subseteq \mathbb{R}^n$*. We term* S ***compact*** *if it is both closed and bounded.*

Definition 3.7 (Accumulation point)**.** *Let* $\{x_i\}$ *be a sequence.* x *is an* ***accumulation point*** *if* $\exists$ $\{x_{n_i}\}$ *subsequence of* $\{x_i\}$ *such that* $\{x_{n_i}\} \to x \equiv \ \liminf_{i \to \infty} d(x_i, \ x) = 0$.

Fact 3.1. *Let S be a compact set and let $\{x_i\} \subseteq S$ minimizing sequence. (Bolzano-Weierstrass) $\{x_i\}$ has an* ***accumulation point*** *$x \in S$, thus the limit of the sequence has to be a feasible solution.*

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.

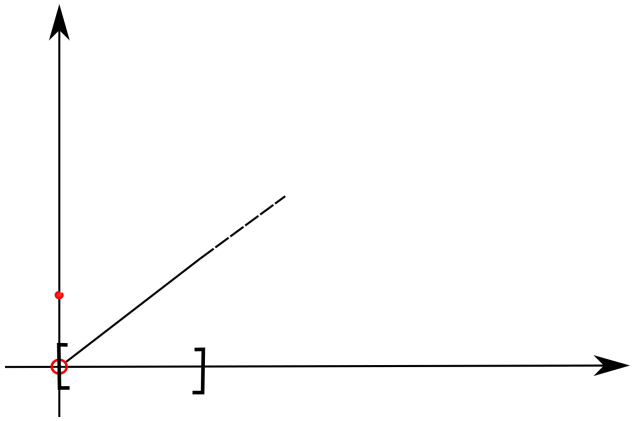


FIGURE 3.1: Case of non-continuity of the objective function in the border point $(0, 0)$.

Definition 3.8 (Domain)**.** *Let $f : D \to \mathbb{R}$. We term D* ***domain****: $D = dom(f)$.*

In this course we will not take into account the domain of functions, because we can make functions defined in the whole space as follows:
$f : \mathbb{R}^n \to \bar{\mathbb{R}}$, where $f(x) = \infty$ for $x \notin D$.

Definition 3.9 (Graph and epigraph)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$. We term* ***graph*** *$gr(f) = \{(f(x),\ x)\ :\ x \in dom(f)\}$.*
On the other hand, we term ***epigraph*** *$epi(f) = \{(v,\ x)\ :\ x \in dom(f)\ \wedge\ v \geq f(x)\}$.*

A graphic example of epigraph is displayed in Figure 3.2.

Definition 3.10 (Level and sublevel set)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$. We term* ***level set*** *$L(f, v) = \{x \in dom(f)\ :\ f(x) = v\}$.*
On the other hand, we term ***sublevel set*** *$S(f, v) = \{x \in dom(f)\ :\ f(x) \leq v\}$.*

Intuitively, the level sets are projections of the points of the domain and they are used in order to "look" at the function even if we are working in $\mathbb{R}^n$.
A graphic example of level set is displayed in Figure 3.3.

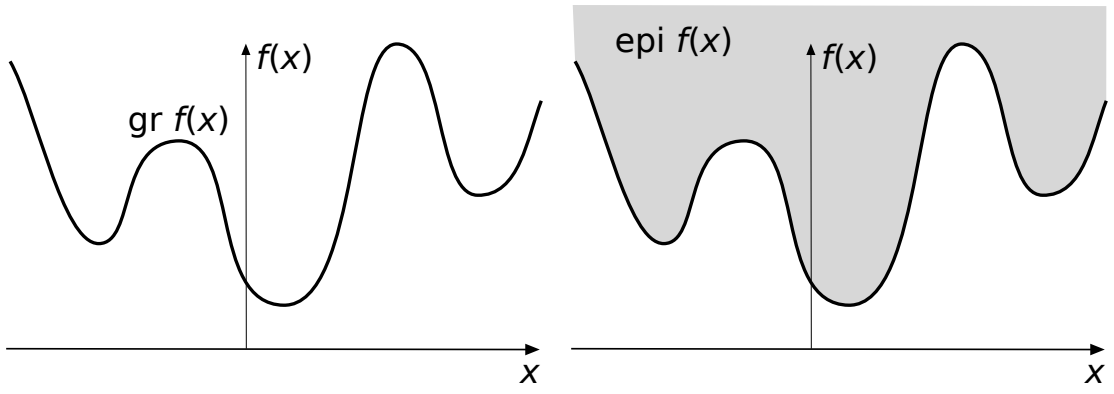


FIGURE 3.2: On the left-hand side the graph of the function $f(x)$, while on the right-handed side the epigraph of such function.

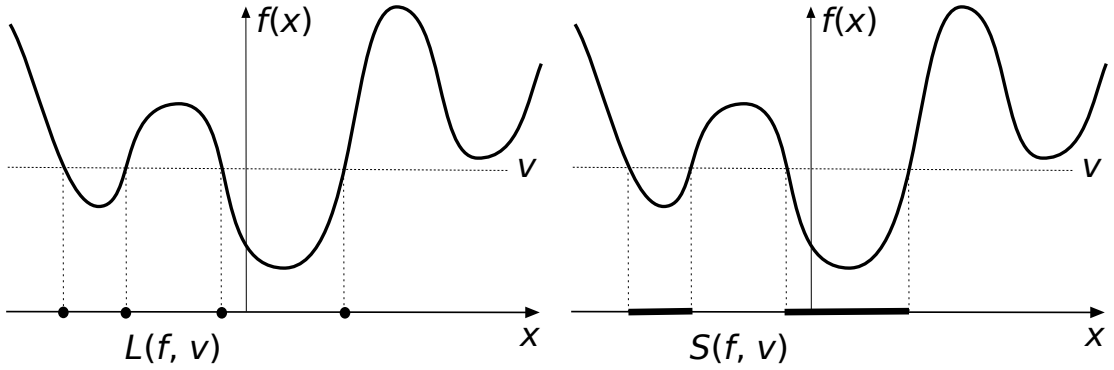


FIGURE 3.3: On the left-handed side of the figure, the level set of the function f, while on the right-hand side the sublevel set of such function.

3.1 Continuity

Definition 3.11 (Continuity)**.** *Let* $f : \mathbb{R}^n \to \mathbb{R}$*. We say that* f *is* ***continuous*** *if* $\forall \varepsilon > 0\ \exists\, \delta > 0$ *such that* $|f(y) - f(x)| < \varepsilon\ \forall y \in \mathcal{B}(x,\ \delta)$*.*

Notice that if f, g continuous at x then:

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

Theorem 3.2 (Intermediate value)**.** *Let* $f : \mathbb{R} \to \mathbb{R}$*.* f *is continuous on* $[a,\ b]$ *if* $\forall v$ $\min\{f(a),\ f(b)\} \leq v \leq \max\{f(a),\ f(b)\}\ \exists\ c \in [a,\ b]$ *such that* $f(c) = v$*.*

Theorem 3.3 (Weierstrass extreme value theorem)**.** *Let $X \subseteq \mathbb{R}^n$ be a compact set and let f continuous on X. Then (P) has an optimal solution.*

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

Definition 3.12 (Lipschitz continuity)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$. We term f* ***Lipschitz continuous*** *(L.c.) on $S \in \mathbb{R}^n$ if $\exists L > 0$ such that:*

$$|f(x) - f(y)| \leq L \|x - y\| \ \ \forall x, y \in S$$

More generally, f is ***globally Lipschitz continuous*** *if $S = \mathbb{R}^n$ and it is* ***locally Lipschitz continuous*** *if at x $\exists\ \varepsilon > 0$ $S = \mathcal{B}(x,\ \varepsilon)$. It's a stronger form of continuity.*

Notice that the L constant value depends on S. The wider the set the smaller L.

Fact 3.4. *Let $f : \mathbb{R}^n \to R$. On a compact set $S \in \mathbb{R}^n$ if f is continuous then it is Lipschitz continuous.*

Definition 3.13 (Lower-upper semi-continuous)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ f is* ***lower****[upper]* ***semi-continuous*** *(l.[u.]s.c.) at x if:*

$$\{ x_i \} \to x \implies f(x) \leq \liminf_{i \to \infty} f(x_i) \ \ [\, f(x) \geq \limsup \dots \,]$$

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

3.2 Derivatives

In this section we address the problem of inferring information on a complicated function arround a certain value $\bar{x}$ using very simple functions, that provide information which is reliable close to the point we are studying. Those functions are called "derivatives".

It goes without saying that it is not possible to compute such simple representation in any kind of point.

Definition 3.14 (Left and right derivative)**.** *Let $f : \mathbb{R} \to \mathbb{R}$. We term* ***left derivative*** $f'_-(x) = \lim_{t \to 0_-} \frac{f(x+t) - f(x)}{t}$.

On the other hand, ***right derivative*** $f'_+(x) = \lim_{t \to 0_+} \frac{f(x+t) - f(x)}{t}$.

Definition 3.15 (Differentiable)**.** *Let $f : \mathbb{R} \to \mathbb{R}$. We say that f is* ***differentiable*** *at $x \in dom(f)$ if $f'_-(x) = f'_+(x) \Leftarrow$ they $\exists$.*

Fact 3.5. *Let $f : \mathbb{R} \to \mathbb{R}$ and differentiable in $x \in dom(f)$ then f is continuous in x.*

Theorem 3.6 (Mean value theorem)**.** *Let $f : \mathbb{R} \to \mathbb{R}$ 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 3.7 (Rolle's theorem)**.** *Let $f : \mathbb{R} \to \mathbb{R}$. If $f(a) = f(b)$ then $\exists\, c \in (a, b)$ s.t. $f'(c) = 0$*

Corollary 3.8. *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 3.16 (Partial derivative)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$. We term **partial derivative** of f w.r.t. x_i at $x \in \mathbb{R}^n$ as*

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

In other words, it is just $f'(x_1, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_n)$, treating x_j for $j \neq i$ as constants.

Definition 3.17 (Gradient)**.** *Given $f : \mathbb{R}^n \to \mathbb{R}$. We term **gradient** of f the vector of all the partial derivatives.*

Definition 3.18 (Directional derivative)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$. We term **directional derivative** of f in point $x \in dom(f)$ along direction $d \in \mathbb{R}^n$:*

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

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.

We are now ready to introduce the notion of multivariate differentiability.

Definition 3.19 (Differentiable)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$. We say that f is **differentiable** at x if $\exists$ a linear function $\phi(h) =< c, h > +f(x)$ s.t.:*

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

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

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.

Fact 3.9. *Let $f : \mathbb{R}^n \to \mathbb{R}$ differentiable at x. Then f is locally Lipschitz continuous at x.*

Corollary 3.10. *Let $f : \mathbb{R}^n \to \mathbb{R}$ differentiable at x. Then f is continuous at x.*

Notice that the convers does not hold.

Fact 3.11. *Let $f : \mathbb{R}^n \to \mathbb{R}$ and let us assume $\exists \delta > 0$ s.t. $\forall i \ \frac{\partial f}{\partial x_i}(y)$ is continuous $\forall y \in \mathcal{B}(x, \delta)$. Then f differentiable at point x.*

Notice that the converse of Proposition 3.11 does not hold either.

We are now ready to introduce a class of functions that allows good results for optimization: $\mathcal{C}^1 := \nabla f(x)$ continuous.

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

Figure 3.4, Figure 3.5 and Figure 3.6 picture some functions which are not differentiable in the minimum.

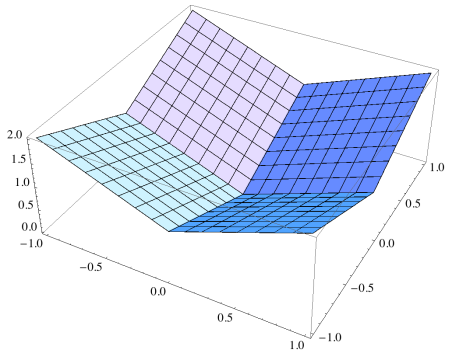


FIGURE 3.4: The function $f(x_1, x_2) = \|[x_1, x_2]\| = |x_1|+|x_2|$ has some kinks.

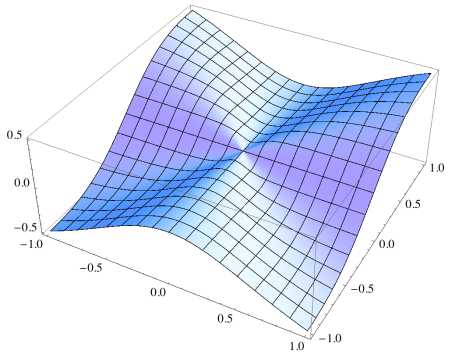


FIGURE 3.5: The function $f(x_1, x_2) = \frac{{x_1}^2 x_2}{{x_1}^2+{x_2}^2}$ may be put to 0 in $(0, 0)$ for continuity, but it is still not differentiable in $(0, 0)$ although the function admits directional derivatives in $(0, 0)$.

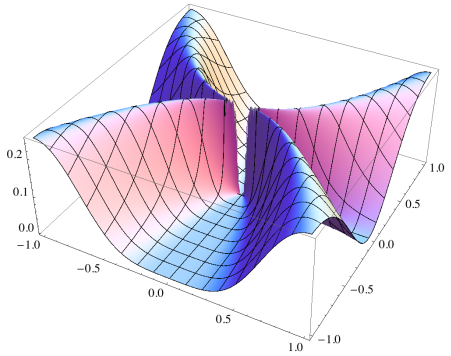


FIGURE 3.6: The function $f(x_1, x_2) = \left(\frac{{x_1}^2 x_2}{{x_1}^4+{x_2}^2}\right)^2$ is not continuous in 0. There are some directions that lead to the limit 0, while there are some other directions (the parabolas above) that do not lead to value 0.

Definition 3.20 (First order model)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ and let $x \in dom(f)$. We term* ***first***

order model *of f at point x:*

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

Fact 3.12. *Let $f : \mathbb{R}^n \to \mathbb{R}$ and let $x \in dom(f)$ such that f is differentiable at x. Then $(L_x, f(x)) \perp S(f, f(x)) \perp \nabla f(x)$.*

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

4 3rd of October 2018 — A. Frangioni

Example 4.1 (On derivatives)**.** *Let $f : \mathbb{R}^2 \to \mathbb{R}$ such that $f(x, y) = x^2 e^y$. Compute the partial derivatives.*

$\frac{\partial f}{\partial x} = 2xe^y$; $\frac{\partial f}{\partial y} = x^2 e^y$

$\frac{\partial f}{\partial [0,1]} = \lim_{t\to 0} \frac{f(x+t_1, y+t_0)}{t} = \lim_{t\to 0} \frac{f(x+t,y)}{t}$, *which is equivalent to the derivative of the second component.*

Conversely, $\frac{\partial f}{\partial [1,0]}$ is equivalent to the derivative in the first component.

We have that the partial derivative on a certain direction d is the scalar product between the gradient and the direction, formally

$$\frac{\partial f}{\partial d} =< \begin{pmatrix} 2xe^y \\ x^2 e^y \end{pmatrix}, \begin{pmatrix} d_1 \\ d_2 \end{pmatrix} >$$

Definition 4.1 (Vector-valued function)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}^m$ is termed **vector-valued function**, where $f(x) = [f_1(x),\ f_2(x), \ldots,\ f_m(x)]$.*

For such functions the computation of the derivative requires to specify not only the component, but also the index of the function.

Definition 4.2 (Partial derivative for vector-valued functions)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}^m$, the partial derivative of the j-th function of the i-th component is*

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

Definition 4.3 (Jacobian)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}^m$ we call **Jacobian** the matrix of all its first-order partial derivatives.*

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

Notice that when the number of variables increases the first derivative is a vector of numbers, the second derivative contains n^2 numbers.

Example 4.2. *Let $f : \mathbb{R}^2 \to \mathbb{R}$ such that $f(x, y) = x^2 e^y$. The gradient was computed above, resulting in*

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

Let us compute the second derivative of this function:

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

Definition 4.4 (Hessian)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}^m$, we call* ***Hessian*** *of f*

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

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

In optimization, we like to work with Hessians, but they need to be handled, because they are very large, hence a trade off is needed.

In most cases, the Hessian is symmetric, meaning that the order in which we derive is not relevant and this happens when $\exists \delta > 0$ s.t. $\forall y \in \mathcal{B}(x, \delta)$ $\frac{\partial^2 f}{\partial x_j \partial x_i}(y)$ and $\frac{\partial^2 f}{\partial x_i \partial x_j}(y)$ exist and $\frac{\partial^2 f}{\partial x_j \partial x_i}(y)$ and $\frac{\partial^2 f}{\partial x_i \partial x_j}(y)$ are continuous at x.

Definition 4.5 (C^2 functions)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}^m$. We say that f* ***belongs to*** *C^2* ***class*** *iff $\nabla^2 f(x)$ is continuous.*

Property 4.1. *Let $f : \mathbb{R}^n \to \mathbb{R}^m$ such that $f \in C^2$, then*

- *$\nabla^2 f(x)$ symmetric*
- *$\nabla f(x)$ continuous*

Definition 4.6 (First order Taylor model)**.** *Let $f : \mathbb{R}^n \to \mathbb{R} \in C^1$ in some $\mathcal{B}(x, \delta)$, we define the* ***first-order Taylor's formula*** *$\forall y \in \mathcal{B}(x, \delta)$ $\exists$ $\alpha \in (0, 1)$ s.t.*

$$f(y) =< \nabla f(\alpha x + (1 - \alpha)y,) > y - x + f(x)$$

Intuitively, it is a linear approximation of the function f in a neighbourhood of x.

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

$$f(y) =< \nabla f(x), y - x > + f(x) + R(y - x)$$

where $\lim_{h \to 0} \frac{R(h)}{\|h\|} = 0$, in other words the error that we make is at most quadratic.

Notice that this is a completely local thing, the furthest we get from x the mor distant the function and the model are.

Example 4.3. *Let $f : \mathbb{R}^2 \to \mathbb{R}$ such that $f(x, y) = x^2 e^y$. The gradient was computed above and now we have $f(1, 0) = 1$, $\nabla f(1, 0) = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$.*

The linear model has the following shape $L_{(1,0)}(x, y) = 1 + < \begin{pmatrix} 2 \\ 1 \end{pmatrix}, \begin{pmatrix} x - 1 \\ y - 0 \end{pmatrix} >= -1 + 2x + y$.

And the quadratic model is:

$$\begin{aligned} Q_{[1,0]}(x,y) &= -1 + 2x + y + \frac{1}{2}\begin{pmatrix} x-1 & y-0 \end{pmatrix}\begin{pmatrix} 2 & 2 \\ 2 & 1 \end{pmatrix}\begin{pmatrix} x-1 \\ y-0 \end{pmatrix} \\ &= -1 + 2x + y + \frac{1}{2}\begin{pmatrix} 2x-2+2y, & 2x-2+y \end{pmatrix}\begin{pmatrix} x-1 \\ y-0 \end{pmatrix} \\ &= -1 + 2x + y + \frac{1}{2}(2x-2+2y)(x-1) + (2x-x+y)y \end{aligned} \tag{4.1}$$

Definition 4.7 (Second order Taylor model)**.** *Let $f : \mathbb{R}^n \to \mathbb{R} \in C^1$ in some $\mathcal{B}(x,\delta)$, we define the* ***second-order Taylor's formula***

$$f(y) = L_x(y) + \frac{1}{2}(y-x)^T \nabla^2 f(x)(y-x) + R(y-x)$$

with $\lim_{h\to 0} R(h)\|h\|^2 = 0 \equiv$ the error is $O(\|y-x\|^3) \equiv$ the remainder vanishes at least cubically.

Notice that the k-th order Taylor expansion with k-th order derivatives, but $\nabla^k f(x)$ tensor of order $k \equiv n^k$ numbers. Often $k > 2$ is unfeasible, so this approach cannot be followed.

Fact 4.2. *Let $f : \mathbb{R}^n \to \mathbb{R} \in C^1$. If f Lipschitz continuous on S, then $\sup\{\|\nabla f(x)\| \ : \ x \in S\} \leq L$ ($\iff$, and $= L$ if S convex).*

Moreover, we can prove

Fact 4.3. *Let $f : \mathbb{R}^n \to \mathbb{R} \in C^1$. f is Lipschitz continuous on S iff $\nabla^2 f$ is bounded on S.*
Equivalently, iff $\lambda_1(\nabla^2 f)$ is bounded.

Fact 4.4. *Let $f : \mathbb{R}^n \to \mathbb{R} \in C^1$. If ∇f is Lipschitz continuous, then $f(y) \leq L_x(y) + \frac{L}{2}\|y-x\|^2$.*

The approximations are good because they give an hint about where the function is decreasing. The limitation is that we cannot take too be steps, because the derivative information is accurate only locally.

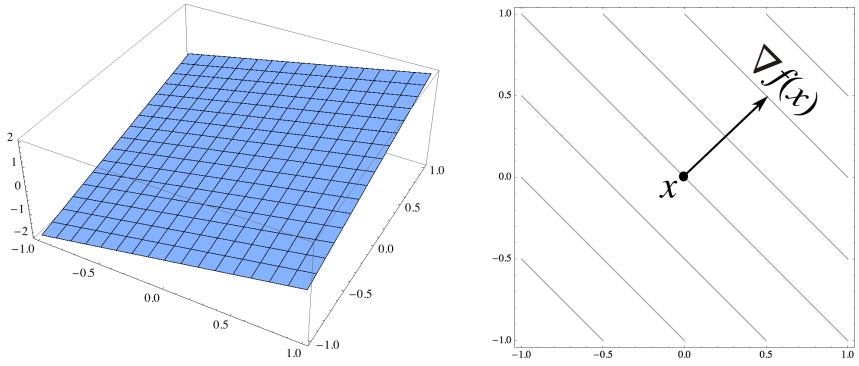
Moreover, we need to take into account the fact that if the model is too simple (but efficient) the approximation is not very good; on the other hand, complex and accurate approximations are computationally heavy.

4.1 Simple functions

4.1.1 Linear functions

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

It holds that $\nabla f(x) = c$, $\nabla^2 f(x) = 0$ and that level sets are parallel hyperplanes orthogonal to c ($= [\,1\,,\,1\,]$ here)



(A) Linear function (B) Level sets

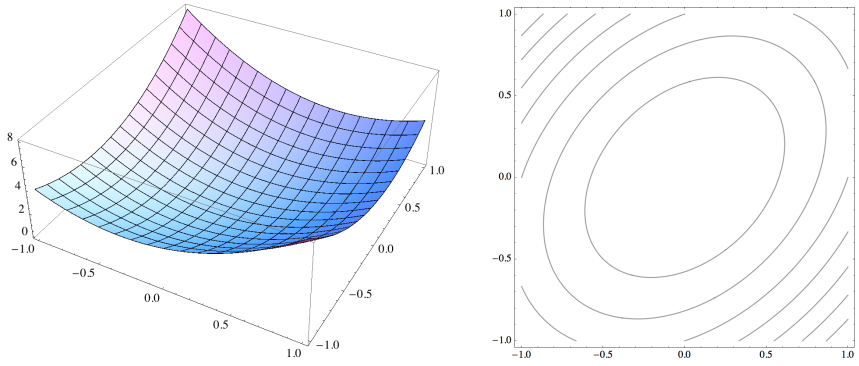
FIGURE 4.1: Graphical example of linear function.

4.1.2 Quadratic functions

Let us move to more interesting objects, aka quadratic functions: $f(x) = \frac{1}{2}x^T Qx + qx$, where $Q \in \mathbb{R}^{n\times n}$, $q \in \mathbb{R}^n$.

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

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



(A) Quadratic function (B) Level sets

FIGURE 4.2: Graphical example of a quadratic function.

In quadratic functions the gradient is a linear function on x ($\nabla f(x) = Qx + q$), while the Hessian is just Q ($\nabla^2 f(x) = Q$) 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 4.5. *We can always assume that if Q is symmetric, then it has spectral decomposition.*

Proof. $x^T Qx = \frac{[(x^T Qx)+(x^T Qx)^T]}{2} = x^T[\frac{(Q+Q^T)}{2}]x = H\Lambda H^T$ □

We will almost always assume that Q is symmetric.

Let us consider the "easy case", where Q is non singular (aka $\lambda_i \neq 0 \forall i$).

In this case, $\bar{x} = -Q^{-1}q$, where $\bar{x}$ is called the center of the ellipsoid. Let us assume that we moved the origin in such $\bar{x}$, so $y = x - \bar{x}$ and $f_{\bar{x}}(y) = \frac{1}{2}y^T Q y$.

Fact 4.6. *Along H_i: $f(\alpha) = f_{\bar{x}}(\alpha H_i) = \alpha^2 \lambda_i$*

Moreover, the size of the axes of the level curves is proportional to $\sqrt{1/\lambda_i}$. If the eigenvalues are 0 then the axes get very long (but we would be in the case of Q singular), on the other hand, if $\lambda_i < 0$, we no longer have axes.

Fact 4.7. *We can state the following:*

- $\forall i \, \lambda_i > 0 \equiv Q \succ 0 \Longrightarrow \bar{x}$ *minimum of f;*
- $\exists i \lambda_i < 0 \Longrightarrow f$ *unbounded below.*

5 5th of October 2018 — A. Frangioni

5.1 Unconstrained optimization

Until now we stated that the best conditions 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 (Unconstrained optimization problem)**.** *We want to solve:*

$$f_* = \min\{f(x) \ : \ x \in X\} \ (P)$$

where $X = \mathbb{R}^n$*.*

If $\mathbb{R}^n$ is not bounded, Weierstrass theorem does not apply, hence even if a (global) minimum x_* exists, finding it is a NP problem.

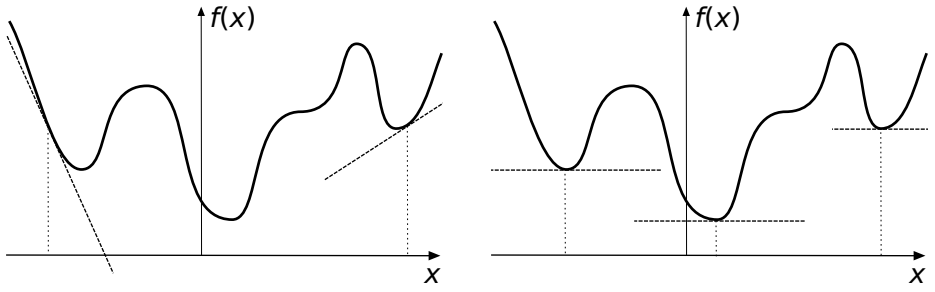
Let us use a weaker condition to ease things a little: x_* is a **local minimum** if it solves:

$$\min\{f(x) \ : \ x \in \mathcal{B}(x_*, \varepsilon)\} \text{for some } \varepsilon > 0$$

aka, the minimum we found is a global minimum in a ball around x^*.

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

To test these conditions derivatives help, as an example see Figure 5.1



(A) Not minima (B) Minima

FIGURE 5.1: In the leftmost plot, we can see that if the derivatives are non zero the point is not a minimum. Such a condition is satisfied in the right handed plot.

If $f'(x) < 0$ or $f'(x) > 0$, x clearly cannot be a local minimum.

Hence, $f'(x) = 0$ in all local minima, so this holds in the global one as well.

5.1.1 First order model

Do you recall?

The first order model of f is $L_x(y) = f(x) + \nabla f(x)(y-x)$, such that $f(y) = f(x) + \nabla f(x)(y-x) + R(y-x)$.
We already stated last lecture that if the norm of the argument of the residual is going to 0, then the residual is going to 0 faster (quadratically), formally $\lim_{\|h\|\to 0} \frac{R(h)}{\|h\|} = 0$.

Fact 5.1. *Let f be differentiable, if x is a local minimum, then $\nabla f(x) = 0$.*

In which direction shall we move in order to get closer to the minimum, provided that we are sitting in x? $x(\alpha) = x - \alpha\nabla f(x)$, hence we should take a step along the anti-gradient $-\nabla f(x)$.

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

In our case, $y = x - \alpha\nabla f(x)$, so we get $f(x-\alpha\nabla f(x)) = f(x) - \alpha \|\nabla f(x)\|^2 + R(-\alpha\nabla f(x))$.

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

Take $\varepsilon < \|nablaf(x)\|$ to get $R(-\alpha\nabla f(x)) < \alpha \|\nabla f(x)\|^2$, then

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

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

Notice that the optimality condition also tells us how to move to get closer to the minimum.

An attentive reader may notice that the gradient is 0 in minima, maxima and saddle points (aka stationary point), hence how to discriminate among those?

We need to take infot account second derivatives, namely such second derivative should be positive for a minimum point.

5.1.2 Second order model

Fact 5.2. *Let $f \in C^2$. If x is a local minimum then $\nabla^2 f(x) \succeq 0$, in words the gradient is positive semidefinite.*

Proof by contraddiction. Our contraddictory hypothesis is that we are in a local minimum, but the Hessian is not positive semidefinite (formally, $\exists d$ s.t. $d^T\nabla^2 f(x)d < 0$ or equivalently, $\exists\lambda_i < 0$, noticing that $\bar{f}(\alpha) = tr(\alpha H_i)\nabla f(x)(\alpha H_i) = \alpha^2\lambda_i < 0$).

Obs: saying that Hessian is not positive semidefinite means saying that there is a direction of negative curvature.

Just like in previous case, we take the direction d normalized ($\|d\| = 1$).

Let us consider a step $x(\alpha) = x + \alpha d$ and then take the second-order Taylor formula (since $\nabla f(x) = 0$ there is no linear term involved)

$$f(x(\alpha)) = f(x) + \frac{1}{2}\alpha^2 d^T \nabla^2 f(x) d + R(\alpha d)$$

with $\lim_{\|h\|\to 0} \frac{R(h)}{\|h\|^2} = 0$, which means that the residual should go to 0 at least cubically.

Since $h = x - x(\alpha)$ we get that $\lim_{\alpha\to 0} \frac{R(\alpha d)}{\alpha^2} = 0$ or equivalently $\forall \varepsilon > 0 \, \exists \bar{\alpha} > 0$ s.t. $R(\alpha d) \leq \varepsilon \alpha^2 \forall 0 \leq \alpha < \bar{\alpha}$.

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

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

Hence x cannot be a local minimum. □

In a local minimum, there cannot be directions of negative curvature "when the first derivative is 0, second-order effects prevail".

As far as sufficient conditions are concerned, we can prove the following

Fact 5.3. *Let $f \in C^2$ and let the Hessian be symmetric (hence real eigenvalues). If $\nabla f(x) = 0$ and the Hessian is strictly positive definite ($\nabla^2 f(x) \succ 0$) then x is a local minimum.*

Proof. Since the gradient is 0 we get the following second order Taylor approximation

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

Hence, by definition of limit $\forall \varepsilon > 0 \; \exists \; \delta > 0$ s.t. $R(d) \leq \varepsilon \|d\|^2 \;\; \forall d$ s.t. $\|d\| < \delta$.

Since the Hessian is strictly positive definite $\lambda_{\min} > 0$ minimum eigenvalue of $\nabla^2 f(x)$, hence the variational caracterization of eigenvalues $d^T \nabla^2 f(x) d \geq \lambda_{\min} \|d\|^2$.

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

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

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

In the remaining part of this lecture we will look for conditions that ensure that one a local minimum is found, it is also a global minimum.

Until now, 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 x is to have that the Hessian is positive semidefinite everywhere ($\forall x \in \mathbb{R}^n$) aka f is a convex function.

5.2 Convexity

Let us introduce some preliminaries to the hypothesis of convex functions.

Definition 5.2 (Convex hull)**.** *Let $x, y \in \mathbb{R}^n$ we term **convex hull** and denote $conv(x, y) = \{z = \alpha x + (1 - \alpha)y \ : \ \alpha \in [0, 1]\}$ the segment joining x and y.*

Definition 5.3 (Convex set)**.** *We term **convex set** if for each couple in the set, the line linksing such points belongs to the set.*

*Formally, $C \subset \mathbb{R}^n$ is a **convex set** if $\forall\ x, y \in C$ $conv(x, y) \subseteq C$.*

Notice that "disconnected sets" cannot be convex sets.

Definition 5.4 (Convex hull of a set)**.** *Given a set S, we can "complete" it to a convex set:*

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

Equivalently, the convex hull of S = iterated convex hull of all $x, y \in S$ or the smallest convex set containing S

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

Fact 5.4. *A convex set is equal to its convex hull, formally C is convex $\iff$ $C = conv(C)$.*

Note

A more general definition of a convex hull is the following: $\text{conv}(\{x_1, \ldots, x_k\}) = \{x = \sum_{i=1}^{k} \alpha_i x_i \ : \ \sum_{i=1}^{k} \alpha_i = 1,\ \alpha_i \geq 0\ \forall i\}$

Definition 5.5 (Unitary simplex)**.** *We term **unitary simplex** the set of k non-negative numbers summing to 1, formally*

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

A few graphical examples are displayed in Figure 5.2.

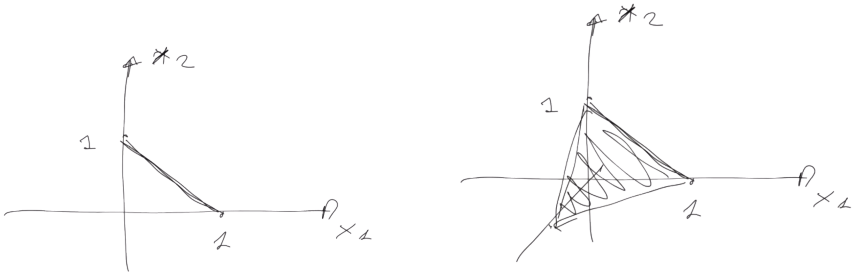
We are interested in sufficient conditions for convexity.

Definition 5.6 (Cone)**.** *We term **cone** the set $\mathcal{C} = \{x \ : \ \alpha x \in \mathcal{C} \forall \alpha \geq 0\}$.*

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.

The following sets are convex:

- Convex polytope $conv(\,\{x_1, \ldots, x_k\}\,)$, unitary simplex Θ



(A) In $\mathbb{R}^2$ (B) In $\mathbb{R}^3$

FIGURE 5.2: Unitary simplexes.

- Affine hyperplane: $\mathcal{H} := \{x \in \mathbb{R}^n \ : \ ax = b\}$
- Affine subspace: $\mathcal{S} := \{x \in \mathbb{R}^n \ : \ ax \leq b\}$
- Ball in p-norm, $p \geq 1$: $\mathcal{B}_p(\, x \,,\, r \,) = \{\, y \in \mathbb{R}^n \ : \ \| \, y - x \, \|_p \leq r \,\}$
- Ellipsoid: $\mathcal{E}(Q, x, r) := \{y \in \mathbb{R}^n \ : \ (y-x)^T Q (y-x) \leq r\}$ with $Q \succeq 0$. Notice that ellipsoids are levelsets of quadratic functions.
- Open versions by substituting "$<$" to "$\leq$"
- Cones
- Conical hull of a finite set of directions: $cone(\{d_1, \ldots, d_k\}) = \left\{ d = \sum\limits_{i=1}^{k} \mu_i d_i \ : \ \mu_i \geq 0 \ \forall \, i \right\}$
- Lorentz (ice-cream) cone: $\mathbb{L} = \left\{ x \in \mathbb{R}^n \ : \ x_n \geq \sqrt{\sum_{i=1}^{n-1} x_i^2} \right\}$
- Cone of positive semidefinite matrices: $\mathbb{S}_+ = \{\, A \in \mathbb{R}^{n \times n} \ : \ A \succeq 0 \}$

Fact 5.5. *The following operations preserve convexity.*

1. *Given a possibly infinite family of convex sets* ($\{C_i\}_{i \in I}$)*, the intersection* ($\bigcap_{i \in I} C_i$) *convex;*
2. *If we have convex sets in different subspaces, their cartesian product is a convex set* ($C_1, \ldots, C_k$ *convex* $\iff$ $C_1 \times \cdots \times C_k$ *convex);*
3. *Given a convex set, its image under a linear mapping (aka scaling, translation, rotation) is a convex set. Formally,* C *convex* $\implies A(C) := \{x = Ay + b \ : \ y \in C\}$ *convex;*
4. C *convex* $\implies A^{-1}(C) := \{x \ : \ Ax + b \in C\}$ *convex (inverse image under a linear mapping);*

5. *Let C_1 and C_2 convex and let $\alpha_1, \alpha_2 \in \mathbb{R}$. Then $\alpha_1 C_1 + \alpha_2 C_2 := \{x = \alpha_1 x_1 + \alpha_2 x_2 : x_1 \in C_1, x_2 \in C_2\}$ convex;*

6. *$C \subseteq \mathbb{R}^n = \mathbb{R}^{n_1} \times \mathbb{R}^{n_2}$ convex $\Longrightarrow$*

   SLICE: *$C(y) := \{x \in \mathbb{R}^{n_1} \ : \ (x, y) \in C\}$ convex;*

   PROJECTION: *$C^1 := \{x \in \mathbb{R}^{n_1} \ : \ \exists y \text{ s.t. } (x, y) \in C\}$ convex*

   A pictorial example in Figure 5.3;

7. *C convex $\Longrightarrow$ $int(C)$ and $cl(C)$ convex*

(A) Slice on $\bar{y}$ (B) Project on the x component

FIGURE 5.3: Pictorial examples of slicing and projecting.

Theorem 5.6. *$\mathcal{P}$ is a polyhedron iff $\exists \{x_1, \ldots, x_k\}$ and $\{d_1, \ldots, d_h\}$ s.t. $\mathcal{P} = conv(\{x_1, \ldots, x_k\}) + cone(\{d_1, \ldots, d_h\})$.*

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

Definition 5.7 (Convex function)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}^m$ be a function. We say that f is **convex** if $\forall x, y \in \mathbb{R}^n$, the segment that joins $f(x)$ and $f(y)$ lies above the function.*

*In other words, f is **convex** iff $epi(f)$ is convex, where epi denotes the epigraph of the function, graphically speaking, the region which is above the function line (in the plot).*

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

Equivalently, $\forall x^1, \ldots, x^k$, $\alpha \in \Theta^k$

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

Definition 5.8 (Sublevel graph)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}^m$ be a function. We term **sublevel graph** of $f(x)$ the projection on the x axis of the portions of the epigraph which lie below the constant $y = \bar{x}$.*

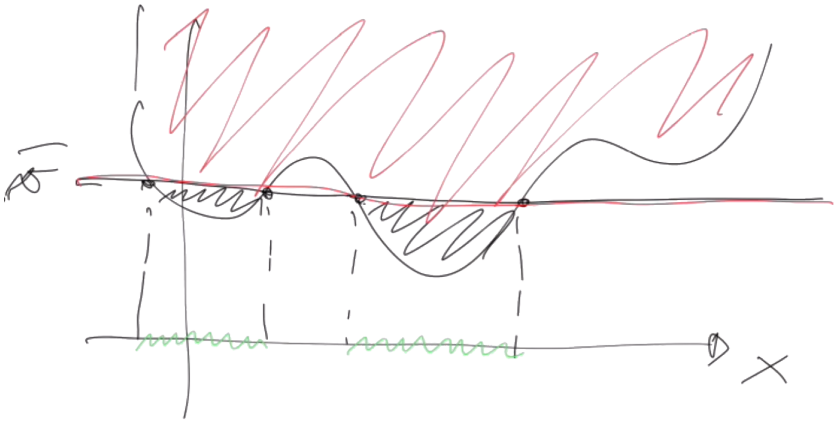


FIGURE 5.4: Pictorial example of sublevel graph. Such a graph is drawn in green in the figure.

Fact 5.7. *The following holds:*

- *Let f convex. Then $S(f, v)$ convex $\forall v \in \mathbb{R}$;*
- *f is concave if $-f$ is convex ("convex analysis is a one-sided world").*

The second statement of Proposition 5.7 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 5.9 (Strict convexity)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}^m$. We term f* ***strictly convex*** *iff $\alpha f(x) + (1-\alpha)f(y) > f(\alpha x + (1-\alpha)y)$.*

Definition 5.10 (Strong convexity)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}^m$. We term f* ***strongly convex modulus*** *$\tau > 0$ iff $f(x) - \frac{\tau}{2}\|x\|^2$ is convex.*

Formally,

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

Next lecture we will talk about how we can check tat a function is convex, operationally.

6 11th of October 2018 — A. Frangioni

Last lecture we left with the task of understanding how to check if a function is convex.

As we already stated for convex sets, we can prove that a function is convex deriving from convex functions, though "convex friendly" operations.

Note

There is a softare called CVX, designed to model convex object. 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.

The following functions are convex:

1. $f(x) = wx$: linear functions are both convex and concave;
2. $f(x) = \frac{1}{2}xQx + qx$ if convex iff $Q \succeq 0$;
3. $f(x) = e^{ax}$ for any $a \in \mathbb{R}$ and $x \in \mathbb{R}$
4. $f(x) = -\log(x)$ for $x > 0$
5. $f(x) = x^a$ for $a \geq 1$ or $a \leq 0$ on $x \geq 0$;
6. $f(x) = \|x\|_p$ for $p \geq 1$;
7. $f(x) = \max\{x_1, \ldots, x_n\}$;
8. for any convex set C, its indicator function

$$\imath_C(x) = \begin{cases} 0 & \text{if } x \in C \\ +\infty & \text{if } x \notin C \end{cases} \quad (\text{l.s.c.} \iff C \text{ closed})$$

9. $A \in \mathbb{R}^{n \times n}$ symmetric, eigenvalues customarily ordered $\lambda_1 \geq \lambda_2 \geq \ldots \lambda_n$: $f_m(A) = \sum_{i=1}^{m} \lambda_i$ (sum of m largest eigenvalues)

Fact 6.1. *The following operations preserve convexity:*

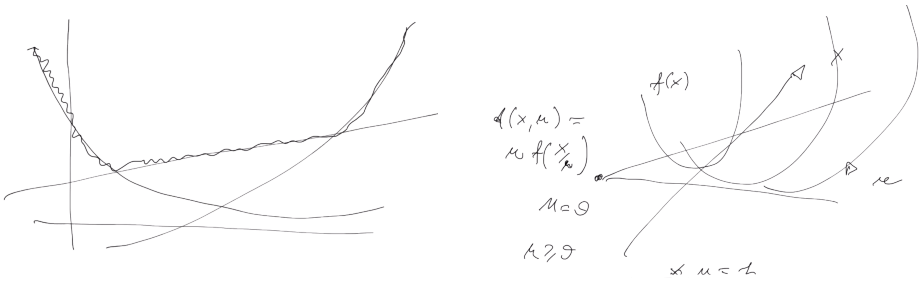
1. *f, g convex, α, $\beta \in \mathbb{R}_+ \implies \alpha f + \beta g$ convex (non-negative combination);*
2. *$\{f_i\}_{i \in I}$ (infinitely many) convex functions $\implies f(x) = \sup_{i \in I} f_i(x)$ convex, see Figure 6.1a;*
3. *Pre-composition with linear function is convex, formally f convex $\implies f(Ax+b)$ convex;*
4. *Post-composition with increasing convex function is convex. Formally, $f : \mathbb{R}^n \to \mathbb{R}$ convex, $g : \mathbb{R} \to \mathbb{R}$ convex increasing $\implies g \circ f = g(f(x))$ is convex;*

5. f_1, f_2 *convex* $\Longrightarrow$ $f(x) = \inf\{ f_1(x_1) + f_2(x_2) \ : \ x_1 + x_2 = x\}$ *convex (infimal convolution);*

6. g *convex* $\Longrightarrow f(x) = \inf\{g(y) \ : \ Ay = x\}$ *convex (image under a linear mapping, aka value function of convex constrained problem);*

7. $g(x, y) : \mathbb{R}^{n+m} \to \mathbb{R}$ *convex* $\Longrightarrow$ $f(x) = \inf\{g(x, y) \ : \ y \in \mathbb{R}^m\}$ *convex (partial minimization);*

8. $f(x)$ *convex* $\Longrightarrow$ $\tilde{f}(x, u) = uf(x/u)$ *when* $u > 0$, $\tilde{f}(x, u) = \infty$ *otherwise, convex (perspective or dilation function of* f*), see Figure 6.1b.*



(A) Suprema — second property (B) Perspective — eighth property

FIGURE 6.1: Graphic hints.

Fact 6.2. *Let* $f : \mathbb{R}^n \to \mathbb{R} \cup \infty$ *a convex function. If* $\exists \bar{x} \in dom(f)$ *such that* $f(\bar{x}) = -\infty$*, then* $f \equiv -\infty$*.*

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

Fact 6.3. *Let* $f : \mathbb{R}^n \to \mathbb{R}$ *be a convex function. Then* f *is Lipschitz continuous* $\forall$ *bounded convex set* $S \subseteq int(dom(f))$ *but on* $\partial dom(f)$ *(the border of the domain) anything can happen.*

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

A couple of examples of Proposition 6.3 can be found in Figure 6.2.

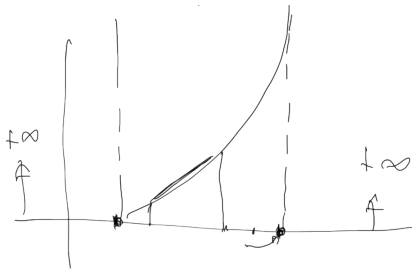
Fact 6.4. *Let convex* $f : \mathbb{R}^n \to \mathbb{R}$*. Then it is Lipschitz continuous on any bounded set and continuous everywhere.*

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

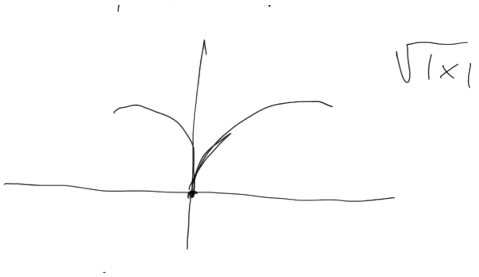
Theorem 6.5 (Convexity characterization)**.** *Let* $f \in C^1$*. It is convex on* C *convex iff*

$$f(y) \geq f(x) + \nabla f(x)(\, y - x\,) \forall\, x, y \in C$$

In other words, given a point x*, we compute the derivative and the value fo the function is above the derivative in that point.*



(A) Take a compact set in the interior of the domain (far from the boundaries) there the function is Lipschitz continuous.



(B) If a function is not Lipschitz on a compact subset it is not convex.

FIGURE 6.2

Proof.

$\Rightarrow$) $\alpha(f(y) - f(x)) \geq f(\alpha(y - x) + x) - f(x)$, send $\alpha \to 0$

$\Leftarrow$) TODO

□

Theorem 6.6. *Let $f \in C^1$ convex. x is a stationary point iff x is a global minimum.*

Fact 6.7. *Let f be twice differentiable (aka has Hessian). f is convex iff the Hessian is positive semidefinite.*

Formally, let $f \in C^2$. f is convex on the open set S iff $\nabla^2 f(x) \succeq 0 \ \forall x \in S$.

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 $\mathbb{R}$ is to solve the following problem:

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

thanks to

Theorem 6.8 (Essential objective)**.** *x_* optimal for $(P) \iff x_*$ local minimum of f_X.*

6.0.1 Subgradients and subdifferentials

Definition 6.1 (Subgradient)**.** *For each $s \in \mathbb{R}$ we term **s-subgradient** of f at x as:*

$$f(y) \geq f(x) + s(y - x) \qquad \forall\, y \in \mathbb{R}^n$$

Let us assume that the minimum of the non differentiable function resides in one of its kinky points, then for $s = 0$ we have a subgradient which is flat and this is a sufficient condition for minimality, for a pictorial example see Figure 6.4.

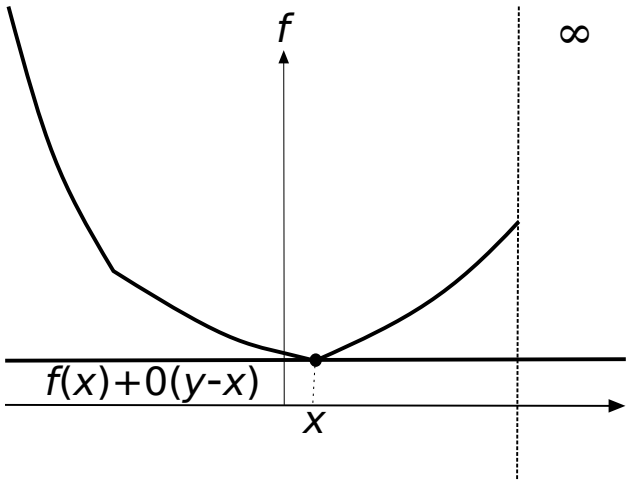


FIGURE 6.3: Pictorial example of subgradients of a non diffrentaible function.

The issue here is that it is unfeasible to check if the subgradient with $s = 0$ is a subgradient for f.

Definition 6.2 (Subdifferential)**.** *We term* ***subdifferential*** *the set of all possible subgradients. Formally:*

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

Theorem 6.9. *x global minimum* $\iff 0 \in \partial f(x)$.

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,$\partial f(x)$ closed and convex, compact $\forall\, x \in int\, dom(f)$.

Moreover, we can prove the following

Fact 6.10. $\partial f(x) = \{\nabla f(x)\} \iff f$ *differentiable at* x.

An attentive reader may have noticed that in the case of non differentiable functions it is not possible to derive the directional derivative from the gradient ($\frac{\partial f}{\partial d}(x) =< \nabla f(x), d >$), but we can prove the following

Fact 6.11. $\frac{\partial f}{\partial d}(x) = \sup\{< s, d > : \ s \in \partial f(x)\} \implies d$ *is a descent direction* $\iff < s, d >< 0 \ \forall \ s \in \partial f(x)$.

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

Example 6.1. *Le us assume we are in x and we want to move towards x^* knowing only the subdifferentials. $(-)\partial f(x)$ is convex and compact and All $(-)g \in \partial f(x)$ "point towards x_*":* $< g, x - x_* >< 0$.

Not all of them are descent directions, but the (opposite to the) minimum-norm one is a descent direction.

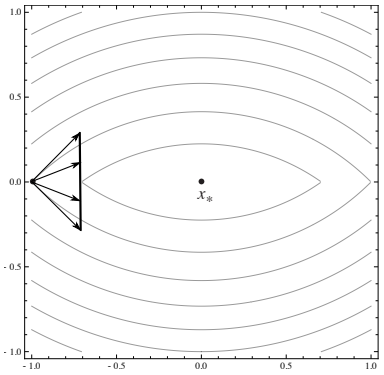


FIGURE 6.4: There are many different subgradients in x. We pick the one with minimum norm among the ones which have a negative scalar product with $x - x^*$.

Notice that in $\mathbb{R}^2$ if we take a function and compute the subdifferential, we can scale both the function and the subdifferential by any positive constant (negative contants would lead to concave functions). Formally,

Fact 6.12. *Let $f, g : \mathbb{R}^n \to \mathbb{R}$ and take $\alpha, \beta \in \mathbb{R}_+$, then $\partial[\alpha f + \beta g](x) = \alpha \partial f(x) + \beta \partial g(x)$.*

Fact 6.13 (Chain rule)**.**
- *Let $f : \mathbb{R}^n \to \mathbb{R}$ and let $A \in M(n, \mathbb{R})$ and $b \in R^n$ then $\partial[f(Ax + b)] = A^T[\partial f](Ax + b)$;*
- *Let $g : \mathbb{R} \to \mathbb{R}$ increasing, then $\partial[g(f(x))] = [\partial g](f(x))[\partial f](x)$.*

Definition 6.3 (ε-subgradient)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$. s is ε-**subgradient** at x if:*

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

support hyperplane passing ε below epi(f).

Fact 6.14. *Given a point x, the value of the finction in x cannot be further from ε the minimum value fo the function. Formally, $0 \in \partial_\varepsilon f(x) \iff x$ is ε-optimal.*

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

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

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

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 form an initial guess x_0 and go on $x^i \rightsquigarrow x^{i+1}$. We want to move towards the otimum.

How to be sure we are in an optimum?

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

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

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

TRUST REGION: first choose α^i (trust radius), then choose d^i.

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

7.1.1 Gradient method for quadratic functions

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

We should not move too far from x^i, so we want $\alpha_i \to 0$ and then $d_i = argmin\{\lim_{t\to 0} \frac{f(x+td)}{t}\} = -\nabla f(x^i)$, aka the steepest descent direction.

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

At each step we want $\alpha^i \in \text{argmin}\{f(x^i + \alpha d^i) \ : \ \alpha \geq 0\}$.

Linear functions are unbounded below, so we would like to use a different family of functions. The easiest family of functions which are still more complex than linear ones are quadratic functions.

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

where $Q \succeq 0$ otherwise f is unbounded below.

The minimum here is the point in which the gradient ($\nabla f(x) = Qx + q$) is 0.

ALGORITHM 7.1 Pseudocode for quadratic functions local minimum detection.

```
1: procedure SDQ(f, x, ε)
2:     while (‖∇f(x)‖ > ε) do
3:         d ← −∇f(x);
4:         α ← ‖d‖² / (dᵀQd);
5:         x ← x + αd;
6:     end while
7: end procedure
```

For the time being we do not go into detail of how to chose the ε such that we can stop when the norm of the gradient is smallert than such a constant.

Let us see how to obtain the formula for α.

We are interested in computing the minimum of $\{f(x^i + \alpha d^i) : \alpha \geq 0\}$.

Let us do come algebra to describe better such an f:

$$\begin{aligned} f(x^i + \alpha d^i) &= \frac{1}{2}(x^i + \alpha d^i)^T Q(x^i + \alpha d^i) + q(x^i + \alpha d^i) \\ &= \cancel{\frac{1}{2}(x^i)^T Q x^i} + (x^i)^T Q(\alpha d^i) + \frac{1}{2}(d^i)^T Q d^i + \cancel{q x^i} + \alpha(q d^i) \\ &= [\frac{1}{2}(d^i)^T Q d^i]\alpha^2 + \alpha[(x^i Q + q)d^i] \\ &\overset{(1)}{=} [\frac{1}{2}(d^i)^T Q d^i]\alpha - \left\|d^i\right\| \end{aligned} \tag{7.1}$$

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

Can we prove that the sequence of iterates is moving towards the optimum?

For this proof let us assume that $\varepsilon = 0$, hence the procedure will neve stop. We want to prove that the sequence $\{x^i\}$ is (or contains) a minimizing sequence.

First of all we can state that the sequence is monotone, so it has a limit for sure.

What we can prove is that the the point where the sequence is converging is a stationary point. $\lim_{i\to\infty} < \nabla f(x^i), \nabla f(x^{i+1}) >= 0 \overset{(*)}{=} < \nabla f(x), \nabla f(x) >$ and this holds because of the fundamental relationship $< \nabla f(x^i), \nabla f(x^{i+1}) >= 0$.

Notice that $\overset{(*)}{=}$ follows from the fact that the gradient is continuous.

How fast is the convergence?

In general, showing how fast $\| x^i - x_* \|$ decreases is more involved that showing how fast $f(x^i) - f_*$ decreases, but we do not know f_*.

We concentrate on computing $\lim_{i\to\infty} \frac{f(x^{i+1}) - f_*}{f(x^i) - f_*{}^p} = R$.

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 $x^* = -Q^{-1}q$, we get that $f(x^*) = \frac{1}{2}q^TQ^{-1}QQ^{-1}q - q^TQ^{-1}q = \frac{1}{2}q^TQ^{-1}q - q^TQ^{-1}q = -\frac{1}{2}q^TQ^{-1}q$.

Let us use a nifty trick: let us define:

$$\begin{aligned}
\bar{f}(x) &= \frac{1}{2}(x - x_*)^TQ(x - x_*) \\
&= \frac{1}{2}x^TQx + \frac{1}{2}{x_*}^TQx_* - x^T(Qx) \\
&\overset{(2)}{=} \frac{1}{2}x^TQx + \frac{1}{2}Q^{-1}{q}^TQ(Q^{-1}q) + qx \\
&= \frac{1}{2}x^TQx + \frac{1}{2}q^T\cancel{Q^{-1}}\cancel{Q}Q^{-1}q + qx \\
&= \frac{1}{2}x^TQx + \frac{1}{2}q^TQ^{-1}q + qx \\
&= f(x) - f_*
\end{aligned} \tag{7.2}$$

8 19th of October 2018 — A. Frangioni

8.1 Gradient method for quadratic functions

This is the simplest possible family of functions where a minumum exists.

Do you recall?

A quadratic function is defined as: $f(x) = \frac{1}{2}x^T Qx + qx$, so its gradient is the following $\nabla f(x) = Qx + q$

We are interested in finding local minima of the function f.

Fact 8.1. *A quadratic function $f : \mathbb{R}^n \to \mathbb{R}^n$, s.t. $f(x) = \frac{1}{2}x^T Qx + qx$ admits a minimum iff $Q \succeq 0$ (Q is positive semidefinite).*

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

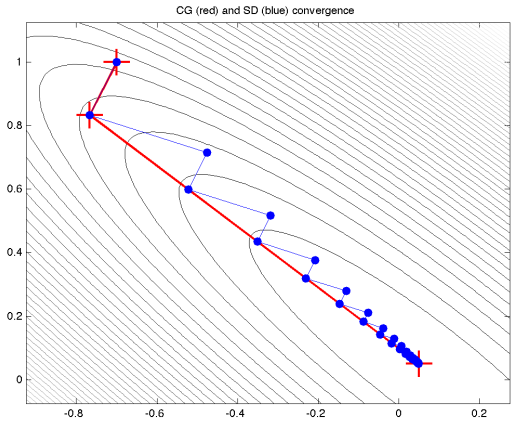


FIGURE 8.1: Some iterations of the gradient method

Fact 8.2. $\forall i \ < d^i, d^{i+1} > = 0$.

Proof. TODO □

$i\alpha^i = \frac{\|d^i\|^2}{d^{iT}Qd^i}$, $x^i \to \overline{x}$

Our aim is to estimate how fast this converges.

$f(x^i) - f_* = -\frac{1}{2}q^T Q^{-1} q$ we know everything for this function, from last lecture ($x^* = -Q^{-1}q$).

What can we say about $f(x^{i+1}) - f_* = \frac{1}{2}(x^{i+1} - x_*)^T A(x^{i+1} - x_*)$?

I want it in terms of $x^{i+1} = x^i - \frac{\|Qx^i - q\|^2}{(Qx^i-q)^T Q(Qx^i-q)}$

From this fomula we can say that the error at the current step $(i+1)$ is equal to the error of the step before (i) divided by something. More precisely, $f_*(x) = \frac{1}{2}(x - x_*)^T Q(x - x_*) = f(x) + \frac{1}{2}{x_*}^T Q x_* = f(x) - f_*$.

If the quantity after the minus is positive but less than 1 the error at the next step will be less than the error at the previous step. 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 also at the beginning.

Fact 8.3. *If Q is positive semidefinite $d^T Q d = \|d\|_Q^2$.*

Fact 8.4. *If Q is positive definite we can say that the error goes like $1 - \left(\frac{\|d_i\|^2}{(d^{i\,T} Q d^i)(d^{i\,T} Q^{-1} d^i)}\right)$ or, equivalently, $1 - \frac{{\|d_i\|_I}^2}{{\|d_i\|_Q}^2} \frac{{\|d_i\|_I}^2}{{\|d_i\|_{Q^{-1}}}^2}$.*

We are measuring a vector with three different norms.

We would like to estimate $\frac{<d_i, d_i>}{{d_i}^T Q d_i}$.

Given λ_i the eigenvalues of Q, $\frac{1}{\lambda_i}$ are the eigenvalues of the matrix Q^{-1}.

We can say that $\lambda^n \|x\|^2 \leq x^T Q x \leq \lambda^1 \|x\|^2$, 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 reursivley 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 upperbound. A coarse upperbound is $(1 - \frac{\lambda^n}{\lambda^1})$, but we can prove more:

$$\forall\, x \in \mathbb{R}^n \quad \frac{\|x\|^4}{(x^T Q x)(x^T Q^{-1} x)} \geq \frac{4\lambda^1 \lambda^n}{(\lambda^1 + \lambda^n)^2}$$

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 eachother the algorithm is converging fast. We can say that, since the eigenvalues are the axis of the ellipsoid, the algorithm is converging fast when the ellipsoid has a round shape.

We can provide an even better estimate, which is $\left(\frac{\lambda^1 - \lambda^n}{\lambda^1 + \lambda^n}\right)^2$

How can we understand if the number of iterations needed to converge is a good result? Well, it depends on how that number is obtained. If it's dimensional independent it's 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.

Obs: We found 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 upperbound 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, becomes 0 on the machine, so we can't give an upperbound.

We will see how to deal with this case.

8.2 MatLab implementation

Let's talk about the implementation. First of all we need to set a proper value for ϵ. A good idea would be the norm of the gradient. We need to give also some performance bound, like maximum number of iterations or the maximum amount of time.

MatLab call:

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

where fStar is the optimal value, which should give us an idea of the convergence.

Note

- Always check the coherence of input values (check if the user passed allowed parameters);
- Always 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);
- Always check in your code the accepted values of variables during calculations: for example if you need to divide 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's happening at each step.

In the code of that function we also kept track of the actual ratio between the error at one step and the error at the next one. This information tells us how many orders of magnitude the error decrease. We can find starting points where the ratio is exaclty 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 be worse.

We saw from some examples that if the conditioning grows the number of iterations needed to find the minumum increases as well.

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

8.2.1 Error

When we are running an algorithm, starting from a point that will not lead to the optimum we would like to stop anyway, at a certain point, when we are close enough to the solution.
$\varepsilon_A = f(x^i) - f_* \leq \varepsilon$ (absolute error)

In this context, we introduce the consept of **relative error**, since the error may be big or small if compared with the value of the function.

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

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

The problem is that often we don't know f^*, so it's impossible to compute this kind of error.

In this very common case a good lower bound $\underline{f} \leq f^*$for f^*. In this course we won't focus on finding $\underline{f}$.

Another stopping conditions may be the following:

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

The sencond stopping condition is expressed in relation to the first value for the norm of the gradient.

We usually chose the norm of the gradient as a threshold for precision, but we don't know how this quantity relates to ϵ_A or ϵ_B.

Example 8.1. *for $X = \mathcal{B}(0, r)$ and f convex, estimate ε_A when $\| \nabla f(x^i) \| \leq \varepsilon$*

I've got a convex function and I know the the minimum is in a ball. We can minimize the linear function in the range of the ball and that minumum is surely a lower bound.

9 24th of October 2018 — A. Frangioni

9.1 Gradient method for non-quadratic functions

We want to move from quadratic functions to wider families of functions.

Do you recall?

The step size α in the quadratic case is defined as follows: $\alpha = \frac{\|d\|^2}{d^T Q d}$.

We would like to find a more general form for the step size, which doesn't 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.

We will see that, differently from the quadratic case (where the gradient was $\nabla f(x) = Qx + q$) computing the gradient in this more general case isn't easy at all.

We may recall from last lecture that the proof of the orthogonality of the gradient doesn't depend on the quadratic nature of our functions, so it works in this case too.

9.1.1 How fast does it converge?

Given Algorithm 9.1 for finding minima of quadratic functions (that differs from the one provided for quadratic ones for the step size) we would like to understand how fast the convergence is.

ALGORITHM 9.1 Pseudocode for non-quadratic functions local minimum detection.

```
1: procedure SDQ(f, x, ε)
2:     while (‖∇f(x)‖ > ε) do
3:         d ← −∇f(x);
4:         α ← argmin{f(x + αd)};
5:         x ← x + αd;
6:     end while
7: end procedure
```

Theorem 9.1. *Let f be a function in C^2 and let x_* be a local minimum for f s.t. $\nabla^2 f(x_*) \succ 0$ (which means that the Hessian matrix is strictly positive definite):*

$$\{ x^i \} \to x_* \Longrightarrow \{ f(x^i) \} \to f(x_*)$$

linearly, with same R as the quadratic case, depending on λ_1 and λ_n of $\nabla^2 f(x_*)$.

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 we recall from the previous lecture, we must pay attention to **conditioning**.

At this point we need to work on finding the local minimum of the one dimensional function φ^i, s.t.:

$$\varphi^i(\alpha) = f(x^i + \alpha d^i)$$

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

Let's omit the i, since we are concentrating on a single iteration.

We are interested in finding a α^* such that $\varphi'(\alpha^*) = 0$.

Example 9.1. *Let's suppose we are in* $\mathbb{R}^2$ *and* $f(x, y) = x^2 e^y$. *We can differentiate* F *and* $\nabla f(x, y) = (2xe^y, x^2 e^y)$.

At the i-th iteration $(x, y) = (1, 0)$, *so* $\nabla f(1, 0) = (2, 1)$. *Now* $x(\alpha) = (1, 0) - \alpha(2, 1) = (1 - 2\alpha, 0 - \alpha)$.

At this point we obtain $\varphi(\alpha) = f(x(\alpha)) = (1 - 2\alpha)^2 e^{-\alpha}$.

Chi è $x(\alpha)$?

It's hard to find the roots of this function. The points we can find are not $\varphi'(\alpha) = 0$, but instead $|\varphi'(\alpha) \leq \varepsilon'|$, where the meaning of ε is bounding the directional derivative to be small.

Fact 9.2. *Let* $\varphi : \mathbb{R} \to \mathbb{R}$, *such that* $\varphi(\alpha) = f(x^i + \alpha d^i)$, $\varphi'(\alpha) =< \nabla f(x^i + \alpha d^i), d >$.

Proof.
TODO: using the **chain rule**: $f : \mathbb{R}^m \to \mathbb{R}^k$, $g : \mathbb{R}^n \to \mathbb{R}^m$, $h : \mathbb{R}^n \to \mathbb{R}^k$ such that $h(x) = f(g(x)) \Rightarrow \mathbf{Jh(x)} = \mathbf{Jf(g(x))} \cdot \mathbf{Jg(x)}$

Obs: $Jf \in \mathbb{R}^{k \times m}$, $Jg \in \mathbb{R}^{m \times n}$ and $Jh \in \mathbb{R}^{k \times m} \cdot \mathbb{R}^{m \times n} = \mathbb{R}^{k \times n}$. □

Fact 9.3. *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 d^i without perturbing the behaviour of the algorithm: $d^i = -\frac{\nabla f(x^i)}{\|\nabla f(x^i)\|}$. 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 $\|d^i\| = 1$ and

$$\begin{aligned}
\varphi'(0) &= \frac{\partial f}{\partial d}(x) \\
\text{(From Proposition 9.2)} &=< \nabla f(x), d > \\
&=< \nabla f(x), \frac{-\nabla f(x)}{\|\nabla f(x)\|} > \\
&= -\frac{< \nabla f(x), \nabla f(x) >}{\|\nabla f(x)\|} \\
&= -\frac{\|\nabla f(x)\|^2}{\|\nabla f(x)\|} \\
&= -\left\|\nabla f(x^i)\right\|
\end{aligned} \tag{9.1}$$

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

Hence, if $\{x^i\} \to x$:

$$\lim_{i\to\infty}\left|< \frac{\nabla f(x^i)}{\|\nabla f(x^i)\|}, \nabla f(x^{i+1}) >\right| =< \frac{\nabla f(x)}{\|\nabla f(x)\|}, \nabla f(x) >= \|\nabla f(x)\| \leq \varepsilon \tag{9.2}$$

Since $\|\nabla f(x^i)\| > \varepsilon$ we have the thesis. □

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

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

This convergence isn't the perfect mathematical convergence, since $\varepsilon \neq 0$, because the line search will never terminate.

9.1.2 Exact line search, first orderd approach

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. Since we are talking about numerical algorithms we are going to stop a little before the minimum is reached.

Key idea: We would like to reduce the range in which performing the search, at each step.

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.

Since the gradient is continuous the directional derivative is continue, 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.

The algorithm is the following:

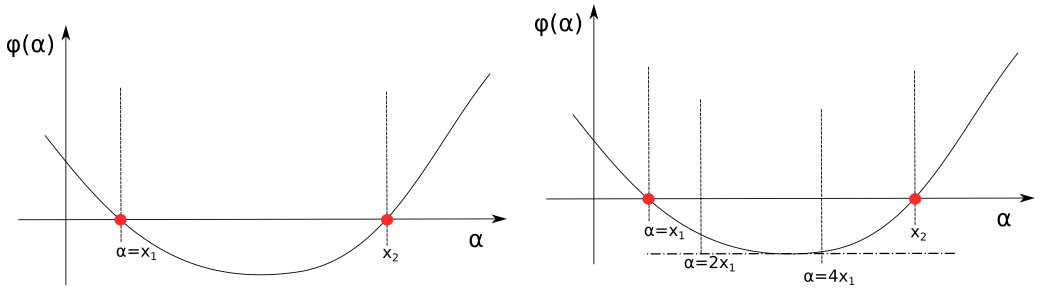


FIGURE 9.1: First, we restrict from $\mathbb{R}$ to $[x_1, x_2]$, then double α until the derivative is greater than 0.

ALGORITHM 9.2 First algorithm for exact line search

```
1: ᾱ ← x_1; # or whatever value > 0
2: while (φ'(ᾱ) < 0) do
3:     ᾱ ← 2ᾱ; #or whatever factor > 1
4: end while
```

We'll stop when $\alpha < -10^{308}$, which is the smallest value for a double, since we will get a numerical error and stop.

Works if φ **coercive**: $\lim_{\alpha\to\infty} \varphi(\,\alpha\,) = \infty$ (e.g. f strongly convex).

Exercise 9.1. *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.

An alternative to the algorithm presented above may be the bisection algorithm, which follows.

ALGORITHM 9.3 Bisection algorithm

```
1: procedure LSBM((φ', ᾱ, ε))
2:     α_- ← 0;
3:     α ← α_+;
4:     α_+ ← ᾱ;
5:     while (|φ'(α)| > ε) do
6:         α ← (α_+ + α_-)2;
7:         if (φ'(α) < 0) then
8:             α_- ← α;
9:         else
10:            α_+ ← α;
11:        end if
12:    end while
13: end procedure
```

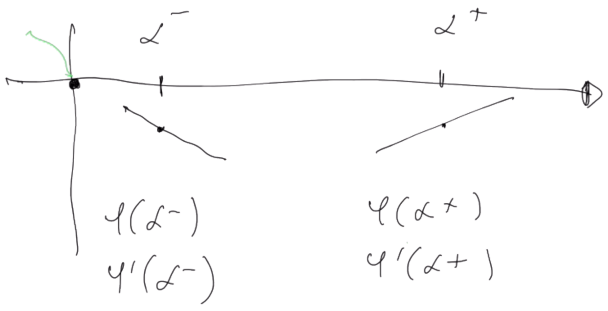


FIGURE 9.2: The information we have about function φ

We would like to improve this algorithm too, finding a better point in the middle than the middle point.

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:

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

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.

We can say something more, since the following fact holds:

Fact 9.4. *Let $\varphi \in C^3$, then quadratic interpolation has convergence of order $1 < p < 2$ (superlinear).*

In Figure 9.3 we can observe a situation in which the hypothesis of Proposition 9.4 aren't satisfied.

FIGURE 9.3: If the function isn't C^3 and one derivative is very big, then the range doesn't shrink much.

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

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

Another idea is, since we have four equations, to build a cubic function, although the operation is very long. In this case the convergence get quadratic, which is better than linear.

10 25th of October 2018 — A. Frangioni

In order to choose a descending direction, we have more choices than the opposite of the gradient.

We want the derivative to be almost zero, given a tolerance value.

Let's see another variant of line search.

10.0.1 Line search: second order approaches

Theorem 10.1. *Given $f \in C^2$, $\exists \varphi''(\alpha) = d^T \nabla^2 f(x + \alpha d) d$ and it's continuous.*

Proof. Via chain rule. □

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 10.1 (Model−Newton tangent method)**.** *Our model, in this case is:*

$$\varphi'(\alpha) \approx \varphi'(\alpha^k) + \varphi''(\alpha^k)(\alpha - \alpha^k)$$

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

Algorithm 10.1 Pseudocode for Newton method.

```
1: procedure LSNM(φ', φ'', α, ε)
2:     while (|φ'(α)|> ε) do
3:         α ← α − φ'(α)/φ''(α);
4:     end while
5: end procedure
```

We need to understand when and why $\varphi''(\alpha) \neq 0$ and when and why this method converges.

Theorem 10.2. *Let $\varphi \in C^3$ such that $\varphi'(\alpha_*) = 0$ and $\varphi''(\alpha_*) \neq 0$. $\exists\ \delta > 0$ s.t. if $\alpha^0 \in [\alpha_* - \delta, \alpha_* + \delta]$ then $\{\alpha^k\} \to \alpha_*$, with $p = 2$.*

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

We want to compute how much the error is, if compared to the error at the previous iteration. Since $\varphi(\alpha_*) = 0$ we can use a dirty trick.

1. Since $\alpha^{k+1} \overset{(1)}{=} \alpha^k - \frac{\varphi'(\alpha_*)}{\varphi''(a^k)}$ and $\varphi'(\alpha_*) \overset{(2)}{=} 0$, we obtain:

$$\begin{aligned} \alpha^{k+1} - \alpha_* &\overset{(1)}{=} \alpha^k - \alpha_* - \frac{\varphi'(\alpha^k)}{\varphi''(\alpha^k)} \\ &\overset{(2)}{=} \alpha^k - \alpha_* - \frac{\varphi'(\alpha^k) - \varphi'(\alpha_*)}{\varphi''(\alpha^k)} \\ &= \frac{[\varphi'(\alpha^k) - \varphi'(\alpha_*) + \varphi''(\alpha^k)(\alpha^k - \alpha_*)]}{\varphi''(\alpha^k)} \end{aligned} \tag{10.1}$$

Where the term inside the square parenthesis is the first order model, centered in α_* computed in α^k.

2. Now we can use the first form of Taylor's formula, which says$\exists\ \beta \in [\alpha^k, \alpha^*]$ s.t. $\varphi'(\alpha_*) \overset{(3)}{=} \varphi'(\alpha^k)+\varphi''(\alpha^k)(\alpha^k-\alpha_*)+\varphi'''(\beta)\frac{(\alpha^k-\alpha_*)^2}{2}$. Let's see what happens to $\alpha^{k+1}-\alpha_*$:

$$\begin{aligned}\alpha^{k+1}-\alpha_* &\overset{(5)}{=} \frac{[\varphi'(\alpha^k)-\varphi'(\alpha_*)+\varphi''(\alpha^k)(\alpha^k-\alpha_*)]}{\varphi''(\alpha^k)} \\ &\overset{(3)}{=} \frac{-\varphi'''(\beta)}{2\varphi''(\alpha^k)}(\alpha^k-\alpha_*)^2\end{aligned} \tag{10.2}$$

3. We can say that the quantity $2\varphi''(\alpha^k)$ doesn't become too small and that the numerator $\varphi'''(\beta)$ doesn't 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 $\alpha,\ \beta \in [\,\alpha_*-\delta\,,\,\alpha_*+\delta]$

$$\begin{aligned}|\alpha^{k+1}-\alpha_*| &\overset{(6)}{=} \frac{-\varphi'''(\beta)}{2\varphi''(\alpha^k)}(\alpha^k-\alpha_*)^2 \\ &= |\alpha^{k+1}-\alpha_*| \leq [\frac{k_1}{2k_2}](\alpha^k-\alpha_*)^2\end{aligned} \tag{10.3}$$

We may notice that $[\frac{k_1}{2k_2}]$ may be very large, but it's multiplied for $(\alpha^k-\alpha_*)^2$, which means that if we start close enough to α^* it's ok.

$$\begin{aligned}|\alpha^{k+1}-\alpha_*| = |\alpha^{k+1}-\alpha_*| &\leq [\frac{k_1}{2k_2}](\alpha^k-\alpha_*)^2 \\ &= |\alpha^{k+1}-\alpha_*| \leq [\frac{k_1}{2k_2}](\alpha^k-\alpha_*)(\alpha^k-\alpha_*)\end{aligned} \tag{10.4}$$

Where $[\frac{k_1}{2k_2}](\alpha^k-\alpha_*) < 1$, so $\frac{k_1(\alpha^k-\alpha_*)}{2k_2} \leq 1 \Longrightarrow \left|\alpha^{k+1}-\alpha_*\right| < \left|\alpha^k-\alpha_*\right|$ At this point, if we start from a point a^0 close enough to α^* (according to this formula) then $\{\,\alpha^k\,\} \to \alpha_*$ and the convergence is quadratic.

□

In the end, we may conclude that if we start from the right point we converge with a quadratic speed.

Problem: This solution makes us compute all the derivatives until the third one. We will now see a solution to this issue.

10.0.2 Exact line search: zeroth-order approaches

Can we do line search without computing derivatives at all? Following this approach we can circumvent the problem of the existence of derivatives. In the case of derivatives definite, it's better if we don't have to compute them.

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

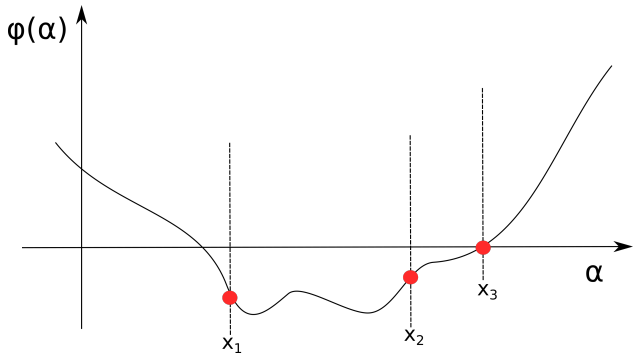


FIGURE 10.1: The interesting interval is $[x_1, x_2]$, since $\varphi(x_2) > \varphi(x_1)$ and we are allowed to exclude the interval $[x_3, +\infty)$ since the value in x_3 is bigger than $\varphi(x_2)$.

Obs: We have to minimize a function we know nothing about, except for its value in a point where we compute it. We would like to reduce to the interval which has as extremes the smallest point.

How can we choose these points? The idea is to choose the points that imply that the interval shrinks as fast as possible.

Obs: We have no guarantee that the interval that we are discarding doesn't contain a very deep minimum.

Elegant solution via golden ratio:

$$r = (\sqrt{5} - 1)/2 \ (\approx 0.618), \ r : 1 = (1 - r) : r$$

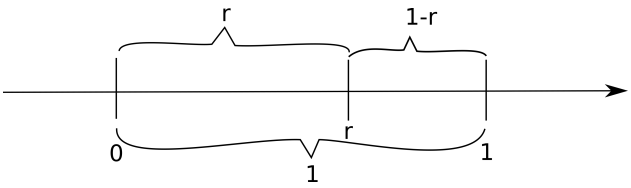


FIGURE 10.2: The relationship between r and $1 - r$ is $r \ : \ 1 = (1 - r) \ : \ r$.

ALGORITHM 10.2 Pseudocode for non differentiable functions for local minimum detection.

procedure LSGRM$(\varphi, \alpha, \varepsilon)$
  $\alpha_{i-} \leftarrow 0$; $\alpha_+ \leftarrow \alpha$;
  $\alpha'_- \leftarrow (1-r)\,\alpha$;
  $\alpha'_+ = r\,\alpha$;
  while $(\alpha_+ - \alpha_- \leq \varepsilon)$ **do** // note: not the same ε
    if $\varphi(\alpha'_-) > \varphi(\alpha'_+)$ **then**
      $\alpha_- \leftarrow \alpha'_-$;
      $\alpha'_- \leftarrow \alpha \leftarrow \alpha'_+$;
      $\alpha'_+ \leftarrow r(\alpha_+ - \alpha_-)$;
    else
      $\alpha_+ \leftarrow \alpha'_+$;
      $\alpha'_+ \leftarrow \alpha \leftarrow \alpha'_-$;
      $\alpha'_- \leftarrow (1-r)(\alpha_+ - \alpha_-)$;
    end if
  end while
end procedure

10.0.3 Inexact line search: Armijo-Wolfe

Key idea: Take your favourite line search (it also suggest a simpler one), and run it, but you don't have to wait for the derivative to become zero.

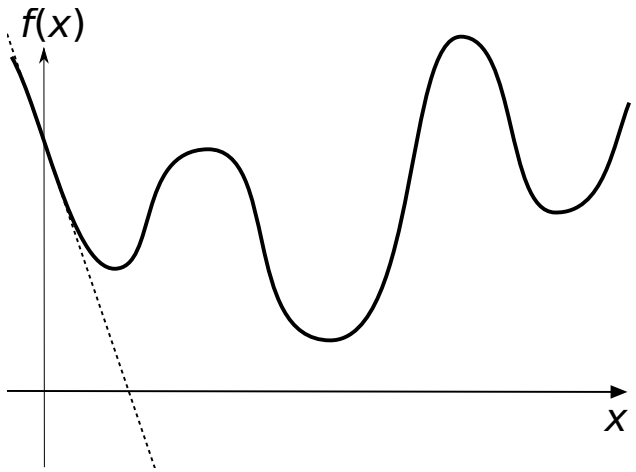


FIGURE 10.3: The dotted line represents the line which has the derivative as slope.

Definition 10.2 (Armijo condition)**.** *Let* $0 < m_1 < (\ll)1$

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

Problem of Armijo condition: small steps satisfy the armijo condition, but make convergence very slow.

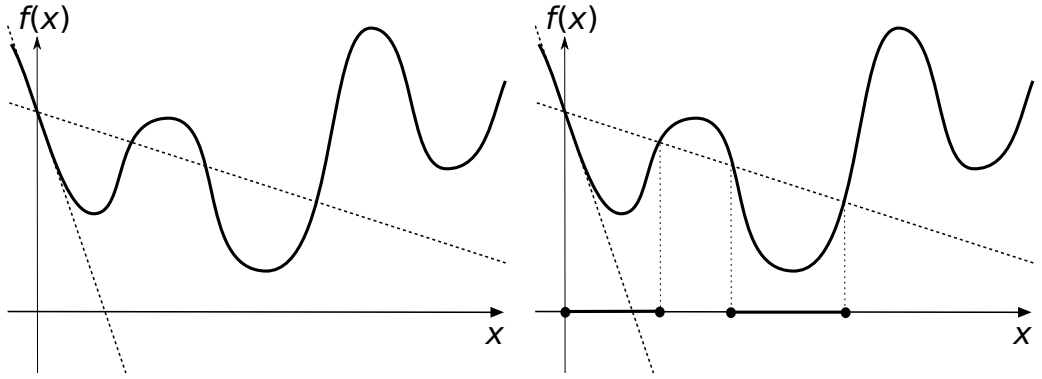


FIGURE 10.4: In the left picture Armijo condition chooses a new line which slope is still negative, but less steep than the original one. In the right one, the ranges where to search are highlighted.

We need another condition, in order to have a lower bound for the step size:

Definition 10.3 (Goldstein condition)**.** *Let* $m_1 < m_2 < 1$

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

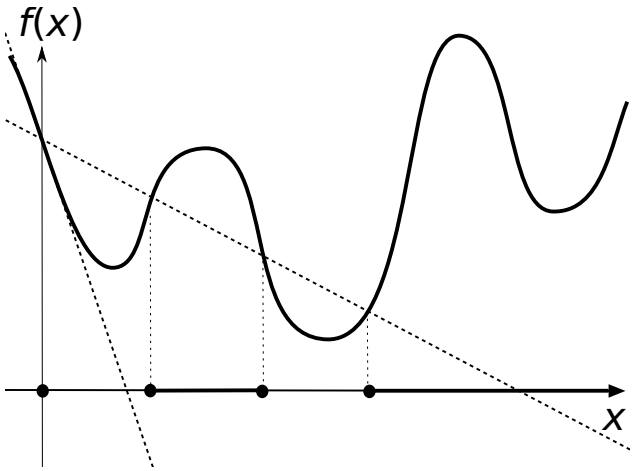


FIGURE 10.5: Goldstein condition chooses a new line which slope is still negative, less steep than the original one, but steeper than the one obtained by Armijo.

Problem: the point that satisfies both Goldstein and Armijo may not contain a local minimum.

To circumvent this problem another condition comes to help us.

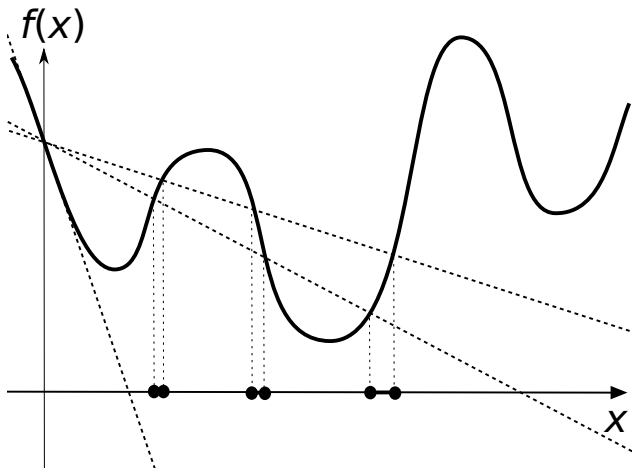


FIGURE 10.6: Here are the intervals that satisfy both Armijo and Goldstein conditions.

Definition 10.4 (Wolfe condition)**.** *Let* $m_1 < m_3 < 1$

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

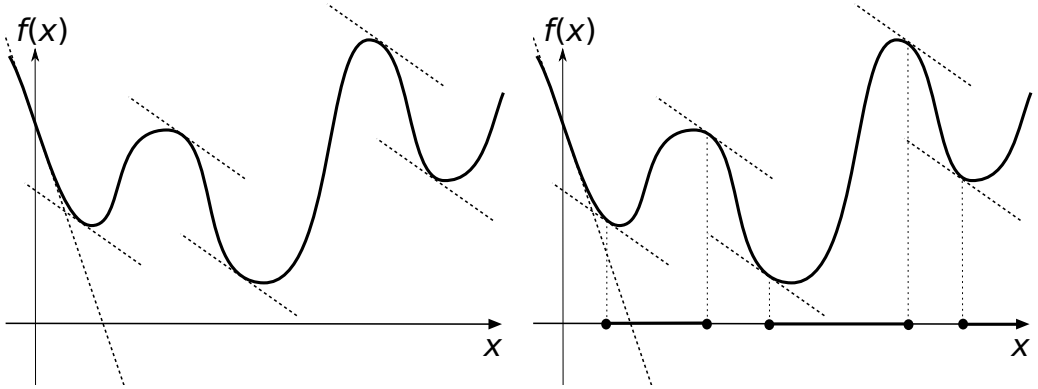


FIGURE 10.7: On the left Wolfe condition (which chooses derivatives that are substantially zero). On the right part the intervals selected by Wolfe.

Another issue of this conditions is that the derivative in the interval is quite big on the right side.

Definition 10.5 (Strong Wolfe condition)**.**

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

Fact 10.3. *If* $\varphi'(\alpha) \not\gg 0$ *and (A)* $\cap$ *(W) / (W') then all local minima (and maxima) are captured unless* m_1 *too close to* 1 *(that's why usually* $m_1 \approx 0.0001$*).*

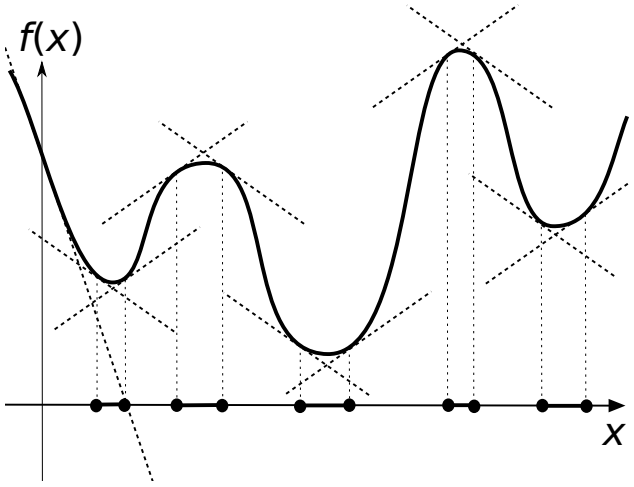


FIGURE 10.8: Strong Wolfe condition.

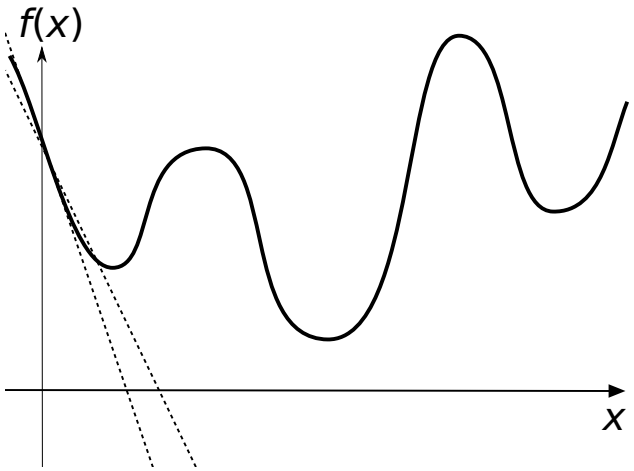


FIGURE 10.9: If the line is too close to the original one only a few points will satisfy Armijo condition which means that the intersection between Armijo and Wolfe will almost be empty.

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 10.4. *Let $\varphi \in C^1$ and $\varphi(\alpha)$ bounded below for $\alpha \geq 0$ then $\exists\ \alpha$ s.t. (A) $\cap$ (W') holds.*

Proof.
$l(\alpha) = \varphi(0) + m_1\alpha\varphi'(0)$, $d(\alpha) = l(\alpha) - \varphi(\alpha) \Longrightarrow$
$d(0) = 0$, $d'(0) = (m_1 - 1)\varphi'(0) > 0$ $(m_1 < 1)$ □

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 $[0, \bar{\alpha}]$.

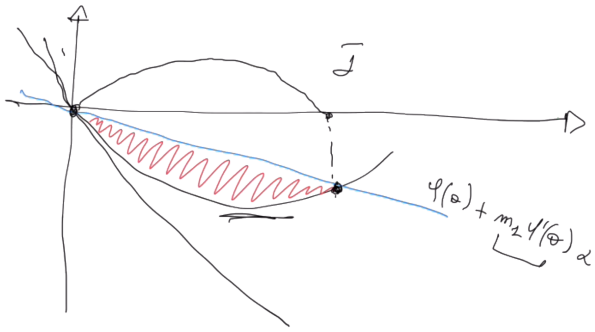


FIGURE 10.10: If the function isn't going to $-\infty$ the blue line and the function will meet again and we denote the value along the x axis $\bar{\alpha}$.

$$d(\alpha) = \varphi(0) + \alpha(m_1\varphi'(0)) - \varphi(\alpha)$$

$$d'(\alpha) = m_1\varphi'(0) + \varphi'(\alpha)$$

So $d'(\alpha^*)$ iif $\varphi'(\alpha^*) = m_1\varphi'(0)$. Then strong Wolfe requests that $|\varphi'(\alpha^*)| \leq m_1|\varphi'(0)|$.

How can we find such a point?

ALGORITHM 10.3 Pseudocode for backtracking line search.

```
1: procedure BLS(φ, φ', α, m_1, τ)
2:     while (φ(α) > φ(0) + m_1 α φ'(0)) do
3:         α ← τ α; τ < 1;
4:     end while
5: end procedure
```

- fundamental assumption: ∇f Lipschitz $\implies$ φ' Lipschitz and L does not depend on x^i (**check**)
- recall: $\exists\bar{\alpha}$ s.t. (A) holds $\forall\alpha \in]0, \bar{\alpha}]$ and $\varphi'(\bar{\alpha}) > m_1\varphi'(0) > \varphi'(0)$;
- φ' Lipschitz $\implies$ $\bar{\alpha}$ is "large" if $\|\nabla f(x^i)\|$ is:

$$L(\bar{\alpha} - 0) \geq \varphi'(\bar{\alpha}) - \varphi'(0) > (1 - m_1)(-\varphi'(0)) \implies \bar{\alpha} > (1 - m_1)\frac{\|\nabla f(x^i)\|\,|}{L}$$

  (recall $-\varphi'(0) = \|\nabla f(x^i)\|$);
- fundamental trick: $\bar{\alpha}$ can $\searrow 0$, but only as fast as $\|\nabla f(x^i)\|$ does;
- enough to prove that $\alpha^i \geq \bar{\alpha}$, or "not too smaller".

Now we can prove the following

Theorem 10.5. *If (A) ∩ (W) holds ∀i then either* $\{f(x^i)\} \to -\infty$ *or* $\{\|\nabla f(x^i)\|\} \to 0$.

Proof. **By contraddiction**, we assume $-\varphi'(0) = \|\nabla f(x^i)\| \geq \varepsilon > 0 \forall i$. Then:

1. (W) $\Longrightarrow \alpha^i \geq \bar{\alpha} > (1 - m_1)\frac{\|\nabla f(x^i)\|}{L} \Longrightarrow \alpha^i \geq \delta > 0$;
2. (A) $\Longrightarrow f(x^{i+1}) \leq f(x^i) - m_1 \alpha^i \|\nabla f(x^i)\| \leq f(x^i) - m_1 \delta \varepsilon$;
3. so $\{f(x^i)\} \to -\infty$ (or $\{\|\nabla f(x^i)\|\} \to 0$).

□

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

11 8th of November 2018 — A. Frangioni

11.1 Good practice of designing a project

In this lecture we focused on how to implement the gradient method inn Matlab and some good hints were underlined:

- 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 altough he might not know what a good starting point could be;
- there might be some parameters with the same sematic of hyperparameters (using machine learning terminology), 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 function calls we need to count also the calls to functions inside a single step;
- it's important to check the conditions that should be satisfied by the input data for the theortical 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** (click here).

12 14th of November 2018 — A. Frangioni

Mantra

If you want better convergence, use a better model.

So far we chose the direction for the step as $d^i = -\nabla f(x^i)\,\|\nabla f(x^i)\|$, in particular this is the direction where the decrease of the function is maximum.

We introduced 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, $\varphi'(0) = \frac{\partial f}{\partial d^i}(x^i) =< d^i, \nabla f(x^i) >= - < \nabla f(x^i), \nabla f(x^i) >= - \|\nabla f(x^i)\|$, which implies that $\|\nabla f(x)\| \to 0$ when $\varphi'(0) \to 0$.

Can we take another direction which isn't the opposite of the gradient and have that the same argument holds? Yes, for example if we choose 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 fo the gradient. It's trivial to observe that there are infinite angles that could be chosen, so we have a lot of flexibility.

Note that the angle between d^i and the gradient shouldn't be too close to 90°, otherwise the cosine would get approximately 0.

Theorem 12.1 (Zoutendijk)**.** *Let $f \in C^1$, ∇f Lipschitz and f bounded below.*
If (A) ∩ (W') then $\sum_{i=1}^{\infty} \cos^2(\theta^i)\|\nabla f(x^i)\|^2 < \infty$.

If we have a positive infinite sequence and we have that the corresponding series converges to a number, than the limit of the sequence is going to 0 reasonably fast.

If we choose an angle which is bounded below then the norm of the gradient has to converge very fast. More formally,

Fact 12.2. $\cos(\theta^i) \geq \varepsilon > 0 \Longrightarrow \|\nabla f(x^i)\| \to 0$

Proof. From the observation above, we may recall that the n-th term of the Zoutendijk sum should be approximately 0 $\forall n > \tilde{n}$, which means that one between $\cos(\theta^i)$ and $\|\nabla f(x^i)\|^2$ should be zero.

The proof is obtained from the fact that it can't be the cosine, because it's bounded below. □

12.1 Taylor method

At this point we assume that the function is perfectly convex (Hessian strictly positive definite).

Theorem 12.3. *Let f be a function such that $\nabla^2 f(x^i) \succ 0$. Then the second order model $Q_{x^i}(y)$ admits a minimum.*

Proof.
For simplicity of notation let us forget about the index ^{i} of x^i. The Taylor expansion is the following:

$$\begin{aligned} Q_x(y) &= f(y) + < \nabla f(x), y - x > + \frac{1}{2}(y-x)^T \nabla^2 f(x)(y-x) \\ &= f(x) + < \nabla f(x), y > - < \nabla f(x), x > + \\ &+ \frac{1}{2}\left(y^T \nabla^2 f(x) y - 2x^T \nabla^2 f(x) y + x^T \nabla^2 f(x) y \right) \\ &\overset{*}{=} < \nabla f(x) + x^T \nabla^2 f(x), y > + \frac{1}{2} y^T \nabla^2 f(x) y \end{aligned} \tag{12.1}$$

Where $\overset{*}{=}$ is given by the fact that there are some contant terms.
$qy + \frac{1}{2} y^T Q y \Leftrightarrow q + Qy = 0$ so

$$\nabla f(x) - x^T \nabla^2 f(x) + \nabla^2 f(x) y = 0 \nabla^2 f(x) y = \nabla f^2(x) x - \nabla f(x) y = x - [\nabla^2 f(x)]^{-1} \nabla f(x)$$

□

Scrivere meglio la dim

Corollary 12.4. *Newton's direction is* $d^i \leftarrow -[\nabla^2 f(x^i)]^{-1} \nabla f(x^i)$ *(just* $\mathbb{R}^n$ *version).*

The direction d is obtained taking the opposite of the inverse of the Hessian times the gradient of the function

Observation 12.1. *We need the Hessian to be invertible, which isn't true in general.*

We need to solve a non linear equation, namely a system of equations. 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 $\nabla f(x) \approx \nabla f(x^i) + \nabla^2 f(x^i)(x - x^i)$.

Theorem 12.5. *Let* f *be a function s.t.* $f \in C^3$, $\nabla f(x_*) = 0$ *and* $\nabla^2 f(x_*) \succ 0$. *Then* $\exists \mathcal{B}(x_*, r)$ *s.t.* $x^1 \in \mathcal{B}$, *which implies* $\{x^i\} \to x_*$ *quadratically.*

We may observe that the direction of the Newton method is good not only when we are close to the minimum, but also when we are far.

The scalar product should be negative and it is so, but we also need to ensure that the scalar product isn't too close to 0. When is it that this condition is satisfied? When the function is "reasonable".

Let us now introduce what we mean by "reasonable": a function f 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 don't get too close to zero (numerically speaking) .

Theorem 12.6. *Let* f *be a function that satisfies* $uI \preceq \nabla^2 f \preceq LI$ *and* $\cos(\theta^i) \leq -\lambda^n/\lambda^1 \leq -u/L$. *Then the method converges.*

Proof.

STEP 1 From the definition of d^i we have that:

$$d^i = -[\nabla^2 f(x^i)]^{-1} \nabla f(x^i)$$

or, equivalently:

$$\nabla^2 f(x^i) d^i = -\nabla f(x^i)$$

which implies:

$$d^i \nabla f(x^i) = -(d^i)^T \nabla^2 f(x^i) d^i \leq -\lambda^n \left\| d^i \right\|^2$$

STEP 2 Since we have that $\cos(\theta^i) = d^i \nabla f(x^i)(\|x^i\| \, \|\nabla f(x^i)\|) \leq \delta < 0$, we want to bound the norm of the gradient:

$$\left\| \nabla f(x^i) \right\| = \left\| \nabla^2 f(x^i) d^i \right\| \leq \left\| \nabla^2 f(x^i) \right\| \left\| d^i \right\| = \lambda^1 \left\| d^i \right\|$$

which implies:

$$\cos(\theta^i) \leq -\lambda^n / \lambda^1 \leq -u/L$$

□

Theorem 12.7. *Let f be a function that satisfies $uI \preceq \nabla^2 f \preceq LI$ and $\cos(\theta^i) \leq -\lambda^n/\lambda^1 \leq -u/L$. Then the method not only converges, but we also have that for some iteration onwards, $\alpha^i = 1$ always satisfy (A).*

Proof.

$$\begin{aligned} f(x^i + d^i) &= f(x^i) + d^i \nabla f(x^i) + \frac{1}{2}(d^i)[\nabla^2 f(x^i)] d^i + R(\left\| d^i \right\|) \\ &= f(x^i) - \nabla f(x^i)^T [\nabla^2 f(x^i)]^{-1} \nabla f(x^i) + \\ &+ \frac{1}{2} \nabla f(x^i)^T [\nabla^2 f(x^i)]^{-1} \nabla f(x^i) + R(\left\| d^i \right\|) \\ &= f(x^i) - \frac{1}{2} \nabla f(x^i)^T [\nabla^2 f(x^i)]^{-1} \nabla f(x^i) + R(\left\| d^i \right\|) \\ &= f(x^i) + \frac{1}{2} < \nabla f(x^i), d^i > + R(\left\| d^i \right\|) \end{aligned} \tag{12.2}$$

□

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).

12.1.1 Interpretation of Newton method

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

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

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 ($x = R^{-1}y$) to the direction we get the direction in 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?

12.1.2 Non convex case

If the Hessian isn't positive definite we may take something which is close to the Hessian, which has the bounding property we used before (eigenvalues bounded below and above). In truth, there is no reason to choose exactly the opposite of the inverse of the Hessian times the gradient for the direction, we may use another matrix which is striclty positive definite ad has more or less th same properties of the Hessian.

In particular, given an Hessian that has some negative eigenvalues we can sum to it a multiple of the identity matrix: $H^i = \nabla^2 f(x^i) + \varepsilon^i I \succ 0$. We iterate this procedure until the resulting matrix is strictly positive definite.

How to compute ε numerically? How much larger should ε be? We also want the smallest eigenvalue to be not too close to 0.

$\varepsilon = \max\{0,\ \delta - \lambda^n\}$ for "appropriately chosen smallish δ", in other words we want all the eigenvalues to be at least δ.

The reasons behind the choice of δ (not too small) are bothnumerical (any double $\leq$ `1e-16` "is 0") and algorithmical (if $\lambda^n(\nabla^2 f(x^i) + \varepsilon I)$ "very small" then the axes of $S(Q_{x^i}, \cdot)$ are "very elongated").

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

Observation 12.2. *Note that these contraints 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 norm, changing a bit the algorithm, but we still get convergence. We would need

to solve $\min\{\|H - \nabla^2 f(x^i)\|_F \; H \succeq \delta I\}$, which is performed in two steps:

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

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

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 may notice that Newton method is very fast to go to a local minimum and this may represent a problem, because it misses global minima.

13 16th of November 2018 — A. Frangioni

What happens when the Hessian isn't strictly positive definite? If there are some negative eigenvaues, 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 13.1 (Karush-Khun-Tucker conditions)**.** *Any optimal solution of the problem* x^{i+1} *must satisfy that* $\equiv \exists\, \lambda \geq 0$ *s.t.:*

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

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

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

Where the last condition means that λ *has to be* 0*, unless the solution we get is exaclty 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:

- $\|x^{i+1}\| < r \implies \lambda = 0 \implies$ normal Newton step ($\mathcal{T}$ has no effect);
- $\lambda > 0$(= 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: $m^i(x) = \nabla f(x^i)(x - x^i) + \frac{1}{2}(x - x^i)^T H^i(x - x^i)$, $x^{i+1} = x^i + \alpha^i d^i$.

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

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: $\nabla m^{i+1}(x^i) = \nabla f(x^i)$;

3. $\|H^{i+1} - H^i\|$ is "small".

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

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

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

In order to have a matrix H^i that satifies 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. *Wolf condition implies $s^i y^i > 0$, using the notation we inroduced: (W) $\Longrightarrow$ (C).*

Proof.

$$\varphi'(\alpha^i) = \nabla f(x^{i+1}) d^i \geq m_3 \varphi'(0) = m_3 \nabla f(x^i) d^i$$

$$\Downarrow$$

$$(\nabla f(x^{i+1}) - \nabla f(x^i)) d^i \geq (m_3 - 1)\varphi'(0) > 0$$

□

Observation 13.1. *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 $H^{i+1} = argmin\{\|H - H^i\| \ : \ H \in (\mathrm{S}), \ H \succeq 0\}$ is a good idea and for this minimum problem holds the following:

Theorem 13.2 (Davidson-Feltcher-Powell)**.** *The new matrix is obtained at each step constructing a rank two matrix, obtained from H^i as a rank to 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 fomulas that give us a way to compute $(H^{i+1})^{-1}$ from $H^{i^{-1}}$.

Theorem 13.3 (Sherman-Morrison-Woodbury)**.** *The inverse of a matrix of the form $A + ab^T$ has the following shape: $(A + ab^T)^{-1} = \frac{A^{-1} - A^{-1} a b^T A^{-1}}{1 - b^T A^{-1} a}$.*

Observation 13.2. *From Theorem 13.3 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}: $s^i = B^{i+1}y^i \implies B^{i+1} = \text{argmin}\ \{\|B - B^i\| \ : \ \dots\ \}$.

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

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

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

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

Observation 13.3. *How can we choose H^1? The value we choose will make a differencein 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^is: 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 gradint method, the angle between two consecutive directions is exactly 90°, as can be seen in Figure 13.1.

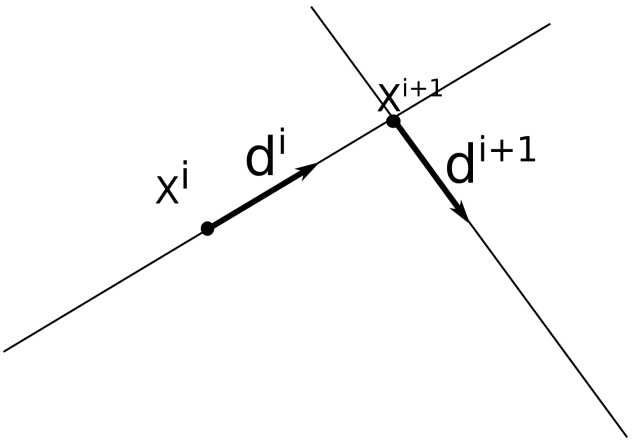


FIGURE 13.1: Geometric idea on how the new direction is chosen.

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

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 13.1 Pseudocode for conjugate gradient method for quadratic functions.

```
1: procedure CGQ(Q, q, x, ε)
2:     d⁻ ← 0;
3:     while (‖∇f(x)‖ > ε) do
4:         if (d⁻ = 0) then
5:             d ← −∇f(x);
6:         else
7:             β = (∇f(x)^T Q d⁻)/((d⁻)^T Q d⁻);
8:             d ← −∇f(x) + β d⁻;
9:         end if
10:        α ← (∇f(x)^T d)/(d^T Q d);
11:        x ← x + α d;
12:        d⁻ ← 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 fomula for β.

The pseudocode of Algorithm 13.2 is referred to Fletcher-Reeves definition of $\beta^i = \|\nabla f(x^i)\| / \|\nabla f(x^{i-1})\|^2$.

This algorithm converges in at most n iterations.

ALGORITHM 13.2 Pseudocode for conjugate gradient method for arbitrary functions

---

procedure CGA(Q, q, x, ε)
  $\nabla f^- = 0$;
  while $(\|\nabla f(x)\| > \varepsilon)$ **do**
    if $(\nabla f^- = 0)$ **then**
      $d \leftarrow -\nabla f(x)$;
    else
      $\beta = \|\nabla f(x^i)\|^2 / \|\nabla f^-\|^2$;
      $d \leftarrow -\nabla f(x) + \beta d^-$;
    end if
    $\alpha \leftarrow \text{AWLS}(f(x + \alpha d))$;
    $x \leftarrow x + \alpha d$;
    $d^- \leftarrow d$;
    $\nabla f^- \leftarrow \nabla f(x)$;
  end while
end procedure

We have three different formulas for β^i, which coincide in the quadratic case.

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

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') $\not\Longrightarrow d^i$ of Polak-Ribière is of descent, unless $\beta^i_{PR} = \max\{\beta^i, 0\}$.

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.

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 prevous 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 metaphore: 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 $\boldsymbol{\beta^i}$ is called **momentum**, $\mathbf{x^i}$ **heavy** and $\nabla \mathbf{f}(\mathbf{x^i})$ **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 we smaller than the prevous 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:

$$\alpha = \frac{4}{\left(\sqrt{\lambda^1} + \sqrt{\lambda^n}\right)^2}, \qquad \beta = \max\left\{\left|1 - \sqrt{\alpha\lambda^n}\right|, \left|1 - \sqrt{\alpha\lambda^1}\right|\right\}^2$$

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:

$$\left\|\{x^{i+1} - x_*\}\right\| \leq \left(\frac{\sqrt{\lambda^1} - \sqrt{\lambda^n}}{\sqrt{\lambda^1} + \sqrt{\lambda^n}}\right) \left\|x^i - x_*\right\|$$

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 alogirthm several times.

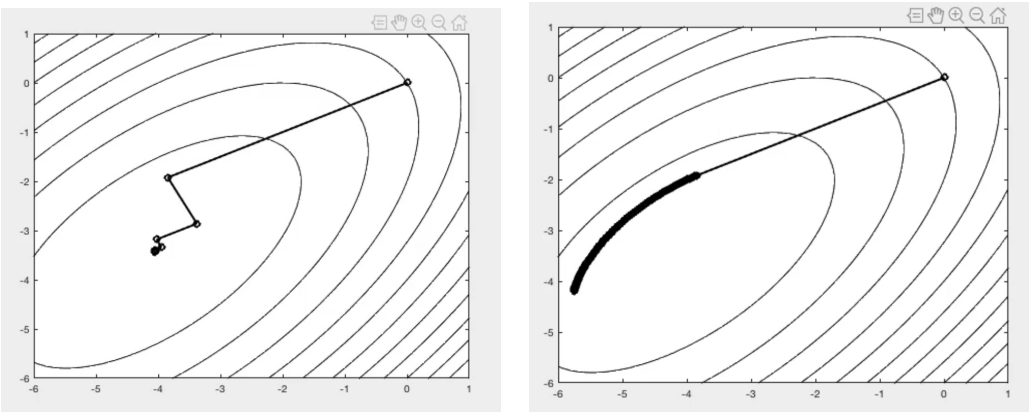


FIGURE 14.1: On the left side Newton method and on the right side the heavy ball method.

The plot of the convergenc 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 Pseudocode for accelerated gradient method.

procedure ACCG$(f, \nabla f, x, \varepsilon)$
  $x_- \leftarrow x$;
  $\gamma \leftarrow 1$;
  repeat
    $\gamma_- \leftarrow \gamma$;
    $\gamma \leftarrow (\sqrt{4\gamma^2 + \gamma^4} - \gamma^2)/2$;
    $\beta \leftarrow \gamma(1/\gamma_- - 1)$;
    $y \leftarrow x + \beta(x - x_-)$;
    $g \leftarrow \nabla f(y)$;
    $x_- \leftarrow x$;
    $x \leftarrow y - (1/L)g$;
  until $(\|g\| > \varepsilon)$
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.

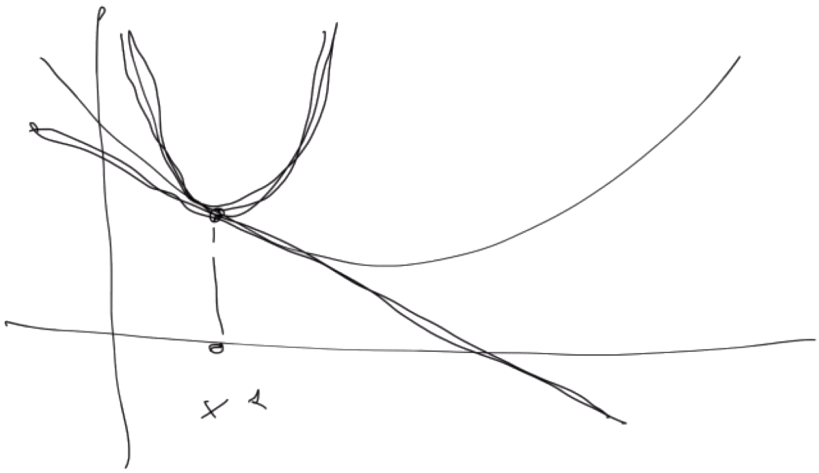


FIGURE 14.2: We build a linear model, which is a lowerbound for the function, then we build a quadratic model, which is above our function.

If we choose γ optimally then the quadratic approximation is very close to the function. We want to find the value for γs 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 (Optimality of accelerated gradient)**.** *If the function isn't strongly convex no algorithm has better convergence than* $|f(x^i) - f_*| = 3L\frac{\left\|x^1 - x_*\right\|^2}{32(i+1)^2}$.

Observation 14.1. *This theorem tells us that this algorithm never gets worse than* $|f(x^i) - f_*| = 3L\frac{\left\|x^1 - x_*\right\|^2}{32(i+1)^2}$*, 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 hypotesis) 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 $\min \{ \sum_{i \in I} l(y^i, < \Phi(X^i), w >) \ : \ w \in \mathbb{R}^n \}$, 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 aplication 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:

$$\min \{ \sum_{i \in I} l(y^i, < \Phi(X^i), w >) + \mu \Omega(w) : w \in \mathbb{R}^n \}$$

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: $\Omega(w) = \|w\|^{2^2}$;
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 $f(w_1, w_2) = (3w_1 + 2w_2 - 2)^2 + 10(|w_1| + |w_2|)$.

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:

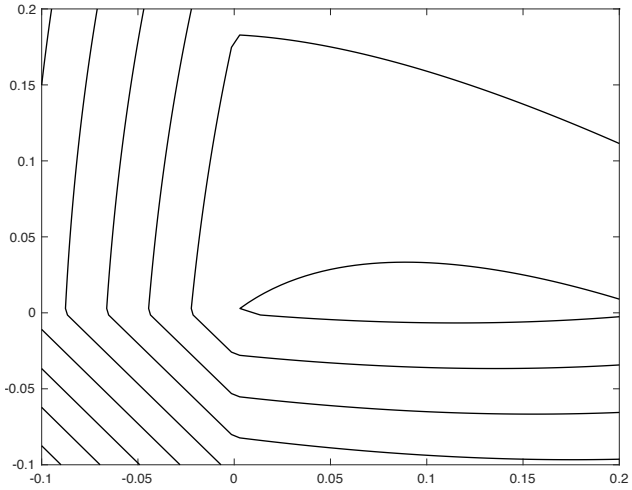


FIGURE 14.3: This function has a lot of kinky points.

Property 14.2. *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, $< x^* - x^i, g >$ 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.

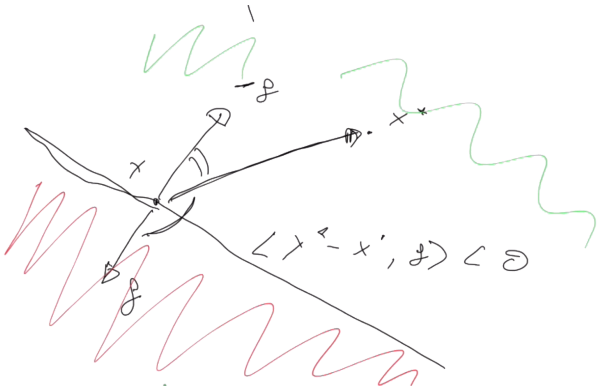


FIGURE 14.4: Since we know that $< g, x^* >$ is negative we get that the angle between g and x^* is larger than 90°. Moreover, we know where the optimum is not (red region), hence we restrict to the green region. At this point we will perform a line search on g direction. Since the function is not smooth, 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^*.

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 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.

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 14.2 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, $f(x_*) \geq f(x) + < g, x_* - x >$, hence $< g, x_* - x > \leq f(x_*) - f(x) \leq 0$.

We want to bound the distance between the point at the "next step" and the optimum:

$$\begin{aligned} \left\|x^{i+1} - x_*\right\|^2 &\overset{(1)}{=} \left\|x^i - \alpha^i d^i - x_*\right\|^2 \\ &= \left\|x^i - x_*\right\|^2 + 2\alpha^i g^i \frac{x_* - x^i}{\|g^i\|} + (\alpha^i)^2 \\ &\overset{(2)}{\leq} \left\|x^i - x_*\right\|^2 - 2\alpha^i \frac{f(x^i) - f(x_*)}{\|g^i\|} + (\alpha^i)^2 \end{aligned} \tag{15.1}$$

Where $\overset{(1)}{=}$ follows from the definition of a normalized step: $x^{i+1} = \frac{x^i - \alpha^i g^i}{\|g^i\|}$ and $\overset{(2)}{\leq}$ follows from the inequality we stated above.

Observation 15.1. *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, $-2\frac{f(x^i) - f(x_*)}{\|g^i\|} < 0 + \alpha^i \searrow$, hence $(\alpha^i)^2 \searrow 0_+$.

An attentive reader may notice that from Equation (15.1) *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 (Diminishing-Square Summable)**.** *We term a series that diverges, while the series of the squares of the terms diverges, as* ***diminisching-square summable****.*

Formally:

$$\sum_{i=1}^{\infty} \alpha^i = \infty \ \wedge \ \sum_{i=1}^{\infty} (\alpha^i)^2 < \infty \ (DSS)$$

Assumptions: function convex, definite in all its domain, hence the norm of the gradient will never become very large.

Fact 15.1. *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:

$$\left\|x^{i+1} - x_*\right\|^2 \leq \left\|x^1 - x_*\right\|^2 - \delta \sum_{k=1}^{i} \alpha^k + \sum_{k=1}^{i} \left(\alpha^k\right)^2$$

The contraddiction is due to the fact that as $i \to \infty$ the right-hand side goes to $-\infty$. □

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 15.2 (Polyak stepsize)**.** *Let f be our convex objective function and let x_* be the optimum for f. We term* ***Polyak stepsize****:*

$$\alpha^i = \beta^i \frac{f(x^i) - f(x_*)}{\|g^i\|} \ \textit{(PSS)}$$

where $\beta \in (0, 2)$.

If we pick a Polyak stepsize, then $\|x^{i+1} - x_*\|^2 < \|x^i - x_*\|^2$, so $\{x^i\} \to x_*$. The best value for β_i is 1. In this case, we obtain what follows by sybstituting $\beta_i = 1$ in Equation (15.1)

$$\frac{\left(f(x^i) - f(x_*)\right)^2}{\|g^i\|^2} \leq \left\|x^i - x_*\right\|^2 - \left\|x^{i+1} - x_*\right\|^2$$

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 vaue ($\underline{f}^i = \min\{f(x^h) \ : \ h = 1, \ldots, i\}$)

$$\frac{\left(\underline{f}^i - f(x_*)\right)^2}{L^2} \leq \frac{\left(f^i - f(x_*)\right)^2}{\|g_i\|^2} \leq \left\|x^i - x_*\right\|^2 - \left\|x^{i+1} - x_*\right\|^2$$

Summing for $i = 1, \ldots, k$ we obtain a telescopic series:

$$\left\|x^1 - x_*\right\| - \cancel{\left\|x^2 - x_*\right\|} + \cancel{\left\|x^2 - x_*\right\|} - \cancel{\left\|x^3 - x_*\right\|} + \cdots + \cancel{\left\|x^k - x_*\right\|} - \left\|x^{k+1} - x_*\right\|$$

Hence resulting in:

$$k\frac{\left(\underline{f}^k - f(x_*)\right)^2}{L^2} \leq \left\|x^1 - x_*\right\|^2 - \left\|x^{k+1} - x_*\right\|^2 \leq \left\|x^1 - x_*\right\|^2 = R$$

which is equivalent to $\left(\underline{f}^k - f(x^*)\right)^2 \leq \frac{R^2L^2}{k}$, 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 15.2. *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 $f_{rec} = \underline{f}$. 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 Pseudocode for target level stepsize.

```
1: procedure SGPTL(f, g, x, i_max, β, δ_0, R, ρ)
2:     r ← 0;
3:     δ ← δ_0;
4:     f_ref ← f_rec ← f(x);
5:     i ← 1;
6:     while (i < i_max) do
7:         g = g(x);
8:         α = β(f(x) − (f_ref − δ))/‖g‖^2;
9:         x ← x − αg;
10:        if (f(x) ≤ f_ref − δ/2) then
11:            f_ref ← f_rec;
12:            r ← 0;
13:        else
14:            if ( r > R ) then
15:                δ ← δρ;
16:                r ← 0;
17:            else
18:                r ← r + α‖ g ‖;
19:            end if
20:        end if
21:    end while
22:    f_rec ← min{ f_rec , f(x) };
23:    i ← 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: $d^i = \gamma^i g^i + (1 - \gamma^i) d^{i-1}$, $x^{i+1} = x^i - \alpha^i d^i$. We can prove that d^i approximates the subgradient. We can also prove that the algorithm converges in the end. The prameters of this algorithm are two: β (stepsize) and γ (deflection). In order to choose them we have two different approaches:

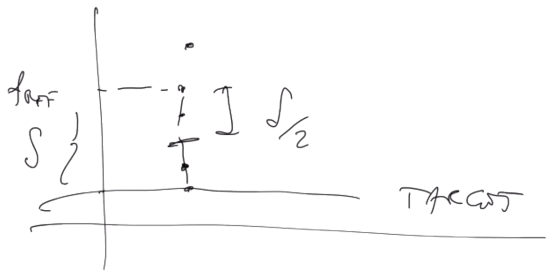


FIGURE 15.1: There are two cases: either the function value is significanlty below the reference (for example $\frac{\delta}{2}$ below the reference) or it's not . If we are in the "happy case" (we just move the reference to the best value). For what conserns the "unhappy case", if we are unhapy for 1 iteration, no problem. Two iterations? no prolem. Many iterations? Problem: after some iterations in which we are not improving it means that we have to decreas 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.

STEPSIZE-RESCRICTED $\equiv$ 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 $\nearrow$, stepsize has to $\searrow$";

DEFLECTION-RESTRICTED $\equiv$ stepsize-first. We first choose γ, then we pick a step size that depends on γ:

$$\text{(DSS)} \ \wedge \ \frac{\alpha^{i-1}\|d^{i-1}\|}{(f(x^i)-f_*)+\alpha^{i-1}}\left\|d^{i-1}\right\| \leq \gamma^i$$

"as $f(x^i) \to f_*$, 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 (Lagrangian function)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a function of the following shape:*

$$f(x) = \max\{x^T Az \ : \ z \in Z\}$$

where Z is convex and bounded.

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.

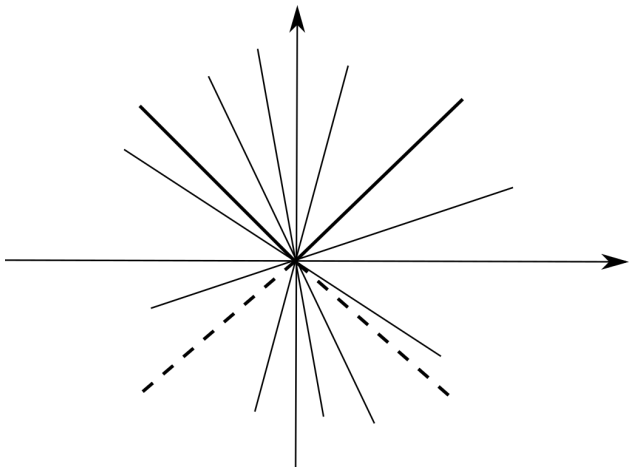


FIGURE 15.2: Let us take the absolute value function $f(x) = |x|$. This function would be differentiable, if it wasn't for some nasty points (in this case 0, were the optimization problem has many optimal solutions). In 0 there are many subgradients, so it has many optima.

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).

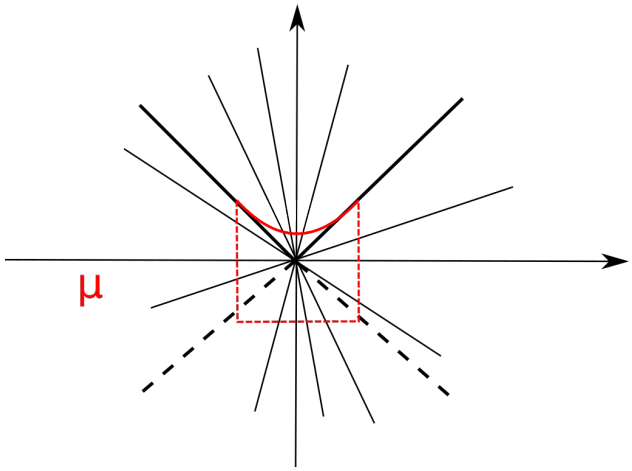


FIGURE 15.3: Geometric intuition of the usage of variable μ.

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 μ where 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 $f_\mu(x) \le f(x) \le f_\mu(x) + \mu R$, such that as $\mu \searrow 0$, "argmin $\{ f_\mu(x) \} \to x_*$".

The new function is not only convex, but it is also Lipschitz continuous: ∇f_μ Lipschitz with $L = \frac{\|A\|^2}{\mu}$, but it is "less and less Lipschitz" as $\mu \searrow 0$.

Fact 15.3. *If $f_* > -\infty$ and picking a very special value of μ ($\mu = \varepsilon/(2R)$), then an appropriate ACCG obtains $f(x^i) - f_* \le \varepsilon$ for $i \ge 4 \|A\| \|x_*\| \frac{\sqrt{R}}{\varepsilon}$.*

We observe that the convergence is much better, because it depends on $O(\frac{1}{\varepsilon})$ instead of $O(\frac{1}{\varepsilon^2})$.

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 th curvature.

The point is that the directional derivatives are defined and (if computed massively) give 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 ($\mathscr{B} = \{(x^i,\ f^i = f(x^i),\ g^i \in \partial f(x^i))\} \equiv$ 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:

$$f_{\mathscr{B}}(x) = \max\{f^i + g^i(x - x^i)\ :\ (x^i, f^i, g^i) \in \mathscr{B}\}$$

At this point we can apply Newton method: first we minimize the model and then use the minimum as next point. We collect information in that specific point and then we repeat.

Notice that the model is always below the objective function ($f_{\mathscr{B}}(x) \le f(x)\ \forall x$), hence $\min\{f_{\mathscr{B}}(x)\} \le f_*$, so $x^*_{\mathscr{B}} \in \text{argmin}\{f_{\mathscr{B}}(x)\} \approx\ x_*$.

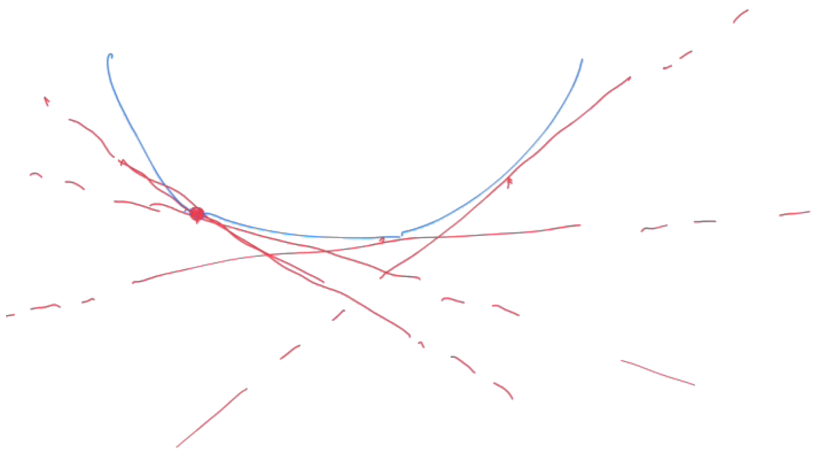


FIGURE 15.5: The idea is that we do not compute a subgradient and discard it. We store it, thus resulting in a model

This function has many kinky points, so how to minimize it? Dirty trick from Ricerca Operativa:

$$\min\{\, f_{\mathscr{B}}(x)\,\} = \min\{\, v \,:\, v \geq f^i + g^i(x - x^i), (x^i, f^i, g^i) \in \mathscr{B}\}$$

And on this problem, we can use the simplex method.

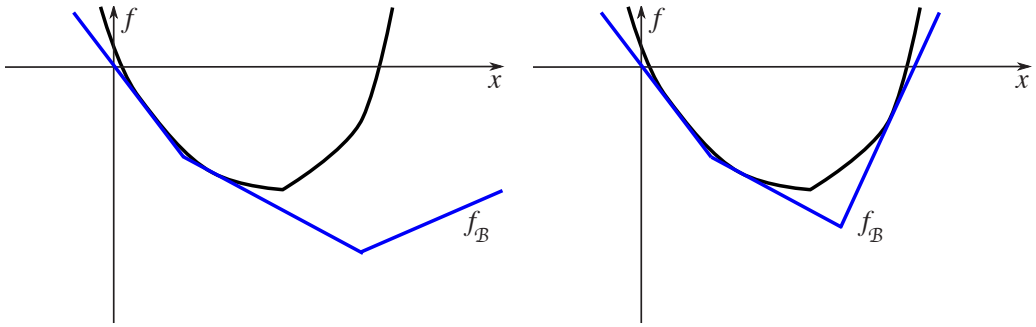


FIGURE 15.6: A geometric representation of how the model (blue line) changes after one iteration.

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 15.4 (Stabilized master problem)**.** *We term* ***stabilized master problem*** *the following:*

$$\min\{f_{\mathscr{B}}(x) + \frac{\mu\|x - \bar{x}\|^2}{2}\}$$

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 the the best encountered so far.

ALGORITHM 15.2 Pseudocode for boundle method.

```
1: procedure PBM(f, g, x̄, m_1, ε)
2:     choose μ;
3:     B ← {(x̄, f(x̄), g(x̄))};
4:     while ( true ) do
5:         x* ← argmin {f_B(x) + μ‖x − x̄‖²/2};
6:         if (μ‖ x* − x̄ ‖_2 ≤ ε) then
7:             break;
8:         end if
9:         if (f(x*) ≤ f(x̄) + m_1(f_B(x*) − f(x̄))) then
10:            x̄ ← x*;
11:            possibly decrease μ;
12:        else
13:            possibly increase μ;
14:        end if
15:        B ← B ∪ ( 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 contrained one, so from next lecture we will start dealing with constrained optimization problems.

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 (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

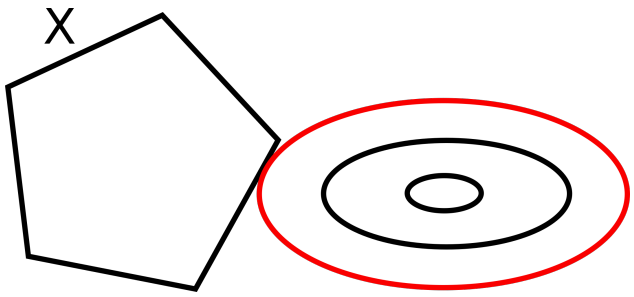


FIGURE 16.1: The red line is level set of the function corresponding to the smallest value that touches the set X. The point in the intersection is not a saddle point of the function f, altough it is the minimum.

There are two kinds of contraints:

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.

$\min\{f(x) \ : \ Ax = b\}$, where the rank of A counts the number of linearly independent rows.

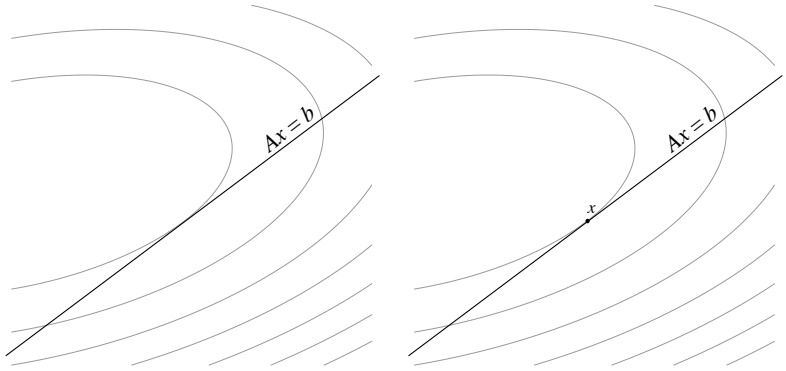


FIGURE 16.2: Linear constraint and a point on the boundary.

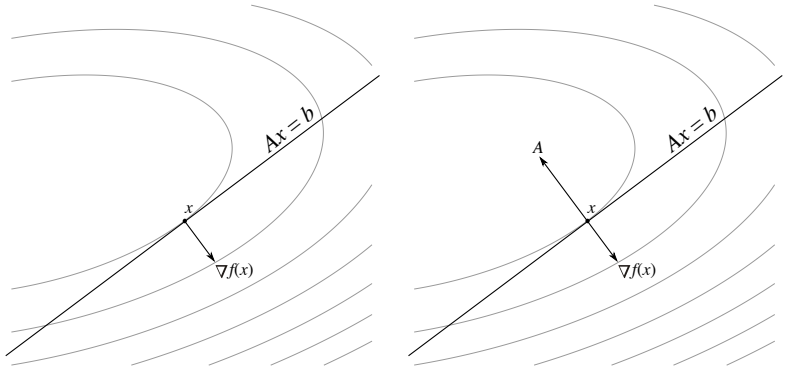


FIGURE 16.3: The gradient is orthogonal to the level set in that point, when the function is smooth. The same holds for matrix A.

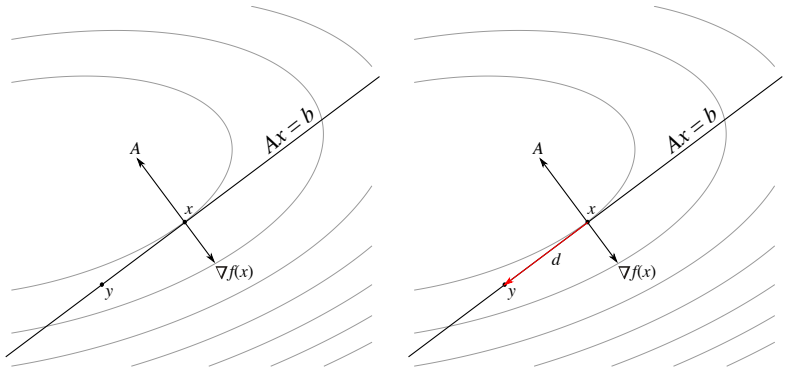


FIGURE 16.4: If we take any other point in the space it has to be orthogonal to A.

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.

$A = [A_B, A_N]$, $x = [x_B, x_N]$, 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 indepedent 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 $\nabla r(w^*) = 0$.

$D = \begin{bmatrix} -A_B^{-1}A_N \\ I \end{bmatrix}$ then $AD = [A_B, A_N] \begin{bmatrix} -A_B^{-1}A_N \\ I \end{bmatrix} = -A_B {A_B}^{-1} A_N + A_N = 0$.

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. *Let $Ax = b$ be a linear system and w such that $x = \left[{A_B}^{-1}(b - A_N w), w\right]$. 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 16.2 (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 16.2. *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 satisy the constraint. Formally, $Ax = b$ is our contstraint 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 contrained 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 contstraints

Definition 16.3 (Tangent cone)**.** *We call **tangent cone** of X at x $\mathbf{T_X(x)} = t$.*

$$\{d \in \mathbb{R}^n \ :\ \exists\ \{z_i \in X\} \to x \ \wedge\ \{t_i \geq 0\} \to 0 \text{ s.t. } d = \lim_{i\to\infty} \frac{z_i - x}{t_i}\}$$

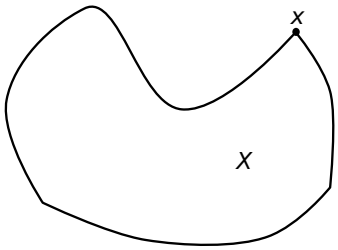


FIGURE 16.5: When the boundary has this shape we can move along directions that point inside the constraints. Tangent directions are not allowed.

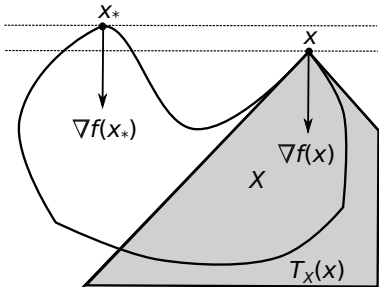


FIGURE 16.6: Geometric representation of tangent cone, which is the region of the space where we can pick the directions. The intuition is to zoom in x and the result of this zooming is a cone.

Theorem 16.3. *Let $\mathcal{C}$ be a cone, $\forall x \in \mathcal{C}\ \alpha x \in \mathcal{C},\ \forall \alpha > 0$.*

Theorem 16.4. *Given a function f, where x is a **local** optimum $< \nabla f(x), d >\geq 0\ \forall d \in T_X(x)$.*

Proof. **Proof by contraddiction:** Assume $\exists d \in T_X(x)$ such that $< \nabla f(x), d >< 0$, but x is a local optimum.

By definition $\exists X \supset \{z_i\} \to x$ and $\{t_i\} \to 0$ such that $d \overset{*}{=} \lim_{i\to\infty} \frac{z_i - x}{t_i}$.

First order Taylor $f(z_i) - f(x) =< \nabla f(x), (z_i - x) > + R(z_i - x)$.

$$\begin{aligned}
\lim_{i\to\infty} \frac{f(z_i - x)}{t_i} &= \lim_{i\to\infty} < \nabla f(x), \frac{z_i - x}{t_i} > + \frac{R(z_i - x)}{t_i} \\
&\overset{*}{=} < \nabla f(x), d > + \lim_{i\to\infty} \frac{R(z_i - x)}{t_i} \\
&\overset{(1)}{=} < \nabla f(x), d > \\
&< 0
\end{aligned} \tag{16.1}$$

Where, $\overset{(1)}{=}$ follows from $\lim_{i\to\infty} \frac{R(z_i - x)}{t_i} = 0$ by Taylor. □

Observation 16.1. *The optimum of Theorem 16.4 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.*

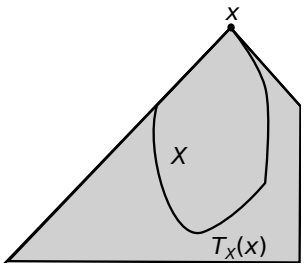


FIGURE 16.7: Convex function.

Observation 16.2. *Notice that the rule $< \nabla f(x), d >\geq 0 \ \forall d \in T_X(x) \Rightarrow x$ 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.*

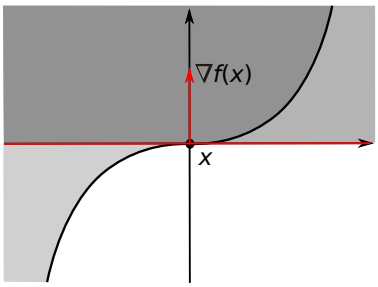


FIGURE 16.8: Let us suppose we pick the direction towards the left part of the function in the saddle point. That direction is promising and actually the value of the function decreases. In this case the problem is that the contraint is not convex.

Now we need a more manageable object for $T_X(x)$.

Definition 16.4 (Cone of feasible directions)**.** *Intuitively, we are in x and we want to find al 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 $F_X(x) = \{d \in \mathbb{R}^n : \exists \bar{\varepsilon} > 0 \ s.t. \ x + \varepsilon d \in X, \ \forall \varepsilon \in [0, \bar{\varepsilon}]\}$.*

Fact 16.5. *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. *$cl(F_X) \subseteq T_X$, where $cl(F_X)$ is the closure of the cone of feasible directions;*

3. if X convex then the cones coincide: T_X and F_X convex and 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 $G = [g_i(x)]_{i\in\mathcal{I}} : \mathbb{R}^n \to \mathbb{R}^{|\mathcal{I}|}$ and $H = [h_i(x)]_{i\in\mathcal{J}} : \mathbb{R}^n \to \mathbb{R}^{|\mathcal{J}|}$;

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)

$$X = \{x \in \mathbb{R}^n \,:\, g(x) = \max\{\, g_i(x) \,:\, i \in \mathcal{I}\,\} \leq 0 \; i \in \mathcal{I},\; h_j(x) = 0 \;\in \mathcal{J}\}$$

Definition 16.5 (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 $G_{\mathcal{B}} = [g_i(x)]_{i\in\mathcal{B}} : \mathbb{R}^n \to \mathbb{R}^{|\mathcal{B}|}$ the corresponding set of inequalities.

Definition 16.6 (First-order feasible direction cone)**.** *We term **First-order feasible direction cone at** $x \in X$:*

$$D_X(x) = \{d \in \mathbb{R}^n \;: < \nabla g_i(x), d > \leq 0 \; i \in \mathcal{A}(x) \;\; < \nabla h_j(x), d > = 0 \; j \in \mathcal{J}\} = \{d \in \mathbb{R}^n : (JG_{\mathcal{A}(x)}(x))d \leq 0,$$

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 contraints.

A visual example of a first order feasible direction cone is displayed in Figure 16.9.

Fact 16.6. *The tangent cone is a subset of the first-order feasible direction cone. Formally, $T_X(x) \subseteq D_X(x)$.*

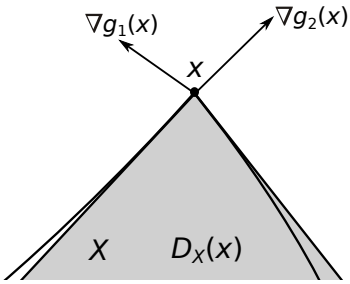


FIGURE 16.9: The first-order feasible direction cone is made of those directions that are orthogonal to the gradient of the constraints $g_1(x)$ and $g_2(x)$ in x.

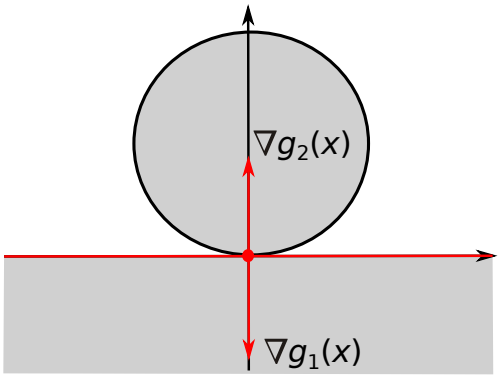


FIGURE 16.10: The circle represents the quadrati constraint, while the semi-plane represents the first degree constraint.

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. *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 16.7. *The following holds:*

AFFINE CONSTRAINTS (AFFC): *Let* g_i *and* h_j *be affine constraints. Then,* $\forall\ i \in \mathcal{I}$ *and* $j \in \mathcal{J}$ $T_X(x) = D_X(x)\ \forall x \in X$.

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 $\Longrightarrow T_X(\bar{x}) = D_X(\bar{x})$. 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 16.7 (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 16.8 (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 16.9 (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 $\mu \in \mathbb{R}^{|\mathcal{I}|}$ 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 16.10. *The Karush-Kuhn-Tucker conditions are also written as:*

FEASIBILITY: $x \in X \equiv g_i(x) \leq 0\ \ i \in \mathcal{I},\ \ h_j(x) = 0\ \ j \in \mathcal{J}$

KKT-G: $\nabla f(x) + \sum\limits_{i \in \mathcal{I}} \lambda_i \nabla g_i(x) + \sum\limits_{j \in \mathcal{J}} \mu_j \nabla h_j(x) = 0$

COMPLEMENTARITY SLACKNESS: $\sum\limits_{i \in \mathcal{I}} \lambda_i g_i(x) = 0$

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 contraints are active.

Fact 16.11. *Let (P) be a convex problem. In this case, if the Karush-Kuhn-Tucker conditions hold then x is a global optimum.*

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 (Lagrangian function)**.** *Let (P) be our minimum problem over $f : \mathbb{R}^n \to \mathbb{R}$. We term* ***Lagrangian function*** *the following:*

$$L(x,\ \lambda,\ \mu) = f(x) + \sum_{i \in \mathcal{I}} \lambda_i g_i(x) + \sum_{j \in \mathcal{I}} \mu_j h_j(x)$$

Fact 17.1. *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 17.2 (Critical cone)**.** *Let us assume (x, λ, μ) satisfies (KKT). We define the* ***critical cone*** *as:*

$$C(x,\ \lambda,\ \mu) = \left\{ d \in \mathbb{R}^n \ : \begin{array}{ll} < \nabla g_i(x), d >= 0 & i \in \mathcal{A}(x)\ s.t.\ \lambda_i^* > 0 \\ < \nabla g_i(x), d > \leq 0 & i \in \mathcal{A}(x)\ s.t.\ \lambda_i^* = 0 \\ < \nabla h_j(x), d >= 0 & i \in \mathcal{J} \end{array} \right\}$$

Theorem 17.2. *Let us assume we have a point $(x,\ \lambda,\ \mu)$ that satisfies the Karush-Khun-Tucker conditions and satisfies the linear independence of the contraints. If x is local optimum then $d^T \nabla^2_{xx} L(x,\ \lambda,\ \mu) d \geq 0\ \forall\ d \in C(x,\ \lambda,\ \mu)$.*

Informally, if the hypothesis holds, then the Hessian of the Lagrangian function is $\succeq 0$ on the critical cone.

Observation 17.1. *$(x,\ \lambda,\ \mu)$ satisfies (KKT) $\wedge$ $\nabla^2_{xx} L(x,\ \lambda,\ \mu) \succ 0$ 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 17.3 (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 17.3. *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 $[\sum_{i\in\mathcal{I}} \nabla g_i(\bar{x}) + \sum_{j\in\mathcal{J}} \nabla h_i(\bar{x})] \in \partial\psi(\lambda,\mu)$*
- *$\forall$ fixed $\lambda \geq 0, \mu, \bar{x} \in X$ $\psi(\lambda,\mu) = \min_x L(x,\lambda,\mu) \leq L(\bar{x},\lambda,\mu) \leq f(\bar{x})$*

$$\psi(\lambda,\mu) = \begin{pmatrix} \lambda_1 g_1(\bar{x}) \\ \vdots \\ \lambda_k g_k(\bar{x}) \\ \lambda_1\mu_1(\bar{x}) \\ \vdots \\ \lambda_p\mu_p(\bar{x}) \end{pmatrix} \text{ is such that } \begin{pmatrix} g_1(\bar{x}) \\ \vdots \\ g_k(\bar{x}) \\ \mu_1(\bar{x}) \\ \vdots \\ \mu_p(\bar{x}) \end{pmatrix} \text{ belongs to the supergradient.}$$

Theorem 17.4 (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

$\psi(\bar{\lambda},\bar{\mu}) = \min_x\{f(x) + \sum_i \lambda_i g_i(x) + \sum_j \mu_j h_j(x) \leq f(\bar{x}) + \sum_i \bar{\lambda}_i g_i(\bar{x}) + \cancel{\sum_j \bar{\mu}_j h_j(\bar{x})}$, 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 $f(x) \leq 0$. □

Observation 17.2. *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.

$\max\{\, \psi(\,\lambda\,,\,\mu\,) \;:\; \lambda \in \mathbb{R}_+^{|\mathcal{I}|} \;\;,\;\; \mu \in \mathbb{R}^{|\mathcal{J}|} \,\}$

Notice that the constraints on λ are very easy, so the problem may be considered somehow uncontrained.

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 fucntion 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 17.1. *Let us take a concave objective function* $\min\{-x^2 \;:\; 0 \leq x \leq 1\}$*, such that its Lagrangian function is* $L(x,\lambda) = -\mathbf{x}^2 + \lambda_1(x-1) - \lambda_2 x$*. 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 17.5. *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.*

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 satifies the KKT conditions then x_* is exactly the minimum, since the Lagrangian function is convex.

Hence, the value of the dual $v(D) \geq \psi(\lambda - *, \mu_*) = L(x_*, \lambda_*, \mu_*) = f(x_*) = v(P) \geq v(D)$. □

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 he solution of ψ is unique and it is also differentiable, bu 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) $\min\{\, cx \,:\, Ax \geq b \,\}$

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) $\max\,\{\,\psi(\,\lambda\,) \,:\, \lambda \geq 0\,\} \equiv \max\,\{\,\lambda b \,:\, \lambda A = c\,,\ \lambda \geq 0\,\}$

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 thwe quadratic case is simpler than the linear case.

(P) $\min\,\{\,\frac{1}{2}\|\,x\,\|_2^2 \colon\, Ax = b\,\}$ (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 stricltly convex since x^2 is striclty 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) $\max\{-\frac{1}{2}\mu^T(AA^T)\mu - \mu b \,:\, \mu \in \mathbb{R}^m\}$

We can generalize with quadratic functions for general problems:

Strictly convex QP: (P)

$\min\,\{\,\frac{1}{2}x^TQx + qx \,:\, Ax \geq b\,\}, Q \succ 0 \Longrightarrow$ (D) $\max\,\{\,\lambda b - \frac{1}{2}v^TQ^{-1}v \,:\, \lambda A - v = q\,,\ \lambda \geq 0\,\}$

strong duality $\equiv v(P) = v(D)$ (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 17.4 (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 17.2. *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.*

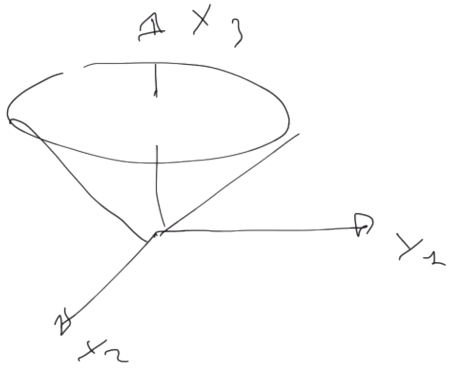


FIGURE 17.1: $x^3 \geq \sqrt{x_1^2 + x_2^2}$ is a convex cone.

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 thrww 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 17.5 (Conic dual)**.** *Conic dual: (D)* $\max\{ yb : yA = c, y \geq_{K^D} 0 \}$*, where* $K^D = \{ z :< z, x >\geq 0 \ \forall x \in K \}$ *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) $\min\{cx : \|D_i x - d_i\|_2 \leq p_i x - q_i \ i = 1, \ldots, m\}$.

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:

$$\max\{\sum_{i=1}^{m} \lambda_i d_i + \nu_i q_i \ : \ \sum_{i=1}^{m} \lambda_i D_i + \nu_i p_i = c, \ \|\lambda_i\|_2 \leq \nu_i, \ i = 1, \ldots, m\}$$

where the nu_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 mroe general form of duality, that we are going to introduce next.

17.4 Fenchel's duality

Definition 17.6 (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. $f(x) = \frac{1}{2}\|x\|_2^2 \Longrightarrow f^*(z) = \frac{1}{2}\|z\|_2^2$ (only function s.t. $f^* = f$);
2. $(\|\cdot\|_1)^*(z) = ß_{\mathcal{B}_\infty(0,1)}(z)$, $(\|\cdot\|_\infty)^*(z) = ß_{\mathcal{B}_1(0,1)}(z)$;
3. $f(x) = \max\{g_i x - \alpha_i \ i \in I\} \Longrightarrow f^*(z) = \min\{\sum_{i\in I}\alpha_i\theta_i \ : \ \sum_{i\in I} g_i\theta_i = z, \ \sum_{i\in I}\theta_i = 1 \ , \ \theta_i \geq 0 \ i \in I\}$.

Fact 17.6. *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)* $-\min\{f^*(z) + g^*(-z)\}$.

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

$$\min \left\{ \frac{1}{2} x^T Q x + qx \ : \ Ax = b \right\}$$

where $Q \succeq 0$.

A way to solve this problem is through Karusch Kuhn Tucker system:

$$\begin{matrix} \text{(a)} \\ \text{(b)} \end{matrix} \begin{bmatrix} Q & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x \\ \mu \end{bmatrix} = \begin{bmatrix} -q \\ b \end{bmatrix} \tag{18.1}$$

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:

$$\begin{cases} Qx + A^T \mu = -q & \text{(a)} \\ Ax = b & \text{(b)} \end{cases}$$

multiply (a) by AQ^{-1}:

$$\begin{cases} A\cancel{Q^{-1}}\cancel{Q}x + AQ^{-1}A^T \mu = -AQ^{-1}q & \text{(a)} \\ Ax = b & \text{(b)} \end{cases}$$

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 $A_B x_B + A_N x_N = b \iff x_B = {A_B}^{-1} \cdot (b - A_N x_N)$, 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

$$\begin{aligned} D^T(Qx + A^T\mu) &= -D^Tq \\ D^TQx + D^TA^T\mu &= -D^Tq \\ D^TQx + (\cancel{AD})^T\mu &= -D^Tq \\ D^TQ(Dx_N + d) &= -D^Tq \end{aligned} \tag{18.2}$$

Where in the last step we applied the definition $x = Dx_N + d$ and hence, $(D^TQD)x_N = -D^T(Qd + q)$.

We term $H = D^TQD \in M(n-m, \mathbb{R})$ 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 contrained 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:

$$\min\{f(x)\ :\ Ax \leq b\}\ (P)$$

18.2.1 Projected gradient method

The intuition behind this kind of algorithm is that we are interested in finding as 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:

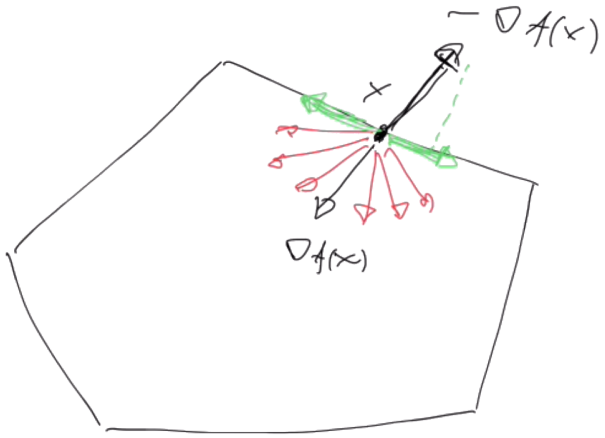


FIGURE 18.1: Geometric idea of how to choose the direction in case of lying on the boundary.

$$d = argmin\{\|\nabla f(x) - d\|^2 : d \in \mathcal{D}_X(x)\}$$

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 contraints become equality constraints:

$$\min\left\{\frac{1}{2}\|\nabla f(x) + d\|^2 = \frac{1}{2}d^T I d + \nabla f(x)d \ : \ \bar{A}d = 0\right\}$$

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:

$$\begin{aligned} d &= -\nabla f(x) - A^T\mu \\ \cancel{Ad} &= -A\nabla f(x) - AA^T\mu \\ AA^T\mu &= -A\nabla f(x) \end{aligned} \tag{18.3}$$

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 contraints is small, solving the system is very fast.

Fact 18.1. *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 contraints 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 18.1, where at each stage we remove one of the contraints 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 18.1 Pseudocode for projected gradient method for quadratic functions.

procedure PGM$(f, A, b, x, \varepsilon)$
  for $(;;)$ **do**
    $B \leftarrow$ maximal $\subseteq \mathcal{A}(x)$ s.t. $rank(A_B) = |B|$;
    for $(;;)$ **do**
      $d \leftarrow (I - A_B{}^T(A_B A_B^T)^{-1} A_B)(-\nabla f(x))$;
      if $< \nabla f(x), d > \leq \varepsilon$ **then**
        $\mu_B \leftarrow -(A_B A_B^T)^{-1} A_B \nabla f(x)$;
        $\mu_i \leftarrow 0 \; \forall \, i \notin B$;
        if $\mu_B \geq 0$ **then return**
        end if
        $h \leftarrow \min\{i \in B \; : \; \mu_i < 0\}$;
      end if
      $\bar{\alpha} \leftarrow \min\{\alpha_i = (b_i - A_i x)/A_i d \; : \; A_i d > 0, \; i \notin B\}$;
      $x \leftarrow x + \alpha d$;
      if $\bar{\alpha} > 0$ **then**
        break;
      end if
      $k \leftarrow \min\{\, i \notin B \; : \; A_i d > 0 \; : \; \alpha_i = 0 \,\}$;
      $B \leftarrow B \cup \{\, k \,\}$;
    end for
    $\alpha \leftarrow$ Line_Search$(f, x, d, \bar{\alpha})$;
    $x \leftarrow x + \alpha d$;
  end for
end procedure

Fact 18.2. *The following holds:*

1. $d = 0 \wedge \mu \geq 0 \Longrightarrow x$ *optimal (from KKT);*
2. $d = 0 \wedge \exists \, h \in B$ *s.t.* $\mu_h < 0 \Longrightarrow \; \exists \, x' \in \{\, x \in \mathbb{R}^n \; : \; A_{B\setminus\{h\}} x = b_{B\setminus\{h\}} \; , \; A_h x \leq b_h \,\}$ *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* $HH = H^T H = H \Longrightarrow < d, \nabla f(x) >< 0$.

Proof.

- The intuition is that if we remove h from B next time we will have a descent direction. $\exists \, d$ s.t. $A_{B\setminus\{h\}} d = 0 \wedge A_h d < 0 \; \Longrightarrow < \nabla f(x), d >=< -\mu A_B, d >= -\mu_h A_h d < 0$;
- $< d, \nabla f(x) >=< -H \nabla f(x), \nabla f(x) >= -\nabla f(x)^T H \nabla f(x) = -(H \nabla f(x))^T H \nabla f(x)$.

□

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 contstraints 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:

$$\text{(P)} \ \min\left\{ f(x) \ : \ l \leq x \leq u \right\}$$

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.

ALGORITHM 18.2 Pseudocode for projected gradient method for quadratic functions in the case of box constraints.

```
1: procedure PGMBC(f, l, u, x, ε)
2:     d = −∇f(x);
3:     ᾱ = ∞;
4:     for (i = 1 . . . n s.t. d_i ≠ 0) do
5:         if (d_i < 0) then
6:             if (x_i = l_i) then
7:                 d_i = 0;
8:             else
9:                 ᾱ ← min{ᾱ(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:                ᾱ ← min{ᾱ(u_i − x_i)/d_i};
16:            end if
17:        end if
18:        if (< ∇f(x), d >≤ ε) then return
19:        end if
20:        α ← Line_Search(f, x, d, ᾱ);
21:        x ← x + αd;
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:

$$\min\left\{\frac{1}{2}x^TQx + qx \ : \ Ax \leq b\right\}$$

where an important hypothesis is that the problem is quadratic, once we know $\mathcal{A}(x_*)$.

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 18.3.

ALGORITHM 18.3 Pseudocode for active set method for quadratic functions.

```
1: procedure ASMQP(Q, q, A, b, x, ε)
2:     for (B ← A(x); ;) do
3:         solve (P_B) min{½x^TQx + qx : A_Bx = b_B} for (x̄, μ̄_B);
4:         if (A_i x̄ ≤ b_i ∀ i ∉ B) then
5:             if (μ_B ≥ 0) then return
6:             end if
7:             h ← min{i ∈ B : μ_i < 0};
8:             B ← B\{h};
9:             continue;
10:        end if
11:        d ← x̄ − x;
12:        ᾱ ← min{α_i = (b_i − A_i x)/A_i d : A_i d > 0, i ∉ B};
13:        x ← x + ᾱd;
14:        B ← 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.

$\{1, \ldots, n\}\backslash(L \cup U)$ is the set of the indexes of the free variables.

Thanks to this consideration the problem to solve becomes:

$$\min\left\{\frac{1}{2}x_F{}^TQ_{FF}x_F + (q_F + u_U{}^TQ_{UF})x_F\right\} \ [+\frac{1}{2}x_U{}^TQ_{UU}x_U + q_Uu_U]$$

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 18.4 Pseudocode for Frank-Wolf method for non linear functions.

```
1: procedure FWM(f, A, b, x, ε)
2:     while (‖∇f(x)‖ > ε) do
3:         x̄ ← argmin {< ∇f(x), y > :  Ay ≤ b};
4:         d ← x̄ − x;
5:         α ← Line_Search(f,  x,  d,  1);
6:         x ← x + αd;
7:     end while
8: end procedure
```

Provided that the computation at line 3 of Algorithm 18.4 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.

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:

$$\min\left\{\frac{1}{2}x^TQx + qx \ : \ Ax \leq b\right\}$$

We would like to work with dual feasibility.

$\forall$ fixed $\lambda \geq 0$, this is the lagrangian problem $\psi(\lambda) = \min_x\{\frac{1}{2}x^TQx + qx + \lambda(b - Ax)\} \leq v$ and the Lagrangian dual is the following:

$$\max\{\psi(\lambda) \ : \ \lambda \geq 0\} \ (D)$$

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. Othewise 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:

$$\min\{f_\mu(x) = f(x) - \mu \sum_{i=1}^{m} \log(b_i - A_i x)\} \ (P_\mu)$$

The parameter μ is there to weight the proximity to the boundary.

Property 19.1.

- *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_μ optimal of (P_μ), since $\mu \sum_{i=1}^{m} \log(b_i - A_i x)$ is striclty convex;*
- *as $\mu \to 0$ x_μ converges to the analytic center of the optimal face. An example of this behaviour may be seen in Figure 19.1.*

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.

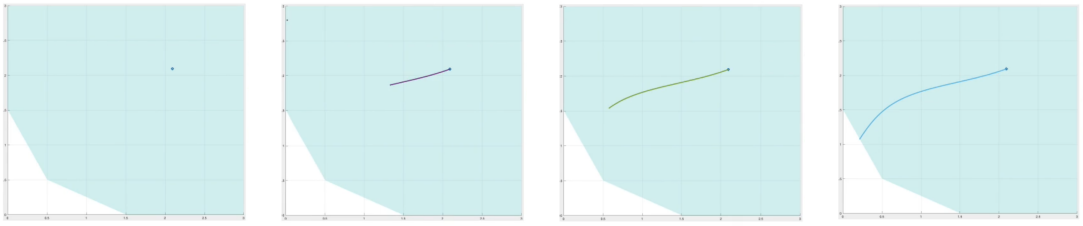


FIGURE 19.1: The trajectory converges to the optimal solution of the problem, when $\mu \to 0$.

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

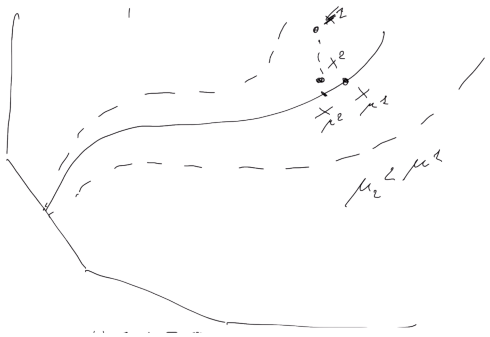


FIGURE 19.2: The dotted line represents a region where the Newton method is very efficient. We are starting from a point x_1, which belongs to that region and we want to move towards x_{μ^1}. At next iterate x_2 is closer to x_{μ^2} than the current iterate.

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 Karusch-Kuhn-Tucker conditions:

PRIMAL FEASIBILITY: $Ax + s = b,\ s \geq 0$;

DUAL FEASIBILITY: $Qx + \lambda A = -q,\ \lambda \geq 0$;

COMPLEMENTARY SLACKNESS: $\lambda_i s_i = 0,\ i = 1, \ldots, m$.

We can write a slackened version of KKT, imposing $\lambda_i s_i = \mu,\ i = 1, \ldots, m$, where $\mu \in \mathbb{R}^n$ and should be decreased over iterations until it gets closer engough 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:

- $x \to x + \Delta x$
- $s \to s + \Delta s$
- $\lambda \to \lambda + \Delta\lambda$

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: $\lambda_i s_i + \lambda_i \Delta s_i + s_i \Delta\lambda_i + \Delta\lambda_i \Delta s_i = \mu,\ i = 1, \ldots, m$.

In this new system of coordinates the first two KKT remain linear, while the third one is no longer linear ($\Delta\lambda_i \Delta s_i$).

$$\begin{bmatrix} Q & A^T & 0 \\ A & 0 & I \\ 0 & S & \Lambda \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \\ \Delta s \end{bmatrix} \overset{(1)}{=} \begin{bmatrix} -(Qx+q) - \lambda A \\ b - Ax - s \\ \mu u - \Lambda S u - \Delta\Lambda\Delta S u \end{bmatrix} \approx \begin{bmatrix} 0 \\ 0 \\ \mu u - \Lambda S u \end{bmatrix} \tag{19.1}$$

Where (1) holds since $\Delta\lambda A = A^T \Delta\lambda^T$, although $\Delta\lambda$ is written without the "transpose" sintax to ease notation.

Notice that $u = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \in \mathbb{R}^m$ and has the purpose of adjusting dimension:

$$S\Delta\lambda = \begin{bmatrix} s_1\Delta\lambda_1 \\ \vdots \\ s_m\Delta\lambda_m \end{bmatrix} \in \mathbb{R}^m;\ \Lambda\Delta s = \begin{bmatrix} \lambda_1\Delta s_1 \\ \vdots \\ \lambda_m\Delta s_m \in \mathbb{R}^m \end{bmatrix} \in \mathbb{R}^m;\ \mu u = \begin{bmatrix} \mu \\ \vdots \\ \mu \end{bmatrix} \in \mathbb{R}^m;$$

$$\Lambda S u = \begin{bmatrix} -s_1\lambda_1 \\ \vdots \\ -s_m\lambda_m \end{bmatrix} \in \mathbb{R}^m;\ \Delta\Lambda\Delta s u = \begin{bmatrix} -\Delta\lambda_1\Delta s_1 \\ \vdots \\ -\Delta\lambda_m\Delta s_m \in \mathbb{R}^m \end{bmatrix} \in \mathbb{R}^m$$

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:

$$\max\{-\lambda b - \frac{1}{2} x^T Q x \ :\ Qx + \lambda A = -q,\ \lambda \geq 0\}\ (D)$$

thus obtaining both a lower and upper bound for the solution x.

We term **complementarity gap** $(\frac{1}{2}x^TQx + qx) - (-\lambda b - \frac{1}{2}x^TQx) = \lambda s = \mu$.

Once we found a solution for Equation (19.1), 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 (19.1). 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 (19.1):

$$\begin{aligned} 0\Delta x + S\Delta\lambda + \Lambda\Delta s &= \mu u - \Lambda S u \\ \Lambda\Delta s &= \mu u - \Lambda S u - S\Delta\lambda \\ \Delta s &= \Lambda^{-1}\mu u - \cancel{\Lambda^{-1}\Lambda} S u - \Lambda^{-1}S\Delta\lambda \\ \Delta s &= \Lambda^{-1}\mu u - Su - \Lambda^{-1}S\Delta\lambda \\ \Delta s &= \Lambda^{-1}(\mu u - S\Delta\lambda) - Su \\ \Delta s &= \Lambda^{-1}(\mu u - S\Delta\lambda) - s \end{aligned} \tag{19.2}$$

We obtain the modified normal equations (or KKT system)

$$\begin{bmatrix} Q & A^T \\ A & -\Lambda^{-1}S \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \end{bmatrix} = \begin{bmatrix} 0 \\ s - \mu\Lambda^{-1}u \end{bmatrix} \tag{19.3}$$

where the last row is derived working on the second row of Equation (19.1) ($A\Delta x + 0\Delta\lambda + I\Delta s = 0 \Leftrightarrow A\Delta x = -\Delta s$), substituting Equation (19.2):

$$\begin{aligned} A\Delta x - [\Lambda^{-1}S]\Delta\lambda &= -\Delta s - [\Lambda^{-1}S]\Delta\lambda \\ &= -\Lambda^{-1}(\mu u - S\Delta\lambda) + s - \Lambda^{-1}S\Delta\lambda \\ &= -\Lambda^{-1}\mu u + \cancel{\Lambda^{-1}S\Delta\lambda} + s - \cancel{\Lambda^{-1}S\Delta\lambda} \\ &= s - \mu\Lambda^{-1}u \end{aligned} \tag{19.4}$$

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 striclty 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).

$$\begin{cases} Q\Delta x + A^T\Delta\lambda = 0 \\ (Q + A^T\Lambda S^{-1}A)\Delta x = A^T(\lambda - \mu S^{-1}u) \end{cases} \tag{19.5}$$

where the first set of equations follows from the expansion of the first row of the KKT system (Equation (19.3)) and the second one is obtained taking the same row of the same system and substituting the value of $\Delta\lambda$ as follows:

$$
\begin{aligned}
A\Delta x - \Lambda^{-1}S\Delta\lambda &= s - \mu\Lambda^{-1}u \\
\Lambda^{-1}S\Delta\lambda &= A\Delta x - s + \mu\Lambda^{-1}u \\
\Delta\lambda &= (\Lambda^{-1}S)^{-1}A\Delta x - (\Lambda^{-1}S)^{-1}s + (\Lambda^{-1}S)^{-1}\mu\Lambda^{-1}u \\
\Delta\lambda &= S^{-1}\Lambda A\Delta x - S^{-1}\Lambda s + \mu S^{-1}\cancel{\Lambda}\cancel{\Lambda^{-1}}u \\
\Delta\lambda &= S^{-1}\Lambda A\Delta x - S^{-1}\Lambda s + \mu S^{-1}u \\
\Delta\lambda &= \mu S^{-1}u + \Lambda S^{-1}A\Delta x - \Lambda S^{-1}s \\
\Delta\lambda &= \mu S^{-1}u + \Lambda S^{-1}A\Delta x - \Lambda u \\
\Delta\lambda &= \mu S^{-1}u + \Lambda S^{-1}A\Delta x - \lambda
\end{aligned}
\tag{19.6}
$$

Hence:

$$
\begin{aligned}
Q\Delta x + A^T\Delta\lambda &= 0 \\
Q\Delta x + A^T(\mu S^{-1}u + \Lambda S^{-1}A\Delta x - \lambda) &= 0 \\
Q\Delta x + A^T\mu S^{-1}u + A^T\Lambda S^{-1}A\Delta x - A^T\lambda &= 0 \\
Q\Delta x + A^T\Lambda S^{-1}A\Delta x &= -A^T\mu S^{-1}u + A^T\lambda \\
(Q + A^T\Lambda S^{-1}A)\Delta x &= A^T(\lambda - \mu S^{-1}u)
\end{aligned}
\tag{19.7}
$$

We term $M = Q + A^T\Lambda S^{-1}A$ and the following holds.

Fact 19.2. *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 bottlneck again is solving the system in Equation (19.1).

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 contraints, hence our problem becomes:

$$
\min\left\{\frac{1}{2}x^TQx + qx \;:\; Ax = b,\; 0 \leq x \leq u\right\} \; (P)
$$

In this special case, things simplify a lot.