Optimization Methods and Game Theory

Francecsco Grillea

A.A. 2021-2022
Last Update: 10 settembre 2022

Preface

I wrote these notes watching the lectures of Professor M. Passacantando during the A.A. 2021-2022. Every suggestion is welcome.

Indice

Capitolo 1

Preliminaries on convex sets and convex functions

1.1 Convex sets

Given two vectors $x, y \in \mathbb{R}^n$, a *linear combination* of x and y is a point $\alpha x + \beta y$ where $\alpha, \beta \in \mathbb{R}$.

Subspace A set $C \subseteq \mathbb{R}^n$ is a *subspace* if it contains all the linear combination of any two points in C. In other words given $u, v \in C$, $C \subseteq \mathbb{R}^n$ is a subspace if $\alpha u + \beta v \in C$ $(\forall \alpha, \beta \in \mathbb{R})$.
For example a subspace is:

- a line which pass through the origin;
- a plane which pass through the origin;
- the solution set of a homogeneous system of linear equation

$$C = \{x \in \mathbb{R}^n : Ax = 0\}$$

  where $A \in \mathbb{R}^{m \times n}$.

Affine Combination Given two vectors $x, y \in \mathbb{R}^n$, an *affine combination* of x and y is a point $\alpha x + \beta y$ where $\alpha + \beta = 1$.

Affine Set A set $C \subseteq \mathbb{R}^n$ is an *affine set* if it contains all the affine combinations of any two points in C. In other words C is an affine set if the line through any two distinct points in C lies in C.
Notice that the affine sets are subspaces with one more constraint, so any subspace is an affine set.

For example an affine set is:

- any point, line or subspace;
- the solution set of a system of linear equation

  $$C = \{x \in \mathbb{R}^n : Ax = b\}$$

  where $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$.
  Dim. Suppose $x_1, x_2 \in C$, so $Ax_1 = b$ and $Ax_2 = b$: we must prove that the affine combination $\alpha x_1 + \beta x_2$ is also in C, so that $A(\alpha x_1 + \beta x_2) = b$.

  $$\begin{aligned}
  &A(\alpha x_1 + \beta x_2) = b \\
  &\alpha A x_1 + \beta A x_2 = b \\
  &\alpha b + \beta b = b && Ax_1 = b \text{ and } Ax_2 = b \\
  &\alpha b + (1-\alpha) = b && \text{is an affine combination, so } a + b = 1 \\
  &\alpha b + b - \alpha b = b \\
  &b = b
  \end{aligned}$$

  which shows that the affine combination $\alpha x_1 + \beta x_2$ is also in C.

Convex Combination Given two vectors $x, y \in \mathbb{R}^n$, a *convex combination* of x and y is a point $\alpha x + \beta y$ where $\alpha + \beta = 1$ and $0 \leq \alpha, \beta \leq 1$.

Convex Set A set $C \subseteq \mathbb{R}^n$ is a *convex set* if it contains all the convex combinations of any two points in C. In other words a set C is convex if the line segment between any two points in C lies in C.

The figure 1.1 shows that the first set, which includes its boundary (shown darker), is convex; the second set is not convex, since the line segment between the two points is not contained in the set; the third set contains some boundary points but not others, so is not convex.

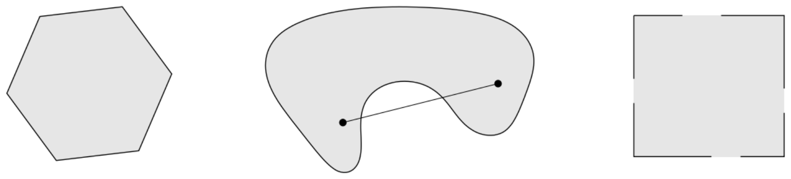


Figura 1.1

Convex Hull The *convex hull* ($conv(C)$) of a set C is the set of all convex combinations of points in C:

$$conv(C) = \{\theta x_1 + ... + \theta_k x_k \Big| x_i \in C, \theta_i \geq 0, \sum \theta_i = 1\}$$

This set is always convex and is the smallest convex set containing C.
For example a convex set is:

- any subspace or affine set;
- any line segment;
- halfspace $\{x \in \mathbb{R}^n : a^T x \leq b\}$
- polyhedron: intersection of a finite number of subspaces. Polyhedron or square are not affine sets but are convex sets. In other words the solution set of a system of linear inequality

  $$P = \{x \in \mathbb{R}^n : Ax \leq b\}$$

  is a convex set, where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$.

Operation that preserve convexity:

- sum: $C_1 + C_2 = \{x + y : x \in C_1, y \in C_2\}$ is convex;
- difference: $C_1 - C_2 = \{x - y : x \in C_1, y \in C_2\}$ is convex;
- intersection: if C_1, C_2 are convex, then $C_1 \cap C_2$ is convex. If $\{C_i\}_{i \in I}$ is any possible infinite family of convex sets, then $\bigcap_{i \in I} C_i$ is convex;
- Warning: union does not preserve convexity, just think two ball with one point in common.
- interior: if C is convex, then $int(C)$ is convex (where $int(C)$ is the set of point inside the "shape" of the set);
- closure: if C is convex, then $cl(C)$ is convex (where $cl(C)$ is the interior plus the set of point on the boundaries of the shape).

Affine Functions A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is affine if is the sum of a linear combination and a constant. Some examples are:

- $f(x) = \alpha x$ with $\alpha > 0$
- $f(x) = x + b$ with $b \in \mathbb{R}^n$
- $f(x) = Ax + b$ with $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$

An affine function can be applied to each element of a convex set. Operations that preserve convexity based on affine functions are:

- if $C \subseteq \mathbb{R}^n$ is convex, then $f(C) = \{f(x) : x \in C\}$ is convex;
- if if $C \subseteq \mathbb{R}^m$ is convex, then $f^{-1}(C) = \{f(x) \in \mathbb{R}^n : f(x) \in C\}$ is convex.

In this way, using an affine function, we can map a convex set in another convex set with a different space.

Cones A set $C \subseteq \mathbb{R}^n$ is a *cone* (with the vertex at the origin) if for any $x \in C$ and $\alpha \geq 0$, $\alpha x \in C$. The constraint $\alpha \geq 0$ means that the cone always starts from the origin and goes to infinity. In other words a set of vectors $x_1, ..., x_k$ can be used to generate a convex cone:

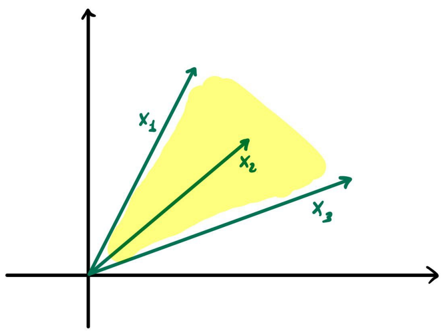


Figura 1.2: A cone made from the vectors x_1, x_2, x_3.

A *polyhedral cone* is a polyhedron that pass through the origin, so can be represented as

$$P = \{x \in \mathbb{R}^n : Ax \leq 0\} \tag{1.1}$$

The cones can be convex or not:

- $\mathbb{R}^n_+$ is a convex cone: is a cone because $\forall x \in \mathbb{R}^n_+, \alpha \geq 0$, $\alpha x \in \mathbb{R}^n_+$ and is convex because contains all the convex combinations between any two points in the set as shown in fig. 1.3 (a);
- the union of the first and third quadrant is a not-convex cone: because the segment between two points (one in the first quadrant, the second in the third) is not contained in the set as shown in fig. 1.3 (b).

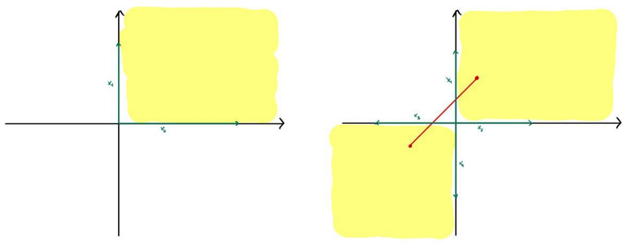


Figura 1.3: (a) a convex cone; (b) a non-convex cone

Recession cone of a Polyhedron The recession cone of a polyhedron $P = \{x : Ax \leq b\}$ is defined as

$$rec(P) = \{d : x + \alpha d \in P \ \ \forall x \in P, \alpha \geq 0\} \tag{1.2}$$

Or equivalently:

$$rec(P) = \{x : Ax \leq 0\} \tag{1.3}$$

so all the edges of the polyhedron are shifted to the origin (from $Ax \leq b$ to $Ax \leq 0$) as shown in figure 1.4.

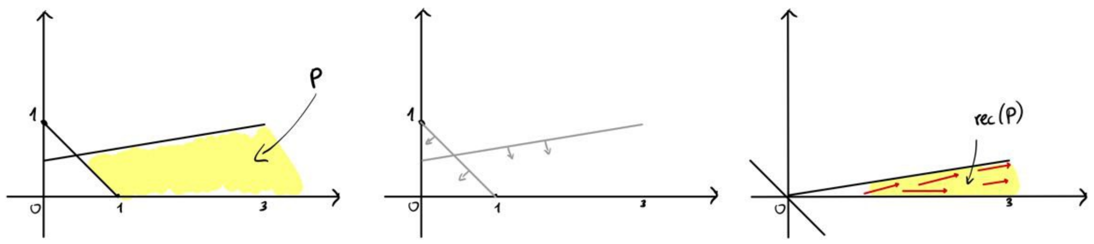


Figura 1.4: A polyhedron P and its recession cone rec(P)

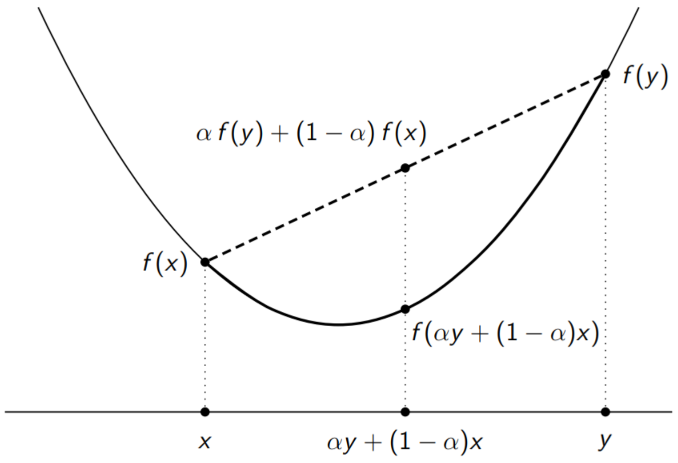
In other words the recession cone is the set of all direction that leads to infinity: this is very important from an optimization prospective because if the problem is unbounded the set will contain the potential direction to reach $+\infty$ (for maximization problem) or $-\infty$ (for minimization problem).

The cones can be polyhedral or not: if we use first order functions, the edges of the shape of are lines so is a polyhedron; if we don't use first order functions, the edges of the shapes are curves so is *not* a polyhedron. For example $\{x \in \mathbb{R}^3 : x_3 \geq \sqrt{x_1^2 + x_2^2}\}$ is the "ice cream cone" and is not a polyhedron because its surface is "rounded".

1.2 Convex functions

Def 1.2.1. Given a convex set $C \subseteq \mathbb{R}^n$, a function $f : C \to \mathbb{R}$ is convex if $\forall x, y \in C$, $x \neq y$ and $\forall \alpha \in (0,1)$ the inequality is true

$$f(\alpha y + (1-\alpha)x) \leq \alpha f(y) + (1-\alpha) f(x) \tag{1.4}$$



$f(y)$
$\alpha\, f(y) + (1-\alpha)\, f(x)$
$f(x)$
$f(\alpha y + (1-\alpha)x)$
x
$\alpha y + (1-\alpha)x$
y

Figura 1.5: Convex function

Note that the right side of the inequality is the convex combination between $f(y)$ and $f(x)$. Geometrically, this inequality means that the line segment between $(x, f(x))$ and $(y, f(y))$, which is the segment from x to y, lies above the graph of f (fig. 1.5).

If $-f$ is convex, we say that f is concave.

Strictly Convex Functions A function $f : C \to \mathbb{R}^n$ is strictly convex if $\forall x, y \in C$, $x \neq y$ and $\forall \alpha \in (0,1)$ the inequality is true

$$f(\alpha y + (1-\alpha)x) < \alpha f(y) + (1-\alpha) f(x)$$

Note that is the same inequality of convex functions but here there's a strict sign ($<$) that doesn't allow linear or constant part of the function.

Strongly Convex Functions A function $f : C \to \mathbb{R}^n$ is strongly convex if exists $\tau > 0$ s.t.

$$f(\alpha y + (1-\alpha)x) < \alpha f(y) + (1-\alpha) f(x) - \frac{\tau}{2}\alpha(1-\alpha)\|y - x\|_2^2$$

$\forall x, y \in C$, $x \neq y$ and $\forall \alpha \in (0,1)$. The constant τ is useful to control the curvature of the function f but we'll talk about that later.

Theorem 1.2.1. *f is strongly convex $\Leftrightarrow \exists \tau > 0$ s.t. $f(x) - \frac{\tau}{2}\|x\|_2^2$ is convex.*

Note that

$$\textit{Strong Convexity} \Rightarrow \textit{Strict Convexity} \Rightarrow \textit{Convexity}$$

1.2.1 First order conditions

Assume $C \subseteq \mathbb{R}^n$ is open and convex, $f : C \to \mathbb{R}$ is continuously and differentiable (its gradient ∇f exists at each point in the domain), we can re-define:

Convex Functions f is convex if and only if

$$f(y) \geq f(x) + (y - x)^T \nabla f(x)$$

$\forall x, y \in C$. Note that the right side of the inequality is the first-order Taylor approximation of f near x, that represent the global underestimator of the function. Conversely, if the first-order Taylor approximation of a function is always a global underestimator of the function, then the function is convex.

Strictly Convex Functions

f is strictly convex if and only if

$$f(y) > f(x) + (y - x)^T \nabla f(x)$$

$\forall x, y \in C$ and $x \neq y$.

Strongly Convex Functions f is strongly convex if and only if exists $\tau > 0$ such that

$$f(y) \geq f(x) + (y - x)^T \nabla f(x) - \frac{\tau}{2}\|y - x\|_2^2$$

$\forall x, y \in C$ and $x \neq y$. Note that the right side is quadratic, not linear.

Second order conditions

Assume $C \subseteq \mathbb{R}^n$ is open and convex, $f : C \to \mathbb{R}$ is *twice* continuously and differentiable (its second derivative $\nabla^2 f$ exists at each point in the domain), we can re-define:

Convex Functions f is convex if and only if $\forall x \in C$ the Hassian matrix $\nabla^2 f(x)$ is positive semi-definite, this means that

$$v^T \nabla^2 f(x) v \geq 0$$

$\forall v \neq 0$. Note that the Hassian matrix is positive semi-definite if and only if the eigenvalues of $\nabla^2 f(x)$ are ≥ 0. Calculating $\lambda_1 \lambda_2 = det(\nabla^2 f(x))$ and $\lambda_1 + \lambda_2 = Tr(\nabla^2 f(x))$, we can easily determine if the eigenvalues are positive.

Strictly Convex Functions If the Hessian matrix $\nabla^2 f(x)$ is positive definite for all $x \in C$, then f is strictly convex. Note that the matrix is positive definite if and only if

$$v^T \nabla^2 f(x) v > 0$$

Strongly Convex Functions The function f is strongly convex function if and only if exists $\tau > 0$ such that $\nabla^2 f(x) - \tau I$ is positive semi-definite for all $x \in C$ this means that:

$$v^T \nabla^2 f(x) v \geq \tau \|v\|_2^2$$

$\forall x \neq 0$. Similarly to before, the Hassian matrix is positive semi-definite if and only if the eigenvalues of $\nabla^2 f(x) - \tau I$ are ≥ 0, or equivalently the eigenvalues of $\nabla^2 f(x) \geq \tau$.

Examples

- $f(x) = c^T x$ (the linear function) is both convex and concave;
- $f(x) = \frac{1}{2} x^T Q x + c^T x$ (the quadratic function) is:
  - convex $\Leftrightarrow$ Q is positive semi-definite;
  - strongly convex $\Leftrightarrow$ Q is positive definite;
  - concave $\Leftrightarrow$ Q is negative semi-definite;
  - strongly concave $\Leftrightarrow$ Q is negative definite.
- $f(x) = e^{ax}$ is strictly convex because $f''(x) = f(x) = e^x > 0$ for all $x \in C$, but not strongly convex because $\exists \tau$ such that $e^x \not> \tau$.
- $f(x) = log(x)$ is strictly concave because $f''(x) = -\frac{1}{x^2} < 0$. Also the function is not strongly concave for the same reason of the exponential function.

- $f(x) = \|x\|$ is convex but not strictly convex. Since the norm function is not continuously and differentiable everywhere (for example in 0), we can prove the convexity using the equation 1.4

$$\|\alpha x + (1-\alpha)y\| \leq \|\alpha x\| + \|(1-\alpha)y\|$$

- $f(x) = max\{x_1, ..., x_n\}$ is convex but not strictly convex because is linear.

Operation that preserve convexity:

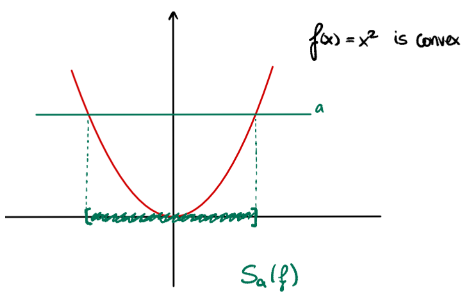
- if f is convex and $\alpha > 0$, then αf is convex;
- if f_1 and f_2 are convex, then $f_1 + f_2$ is convex;
- if f is convex, then $f(Ax + b)$ is convex. Note that $Ax + b$ is an affine function;
- if $f_1, ..., f_n$ are convex, then $max\{f_1, ..., f_n\}$ is convex;

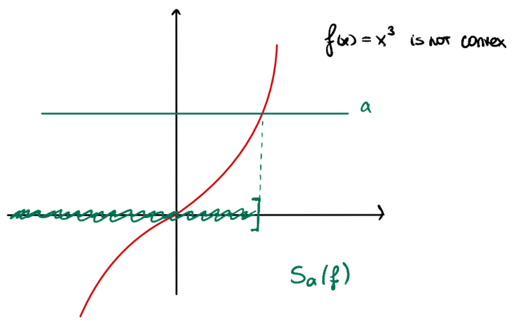
1.2.2 Sublevel Sets

Given $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and $\alpha \in \mathbb{R}$, the set

$$S_\alpha(f) = \{x \in \mathbb{R}^n : f(x) \leq \alpha\}$$

is called $\alpha-$sublevel set of f. Note that if f is convex, then $S_\alpha(f)$ is a convex set for any value of α (the opposite is not true).





Given $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and $\alpha \in \mathbb{R}$, the set

$$S_\alpha(f) = \{x \in \mathbb{R}^n : f(x) = \alpha\}$$

is called α-level set of f. Note that if f is affine, then $S_\alpha(f)$ is a convex set for any value of α (the opposite is not true).

This tool is the link between convexity in function and convexity in sets because we can use the sublevel set property to establish convexity of a set, by expressing it as a sublevel set of a convex function.

1.2.3 Quasiconvex Functions

Given a convex set $C \subseteq \mathbb{R}^n$, a function $f : C \to \mathbb{R}$ is quasiconvex if the $\alpha-$sublevel sets are convex for all $\alpha \in \mathbb{R}$. In the same way of convex function, f is said quasiconcave function if $-f$ is quasiconvex. In other words a continuous funciton f is quasiconvex if and only if at least one of the following conditions holds:

- f is nondecreasing
- f is nonincreasing
- exists a point $c \in dom f$ such that for $t \leq c$ (and $t \in dom f$), f is nonincreasing, and for $t \geq c$, f is nondecreasing (fig. 1.6).

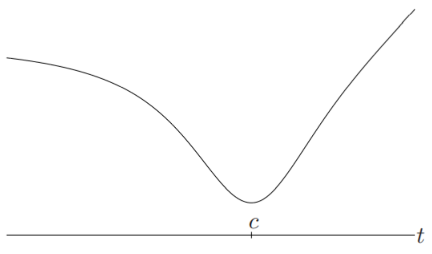


Figura 1.6: The function is nonincreasing for $t \leq c$ and nondecreasing for $t \geq c$. Note that f is quasiconvex but not convex.

Examples:

- $f(x) = \sqrt{|x|}$ is quasiconvex on $\mathbb{R}$;
- $f(x_1, x_2) = x_1 x_2$ is quasiconvex on $\{x \in \mathbb{R}^2 : x_1 > 0, x_2 > 0\}$;
- $f(x) = log(x)$ is quasiconvex and quasiconcave;
- $f(x) = ceil(x) = inf\{z \in \mathbb{Z} : z \geq x\}$ is quasiconvex and quasiconcave.

Now we can extend the previous sequence of implications:

$$Strong\ Convexity \Rightarrow Strict\ Convexity \Rightarrow Convexity \Rightarrow Quasiconvexity$$

Capitolo 2

Existence of optimal solutions and optimality conditions

2.1 Definitions

An optimization problem is defined as follows:

$$\begin{cases} \min f(x) \\ g(x) \leq 0 \\ h(x) = 0 \end{cases} \tag{2.1}$$

where

- $f : \mathbb{R}^n \to \mathbb{R}$ is the objective function (or cost function) that we want to minimize;
- $g(x) = (g_1(x), ..., g_m(x))$, where $g_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, ..., m$ are the inequality constraints functions;
- $h(x) = (h_1(x), ..., h_p(x))$, where $h_j : \mathbb{R}^n \to \mathbb{R}$, $i = 1, ..., p$ are the equality constraints functions.

The *domain* of the problem is the set of points where the functions are defined

$$\mathcal{D} = dom(f) \cap \bigcap_{i=1}^{m} dom(g_i) \cap \bigcap_{j=1}^{p} dom(h_j)$$

The *feasible region* is the set of point that can be the solution of the problem: the problem is feasible if there exists at least one feasible point

$$\Omega = \{x \in \mathcal{D} : g(x) \leq 0, h(x) = 0\}$$

Note that every minimization problem can be "reversed" to maximization problem as

$$min\{f(x) : x \in \Omega\} = max\{-f(x) : x \in \Omega\}$$

The *optimum value* v^* of the problem is defined as

$$v^* = inf\{f(x) : x \in \Omega\}$$

which can be a real value ($v^* \in \mathbb{R}$) if the problem is bounded below, $v^* = -\infty$ if the problem is unbounded below or $v^* = \infty$ is the problem is infeasible ($\Omega = \emptyset$).

The *global optimal solution* is a feasible point $x^* \in \Omega$ s.t.

$$f(x^*) \leq f(x) \qquad \forall x \in \Omega$$

Sometimes is very difficult to find the *global* optimal solution, but we can easily find the *local* optimal solution: a feasible point x^* such that

$$f(x^*) \leq f(x) \qquad \forall x \in \Omega \cap B(x^*, R)$$

2.1.1 Convex Problems

An optimization problem defined as 2.1 is said *convex* if the following condition holds:

- f is convex
- $g_1, ..., g_m$ are convex functions
- $h_1, ..., h_p$ are affine functions (i.e. $h_j(x) = c^T x + d$)

The following problem is convex

$$\begin{cases} \min\ x_1^2 + x_1 x_2 + 3x_2^2 + 4x_1 + 5x_2 \\ x_1^2 + x_2^2 - 4 \leq 0 \\ x_1 + x_2 - 2 = 0 \end{cases}$$

The following problem is not convex: f is convex, g is not convex and h is not an affine function.

$$\begin{cases} \min\ x_1^2 + x_2^2 \\ x_1/(1+x_2^2) \leq 0 \\ (x_1+x_2)^2 = 0 \end{cases}$$

but can be written in an equivalent way that makes the problem convex:

$$\begin{cases} \min\ x_1^2 + x_2^2 \\ x_1 \leq 0 \\ x_1+x_2 = 0 \end{cases}$$

In general, two problems are equivalent when the solution sets are equals, sometimes is useful re-write the problem in an equivalent (and easier) form.

Theorem 2.1.1. *In any convex optimization problem the feasible region is a convex set.*

Is easy to prove that because the feasible region is the intersection of all the constraints (that are convex): the sublevel sets of convex functions $(g_1, ..., g_m)$ are convex and the level sets of affine functions $(h_1, ..., h_p)$ are convex, so the intersection of the sets is convex.

Theorem 2.1.2. *In any convex optimization problem any local optimum is a global optimum.*

2.2 Existence of optimal solutions

Theorem 2.2.1 (Weierstrass)**.** *If the objective function f is continuous and the feasible region Ω is closed and bounded, then (at least) a global optimum exists.*

Corollary 1 If all the functions f, g_i, h_j are continuous, the domain $\mathcal{D}$ is closed and the feasible region Ω is bounded, then there (at least) a global optimum exists.

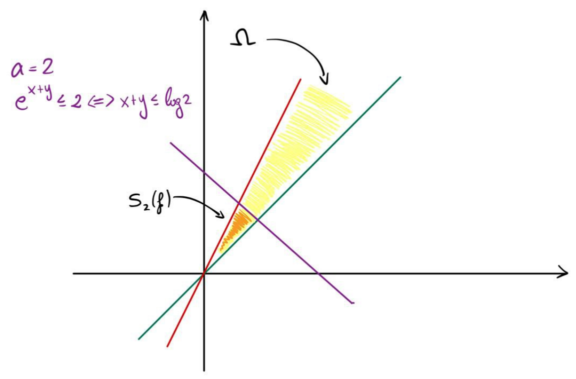
Corollary 2 If the objective function f is continuous, the feasible region Ω is closed (even if is not bounded) and there exists $\alpha \in \mathbb{R}$ such that the α-sublevel set

$$S_\alpha(f) = \{x \in \Omega : f(x) \leq \alpha\}$$

is nonempty and bounded, then (at least) a global optimum exists. This corollary is useful even when the feasible region is *not* bounded, so the intuition is to reduce the feasible region Ω to a subset S_α that is bounded. For example

$$\begin{cases} \min\ e^{x_1+x_2} \\ x_1 - x_2 \leq 0 \\ -2x_1 + x_2 = 0 \end{cases}$$

f is continuous, Ω is unbounded (above), but the sublevel set $S_2(f) = \{x \in \Omega : f(x) \leq 2\}$ is nonempty and bounded, thus a global optimum exists.



Corollary 3 If the objective function f is continuous and *coercive*, i.e.

$$\lim_{\|x\| \to \infty} f(x) = +\infty$$

and the feasible region Ω is closed, then (at least) a global optimum exists. Is easy to prove because any sublevel set of a coercive function f is bounded by definition, and then we can use the Corollary 2. Let's do an example:

$$\begin{cases} \min\ x^4 + 3x^3 - 5x^2 + x - 2 \\ x \in \mathbb{R} \end{cases}$$

even if the feasible region is unbounded (because there are no constraints), we're sure that a global optimum exists since the objective function is continuous and coercive. Note that we need that $\|x\| \to \infty$, this means that we have to calculate it with $x \to \infty$ and $x \to -\infty$ and the result needs to go towards $+\infty$ in both cases.

Corollary 3 v2 If f is strongly convex and the feasible region Ω is closed, then there exists a global optimum. Is a consequence of the previous corollary because strongly convex functions are continuous and coercive, so we can apply the Corollary 3.

Corollary 4 If f is strongly convex and Ω is closed *and convex*, then there exists a *unique* global optimum.

2.2.1 Existence of global optima for quadratic programming problems

Considering the following quadratic problem

$$\begin{cases} \min \frac{1}{2}x^TQx + c^Tx \\ Ax \leq b \end{cases} \tag{2.2}$$

and the recession cone of Ω as

$$rec(\Omega) = \{d : Ad \leq 0\}$$

Theorem 2.2.2 (Eaves). *There exists a global optimum for 2.2 if and only if the following conditions hold:*

(a) $d^TQd \geq 0$ for any $d \in rec(\Omega)$

(b) $d^T(Qx + c) \geq 0$ for any $x \in \Omega$ and any $d \in rec(\Omega)$ such that $d^TQd = 0$

Special cases:

- if Q is positive definite, then
  - (a) is satisfied because if Q is positive definite, then d^TQd is always ≥ 0;
  - (b) is satisfied because if Q is positive definite, the only direction $d \in rec(\Omega)$ such that $d^TQd = 0$ is the direction $d = 0$. If $d = 0$ then $d^T(Qx + c) = 0$.
- if Ω is bounded, then the recession cone $rec(\Omega)$ is given only by the vector $d = 0$, so
  - (a) is satisfied because if $d = 0$, then $d^TQd = 0$;
  - (b) is satisfied because if $d = 0$, then $d^T(Qx + c) = 0$;
- if $Q = 0$ then we have a linear programming problem, so the problem 2.2 has a global optimum if and only if $d^Tc \geq 0$ for any $d \in rec(\Omega)$. This means that we have no recession direction that is a descend direction.

2.3 First order optimality condition

2.3.1 Unconstrained problems

Consider a minimization problem without constraints: $min\{f(x) : x \in \mathbb{R}\}$ where f is continuously differentiable.

Theorem 2.3.1. *If x^* is a local optimum* $\Rightarrow \nabla f(x^*) = 0$

Proof: by contradiction, assume that $\nabla f(x^*) \neq 0$. Choose a (descent) direction $d = -\nabla f(x^*)$ and define $\varphi(t) = f(x^* + td)$,

$$\varphi'(0) = d^T \nabla f(x^*) = -\|\nabla f(x^*)\|^2 < 0$$

thus $f(x^* + td) < f(x^*)$ for all t small enough, which is impossible because x^* is a local optimum.

Theorem 2.3.2. *If f is convex, then x^* is a global optimum* $\Leftrightarrow \nabla f(x^*) = 0$

The initial condition is very important, otherwise if f was not convex the derivative could be 0 in more than one point (for example $x^3 + 2x^2$), or that point could not be a minimum or maximum (for example x^3).

$$\text{Stationary Points} \subseteq \text{Local Minima} \subseteq \text{Global Minima}$$

2.3.2 Constrained problems

To generalize the previous theorems to a constrained problem we have to introduce first the tangent cone.

Tangent Cone Given $x \in \Omega$, the set

$$T_\Omega(x) = \left\{ d \in \mathbb{R}^n : \exists\{z_k\} \subset \Omega, \exists\{t_k\} > 0, z_k \to x, t_k \to 0, \lim_{k\to\infty} \frac{z_k - x}{t_k} = d \right\}$$

is called the *tangent cone* to Ω at x. It is the set of tangent direction to the feasible region starting from the point x^*. If x^* is inside the feasible region, the tangent cone is the whole plane.

Theorem 2.3.3. *If x^* is a local optimum, then*

$$d^T \nabla f(x^*) \geq 0 \quad \forall d \in T_\Omega(x^*)$$

in other words if x^* is not a local optimum, will exist a direction d such that the tangent is negative ($d^T\nabla f(x^*) < 0$): this means that exists a point x^{**} such that $f(x^{**}) < f(x^*)$. Note that this inequality has to be satisfied for all the vectors in the tangent cone, so we may have an infinite number of inequalities to satisfy and it's not very useful in practice.

Theorem 2.3.4. *If Ω is convex, then $\Omega \subseteq T_\Omega(x) + x$ for any $x \in \Omega$.*

Theorem 2.3.5. *If the optimization problem is convex, then x^* is a global optimum if and only if*

$$(y - x^*)^T \nabla f(x^*) \geq 0 \quad \forall y \in \Omega$$

To understand how to use the tangent cone, we need another kind of cone that is the first-order feasible direction cone.

First Order Feasible Direction Cone Given $x \in \Omega$, the set $\mathcal{A}(x) = \{i : g_i(x) = 0\}$ denotes the set of inequality constraints which are active at x. The set

$$D(x) = \left\{ d \in \mathbb{R}^n : \begin{array}{l} d^T\nabla g_i(x^*) \leq 0 \ \ \forall i \in \mathcal{A}(x^*) \\ d^T\nabla h_j(x^*) = 0 \ \ \forall j = 1,...,p \end{array} \right\}$$

is called the first order feasible direction cone at x.

Theorem 2.3.6.

$$T_\Omega(x) \subseteq D(x) \quad \forall x \in \Omega$$

Abadie Constraints Qualification (ACQ) If $T_\Omega(x) = D(x)$, then the Abadie Constraints Qualification holds at x, but in practice this condition is very difficult to check. In general ACQ doesn't hold at any $x \in \Omega$, but ACQ holds (in one or any point) if any of the three sufficient condition is satisfied:

a) [Affine Constraints] If g_i and h_j are affine for all $i = 1,...,m$ and $j = 1,...,p$, then ACQ holds at any $x \in \Omega$;

b) [Slater Condition] If g_i are convex for all $i = 1,...,m$, h_j are affine for all $j = 1,...,p$ and there exists $\bar{x} \in int(\mathcal{D})$ s.t. $g(\bar{x}) < 0$ and $h(\bar{x}) = 0$, then ACQ holds at any $x \in \Omega$;

c) [Linear independence of the gradients of active constraints] If $\bar{x} \in \Omega$ and the vectors
$$\begin{cases} \nabla g_i(\bar{x}) & \forall i \in \mathcal{A}(\bar{x}), \\ \nabla h_j(\bar{x}) & \forall j = 1, ..., p \end{cases}$$
are linear independent, then ACQ holds at $\bar{x}$. Note that this condition is local for $\bar{x}$ and not for all x.

The following theorem makes the ACQ important.

Karush-Kuhn-Tucker Theorem If x^* is a local optimum and ACQ holds at x^*, then there exist $\lambda^* \in \mathbb{R}^m$ and $\mu \in \mathbb{R}^p$ s.t. (x^*, λ^*, μ^*) satisfies the KKT system:

$$\begin{cases} \nabla f(x^*) + \sum_{i=1}^m \lambda_i^* \nabla g_i(x^*) + \sum_{j=1}^p \mu_j^* \nabla h_j(x^*) = 0 \\ \lambda_i^* g_i(x^*) = 0 \qquad \forall i = 1, ..., m \\ \lambda^* \geq 0 \\ g(x^*) \leq 0 \\ h(x^*) = 0 \end{cases}$$

so using the KKT Theorem, an optimization problem can be written in a different form using vectors λ^* and μ^* in the local minimum x^*. If the KKT-system has more than one solution, we can find the global optimum (because we're sure it exists) just taking the point with the lowest value of f.

Note that it is a necessary condition: even if the system has no solution, the global optimum could exists, and for the same reason if (x^*, λ^*) solves the KKT system, x^* could not be a local optimum. Here some examples:

- $$\begin{cases} \min x_1 + x_2 \\ (x_1 - 1)^2 + (x_2 - 1)^2 - 1 \leq 0 \\ x_2 \leq 0 \end{cases}$$
  $\Omega = \{(1, 0)\} \Rightarrow x^* = (1, 0)$ is the global optimum.
  $T_\Omega(x^*) = \{0\}$ and $D(x^*) = \{d \in \mathbb{R}^2 : d_2 = 0\} \Rightarrow$ ACQ doesn't hold at x^*.
  $\nabla g_1(x^*) = (0, -2)$, $\nabla g_2(x^*) = (0, 1)$, $\nabla f(x^*) = (1, 1) \Rightarrow \not\exists \lambda^*$ s.t. (x^*, λ^*) solves the KKT system, even if the global optimum exists!

- $$\begin{cases} \min\, x_1 + x_2 \\ -x_1^2 - x_2^2 + 2 \leq 0 \end{cases}$$

  $x^* = (1,1)$, $\lambda^* = \frac{1}{2}$ solves the KTT system, but x^* is not a local optimum.

KKT Theorem for convex problems If the optimization problem is convex and (x^*, λ^*, μ^*) solves the KKT system, then x^* is a global optimum.

The Euclidean Projection of a point $z \in \mathbb{R}^n$ on a set $\Omega \subset \mathbb{R}^n$ is the solution set of the following optimization problem:

$$\begin{cases} \min\, \|x - z\|_2 \\ x \in \Omega \end{cases} \qquad \text{equivalently} \qquad \begin{cases} \min\, \frac{1}{2}\|x - z\|_2^2 \\ x \in \Omega \end{cases}$$

in other words x is the closest value to z.

2.4 Second order optimality condition

In general the second order optimality conditions are used for non-convex problem. Using the first derivative we're not always able to find the minimum of the function: for example let's find the minimum of $f_1(x) = x^2$ and $f_2(x) = -x^2$. If we calculate the first derivative of the two function on the point 0, we can notice that for $\nabla f_1(0) = 0$, the point represent the global minimum of the function; while for $\nabla f_2(0) = 0$ the same point represent the global maximum of the function. So we have to consider the second derivative to calculate the convexity or concavity of the function. First of all let's introduce the critical cone.

Critical Cone If (x^*, λ^*, μ^*) solves the KKT system, then the critical cone at (x^*, λ^*, μ^*) is defined as

$$C(x^*, \lambda^*, \mu^*) = \left\{ d \in \mathbb{R}^n : \begin{array}{c} d^T \nabla g_i(x^*) = 0 \;\; \forall i \in \mathcal{A}(x^*) \text{ con } \lambda_i^* > 0 \\ d^T \nabla g_i(x^*) \leq 0 \;\; \forall i \in \mathcal{A}(x^*) \text{ con } \lambda_i^* = 0 \\ d^T \nabla h_j(x^*) = 0 \;\; \forall j = 1, ..., p \end{array} \right\}$$

This cone is called critical since it's the set of directions where the objective function could be decreasing, so if we find a sequence of points along f that is decreasing the point is just a candidate but not a local minimum. Another equivalently definition is

$$C(x^*, \lambda^*, \mu^*) = \Big\{ d \in D(x^*) : d^T \nabla f(x^*) = 0 \Big\}$$

that is the set of direction in the first order feasible direction cone that are orthogonal to the gradient of $f(x^*)$.

Lagrangian Function The Lagrangian function is defined as

$$L(x, \lambda, \mu) := f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) + \sum_{j=1}^{p} \mu_j h_j(x) \tag{2.3}$$

Necessary optimality condition Assume that (x^*, λ^*, μ^*) solves the KKT system and the gradients of active constraints at x^* are linear independent. If x^* is a local optimum, then

$$d^T \nabla^2_{xx} L(x^*, \lambda^*, \mu^*) d \geq 0 \ \ \forall d \in C(x^*, \lambda^*, \mu^*)$$

where $\nabla^2_{xx} L(x^*, \lambda^*, \mu^*)$ is the Hassian matrix of $L(\cdot, \lambda^*, \mu^*)$ at x^*.

But if the problem is unconstrained we can consider the following special case: if x^* is a local optimum, then $\nabla^2 f(x^*)$ is positive semi-definite.

Sufficient optimality condition If (x^*, λ^*, μ^*) solves the KKT system and

$$d^T \nabla^2_{xx} L(x^*, \lambda^*, \mu^*) d \geq 0 \ \ \forall d \in C(x^*, \lambda^*, \mu^*), d \neq 0$$

then x^* is a local optimum.

But if the problem is unconstrained we can consider the following special case: if $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive definite, then x^* is a local optimum.

Capitolo 3

Lagrangian duality

Consider the general optimization problem 2.1 where $x \in \mathcal{D}$ (the domain of the problem, so the intersection of the domains of the data) and $v(P)$ is the optimal value for the problem P and the Lagrangian Function $L : \mathcal{D} \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ defined in 2.3.

Lagrangian Relaxation Given $\lambda \geq 0$ and $\mu \in \mathbb{R}^p$, the problem

$$\begin{cases} \min \ L(x, \lambda, \mu) \\ x \in \mathcal{D} \end{cases}$$

is called Lagrangian Relaxation of the problem P and the basic idea in Lagrangian duality is to put the constraints g, h inside the objective function (giving a weight at each of them) and relaxing the new constraints. In other words this is an easier and equivalent form of the primal problem.

Lagrangian Dual Function The Lagrangian dual function is defined as

$$\varphi(\lambda, \mu) = \min_{x \in \mathcal{D}} L(x, \lambda, \mu) = L(x^*, \lambda, \mu)$$

i.e. the minimum value of the Lagrangian over x. The lagrangian dual function φ

- is concave even when the "original" problem is not convex because;
- may be equal to $-\infty$ at some point if the lagrangian is unbounded below;
- may be not differentiable at some point

Theorem 3.0.1. *For any $\lambda \geq 0$ and $\mu \in \mathbb{R}^p$, we have $\varphi(\lambda, \mu) \leq v(P)$.*

this means that the Lagrangian dual function provides a lower bound to the optimal value of the primal $v(P)$. Is easy to prove this theorem: let $x \in \Omega$, $g(x) \leq 0$ and $h(x) = 0$, then

$$L(x, \lambda, \mu) := f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) \leq f(x)$$

$$\varphi(\lambda, \mu) = \min_{x \in \mathcal{D}} L(x, \lambda, \mu) \leq \min_{x \in \Omega} L(x, \lambda, \mu) \leq \min_{x \in \Omega} f(x) = v(P)$$

Lagrangian Dual Problem The problem

$$\begin{cases} \max \varphi(\lambda, \mu) \\ \lambda \geq 0 \end{cases}$$

is called Lagrangian dual problem of P. In other words the dual problem D consists in finding the best lower bound of $v(P)$. Note that D is always a convex problem (maximize a concave function is equivalent to minimize a convex function), even if P is a non-convex problem.

Weak Duality For any optimization problem P, we have $v(D) \leq v(P)$.

Strong Duality Strong duality ($v(D) = v(P)$) does not hold in general. Suppose f, g, h are continuously differentiable, the primal problem P is convex, there exists a global optimum x^* and ACQ holds at x^*. Then

- $v(D) = v(P)$
- (λ^*, μ^*) is optimal for D if and only if (λ^*, μ^*) is a KKT multiplier vectors associated to x^*.

[Proof not written]. Note that strong duality may hold also for some non-convex problems.

Capitolo 4

Support Vector Machines

Support Vector Machines are machine learning models used in regression and classification supervised problems.

4.1 Linear SVM

Let's consider a supervised binary classification problem: we have two finite sets $A, B \subset \mathbb{R}^n$ with known labels (1 for points $\in A$, -1 for points $\in B$). $\mathbb{R}^n$ is the input space, $A \cup B$ is the training set.

Assume that A and B are linear separable i.e. exists a separating hyperplane $H = \{x \in \mathbb{R}^n : w^T x + b = 0\}$ such that

$$w^T x + b > 0 \; \forall x_i \in A$$

$$w^T x + b < 0 \; \forall x_i \in B$$

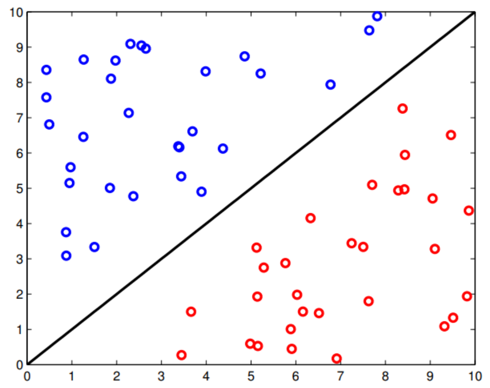


Figura 4.1: linear separable dataset

When comes in a new data x, we use the decision function to decide which class the data belongs to.

$$f(x) = sign(w^T x + b) = \begin{cases} 1 & \text{if } w^T x + b > 0 \\ -1 & \text{if } w^T x + b < 0 \end{cases}$$

Using the properties of convexity (Chapter 1) can be defined a necessary and sufficient condition for A and B to be linearly separable:

$$conv(A) \cap conv(B) = \emptyset \Leftrightarrow A, B \text{ are linearly separable}$$

To choose the best separating hyperplane when we have multiple choices, we have to introduce first the concept of margin.

Margin of Separation If H is a separating hyperplane, then the margin of separation of H is defined as the minimum distance between H and the closest training point to H,

$$\rho(H) = \min_{x \in A \cup B} \frac{|w^T x + b|}{\|w\|}$$

so we look for the separating hyperplane with the maximum margin of separation.

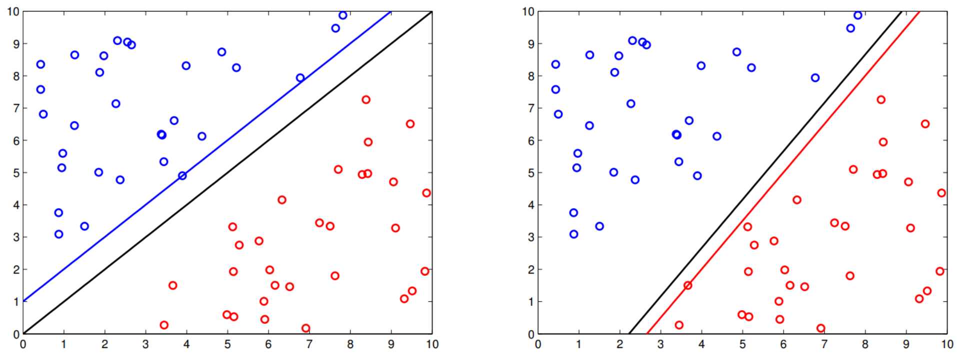


Figura 4.2: Margin of separation of two sets

Finding the separating hyperplane with the maximum margin of separation is equivalent to solve the following convex quadratic programming problem:

$$\begin{cases} \min\limits_{w,b} \frac{1}{2}\|w\|^2 \\ w^T x^i + b \geq 1 & \forall x^i \in A \\ w^T x^j + b \leq -1 & \forall x^j \in B \end{cases} \tag{4.1}$$

[Proof on iPad]. Note that f is a convex function (not strongly because b doesn't appear) and g are affine functions: this means that ACQ holds for any $x \in \Omega$ and we can apply the KKT system to find the global (unique) solution (w^*, b^*) for the problem 4.1.

Let y^i the label for the data x^i ($y^i = 1$ if $x^i \in A$ or $y^i = -1$ if $x^i \in B$), we can re-write the problem 4.1 as

$$\begin{cases} \min\limits_{w,b} \frac{1}{2}\|w\|^2 \\ 1 - y^i(w^T x^i + b) \leq 0 \end{cases} \tag{4.2}$$

and is often useful to consider the Lagrangian dual of this new problem where the Lagrangian Function is

$$\begin{aligned} L(w,b,\lambda) &= \frac{1}{2}\|w\|^2 + \sum_{i=1}^{l} \lambda_i[1 - y^i(w^T x^i + b)] \\ &= \frac{1}{2}\|w\|^2 - \sum_{i=1}^{l} \lambda_i y^i w^T x^i - b\sum_{i=1}^{l} \lambda_i y^i + \sum_{i=1}^{l} \lambda_i \end{aligned}$$

note that:

- if $\sum_{i=1}^{l} \lambda_i y^i \neq 0$ the problem is unbounded below, then $\min\limits_{w,b} L(w,b,\lambda) = -\infty$ because b is linear and without constraints (we can decrease to infinity);
- if $\sum_{i=1}^{l} \lambda_i y^i = 0$, then L does not depend on b, so L is strongly convex (because L depends only on w that is quadratic) and has a unique global minimum that is given by the stationary point

$$\nabla_w L(w,b,\lambda) = w - \sum_{i=1}^{l} \lambda_i y^i x^i = 0$$

Therefore the dual function is [computations on iPad]:

$$\varphi(\lambda) = \begin{cases} -\infty & \text{if } \sum_{i=1}^{l} \lambda_i y^i \neq 0 \\ -\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} y^i y^j (x^i)^T x^j \lambda_i \lambda_j \sum_{i=1}^{l} \lambda_i & \text{if } \sum_{i=1}^{l} \lambda_i y^i = 0 \end{cases}$$

Note that φ is a quadratic function because depends on λ. Now we can write the dual of 4.2 as

$$\begin{cases} \max\limits_{\lambda} -\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} y^i y^j (x^i)^T x^j \lambda_i \lambda_j \sum_{i=1}^{l} \lambda_i \\ \sum_{i=1}^{l} \lambda_i y^i = 0 \\ \lambda \geq 0 \end{cases}$$

or

$$\begin{cases} \max\limits_{\lambda} -\frac{1}{2}\lambda^T X^T X \lambda + e^T \lambda \\ \sum_{i=1}^{l} \lambda_i y^i = 0 \\ \lambda \geq 0 \end{cases} \tag{4.3}$$

where $X = (y^1x^1, y^2x^2, ..., y^lx^l)$ is the first column of a $n \times l$ matrix (remember that $x, y \in \mathbb{R}^n$) and $e = (1, .., 1)$ is a vector. Note that we have only one linear constrain and the objective function is concave because $X^T X$ is always positive semidefinite, so $-X^T X$ is always negative semidefinite.

Here some property of Linear SVM dual problem:

- Dual problem is a convex quadratic programming problem;
- Dual constraints are simpler than primal constraints;
- The dual has optimal solutions, because each KKT multiplier associated to the primal optimum is a dual optimum;
- If λ^* is a dual optimum, then

  $$w = \sum_{i=1}^{l} \lambda_i y^i x^i$$

  is the slope of the separating hyperplane: if $\lambda_i > 0$ the corresponding x_i is called support vector because it's supporting the final result of w^*. We can therefore say that w is the linear combination of its support vectors.
- b^* is obtained using the complementary conditions:

  $$\lambda_i^*[1 - y^i((w^*)^T x^i + b^*)] = 0$$

  this means that if i such that $\lambda_i > 0$ we have

  $$b^* = \frac{1}{y^i} - (w^*)^T x^i$$
- Finally, the decision function is

  $$f(x) = sign((w^*)^T x + b^*)$$

4.1.1 SVM with soft margin

When the sets A and B are not linear separable, we can "ignore" some misclassified points to make the two sets linear separable: to do this we have to introduce the *slack variables* $\xi_i \geq 0$ that represents the tolerance for each training data that could be misclassified because of the noise. Having this tolerance in the separation process, the margin is called "soft". For this approach we consider the following relaxed system

$$\begin{cases} 1 - y^i(w^T x^i + b) \leq \xi_i & i = 1, ..., l \\ \xi_i \geq 0 & i = 1, ..., l \end{cases}$$

note that if x^i is misclassified, then $\xi_i > 1$. This means that $\sum_{i=1}^{l} \xi_i$ is an upper bound of the numbers of misclassified points.

Using the soft margin, the problem 4.2 is replaced with

$$\begin{cases} \min\limits_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l} \xi_i \\ 1 - y^i(w^T x^i + b) \leq \xi_i & i = 1, ..., l \\ \xi_i \geq 0 & i = 1, ..., l \end{cases} \tag{4.4}$$

where $C > 0$ is a parameter.

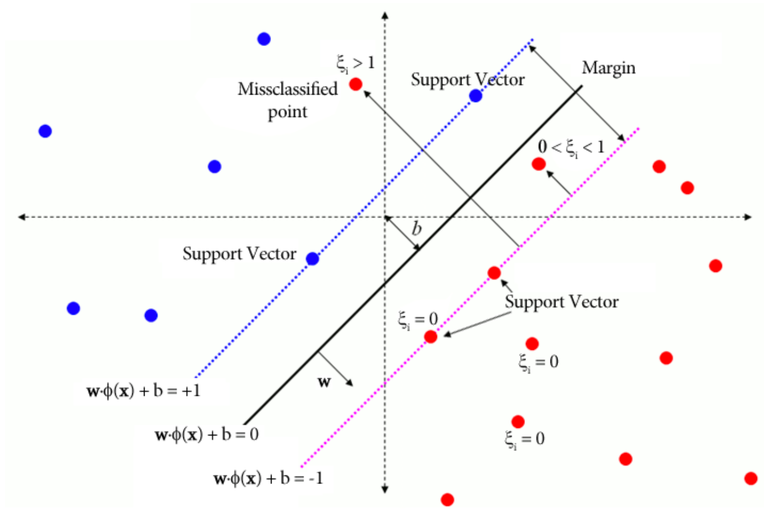


Figura 4.3: SVM with soft margin

Its dual problem is

$$\begin{cases} \max\limits_{\lambda} -\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} y^i y^j (x^i)^T x^j \lambda_i \lambda_j + \sum_{i=1}^{l} \lambda_i \\ \sum_{i=1}^{l} \lambda_i y^i = 0 \\ 0 \leq \lambda_i \leq C & i = 1, ..., l \end{cases} \tag{4.5}$$

note that is the same dual of (normal) linear SVM, but here we have an upper bound for λ. If λ^* is the optimal value for the dual, we can calculate w^* and b^* as follows:

$$w^* = \sum_{i=1}^{l} \lambda_i^* y^i x^i$$

to find b^* we have to choose i s.t. $0 < \lambda_i^* < C$ and using the complementary conditions:

$$\begin{cases} \lambda_i^*[1 - \lambda^i((w^*)^T x^i + b^*) - \xi_i^*] = 0 \\ (C - \lambda_i^*)\xi_i^* = 0 \end{cases}$$

this means that

$$b^* = \frac{1}{y^i} - (w^*)^T x^i$$

4.2 Non-Linear SVM

Consider now two sets A and B shown in figure 4.4 which are not linearly separable: is easy to notice that even using the soft margin we have a bad classification. So we need to use a different approach to this kind of problems.

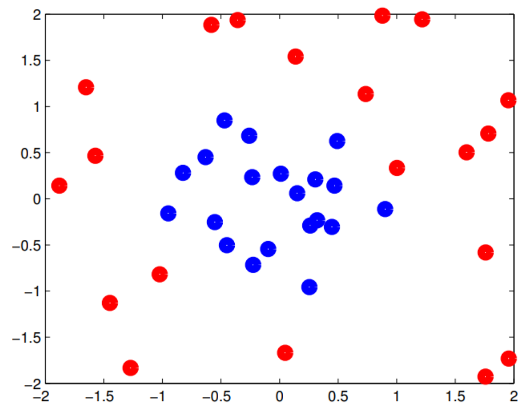


Figura 4.4: Non linearly separable sets

The idea is to map the input space in an higher dimensional (maybe infinite) space called *features space* to make the two sets linearly separable in it, finally is possible to use the SVM with soft margin to classify the points. So let $\phi : \mathbb{R}^n \to \mathcal{H}$ the map function for each input x^i, $i = 1, ..., l$, we can define:

Primal:

$$\begin{cases} \min\limits_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l} \xi_i \\ 1 - y^i(w^T\phi(x^i) + b) \leq \xi_i & i = 1, ..., l \\ \xi_i \geq 0 & i = 1, ..., l \end{cases}$$

Dual:

$$\begin{cases} \max\limits_{\lambda} -\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} y^i y^j \phi(x^i)^T\phi(x^j)\lambda_i\lambda_j + \sum_{i=1}^{l}\lambda_i \\ \sum_{i=1}^{l}\lambda_i y^i = 0 \\ 0 \leq \lambda_i \leq C \quad i = 1, ..., l \end{cases}$$

note that are the same of linear SVM with soft margin, but here we've $\phi(x)$ and not x; note also that w now is a vector in an high dimensional space (maybe infinite variables), so solving the dual is easier beacuse the number of variables is the number of training data.

Similarly to the linear svm with soft margin, we can compute λ^* solving the dual problem, then we can compute w^* as

$$w^* = \sum_{i=1}^{l} \lambda^* y^i \phi(x^i)$$

and compute b^* using any λ_i^* s.t. $0 < \lambda_i^* < C$ as

$$y^i \left[\sum_{j=1}^{l} \lambda_j^* y^j \phi(x^j)^T\phi(x^i) + b^*\right] - 1 = 0$$

and finally the decision function is

$$f(x) = \text{sign}((w^*)^T\phi(x) + b^*) = \text{sign}\left(\sum_{i=1}^{l} \lambda_i^* y^i \phi(x^i)^T\phi(x) + b^*\right)$$

note that the decision function depends on λ^* (that depends on $\phi(x^i)^T\phi(x^j)$), on $\phi(x^i)^T\phi(x^j$ and on b^* (that depends on $\phi(x^i)^T\phi(x^j)$: so we don't need to know $\phi(x)$ but only $\phi(x^i)^T\phi(y)$ to solve the dual that is in general easier to solve. To do that we use the kernel functions.

Kernel Functions A function $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is called kernel if there exists a map $\phi : \mathbb{R}^n \to \mathcal{H}$ such that

$$k(x, y) = \langle \phi(x), \phi(y) \rangle$$

where $\langle \cdot, \cdot \rangle$ is a scalar product in $\mathcal{H}$. Here some examples:

- $k(x, y) = x^T y$
- $k(x, y) = (x^T + 1)^p$, with $p \geq 1$ (polynomial)
- $k(x, y) = e^{-\gamma \|x-y\|^2}$ (Gaussian)
- $k(x, y) = tanh(\beta x^T y + \gamma)$ with suitable β and γ.

Theorem 4.2.1. *If $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is a kernel and $x^1, ..., x^l \in \mathbb{R}^n$, then the matrix K defined as follows*

$$K_{i,j} = k(x^i, x^j)$$

is positive semidefinite.

Now we can choose a kernel and apply it to the non-linear svm's dual problem definition:

$$\begin{cases} \max\limits_{\lambda} -\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} y^i y^j k(x^i, x^j)\lambda_i\lambda_j + \sum_{i=1}^{l}\lambda_i \\ \sum_{i=1}^{l}\lambda_i y^i = 0 \\ 0 \leq \lambda_i \leq C \quad i = 1, ..., l \end{cases}$$

in this way we can find the value of λ^* and compute b^* using any λ_i^* s.t. $0 < \lambda_i^* < C$ as

$$y^i \left[\sum_{j=1}^{l} \lambda_j^* y^j k(x^i, x^j) + b^* \right] - 1 = 0$$

and finally the decision function is

$$f(x) = \text{sign}\left(\sum_{i=1}^{l} \lambda_i^* y^i k(x^i, x) + b^* \right)$$

and $f(x) = 0$ is the separating surface: that is linear in the features spaces but non-linear in the input space.

Capitolo 5

Regression problems

5.1 Polynomial Regression

First of all let's introduce what Polynomial Regression means: given $x_1, ..., x_l \in \mathbb{R}$ the input points and $y_1, ..., y_l \in \mathbb{R}$ the experimental data corresponding to observations of the points.

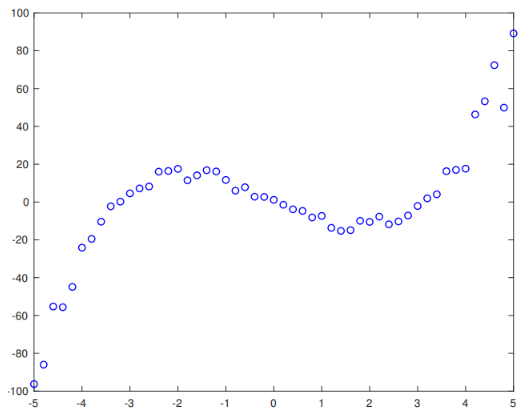


Figura 5.1: Polynomial Regression

We want to find a polynomial function p (of degree $n-1$, with $n \leq l$) that is the best approximation of experimental data. Polynomial p has coefficients $z_1, ..., z_n$:

$$p(x) = z_1 + z_2x + z_3x^2 + ... + z_nx^{n-1}$$

in this context x^i means x raised to i. So the aim is to find the best values of z_i.

The *residual* is the vector $r \in \mathbb{R}^l$ such that

$$r_i = p(x_i) - y_i \text{ with } i = 1, ..., n$$

and represent the error of our approximation: how far is the approximation to the corresponding value y_i. So we want to find the best polynomial approximation such that the residual is minimum: in other words we want to find the best coefficients z such that $\|r\|$ is minimum i.e. solving the following unconstrained problem

$$\begin{cases} \min \|Az - y\| \\ z \in \mathbb{R}^n \end{cases} \tag{5.1}$$

where

$$A = \begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ \vdots & \vdots & \ddots & \vdots & \\ 1 & x_l & x_l^2 & \cdots & x_l^{n-1} \end{pmatrix} \text{ and } y = \begin{pmatrix} y_1 \\ \vdots \\ y_l \end{pmatrix}$$

so our objective function is

$$f(z) = \|Az - y\|$$

and is convex for any norm, let's go into deep:

5.1.1 Polynomial Regression with $\| \cdot \|_2$

Reminding that the Euclidian norm is least squares approximation, we can rewrite the problem 5.1 as

$$\begin{cases} \min \frac{1}{2}\|Az - y\|_2^2 = \frac{1}{2}(Az - y)^T(Az - y) = \frac{1}{2}z^T A^T A z - z^T A^T y + \frac{1}{2}y^T y \\ z \in \mathbb{R}^n \end{cases}$$

note that is a quadratic optimization problem in z where $Q = A^T A$. Also can be proved that $rank(A) = n$, thus $A^T A$ is positive definite so the function is strongly convex and the problem has no constraints: this means that the unique optimal solution is the stationary point of the objective function given by the following system of linear equations:

$$\nabla f = A^T A z - A^T y = 0$$

$$A^T A z = A^T y$$

5.1.2 Polynomial Regression with $\|\cdot\|_1$

$$\begin{cases} \min \|Az-y\|_1 = \sum_{i=1}^{l} |A_i z - y_i| \\ z \in \mathbb{R}^n \end{cases}$$

note that is a linear optimization problem in z but the objective function is not smooth, but this is equivalent to

$$\begin{cases} \min\limits_{z,u} & \sum_{i=1}^{l} u_i \\ u_i & = |A_i z - y_i| \\ & = \max\{A_i z - y_i, y_i - A_i z\} \end{cases} \rightarrow \begin{cases} \min\limits_{z,u} & \sum_{i=1}^{l} u_i \\ u_i & \geq \max\{A_i z - y_i, y_i - A_i z\} \end{cases}$$

$$\rightarrow \begin{cases} \min\limits_{z,u} \sum_{i=1}^{l} u_i \\ u_i \geq A_i z - y_i \quad \forall i = 1, ..., l \\ u_i \geq y_i - A_i z \quad \forall i = 1, ..., l \end{cases}$$

now we have the same linear optimization problem with a smooth objective function.

5.1.3 Polynomial Regression with $\|\cdot\|_\infty$

$$\begin{cases} \min \|Az-y\|_\infty = \max\limits_{i=1,...,l} |A_i z - y_i| \\ z \in \mathbb{R}^n \end{cases}$$

for the same reason as before, we can write the problem as

$$\begin{cases} \min u \\ u = \max\limits_{i=1,...,l} |A_i z - y_i| \end{cases} \rightarrow \begin{cases} \min\limits_{z,u} u \\ u \geq A_i z - y_i \quad \forall i = 1, ..., l \\ u \geq y_i - A_i z \quad \forall i = 1, ..., l \end{cases}$$

5.2 ε-SV Regression

Another way to solve regression tasks is to use support vectors. we know that sv are classification points s.t. the corresponding dual variable is positive, so are the points needed to create the optimal separating hyperplane.

Given a set of training data $\{(x_1, y_1), ..., (x_l, y_l)\}$ where $x_i \in \mathbb{R}^n$ is the input and $y_i \in \mathbb{R}$ is the target, we want to find a function f such that:

1. has at most ε deviation from the targets y_i for all the training data: is a kind of error tolerance;

2. has to be as flat as possible to have good results on generalization.

$$\begin{cases} \min\limits_{w,b} \frac{1}{2}\|w\|^2 \\ y_i \leq w^T x_i + b + \varepsilon \quad \forall i = 1, ..., l \\ y_i \geq w^T x_i + b - \varepsilon \quad \forall i = 1, ..., l \end{cases} \tag{5.2}$$

the objective function represent the 2nd property and the constraints represent the first property. Note that we're not searching for a separating hyperplane, but for the best affine function that approximates the training points: all the training points must be inside the ε-tube (represented by the constraints) as shown in fig 5.2.

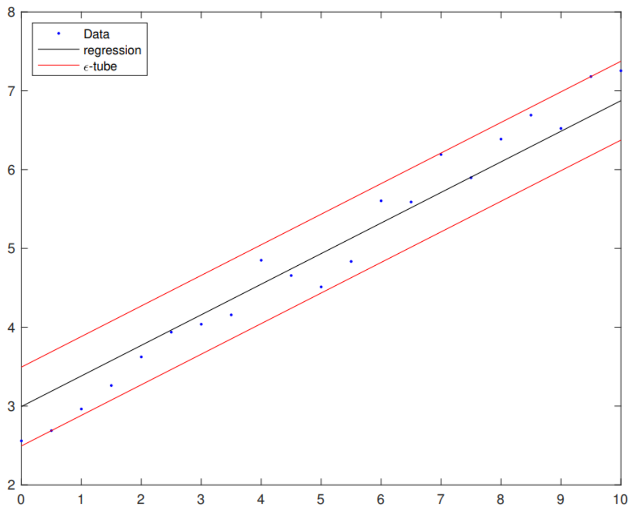


Figura 5.2

If ε is too small the model 5.2 is not feasible: as we did in the SVM with soft margin, we can relax the constraints of the problem introducing some slack variables ξ^+, ξ^- to consider also misclassified points:

$$\begin{cases} \min\limits_{w,b,\xi^+,\xi^-} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}(\xi_i^+ + \xi_i^-) \\ y_i \leq w^T x_i + b + \varepsilon + \xi_i^+ \quad \forall i = 1, ..., l \\ y_i \geq w^T x_i + b - \varepsilon - \xi_i^- \quad \forall i = 1, ..., l \\ \xi_i^+ \geq 0 \\ \xi_i^- \geq 0 \end{cases} \tag{5.3}$$

where parameter C gives the trade-off between the flatness of f and tolerance to deviations larger than ε. Note that we have 2 sets of variables ξ: one is for the upper bound of the tube and the other for the lower bound. Note also that f is quadratic respect w, linear respect to ξ and doesn't depend on b: the function is convex and the constraints are linear. This means that this is another quadratic convex problem.

Now let's define the dual form of problem 5.3. First of all we have to define the Lagrangian Function as

$$L(\underbrace{w, b, \xi^+, \xi^-}_{\text{primal var.}}, \underbrace{\lambda^+, \lambda^-, \eta^+, \eta^-}_{\text{dual var.}}) = \frac{1}{2}\|w\|^2 - w^T\left[\sum_{i=1}^{l}(\lambda_i^+ - \lambda_i^-)x_i\right] - b\sum_{i=1}^{l}(\lambda_i^+ - \lambda_i^-)$$
$$+ \sum_{i=1}^{l}\xi_i^+(C - \lambda_i^+ - \eta_i^+) + \sum_{i=1}^{l}\xi_i^-(C - \lambda_i^- - \eta_i^-)$$

note that if one of the following condition is true,

- $\sum_{i=1}^{l}(\lambda_i^+ - \lambda_i^-) \neq 0$ the minimum respect to b will be $-\infty$;
- $C - \lambda_i^+ - \eta_i^+ \neq 0$ the minimum respect to ξ^+ will be $-\infty$;
- $C - \lambda_i^- - \eta_i^- \neq 0$ the minimum respect to ξ^- will be $-\infty$;

$(\min\limits_{w,b,\xi^+,\xi^-} L = -\infty)$ because using the Lagrangian relaxation, there's not constraints. Otherwise when all of those three terms are equal to 0, we have only to minimize respect to w: so we want to minimize

$$\frac{1}{2}\|w\|^2 - w^T\left[\sum_{i=1}^{l}(\lambda_i^+ - \lambda_i^-)x_i\right]$$

that is a quadratic strongly convex function, so the minimum is given by the stationary point

$$\nabla_w L = w - \sum_{i=1}^{l}(\lambda_i^+ - \lambda_i^-)x_i = 0$$

and finally we can define the dual problem as

$$\begin{cases} \max\limits_{\lambda^+,\lambda^-} & -\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}(\lambda_i^+ - \lambda_i^-)(\lambda_j^+ - \lambda_j^-)(x_i)^T x_j \\ & -\varepsilon\sum_{i=1}^{l}(\lambda_i^+ + \lambda_i^-) + \sum_{i=1}^{l} y_i(\lambda_i^+ - \lambda_i^-) \\ \sum_{i=1}^{l} & (\lambda_i^+ - \lambda_i^-) = 0 \\ \lambda_i^+ & \in [0, C] \\ \lambda_i^- & \in [0, C] \end{cases} \tag{5.4}$$

note that now we deleted η^+ and η^- because we know that $C - \lambda_i^+ - \eta_i^+ = 0$ and $C - \lambda_i^- - \eta_i^- = 0$ so $\eta_i^+ = C - \lambda_i^+$ and $\eta_i^- = C - \lambda_i^-$. If $\lambda_i^+ > 0$ or $\lambda_i^- > 0$, then x_i is a support vector.

If (λ^+, λ^-) is a dual optimum then we can calculate w as

$$w = \sum_{i=1}^{l} (\lambda^+, \lambda^-) x_i$$

and we can calculate b using the complementarity conditions:

$$\lambda_i^+ [\varepsilon + \xi_i^+ - y_i + w^T x_i + b] = 0$$

$$\lambda_i^- [\varepsilon + \xi_i^- + y_i - w^T x_i - b] = 0$$

$$\xi_i^+ (C - \lambda_i^+) = 0$$

$$\xi_i^- (C - \lambda_i^-) = 0$$

hence if there is some i s.t. $0 < \lambda_i^+ < C$, then

$$b = y_i - w^T x_i - \varepsilon$$

or if there is some i s.t. $0 < \lambda_i^- < C$, then $b = y_i - w^T x_i + \varepsilon$

5.2.1 Non-linear ε-SV Regression

To solve non-linear regression problems we have to use the same approach used in non linear classification problems: basically we have to define a map $\phi : \mathbb{R}^n \to \mathcal{H}$ (where $\mathcal{H}$ is called features space and is an higher dimensional space) and find a linear regression for the points $\{(\phi(x_i), y_i)\}$ in the feature space $\mathcal{H}$. The primal problem is defined as

$$\begin{cases} \min\limits_{w,b,\xi^+,\xi^-} \frac{1}{2}\|w\|^2 + C \sum_{i=1}^{l} (\xi_i^+ + \xi_i^-) \\ y_i \leq w^T \phi(x_i) + b + \varepsilon + \xi_i^+ & \forall i = 1, ..., l \\ y_i \geq w^T \phi(x_i) + b - \varepsilon - \xi_i^- & \forall i = 1, ..., l \end{cases} \tag{5.5}$$

but the primal is a very "difficult" problem because w is a vector in a high dimensional space and could have infinite variables. So is easier to solve the dual problem defined as

$$\begin{cases} \max\limits_{\lambda^+,\lambda^-} & -\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}(\lambda_i^+ - \lambda_i^-)(\lambda_j^+ - \lambda_j^-)\phi(x_i)^T\phi(x_j) \\ & -\varepsilon\sum_{i=1}^{l}(\lambda_i^+ + \lambda_i^-) + \sum_{i=1}^{l} y_i(\lambda_j^+ - \lambda_j^-) \\ \sum_{i=1}^{l} & (\lambda_i^+ - \lambda_i^-) = 0 \\ \lambda_i^+, \lambda_i^- & \in [0, C] \end{cases} \tag{5.6}$$

that has only $2l$ variables (l for λ^+ and l for λ^-). As discussed before, we don't need to know the value of $\phi(x_i)$ or $\phi(x_j)$ but the scalar product $\phi(x_i)\phi(x_j)$ that can be calculated easier using the kernel functions

$$\begin{cases} \max\limits_{\lambda^+,\lambda^-} & -\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}(\lambda_i^+ - \lambda_i^-)(\lambda_j^+ - \lambda_j^-)k(x_i, x_j) \\ & -\varepsilon\sum_{i=1}^{l}(\lambda_i^+ + \lambda_i^-) + \sum_{i=1}^{l} y_i(\lambda_j^+ - \lambda_j^-) \\ \sum_{i=1}^{l} & (\lambda_i^+ - \lambda_i^-) = 0 \\ \lambda_i^+, \lambda_i^- & \in [0, C] \end{cases} \tag{5.7}$$

In summary we have to

- chose a kernel k
- solve the dual problem to find (λ^+, λ^-)
- compute b as

  $$\begin{cases} b = y_i - \varepsilon - \sum_{j=1}^{l}(\lambda_j^+ - \lambda_j^-)k(x_i, x_j) & \text{for some } i \text{ s.t. } 0 < \lambda_i^+ < 0 \\ b = y_i + \varepsilon - \sum_{j=1}^{l}(\lambda_j^+ - \lambda_j^-)k(x_i, x_j) & \text{for some } i \text{ s.t. } 0 < \lambda_i^- < 0 \end{cases}$$

  in some sense we are also computing w here, but we don't care about it's value
- calculate the Recession function f as

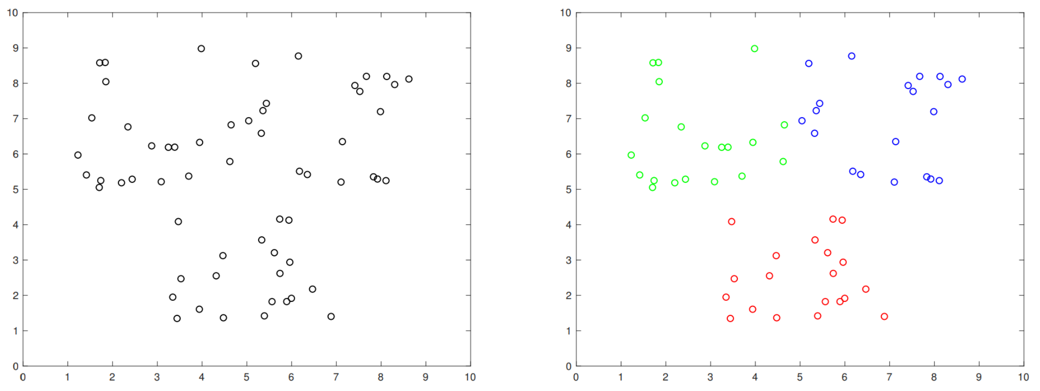
  $$f(x) = \sum_{i=1}^{l}(\lambda^+ - \lambda^-)k(x_i, x) + b$$

  that is *linear* in the features space and *non-linear* in the input space.

Capitolo 6

Clustering problems

Clustering problems are focused on the partition of a set S in k subsets $S_1, ..., S_k$ (called clusters) that are homogeneous and well separated. These kind of problems are used in unsupervisioned machine learning tasks, where datas has no labels and so the model have to find by itself some properties that allow to classify consistently the data.



First of all let's try to write this kind of problem as an optimization problem: assume that patterns are vectors $p_1, ..., p_l \in \mathbb{R}^n$ (dataset) and consider the function $d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ as the distance function between two vectors, for example $d(x, y) = \|x-y\|_2^2$ or $d(x, y) = \|x-y\|_1$. For each cluster S_j introduce a *centroid* $x_j \in \mathbb{R}^n$ (that for the moment is unknown and potentially not included in the dataset) and define the clusters such that each pattern is associated to the closest centroid. So we want to find k centroids in order to minimize the sum of the distances between each pattern and the closest centroid:

$$\begin{cases} \min \sum_{i=1}^{l} \min\limits_{j=1,\dots,k} & d(p_i, x_j) \\ x_j \in \mathbb{R}^n & \forall j = 1, ..., k \end{cases}$$

6.1 Optimization model using the $\|\cdot\|_2$ as distance

Let's consider the distance $d(x, y) = \|x - y\|_2^2$. The optimization problem to solve is

$$\begin{cases} \min\limits_{x} \sum_{i=1}^{l} \min\limits_{j=1,\dots,k} & \|p_i - x_j\|_2^2 \\ x_j \in \mathbb{R}^n & \forall j = 1, ..., k \end{cases} \tag{6.1}$$

- if $k = 1$ (so we have only one cluster), the problem is a convex quadratic programming problem without constraints:

  $$\begin{cases} \min\limits_{x} \sum_{i=1}^{l} \|p_i - x\|_2^2 & = \sum_{i=1}^{l} (x - p_i)^T (x - p_i) \\ x \in \mathbb{R}^n & \forall j = 1, ..., k \end{cases} \tag{6.2}$$

  and the global optimum is given by the stationary point

  $$2lx - 2\sum_{i=1}^{l} p_i = 0 \Leftrightarrow x = \frac{\sum_{i=1}^{l} p_i}{l}$$

- if $k > 1$ (so we have at least two clusters), the problem 6.1 is *non-convex* and *non-smooth* (and also *non-quadratic*), but can be written as the following *non-convex* but smooth problem:

  $$\begin{cases} \min\limits_{x,\alpha} \sum_{i=1}^{l} \sum_{j=1}^{k} \alpha_{ij} \|p_i - x_j\|_2^2 & \\ \sum_{j=1}^{k} \alpha_{ij} = 1 & \forall i = 1, ..., l \\ \alpha_{ij} \geq 0 & \forall i = 1, ..., l,\ j = 1, ..., k \\ x_j \in \mathbb{R}^n & \forall j = 1, ..., k \end{cases} \tag{6.3}$$

  Proof: notice that $\min\limits_{j=1,\dots k} \{a_j\} = \min \left\{ \sum_{j=1}^{k} \alpha_j a_j \ : \ \sum_{j=1}^{k} \alpha_j = 1, \alpha \geq 0 \right\}$

6.2 k-means algorithm

Since the problem 6.3 is difficult to solve, we can fix one variable at time to solve the problem using the k-means algorithm.

- If x_j (the centroid) are fixed, the norm $\|p_i - x_j\|_2$ is a parameter (because we know both values) so we want to minimize a linear function, hence the problem 6.3 can be decomposed into l very simple linear problems (for any $i = 1, ..., l$): so the optimal solution is [step by step on iPad]

$$\alpha^*_{i,j} = \begin{cases} 1 & \text{if } j \text{ is the first index s.t. } \|p_i - x_j\|_2 = \min_{h=1,\dots,k} \|p_i - x_h\|_2 \\ 0 & \text{otherwise} \end{cases}$$

  in other words if $\alpha_{ij} = 1$ the pattern p_i belongs to cluster j.

- If $\alpha_{i,j}$ is fixed, the problem can be decomposed in k very simple convex quadratic problems similarly to 6.2. So we consider only one cluster at time, hence for any $j = 1, ..., k$ the optimal solution is [step by step on iPad]

$$x^*_j = \frac{\sum_{i=1}^l \alpha_{i,j} p_i}{\sum_{i=1}^l \alpha_{i,j}} = \text{mean}(p_i \ : \alpha_{ij} = 1)$$

  the mean of the patterns that belong to cluster j.

The idea of k-means algorithm is to alternate the minimization of $f(x, \alpha) = \sum_{i=1}^l \sum_{j=1}^k \alpha_{i,j} \|p_i - x_j\|_2^2$ with respect the two blocks of variables x and α (fist we update α, then we update x):

1. (Initialization) set $t = 0$ (iteration index) and choose centroids $x^0_1, ..., x^0_k \in \mathbb{R}^n$ and assign patterns to clusters: for any $i = 1, ..., l$

$$\alpha^0_{i,j} = \begin{cases} 1 & \text{if } j \text{ is the first index s.t. } \|p_i - x^0_j\|_2 = \min_{h=1,\dots,k} \|p_i - x^0_h\|_2 \\ 0 & \text{otherwise} \end{cases}$$

   note that the centroids x_j have been already fixed;

2. (Update centroids) for each $j = 1, ..., l$ compute the mean

$$x^{t+1}_j = \frac{\sum_{i=1}^l \alpha^t_{i,j} p_i}{\sum_{i=1}^l \alpha^t_{i,j}}$$

   like the previous step note that we're updating only the centroids x_j because the α_{ij} have been fixed at the previous step.

3. (Update clusters) for any $i = 1, ..., l$ compute

$$\alpha_{i,j}^{t+1} = \begin{cases} 1 & \text{if } j \text{ is the first index s.t. } \|p_i - x_j^{t+1}\|_2 = \min\limits_{h=1,...,k} \|p_i - x_h^{t+1}\|_2 \\ 0 & \text{otherwise} \end{cases}$$

   is the same of step 1;

4. if $f(x^{t+1}, \alpha^{t+1}) = f(x^t, \alpha^t)$ then STOP; otherwise $t = t + 1$ and go to Step 2.

This algorithm stops after a finite number of iterations at a solution (x^*, α^*) of the KKT system of the problem 6.3 such that

$$\begin{aligned} f(x^*, \alpha^*) &\leq f(x^*, \alpha) & \forall \alpha \geq 0 \text{ s.t. } \sum_{j=1}^{k} \alpha_{i,j} = 1 \forall i = 1, ..., l \\ f(x^*, \alpha^*) &\leq f(x, \alpha^*) & \forall x \in \mathbb{R}^{kn} \\ f(x^*, \alpha^*) &\nleq f(x, \alpha) & \forall (x, \alpha) \text{ feasible} \end{aligned}$$

Note that the k-means algorithm *doesn't guarantee to find a global optimum*, but a "partial minimum": it find a minimum of f respect to α when x^* is fixed and a minimum of f respect to x when α^* is fixed, but not the minimum of f respect both x and α.

6.3 Optimization model using the $\|\cdot\|_1$ as distance

Now let's consider the distance $d(x, y) = \|x - y\|_1$. The optimization problem to solve is

$$\begin{cases} \min \sum_{i=1}^{l} \min\limits_{j=1,...,k} & \|p_i - x_j\|_1 \\ x_j \in \mathbb{R}^n & \forall j = 1, ..., k \end{cases} \tag{6.4}$$

- if $k = 1$ (so we have only one cluster), the problem is a convex programming problem without constraints that can be decomposed into n convex problems of one variable (remember that n is the input dimension):

$$\begin{cases} \min\limits_{x} \sum_{i=1}^{l} \|p_i - x\|_1 = \sum_{i=1}^{l} \sum_{h=1}^{n} |x_h - (p_i)_h| = \sum_{h=1}^{n} \underbrace{\sum_{i=1}^{l} |x_h - (p_i)_h|}_{f_h(x_h)} \\ x \in \mathbb{R}^n \end{cases} \tag{6.5}$$

Now let's consider one of the n "sub-problems": if we fix h we can assign $a_i = (p_i)_h$ such that $a_1 < a_2 < ... < a_l$ and re-write the previous problem as the following unconstrained minimization problem

$$\begin{cases} \min \ \sum_{i=1}^{l} |x - a_i| = f(x) \\ x \in \mathbb{R} \end{cases}$$

and now the objective function f is given by the following $l + 1$ cases [steps on iPad]:

$$f(x) \begin{cases} -lx + \sum_{i=1}^{l} a_i & \text{if } x < a_1 \\ (2-l)x + \sum_{i=2}^{l} a_i - a_1 & \text{if } x \in [a_1, a_2] \\ ... & ... \\ (2r-l)x + \sum_{i=r+1}^{l} a_i - \sum_{i=1}^{r} a_i & \text{if } x \in [a_r, a_{r+1}] \\ ... & ... \\ (l-2)x + a_l - \sum_{i=1}^{l-1} a_i & \text{if } x \in [a_{l-1}, a_l] \\ lx - \sum_{i=1}^{l} a_i & \text{if } x > a_l \end{cases}$$

and in the end we can calculate the minimum value of f as the median of $a_1, ..., a_l$:

$$\text{median}(a_1, ..., a_l) = \begin{cases} a_{(l+1)/2} & \text{if } l \text{ is odd} \\ \frac{1}{2}(a_{l/2} + a_{l/2+1}) & \text{if } l \text{ is even} \end{cases}$$

remember that this is only one of the n sub-problems, so we have to compute the median n times.

- if $k > 1$ (we have at least two clusters), the problem 6.4 is *non-convex* and *non-smooth* (and also *non-quadratic*), but can be written as the following problem (like we did using the $\| \cdot \|_2$):

$$\begin{cases} \min\limits_{x,\alpha} \ \sum_{i=1}^{l} \sum_{j=1}^{k} \alpha_{i,j} \|p_i - x_j\|_1 \\ \sum_{j=1}^{k} \alpha_{i,j} = 1 & \forall i = 1, ..., l \\ \alpha_{i,j} \geq 0 & \forall i = 1, ..., l, \ j = 1, ..., k \\ x_j \in \mathbb{R}^n & \forall j = 1, ..., k \end{cases} \tag{6.6}$$

and, just to know, the problem 6.6 is equivalent to the non-convex bilinear problem:

$$\begin{cases} \min\limits_{x,\alpha,u} \sum_{i=1}^{l} \sum_{j=1}^{k} \sum_{h=1}^{n} \alpha_{i,j} u_{i,j,h} & \\ u_{i,j,h} \geq (p_i)_h - (x_j)_h & \forall i = 1, ..., l \; j = 1, ..., k \; h = 1, ..., n \\ u_{i,j,h} \geq (x_j)_h - (p_i)_h & \forall i = 1, ..., l \; j = 1, ..., k \; h = 1, ..., n \\ \sum_{j=1}^{k} \alpha_{i,j} = 1 & \forall i = 1, ..., l \\ \alpha_{i,j} \geq 0 & \forall i = 1, ..., l \; j = 1, ..., k \\ x_j \in \mathbb{R}^n & \forall j = 1, ..., k \end{cases}$$

6.4 k-median algorithm

The k-median algorithm is based on the following properties of problem 6.6: as the k-means algorithm we fix one variable at time to update the other.

- if x_j (the centroid) are fixed, the problem 6.6 can be decomposed into l very simple linear problems (for any $i = 1, ..., l$): so the optimal solution is

$$\alpha^*_{i,j} = \begin{cases} 1 & \text{if } j \text{ is the first index s.t. } \|p_i - x_j\|_1 = \min\limits_{h=1,...,k} \|p_i - x_h\|_1 \\ 0 & \text{otherwise} \end{cases}$$

  in other words if $\alpha_{ij} = 1$ the pattern p_i belongs to cluster j.

- if $\alpha_{i,j}$ are fixed, the problem can be decomposed in k very simple convex problems similarly to 6.5. So we consider only one cluster at time, hence for any $j = 1, ..., k$ the optimal solution is

$$x^*_j = \text{median}(p_i : \alpha_{i,j} = 1)$$

  the median of the patterns that belong to cluster j.

The idea of k-median algorithm is the same of the k-means algorithm: to alternate the minimization of $f(x, \alpha) = \sum_{i=1}^{l} \sum_{j=1}^{k} \alpha_{i,j} \|p_i - x_j\|_1$ with respect the two blocks of variables x and α (fist we update α, then we update x):

1. (Initialization) set $t = 0$ (iteration index) and choose centroids $x^0_1, ..., x^0_k \in \mathbb{R}^n$ and assign patterns to clusters: for any $i = 1, ..., l$

$$\alpha^0_{i,j} = \begin{cases} 1 & \text{if } j \text{ is the first index s.t. } \|p_i - x^0_j\|_1 = \min\limits_{h=1,...,k} \|p_i - x^0_h\|_1 \\ 0 & \text{otherwise} \end{cases}$$

2. (Update centroids) for each $j = 1, ..., l$ compute the median

$$x_j^{t+1} = \text{median}(p_i : \alpha_{i,j}^t = 1)$$

3. (Update clusters) for any $i = 1, ..., l$ compute

$$\alpha_{i,j}^{t+1} = \begin{cases} 1 & \text{if } j \text{ is the first index s.t. } \|p_i - x_j^{t+1}\|_1 = \min_{h=1,...,k} \|p_i - x_h^{t+1}\|_1 \\ 0 & \text{otherwise} \end{cases}$$

4. if $f(x^{t+1}, \alpha^{t+1}) = f(x^t, \alpha^t)$ then STOP; otherwise $t = t + 1$ and repeat from Step 2.

This algorithm stops after a finite number of iterations at a stationary point (x^*, α^*) of the problem 6.6 such that

$$\begin{aligned} f(x^*, \alpha^*) &\le f(x^*, \alpha) & \forall \alpha \ge 0 \text{ s.t. } \sum_{j=1}^{k} \alpha_{i,j} = 1 \forall i = 1, ..., l \\ f(x^*, \alpha^*) &\le f(x, \alpha^*) & \forall x \in \mathbb{R}^{kn} \\ f(x^*, \alpha^*) &\not\le f(x, \alpha) & \forall (x, \alpha) \text{ feasible} \end{aligned}$$

Note that the k-median algorithm *doesn't guarantee to find a global optimum*, but a "partial minimum": it find a minimum of f respect to α when x^* is fixed and a minimum of f respect to x when α^* is fixed, but not the minimum of f respect both x and α.

Capitolo 7

Solution methods for unconstrained optimization problems

In this chapter we discuss methods for solving the unconstrained optimization problem where f is non-linear and smooth enough (continuously differentiable for the first two methods and twice continuously differentiable for the Newton method). In some special cases, for example when f is convex, we can find a solution to the minimization problem by solving analytically $\nabla f(x) = 0$, but usually the problem must be solved by an iterative algorithm. By this we mean an algorithm that computes a sequence of points $x^0, x^1, ... \in \operatorname{dom} f$ (called minimization sequence) with $f(x^k) \to p^*$ as $k \to \infty$. The algorithm is terminated when $f(x^k) - p^* \leq \varepsilon$ where $\varepsilon > 0$ is some specified tolerance.

7.1 Gradient Method

This is the basic approach based on first derivatives. The idea is to start from a point x^0 and search for the direction $d \in \mathbb{R}^n$ such that $\nabla f(x^k)^T d < 0$ has the minimum value (the steepest descent): this direction is given when the scalar product is minimum, so when the direction is the opposite of the gradient because

$$d^k = -\nabla f(x^k)$$

From this idea we can write an iterative method that search the minimum along the direction d and stops only when we reach a stationary point:

```
Choose $x^0 \in \mathbb{R}^n$, set $k = 0$
while $\nabla f(x^k) \neq 0$ do
    [search direction] $d^k = -\nabla f(x^k)$
    [step size] compute an optimal solution $t_k$ of the problem: $\min_{t>0} f(x^k + t\, d^k)$
    $x^{k+1} = x^k + t_k\, d^k$, $k = k + 1$
end
```

note that now we want to minimize f restrict to the half-line given by the direction d, so we're not minimizing anymore a function with n variables but a function with only one variable (t, the step). Note that this method can be very useful if the function is convex, because allow us to find the global minimum.

If f is a quadratic function $\frac{1}{2}x^TQx + c^Tx$, with Q positive definite matrix, the step size is equal to

$$t_k = -\frac{(g^k)^T d^k}{(d^k)^T Q d^k}$$

where $g^k = \nabla f(x^k)$. Explanation: $f(x^k + td^k)$ is strongly convex with respect to t, so $\min_t f(x^k + td^k) \Leftrightarrow \nabla f(x^k + td^k) = 0$ [step by step on iPad]. Now let's see some properties of this method:

- $(d^k)^T d^{k+1} = 0$ for any iteration k, this means that the two directions are orthogonal. This means that the generated sequence has a zig-zag behaviour [proof on iPad];
- if $\{x^k\}$ converges to x^*, then $\nabla f(x^*) = 0$;
- if f is coercive, then for any starting point x^0 the generated sequence $\{x^k\}$ is bounded and any of its cluster points (limit points of the subsequence) is a *stationary point* of f.
- if f is coercive and convex, then for any starting point x^0 the generated sequence $\{x^k\}$ is bounded and any of its cluster points is a *global minimum* of f;
- if f is strongly convex, then for any starting point x^0 the generated sequence $\{x^k\}$ converges to the *global minimum* of f. Because strongly convex functions are always coercive.

Theorem 7.1.1 (Error Bound). *If $f(x) = \frac{1}{2}x^TQx + c^Tx$, with Q positive definite matrix and x^* is the global minimum of f, then the sequence $\{x^k\}$*

satisfies the following inequality:

$$\|x^{k+1} - x^*\|_Q \leq \left(\frac{\frac{\lambda_n}{\lambda_1} - 1}{\frac{\lambda_n}{\lambda_1} + 1} \right) \|x^k - x^*\|_Q$$

$\forall k \geq 0$, where $\|x\|_Q = \sqrt{x^T Q x}$ and $0 < \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$ are the eigenvalues of Q.

in practice the error at the iteration $k+1$ has to be less or equal than the error at the previous iteration multiplied by the ratio (between 0 and 1). So this theorem gives us an upper bound of the error and tell us how "fast" we're moving to the minimum: the bigger the ratio $\frac{\frac{\lambda_n}{\lambda_1}-1}{\frac{\lambda_n}{\lambda_1}+1}$ (near to 1), the slowest the convergence is.

When f is not a quadratic function, the step size is computed in a different (and expensive) way than a quadratic function with the Armijo inexact line search as shown below [step by step on iPad]:

```
Set $\alpha, \gamma \in (0,1)$ and $\bar{t} > 0$. Choose $x^0 \in \mathbb{R}^n$, set $k = 0$.

while $\nabla f(x^k) \neq 0$ do
    $d^k = -\nabla f(x^k)$
    $t_k = \bar{t}$
    while $f(x^k + t_k d^k) > f(x^k) + \alpha t_k (d^k)^T \nabla f(x^k)$ do
        $t_k = \gamma t_k$
    end
    $x^{k+1} = x^k + t_k d$, $k = k + 1$
end
```

Theorem 7.1.2. *If f is coercive, then for any starting point x_0 the generated sequence $\{x^k\}$ is bounded and any of its cluster points is a stationary point of f.*

7.2 Conjugate Gradient Method

This approach is based on first derivatives as the previous but it avoid the zig-zag behaviour of the previous one. To do this we have to change the way we calculate the search direction.

First of all consider the quadratic case $f(x) = \frac{1}{2}x^T Q x + c^T x$ where Q is positive definite and set

$$g = \nabla f(x) = Qx + c$$

At iteration k, the search direction involves the gradient computed at the current iteration and the direction computed at the previous iteration:

$$d^k = \begin{cases} -g^0 & \text{if } k = 0 \\ -g^k + \beta_k d^{k-1} & \text{if } k \geq 1 \end{cases}$$

where β_k is such that d^k and d^{k-1} are *conjugate* with respect to Q, i.e. $(d^k)^T Q d^{k-1}$, so we can easily calculate β_k replacing d^k with its value [step by step on iPad].

$$b_k = \frac{(g^k)^T Q d^{k-1}}{(d^{k-1})^T Q d^{k-1}} \text{ or equivalently } = \frac{\|g^k\|^2}{\|g^{k-1}\|^2}$$

and the step size given by the exact line search is the same of the previous method (because the only thing that changes is the direction)

$$t_k = -\frac{(g^k)^T d^k}{(d^k)^T Q d^k} \text{ or equivalently } = \frac{\|g^k\|^2}{(d^k)^T Q d^k}$$

and finally we can define the pseudo-algorithm of this method

```
Choose $x^0 \in \mathbb{R}^n$, set $g^0 = Q\,x^0 + c$, $k := 0$
while $g^k \neq 0$ do
    if $k = 0$ then $d^k = -g^k$
    else $\beta_k = \frac{(g^k)^T Q\, d^{k-1}}{(d^{k-1})^T Q\, d^{k-1}}$,    $d^k = -g^k + \beta_k\, d^{k-1}$
    end
    $t_k = -\frac{(g^k)^T d^k}{(d^k)^T Q\, d^k}$
    $x^{k+1} = x^k + t_k\, d^k$, $g^{k+1} = Q\, x^{k+1} + c$, $k = k + 1$
end
```

in general the method converges in at most n iterations (where n is the input dimension) but if Q has r distinct eigenvalues, then the method converges in at most r iterations. If this doesn't happen after k iterations (with $k > n$ or $k > r$), then the gradients $\{g^0, g^1, ..., g^k\}$ are orthogonal (g^2 is orthogonal not only to g^1 but also to the previous g^0 and so on) and the directions $\{d^0, d^1, ..., d^k\}$ are conjugate with respect to Q, so x^k is the minimum of f on $x^0 + Span(d^0, d^1, ..., d^k)$ (where $Span(v^1, ..., v^k) = \{c_1 v^1 + ... + c_k v^k \; | \forall c_i \in \mathbb{R}\}$).

Theorem 7.2.1 (Error Bound)**.** *If $0 < \lambda_1 \leq ... \leq \lambda_n$ are the eigenvalues of Q, then the following bounds hold:*

$$\|x^k - x^*\|_Q \leq 2 \left(\frac{\sqrt{\frac{\lambda_n}{\lambda_1}} - 1}{\sqrt{\frac{\lambda_n}{\lambda_1}} + 1} \right)^k \|x^0 - x^*\|_Q$$

$$\|x^k - x^*\|_Q \leq \left(\frac{\lambda_{n-k+1} - \lambda_1}{\lambda_{n-k+1} + \lambda_1} \right) \|x^0 - x^*\|_Q$$

for all $k \geq 0$, where $\|x\|_Q = \sqrt{x^T Q x}$. Note that the first error bound depends only on the lowest and highest eigenvalues, but the second error bound depends on all eigenvalues.

Until now we have considered standard quadratic problems, the algorithm for general non-linear functions is

```
Choose $x^0 \in \mathbb{R}^n$, set $k := 0$

while $\nabla f(x^k) \neq 0$ do
    if $k = 0$ then $d^k = -\nabla f(x^k)$
    else $\beta_k = \frac{\|\nabla f(x^k)\|^2}{\|\nabla f(x^{k-1})\|^2}$,    $d^k = -\nabla f(x^k) + \beta_k \, d^{k-1}$
    end
    Compute the step size $t_k$
    $x^{k+1} = x^k + t_k \, d^k$, $k = k + 1$
end
```

note that the computation of the step size t_k has not an explicit formula:

- if f is quadratic we can consider the same formula as above that use the exact linear search: if t_k is computed by exact linear search, then d^k is a descent direction
- if f is non-quadratic we have to use the inexact linear search: if t_k satisfies the following conditions:

$$\begin{cases} f(x^k + t_k d^k) & \leq f(x^k) + \alpha t_k \nabla f(x^k)^T d^k \\ |\nabla f(x^k + t_k d^k)^T d^k| & \leq -\beta \nabla f(x^k)^T d^k \end{cases}$$

  with $0 < \alpha < \beta < 1/2$, then d^k is a descent direction. Note that the first condition is the Armijo inexact line search and the second condition means that the value of t we're searching for is not too close to 0 [graphical representation on iPad].

Theorem 7.2.2. *If f is coercive, then the conjugate gradient method, where t_k satisfies both condition of inexact linear search, generate a sequence $\{x^k\}$ such that*

$$\lim_{k\to\infty} \textit{inf}\, \|\nabla f(x^k)\| = 0$$

this guarantees that the sequence converges to a stationary point.

7.3 Newton Methods

As we did before we want to find a stationary point $\nabla f(x) = 0$ but now using a different approach that involves second order derivatives. Using the Taylor second order approximation we can write

$$\nabla f(x) \simeq \nabla f(x^k) + \nabla^2 f(x^k)(x - x^k)$$

or equivalently we can "shift of one derivative"

$$f(x) \simeq f(x^k) + (x - x^k)^T \nabla f(x^k) + \frac{1}{2}(x - x^k)^T \nabla^2 f(x^k)(x - x^k)$$

and find the stationary point x^{k+1} solving the linear system

$$\nabla f(x^k) + \nabla^2 f(x^k)(x - x^k) = 0$$

```
Choose $x^0 \in \mathbb{R}^n$, set $k = 0$
while $\nabla f(x^k) \neq 0$ do
    Solve the linear system $\nabla^2 f(x^k) d^k = -\nabla f(x^k)$
    $x^{k+1} = x^k + d^k$, $k = k + 1$
end
```

Theorem 7.3.1 (Convergence). *If x^* is a local minimum of f and $\nabla^2 f(x^*)$ is positive definite, then there exists $\delta > 0$ such that for any $x^0 \in B(x^*, \delta)$ the sequence $\{x^k\}$ converges to x^* and (converges with the following "speed")*

$$\|x^{k+1} - x^*\| \leq C\|x^k - x^*\|^2$$

for all $k > \bar{k}$ and for some $C > 0$ and $\bar{k} > 0$. This means that the convergence is very fast if the starting point x^0 is close enough to the solution x^*. Note that at each iteration we need to compute both the gradient $\nabla f(x^k)$ and the hessian matrix $\nabla^2 f(x^k)$, so it is very expensive as the number of iteration increases. Be careful because this theorem is about local convergence: if x^0 is too far from the optimum x^*, then the generated sequence can be not convergent to x^*.

How to avoid these two drawbacks?

7.3.1 Newton method with line search

To avoid the problem given by the local convergence, we can use the Armijo line search: if f strongly convex, we can combine the direction given by the Newton method and the step size given by the Armijo line search.

```
Set $\alpha, \gamma \in (0,1)$, $\bar{t} > 0$. Choose $x^0 \in \mathbb{R}^n$, set $k = 0$
while $\nabla f(x^k) \neq 0$ do
    [search direction] Solve the linear system $\nabla^2 f(x^k) d^k = -\nabla f(x^k)$
    $t_k = \bar{t}$
    while $f(x^k + t_k d^k) > f(x^k) + \alpha\, t_k\, (d^k)^\mathsf{T} \nabla f(x^k)$ do
        $t_k = \gamma t_k$
    end
    $x^{k+1} = x^k + t_k d^k$, $k = k + 1$
end
```

This approach is better than the simple Newton approach because if f is convex, the sequence $\{x^k\}$ converges to the global minimum of f for any starting point $x^0 \in \mathbb{R}^n$ and not anymore $x^0 \in B(x^0, \delta)$. Moreover if $\alpha \in (0, 1/2)$ and $\bar{t} = 1$, the convergence is quadratic.

7.3.2 Quasi Newton Method

To avoid the problem given by the computational cost of $\nabla^2 f(x^k)$ we can use the so called quasi-newton method. The idea is to approximate $[\nabla^2 f(x^k)]^{-1}$ with a positive definite matrix H_k at each iteration.

```
Choose $x^0 \in \mathbb{R}^n$, a positive definite matrix $H_0$, $k = 0$

while $\nabla f(x^k) \neq 0$ do
    $d^k = -H_k \nabla f(x^k)$
    Compute step size $t_k$
    $x^{k+1} = x^k + t_k d^k$
    update $H_{k+1}$
    $k = k + 1$
end
```

Now let's see how to update H. Starting from the second order Taylor approximation of ∇f

$$\nabla f(x^k) \simeq \nabla f(x^{k+1}) + \nabla^2 f(x^{k+1})(x^k - x^{k+1})$$

we set $p^k = x^{k+1} - x^k$ and $g^k = \nabla f(x^{k+1}) - \nabla f(x^k)$, so we have

$$\nabla^2 f(x^{k+1})p^k \simeq g^k \Rightarrow [\nabla^2 f(x^{k+1})]^{-1} g^k \simeq p^k$$

In conclusion we have to choose H_{k+1} such that $H_{k+1} g^k = p^k$:

$$H_{k+1} = H_k + \frac{p^k (p^k)^T}{(p^k)^T g^k} - \frac{H_k g^k (g^k)^T H_k}{(g^k)^T H_k g^k}$$

known as Davidon-Fletcher-Powell (DFP) method.

Another approach is the Broyden–Fletcher–Goldfarb–Shanno (BFGS) method that consist to find a matrix $B_k = (H_k)^{-1}$ that approximates $\nabla^2 f(x^k)$. Since $\nabla^2 f(x^{k+1})p^k \simeq g^k$ we impose that $B_{k+1} p^k = g^k$, so we can update B_k and finally H_k as

$$H_{k+1} = H_k + \left(1 + \frac{(g^k)^T H_k g^k}{(p^k)^T g^k}\right) \frac{p^k (p^k)^T}{(p^k)^T g^k} - \frac{p^k (g^k)^T H_k + H_k g^k (p^k)^T}{(p^k)^T g^k}$$

7.4 Derivative-Free Methods

There are situations where derivatives of the objective function don't exist or are computationally expensive, so a different approach is to search for the solution without using derivatives.

Def 7.4.1. A positive basis is a set of vectors $\{v^1, ..., v^p\} \subset \mathbb{R}^n$ such that:

- each vector $x \in \mathbb{R}^n$ is a conic combination of the basis $v^1, ..., v^p$: this means that is a linear combination with non-negative coefficients;

- each vector v^i is not a linear combination of the other vectors $v^1, ..., v^p$.

If $\{v^1, ..., v^p\}$ is a positive basis, then for any $w \in \mathbb{R}^n/\{0\}$ there is $i \in \{1, ..., p\}$ such that $w^T v^i < 0$. The key point is to imagine $w = \nabla f(x)$, and the vector v^i is the direction: so if we have $v^T \nabla f < 0$, v is a descent direction. So the search direction are the vectors of the positive basis and knowing that at least one of them is a descent direction, we can use it to find the minimum of f. Below the directional direct-search method

```
Choose starting point $x^0 \in \mathbb{R}^n$, step size $t_0 > 0$, $\beta \in (0,1)$, tolerance $\varepsilon > 0$ and a
positive basis $D$. Set $k = 0$.

while $t_k > \varepsilon$ do
    Order the poll set $\{x^k + t_k d, \quad d \in D\}$
    Evaluate $f$ at the poll points following the chosen order
    If there is a poll point s.t. $f(x^k + t_k d) < f(x^k)$
        then $x^{k+1} = x^k + t_k d$, $t_{k+1} = t_k$ (successful iteration)
    else $x^{k+1} = x^k$, $t_{k+1} = \beta t_k$ (step size reduction)
    end
    $k = k + 1$
end
```

Theorem 7.4.1 (Convergence). *Assuming that all the vectors of the positive basis $D \in \mathbb{Z}^n$, if f is coercive and continuously differentiable, then the generated sequence $\{x^k\}$ has a cluster point x^* such that $\nabla f(x^*) = 0$.*

Note that f continuously differentiable doesn't mean that we can compute the derivatives. Here two properties of the theorem:

- if we consider the "sufficient" iteration condition $f(x^{k+1}) \leq f(x^k) - t_k^2$, we can remove the assumption that vectors in D are in $\mathbb{Z}^n$.

- if we compute $f(x^{k+1}) \leq f(x^k + t_k d)$ for all $d \in D$, then any cluster point of $\{x^k\}$ is a stationary point of f and $\lim_{k\to\infty} \|\nabla f(x^k)\| = 0$.

A variant of the directional direct-search method consist in choosing a set of positive bases as shown below:

```
Choose starting point $x^0 \in \mathbb{R}^n$, step size $t_0 > 0$, $\beta \in (0,1)$, tolerance $\varepsilon > 0$, $\gamma \geq 1$
and a set of positive bases $\mathcal{D}$. Set $k = 0$.

while $t_k > \varepsilon$ do
    Choose a positive basis $D \in \mathcal{D}$
    Order the poll set $\{x^k + t_k d, \quad d \in D\}$
    Evaluate $f$ at the poll points following the chosen order
    If there is a poll point s.t. $f(x^k + t_k d) < f(x^k)$
        then $x^{k+1} = x^k + t_k d$, $t_{k+1} = \gamma t_k$ (successful iteration)
    else $x^{k+1} = x^k$, $t_{k+1} = \beta t_k$ (step size reduction)
    end
    $k = k + 1$
end
```

note that in a successful iteration, we're increasing the step size.

Capitolo 8

Solution methods for constrained optimization problems

In this chapter we'll discuss about three approaches: each one of them is based to solve at each iteration a (simpler) unconstrained problem that is equivalent to the constrained original problem.

8.1 Active-Set Method

Let's consider the following constrained optimization problem with linear equality constraints

$$\begin{cases} \min\ f(x) \\ Ax = b \end{cases}$$

where f is a strongly convex function and twice continuously differentiable and A is a $p \times n$ matrix with $\text{rank}(A) = p$ (the rows of the matrix are linear independent). The idea is to formulate an equivalent unconstrained problem: write $A = (A_B, A_N)$ where A_B is a $p \times p$ matrix with $\det(A_B) \neq 0$, then $Ax = b$ can be written as

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

thus

$$\begin{cases} \min\ f(x) \\ Ax = b \end{cases} \quad \to \quad \begin{cases} \min\ f(A_B^{-1}(b - A_N x_N), x_N) \\ x_N \in \mathbb{R}^{n-p} \end{cases}$$

note that now we have an unconstrained problem with $n-p$ variables. [Example on iPad].

Now consider the following quadratic constrained optimization problem with linear inequalities constraints

$$\begin{cases} \min \frac{1}{2}x^TQx + c^Tx \\ Ax \leq b \end{cases}$$

where Q is positive definite and, for any feasible point x, the vectors $\{A_i : A_ix = b_i\}$ are linearly independent. Note that for linear equality the vectors $\{A_i : A_ix = b_i\}$ are equal to the gradient $\nabla f(x)$. The idea is to reformulate (with the following iterating algorithm) the problem using only equality constraints to solve the problem as the previous one:

```
0. Choose a feasible point $x^0$, set $W_0 = \{i : A_ix^0 = b_i\}$ (working set) and
   $k = 0$.
1. Find the optimal solution $y^k$ of the problem
       $\begin{cases} \min \frac{1}{2}x^TQx + c^Tx \\ A_ix = b_i \quad \forall\, i \in W_k \end{cases}$
2. **If** $y^k \neq x^k$ **then** go to step 3
   **else** go to step 4
3. **If** $y^k$ is feasible **then** $t_k = 1$
   **else** $t_k = \min\left\{\frac{b_i - A_ix^k}{A_i(y^k - x^k)} : i \notin W_k,\ A_i(y^k - x^k) > 0\right\}$,
   **end**
   $x^{k+1} = x^k + t_k(y^k - x^k)$, $W_{k+1} = W_k \cup \{i \notin W_k : A_ix^{k+1} = b_i\}$,
   $k = k + 1$ and go to step 1
4. Compute the KKT multipliers $\mu^k$ related to $y^k$
   **If** $\mu^k \geq 0$ **then** STOP
   **else** $x^{k+1} = x^k$, $\mu_j^k = \min_{i \in W_k} \mu_i^k$, $W_{k+1} = W_k \setminus \{j\}$, $k = k + 1$ and go to step 1
```

note that if y^k is feasible (step 3), $x^{k+1} = y^k$; note also that if $y^k = x^k$ (step 4), it is a candidate to be a the minimum of the original problem: to check this we have to compute the KKT-System multipliers related to y^k. Since we're solving a problem with inequality constraints the multipliers μ must be non-negative.

8.2 Penalty Method

Let's consider the following constrained optimization problem

$$\begin{cases} \min f(x) \\ g_i(x) \leq 0 \quad \forall i = 1, ..., m \end{cases} \tag{P}$$

the idea is also to solve an equivalent unconstrained problem. First of all let's define the quadratic penalty function:

$$p(x) = \sum_{i=1}^{m} (\max\ \{0, g_i(x)\})^2$$

so $p(x)$ is called penalty function because it sums the value of the constraints that are not satisfied at x. Note that p is a function related to the feasible region, so we can write p in the objective function and relax the problem: we can consider the unconstrained penalized problem

$$\begin{cases} \min\ f(x) + \frac{1}{\varepsilon} p(x) \\ x \in \mathbb{R}^n \end{cases} \tag{$P\varepsilon$}$$

note that

$$\frac{1}{\varepsilon} p(x) = p_\varepsilon(x) \begin{cases} = f(x) & \text{if } x \in \Omega \\ > f(x) & \text{if } x \notin \Omega \end{cases}$$

here some properties:

- if f and g_i are continuously differentiable, then p_ε is continuously differentiable and $\nabla p_\varepsilon = \nabla f(x) + \frac{2}{\varepsilon} \sum_{i=1}^{m} \max\ \{0, g_i(x)\} \nabla g_i(x)$
- if f and g_i are convex, then p_ε is convex
- the problem P_ε is a relaxation of the original problem P, so $v(P_\varepsilon) \leq v(P)$ for any $\varepsilon > 0$
- if $0 < \varepsilon_1 < \varepsilon_2$, then $v(P_{\varepsilon_2}) \leq v(P_{\varepsilon_1})$, so the idea is to find a sequence of ε that give us the lower value of $v(P_\varepsilon)$
- if x^*_ε solves P_ε and $x^*_\varepsilon \in \Omega$, then x^*_ε is optimal also for P.

Here the algorithm for the penalty method

Penalty method

```
0. Set $\varepsilon_0 > 0$, $\tau \in (0, 1)$, $k = 0$
1. Find an optimal solution $x^k$ of the penalized problem $(P_{\varepsilon_k})$
2. If $x^k \in \Omega$ then STOP
   else $\varepsilon_{k+1} = \tau\varepsilon_k$, $k = k + 1$ and go to step 1.
```

- if f is coercive, then the sequence $\{x^k\}$ is bounded and any of its cluster points is an optimal solution of P

- if $\{x^k\}$ converges to x^*, then x^* is an optimal solution of P
- if $\{x^k\}$ converges to x^* and the gradients of the active constraints at x^* are linear independent, then x^* is an optimal solution of P and the sequence of vector $\{\lambda^k\}$ defined as follows, converges to a vector λ^* of KKT multipliers associated to x^*.

$$\lambda_i^k = \frac{2}{\varepsilon_k} \max \{0, g_i(x^k)\} \; \forall i = 1, ..., m$$

8.2.1 Exact Penalty Method

Another approach to the penalty method is called Exact Penalty Method. Consider the convex problem

$$\begin{cases} \min \; f(x) \\ g_i(x) \leq 0 \quad \forall i = 1, ..., m \end{cases}$$

and define the penalty function is defined as

$$\tilde{p}(x) = \sum_{i=1}^{m} \max \{0, g_i(x)\}$$

note that now we're not considering the square of the maximum then we're losing the smoothness. Now we can define the penalized problem as an unconstrained, convex and *non-smooth* problem:

$$\begin{cases} \min \; f(x) + \frac{1}{\varepsilon}\tilde{p}(x) \\ x \in \mathbb{R}^n \end{cases} \tag{$\tilde{P}_\varepsilon$}$$

The bad news is that since is a non-smooth problem, we can't use derivatives; on the other hand we don't need a sequence $\varepsilon_k \to 0$ to approximate an optimal solution of P, so we will avoid numerical issues because moving to 0, ε is a very small number.

Suppose that there exists an optimal solution x^* of P and λ^* is a KKT multipliers vector associated to x^*. Then, the sets of optimal solutions of P and $\tilde{P}_\varepsilon$ coincide provided that $\varepsilon \in (0, 1/\|\lambda^*\|_\infty)$.

Exact penalty method

```
0. Set $\varepsilon_0 > 0$, $\tau \in (0, 1)$, $k = 0$
1. Find an optimal solution $x^k$ of the penalized problem $(\widetilde{P}_{\varepsilon_k})$
2. If $x^k \in \Omega$ then STOP
   else $\varepsilon_{k+1} = \tau\varepsilon_k$, $k = k + 1$ and go to step 1.
```

this method stops after a finite number of iterations at an optimal solution of P.

8.3 Barrier Method

Let's consider the following problem

$$\begin{cases} \min f(x) \\ g(x) \leq 0 \end{cases}$$

where

- f, g_i are convex and twice continuously differentiable functions
- there's no isolated point in Ω
- exists an optimal solution
- slater constraint qualifiaction holds: there exists $\bar{x}$ such that $\bar{x} \in \mathrm{dom}(f)$ and $g_i(\bar{x}) < 0 \; \forall i = 1, ..., m$.

all those conditions implies strong duality. Linear programming problems and convex quadratic programming problems are special cases. The idea is always the same: reformulate the constrained problem to an equivalent unconstrained problem like

$$\begin{cases} \min f(x) + \sum_{i=1}^{m} I_-(g_i(x)) \\ x \in \mathbb{R}^n \end{cases}$$

where $I_-(u)$ is called indicator function of $\mathbb{R}_-$ that is neither finite nor differentiable and is defined as

$$I_-(u) = \begin{cases} 0 & \text{if } u \leq 0 \\ +\infty & \text{if } u > 0 \end{cases}$$

note that if x is feasible, then $I_-(g_i(x)) = 0$ for all i (because the constraint g_i are satisfied) and then we have only to minimize $f(x)$ without worrying about the constraints. But if x is not feasible, then the objective function is $+\infty$: how can we manage this "non-numeric" value? A solution is to approximate the indicator function I_- with a smooth convex function

$$u \mapsto -\varepsilon \; log(-u)$$

(called logarithmic barrier) with $\varepsilon > 0$ and the approximation improves as $\varepsilon \to 0$. So we can approximate the problem with

$$\begin{cases} \min\ f(x) - \varepsilon \sum_{i=1}^{m} log(-g_i(x)) \\ x \in \text{int}(\Omega) \end{cases} \tag{8.1}$$

another approximation of the indicator function can be $u \mapsto -\frac{\varepsilon}{u}$ that is a smooth convex function, and from this we could "build" another barrier method based on this approximation.

From the logarithmic barrier we can define the logarithmic barrier function as

$$B(x) = -\sum_{i=1}^{m} \log(-g_i(x))$$

and has the following properties:

- $\text{dom}(B) = \text{int}(\Omega)$
- B is convex because the constraints and the logarithm are convex functions
- B is smooth and

$$\nabla B(x) = -\sum_{i=1}^{m} \frac{1}{g_i(x)} \nabla g_i(x)$$

$$\nabla^2 B(x) = \sum_{i=1}^{m} \frac{1}{g_i(x)^2} \nabla g_i(x) \nabla g_i(x)^T + \sum_{i=1}^{m} \frac{1}{-g_i(x)} \nabla^2 g_i(x)$$

Now let's discuss about the relationship between the unconstrained problem and the original problem: suppose that x_ε^* is the optimal solution of 8.1, then $\nabla F(x_\varepsilon^*) = 0$ (with F the objective function). If we define $\lambda_\varepsilon^* = \left(\frac{\varepsilon}{-g_1(x_\varepsilon^*)}, ..., \frac{\varepsilon}{-g_m(x_\varepsilon^*)}\right) > 0$, then the Lagrangian function

$$L(x, \lambda_\varepsilon^*) = f(x) + \sum_{i=1}^{m} (\lambda_\varepsilon^*)_i g_i(x)$$

is convex and $\nabla_x L(x_\varepsilon^*, \lambda_\varepsilon^*) = \nabla F(x_\varepsilon^*) = 0$ hence

$$f(x_\varepsilon^*) \geq v(P) \geq \varphi(\lambda_\varepsilon^*) = \min_x\ L(x, \lambda_\varepsilon^*) = L(x_\varepsilon^*, \lambda_\varepsilon^*) = f(x_\varepsilon^*) - m\varepsilon$$

where $m\varepsilon$ is called optimality gap (the error). Explanation: if x^*_ε is feasible $f(x^*_\varepsilon) \geq v(P)$ (the optimal value of the primal); the dual function φ is always a lower bound for the primal problem; by definition $\varphi(\lambda^*_\varepsilon) = \min_x L(x, \lambda^*_\varepsilon) = L(x^*_\varepsilon, \lambda^*_\varepsilon)$ because L is convex. Notice that $(x^*_\varepsilon, \lambda^*_\varepsilon)$ solves the following approximation of the KKT system:

$$\begin{cases} \nabla f(x) + \sum_{i=1}^m \lambda_i \nabla g_i(x) = 0 \\ -\lambda_i g_i(x) = \varepsilon \\ \lambda \geq 0 \\ g(x) \leq 0 \end{cases}$$

Logarithmic barrier method

```
0. Set tolerance $\delta > 0$, $\tau < 1$ and $\varepsilon_1 > 0$. Choose $x^0 \in \text{int}(\Omega)$, set $k = 1$
1. Find the optimal solution $x^k$ of
       $\begin{cases} \min\ f(x) - \varepsilon_k \sum_{i=1}^{m} \log(-g_i(x)) \\ x \in \text{int}(\Omega) \end{cases}$
   using $x^{k-1}$ as starting point
2. If $m\varepsilon_k < \delta$ then STOP
   else $\varepsilon_{k+1} = \tau\varepsilon_k$, $k = k+1$ and go to step 1
```

note that a small τ means fewer outer iterations, more inner iterations.

How to find $x^0 \in \text{int}\ \Omega$? We have to consider the auxiliary problem

$$\begin{cases} \min\limits_{x,s}\ s \\ g_i(x) \leq s \end{cases}$$

- take $\tilde{x} \in \mathbb{R}^n$ and find $\tilde{s} > \max\limits_{i=1,..,m} g_i(\tilde{x})$ this means that $(\tilde{x}, \tilde{s})$ is in the interior of the feasible region of the auxiliary problem
- Find an optimal solution (x^*, s^*) of the auxiliary problem using a barrier method starting from $(\tilde{x}, \tilde{s})$
- if $s^* < 0$ then $x^* \in \text{int}(\Omega)$, else $\text{int}(\Omega) = \emptyset$.

Capitolo 9

Multi-objective optimization problems

Until now we considered only one objective function, from now on we'll consider the objective function f as a vector of p elements:

$$f(x) = (f_1(x), f_2(x), ..., f_p(x))$$

Minimizing f means minimizing all the $f_i \in f$ and so the definition of optimality is not obvious: could happen that the f_i are conflicting each other, for example if the value of f_1 decrease, the value of f_2 could increase and vice versa.

First of all we need to define an order in $\mathbb{R}^p$ in order to determine if a vector is better than an other vector: given $x, y \in \mathbb{R}^p$, we say that

$$x \geq y \iff x_i \geq y_i \ \forall i = 1, ..., p$$

this order is called *Pareto Order* and is a partial order, this means that has the following properties:

- reflexivity: $x \geq x$
- asymmetry: if $x \geq y$ and $y \geq x$, then $x = y$
- transitivity: if $x \geq y$ and $y \geq z$, then $x \geq z$

but is not a total order: if $x = (1, 4)$ and $y = (3, 2)$, then $x \ngeq y$ and $y \ngeq x$. Starting from this, we can define the following properties: given a subset $A \subset \mathbb{R}^p$

- $x \in A$ is a *Pareto ideal minimum* of A if $x \leq y$ for any $y \in A$: in other words we have to minimize at the same time all the components;

- $x \in A$ is a *Pareto minimum* of A if $\nexists y \in A$, $y \neq x$ such that $y \leq x$ (for each component of x and y);
- $x \in A$ is a *Pareto weak minimum* of A if $\nexists y \in A$, $y \neq x$ such that $y < x$ (for each component of x and y).

Note that

- $IMin(A) \subseteq Min(A) \subseteq WMin(A)$
- if $IMin(A) \neq \emptyset$ then $Min(A) = WMin(A) = \{\bar{x}\}$

Finally let's define a multiobjective optimization problem as

$$\begin{cases} \min\ f(x) = (f_1(x), f_2(x), ..., f_p(x)) \\ x \in \Omega \end{cases} \tag{P}$$

- $f(x^*)$ is a Pareto ideal minimum of $f(\Omega)$ if $f(x^*) \leq f(x)$ for any $x \in \Omega$ (for each component of $f(x^*)$ and $f(x)$).
  - if $f(x^*)$ is a Pareto ideal minimum of $f(\Omega)$, then $x^* \in \Omega$ is a Pareto ideal minimum of P.
- $f(x^*)$ is a Pareto Minimum of $f(\Omega)$ if there's no $x \in \Omega$, $x \neq x^*$ such that

  $$f_i(x) \leq f_i(x^*) \text{ for any } i = 1, ...p$$
  $$f_j(x) < f_j(x^*) \text{ for some } j \in \{1, ...p\}$$

  In other words, x^* is a Pareto minimum if we can't find another point $x \neq x^*$ that improves all the objective functions.
  - if $f(x^*)$ is a Pareto minimum of $f(\Omega)$, then $x^* \in \Omega$ is a Pareto minimum of P.
- $f(x^*)$ is a Pareto Weak Minimum of $f(\Omega)$ if there's no $x \in \Omega$, $x \neq x^*$ such that

  $$f_i(x) \leq f_i(x^*) \text{ for any } i = 1, ...p$$

  - if $f(x^*)$ is a Pareto weak Minimum of $f(\Omega)$, then $x^* \in \Omega$ is a Pareto weak minimum of P.

Note that we're moving from a space $\mathbb{R}^n$ to $\mathbb{R}^p$, with n the input dimension and p the number of objective functions: to do this we have to write the problem in function of $y = f(x)$, not anymore x. The idea is to find those three types of minimum in $f(\Omega)$ and then move to Ω to find the minimum of the original problem. [Example on iPad].

9.1 Existence and Optimality Conditions

9.1.1 Existence results

- If f_i is continuous for any $i = 1, ..., p$ and Ω is closed and bounded, then there exists a (Pareto) minimum of P.
- If f_i is continuous for any $i = 1, ..., p$, Ω is closed (even if is not bounded) and there are $v \in \mathbb{R}$ and $j \in \{1, ..., p\}$ such that the sublevel set

$$\{x \in \Omega : f_j(x) \leq v\}$$

  is nonempty and bounded, then there exists a minimum of P.
- If f_i is continuous for any $i = 1, ..., p$, Ω is closed and f_j is coercive for some $j \in \{1, ..., p\}$, then there exists a minimum of P.

9.1.2 Optimality conditions

- $x^* \in \Omega$ is a minimum of P if and only if the auxiliary optimization problem has optimal value equal to 0:

$$\begin{cases} \max \sum_{i=1}^{p} \varepsilon_i \\ f_i(x) + \varepsilon_i \leq f_i(x^*) & \forall i = 1, ..., p \\ x \in \Omega \\ \varepsilon \geq 0 \end{cases}$$

- $x^* \in \Omega$ is a weak minimum of P if and only if the auxiliary optimization problem has optimal value equal to 0:

$$\begin{cases} \max v \\ v \leq \varepsilon_i & \forall i = 1, ..., p \\ f_i(x) + \varepsilon_i \leq f_i(x^*) & \forall i = 1, ..., p \\ x \in \Omega \\ \varepsilon \geq 0 \end{cases}$$

Note that those problems are linear programming problems: can be solved using the MatLab function `linprog`.

9.1.3 First-Order Optimality conditions for unconstrained problems

Consider an unconstrained multiobjective problem

$$\begin{cases} \min\ f(x) = (f_1(x), f_2(x), ..., f_p(x)) \\ x \in \mathbb{R}^n \end{cases} \tag{P}$$

where the f_i are continuously differentiable for any $i = 1, ..., p$.

[**Necessary Optimality Condition**] If x^* is a weak minimum of P, then exists $\xi^* \in \mathbb{R}^p$ such that

$$\begin{cases} \sum_{i=1}^{p} \xi_i^* \nabla f_i(x^*) = 0 \\ \xi^* \geq 0 \\ \sum_{i=1}^{p} \xi_i^* = 1 \end{cases} \tag{S}$$

note that this system generalize the stationary point condition (from the scalar case to the vector case) using a convex combination.

[**Sufficient Optimality Condition**] If the problem P is convex (i.e. f_i is convex for any $i = 1, ..., p$), and (x^*, ξ^*) is a solution of the system S, then x^* is a weak minimum of P.

9.1.4 First-Order Optimality conditions for constrained problems

Consider a constrained multiobjective problem

$$\begin{cases} \min\ f(x) = & (f_1(x), f_2(x), ..., f_p(x)) \\ g_j(x) \leq 0 & \forall j = 1, ..., m \\ h_i(x) = 0 & \forall i = 1, ..., q \end{cases} \tag{P}$$

where f_i, g_j, h_k are continuous differentiable for any i, j, k. The idea of this situation is to generalize the KKT system.

[**Necessary Optimality Condition**] If x^* is a weak minimum of P and ACQ holds at x^*, then there exists $\xi^* \in \mathbb{R}^p$, $\lambda^* \in \mathbb{R}^m$ and $\mu^* \in \mathbb{R}^q$ such that

$(x^*, \xi^*, \lambda^*, \mu^*)$ solves the following KKT system

$$\begin{cases} \sum_{i=1}^{p} \xi_i^* \nabla f_i(x^*) + \sum_{j=1}^{m} \lambda_j^* \nabla g_j(x^*) + \sum_{k=1}^{q} \mu_k^* \nabla h_k(x^*) = 0 \\ \xi^* \geq 0 \\ \sum_{i=1}^{p} \xi_i^* = 1 \\ \lambda^* \geq 0 \\ \lambda_j^* g_j(x^*) = 0 \ \forall j = 1, ..., m \end{cases}$$

[**Sufficient Optimality Condition**] If P is convex, i.e. f_i is convex, g_j convex and h_k affine, and $(x^*, \xi^*, \lambda^*, \mu^*$ solves the KKT system, then x^* is a weak minimum of P.

9.2 Scalarization method

The idea is to transform the vector of objective functions into only one function. Let's define a vector of weights associated to the objectives

$$\alpha = (\alpha_1, ..., \alpha_p) \text{ such that } \sum_{i=1}^{p} \alpha_i = 1$$

and consider the following scalar optimization problem

$$\begin{cases} \min \ \sum_{i=1}^{p} \alpha_i f_i(x) \\ x \in \Omega \end{cases} \qquad (P_\alpha)$$

note that in this way the we're giving a weight to each objective function in order to create a "new" objective function with only one component.

Let S_α the set of optimal solution of P_α

- $\bigcup_{\alpha \geq 0} S_\alpha \subseteq \{$weak minima of the original problem $P\}$
- $\bigcup_{\alpha > 0} S_\alpha \subseteq \{$ minima of the original problem $P\}$

note that solving P_α for any possible choice of α does not allow finding all the minima and weak minima.

- if P is a linear problem, then $\{$weak minima of $P\} = \bigcup_{\alpha \geq 0} S_\alpha$ and $\{$minima of $P\} = \bigcup_{\alpha > 0} S_\alpha$
- if P is convex, then $\{$weak minima of $P\} = \bigcup_{\alpha \geq 0} S_\alpha$
- if P is convex and f_i strongly convex for any $i = 1, ..., p$, then $\{$minima of $P\} = \{$weak minima of $P\} = \bigcup_{\alpha \geq 0} S_\alpha$

9.3 Goal method

Using the previous methods we could find an infinite set of points that are minima, but maybe some points could be more efficient than other points: we want to choose only one solution, what is the better one? A solution to this question is given by the goal method.

Let's define an ideal point $z \in \mathbb{R}^p$ as

$$z_i = \min_{x \in \Omega} f_i(x)$$

in other words an ideal point is the image of an ideal minimum. Unfortunately very often the ideal minimum of a problem P doesn't exists ($z \notin f(\Omega)$), so we want to find the point of $f(\Omega)$ which is close as possible to z:

$$\begin{cases} \min \|f(x) - z\|_s \quad \text{with } s \in [1, +\infty] \\ x \in \Omega \end{cases} \tag{G}$$

note that we're not searching for a set of points but only for one point (that is the minimum). Following some important results:

- if $s \in [1, +\infty)$ then any optimal solution of G is a minimum of P;
- if $s = +\infty$ then any optimal solution of G is a weak minimum of P.

Assuming that P is a multi-objective optimization problem, i.e.

$$\begin{cases} \min Cx \\ Ax \leq b \end{cases} \tag{P}$$

where C is a $p \times n$ matrix,

- if $s = 2$, then G is equivalent to the following quadratic programming problem

$$\begin{cases} \min \frac{1}{2}\|Cx - z\|_2^2 = \frac{1}{2}x^T C^T C x - x^T C^T z + \frac{1}{2} z^T z \\ Ax \leq b \end{cases} \tag{G_2}$$

- if $s = 1$, then G is equivalent to the following linear programming problem

$$\begin{cases} \min \ \sum_{i=1}^p y_i \\ y_i \geq C_i x - z_i \quad \forall i = 1, ..., p \\ y_i \geq z_i - C_i x \quad \forall i = 1, ..., p \\ Ax \leq b \end{cases} \tag{G_1}$$

- if $s = \infty$, then G is equivalent to the following linear programming problem

$$\begin{cases} \min\limits_{x,y} \; y \\ y_i \geq C_i x - z_i \quad \forall i = 1, ..., p \\ y_i \geq z_i - C_i x \quad \forall i = 1, ..., p \\ Ax \leq b \end{cases} \tag{G_∞}$$

note that changing norm, the minimum will change and also the complexity of the problem as well. [Example on iPad].

Capitolo 10

Non-cooperative game theory

At the beginning of this course we considered optimization problems with one decision maker and one objective function, then we considered optimization problems with one decision maker and multiple objective function, now we will consider optimization problems with different decision makers (players) which have different interests (more objective functions): game theory deals with that.

Game theory studies the possibility to forecast the strategies that will be chosen by each player which is assumed to be "rational".

A *non-cooperative game* (the players won't help each other) in normal form is given by a set of N players, each player i has a set Ω_i of strategies and a cost function $f_i : \Omega_1 \times ... \times \Omega_N \to \mathbb{R}$ in which each player wants to find the best value for his strategy considering also all the possible strategies of the other players (variables we can not control). The aim of each player i consists in solving the following optimization problem

$$\begin{cases} \min\ f_i(x_1, x_2, ..., x_i, x_{i+1}, ..., x_N) \\ x_i \in \Omega_i \end{cases}$$

With two players the optimization problem can be written as

$$\text{Player 1:} \begin{cases} \min\ f_1(x,y) \\ x \in X \end{cases} \qquad \text{Player 2:} \begin{cases} \min\ f_2(x,y) \\ y \in Y \end{cases}$$

where Player 1 controls x and Player 2 controls y: when Player 1 wants to find the best move (using f_i), he has to consider also the strategies of Player 2. Note that in a cooperative game, we have simply a multiobjective optimization problem (where $f = (f_1, f_2)$) with only one decision maker.

Sometimes the optimal solution found could be good for one player but bad for the other: we want to find an equilibrium in which the two players are happy. So let's define an equilibrium notion known as *Nash Equilibrium*:

Def 10.0.1 (Nash Equilibrium)**.** In a two person non-cooperative game, a pair of strategies $(\bar{x}, \bar{y})$ is a Nash Equilibrium if no player can decrease his cost by unilateral deviation[1] i.e.

$$f_1(\bar{x}, \bar{y}) = \min_{x \in X} f_1(x, \bar{y}) \quad \text{and} \quad f_2(\bar{x}, \bar{y}) = \min_{y \in Y} f_2(\bar{x}, y)$$

in other words, assuming I know the strategy $(\bar{y})$ of the other player, I want to find the best possible strategy $(\bar{x})$ for me AND assuming that the other player knows my strategy $(\bar{x})$, he wants to find the best possible strategy $(\bar{y})$ for him.

An equivalent definition is: $(\bar{x}, \bar{y})$ is a Nash Equilibrium if and only if $\bar{x}$ is the best response of Player 1 to strategy $\bar{y}$ of Player 2 and vice versa $\bar{y}$ is the best response of Player 2 to strategy $\bar{x}$ of Player 1.

10.1 Matrix Games

A matrix game is a two-person non-cooperative game where:

- X and Y are finite sets of strategies: $X = \{1, ..., m\}$, $Y = \{1, ..., n\}$
- $f_2 = -f_1$ (zero sum game).

This class of games is call Matrix Games because can be represented by a $m \times n$ matrix C, where c_{ij} is the amount of "money" (the cost) that Player 1 pays to Player 2 if Player 1 chooses the strategy i and Player 2 chooses the strategy j: if (i, j) and (p, q) are Nash Equilibria of a matrix game [proofs on iPad]:

- $c_{ij} = c_{pq}$
- $(i, q) = (p, j)$ are Nash equilibria as well.

Given a two-person non-cooperative game, a strategy $x \in X$ is *strictly dominated* by $\tilde{x} \in X$ if

$$f_1(x, y) > f_1(\tilde{x}, y) \qquad \forall y \in Y$$

[1]Each player can control only his variables.

and similarly a strategy $y \in Y$ is strictly dominated by $\tilde{y} \in Y$ if

$$f_2(x, y) > f_2(x, \tilde{y}) \qquad \forall x \in X$$

this is an important property because strictly dominated strategies can be deleted from the game.

10.1.1 Mixed Strategies

If the problem has no Nash equilibria and also there are no strictly dominated strategies, we can relax the problem using *mixed strategies*: if C is a $m \times n$ matrix game, then a mixed strategy for Player 1 is a $m-$vector of probabilities and we consider

$$X = \{x \in \mathbb{R}^m : x \geq 0, \sum_{i=1}^{m} x_i = 1\}$$

the set of mixed strategies of Player 1. And similarly

$$Y = \{y \in \mathbb{R}^n : y \geq 0, \sum_{j=1}^{n} y_i = 1\}$$

is the set of mixed strategies of Player 2. If $x_i = 1$ and $\forall k \neq i \ x_k = 0$, the strategy is called *pure* (like the general definition of Nash Equilibrium). Now the expected costs are

$$f_1(x, y) = x^T C y = \sum_{i=1}^{m} \sum_{j=1}^{n} x_i c_{ij} y_j$$

$$f_2(x, y) = -x^T C y = -\sum_{i=1}^{m} \sum_{j=1}^{n} x_i c_{ij} y_j$$

respectively to Player 1 and Player 2.

Now let's re-define the Nash equilibrium using mixed strategies: if C is a $m \times n$ matrix game, then $(\bar{x}, \bar{y}) \in X \times Y$ is a *mixed strategies Nash equilibrium* if

$$\max_{y \in Y} \bar{x}^T C y = \bar{x}^T C \bar{y} = \min_{x \in X} x^T C \bar{y}$$

so $(\bar{x}, \bar{y})$ is a saddle point of the function $x^T C y$.

Theorem 10.1.1. *$(\bar{x}, \bar{y})$ is a mixed strategies Nash equilibrium if and only if*

$$\begin{cases} \bar{x} \text{ is an optimal solution of } \min\limits_{x \in X} \max\limits_{y \in Y} x^T C y \\ \bar{y} \text{ is an optimal solution of } \max\limits_{y \in Y} \min\limits_{x \in X} x^T C y \end{cases}$$

so the Nash equilibrium can be found by solving two optimization problems that are equivalent to the following linear programming problems [proofs on iPad]:

$$\min_{x \in X} \max_{y \in Y} x^T C y = \begin{cases} \min v \\ v \geq \sum_{i=1}^m c_{ij} x_i \quad \forall j = 1, ..., n \\ x \geq 0 \\ \sum_{i=1}^m x_i = 1 \end{cases} \tag{P_1}$$

$$\max_{y \in Y} \min_{x \in X} x^T C y = \begin{cases} \max w \\ w \leq \sum_{j=1}^n c_{ij} y_j \quad \forall i = 1, ..., m \\ y \geq 0 \\ \sum_{j=1}^n y_j = 1 \end{cases} \tag{P_2}$$

note that P_2 is the dual of P_1.

10.2 Bimatrix Games

A bimatrix game is a two-person non-cooperative game where:

- the sets of pure strategies are finite, hence the sets of mixed strategies are:
  - ◇ $X = \{x \in \mathbb{R}^m : x \geq 0, \sum_{i=1}^m x_i = 1\}$
  - ◇ $Y = \{y \in \mathbb{R}^n : y \geq 0, \sum_{j=1}^n y_j = 1\}$
- the cost functions are $f_1(x, y) = x^T C_1 y$ and $f_2(x, y) = x^T C_2 y$ where C_1 and C_2 are $m \times n$ matrix (this is why is call bi-matrix). Note that is not a zero-sum game as before, hence we have to separate the cost of the strategies for Player 1 and Player 2 using C_1 and C_2.

Any bimatrix game has at least a mixed strategies Nash equilibrium.

To *find strictly dominated strategies* we have to compare the rows of C_1 (if row i is greater than row j, we can delete row i from C_1) and the columns of C_2 (if column i is greater than column j, we can delete column i from C_2).

To *find the Nash equilibrium* we have to define the best response mapping $B_1 : Y \to X$ and $B_2 : X \to Y$ as

$$B_1(y) = \{\text{optimal solution of } \min_{x \in Y} x^T C_1 y\}$$

$$B_2(x) = \{\text{optimal solution of } \min_{y \in Y} x^T C_2 y\}$$

$(\bar{x}, \bar{y})$ is a Nash equilibrium if and only if $\bar{x} \in B_1(\bar{y})$ and $\bar{y} \in B_2(\bar{x})$. Graphically means that the Nash equilibria are given by the intersections of the graphs of the best response mappings B_1 and B_2. [Example on iPad].

When the matrix C is bigger than a 2×2 matrix, plotting the graphs could be difficult because we should deal with higher dimensional spaces. A more generic approach is to find the Nash equilibria using the KKT condition:

[**Necessary and Sufficient Condition**] $(\bar{x}, \bar{y})$ is a Nash equilibrium if and only if there are $\mu_1, \mu_2 \in \mathbb{R}$ such that

$$\begin{cases} C_1\bar{y} + \mu_1 e \geq 0 \\ \bar{x} \geq 0 \\ \sum_{i=1}^m \bar{x}_i = 1 \\ \bar{x}_i(C_1\bar{y} + \mu_1 e)_i = 0 \quad \forall i = 1, ..., m \\ C_2^T\bar{x} + \mu_2 e \geq 0 \\ \bar{y} \geq 0 \\ \sum_{j=1}^n \bar{y}_i = 1 \\ \bar{y}_j(C_2^T\bar{x} + \mu_2 e)_j = 0 \quad \forall j = 1, ..., n \end{cases}$$

where $e = (1, ..., 1)^T$ [proof on iPad]. Note that this system is given by the union of the KKT System of $P_1(y)$ and the KKT System of $P_2(x)$.

An equivalent formulation to find the Nash equilibria is the following: first of all let's assume that $C_1 < 0$ and $C_2 < 0$ (just choose two constant k_1, $k_1 \in \mathbb{R}$ such that $C_1 - k_1 < 0$ and $C_2 - k_2 < 0$).

[**Necessary Condition**] If $(\bar{x}, \bar{y})$ is a Nash equilibrium then there are $u > 0$ and $v > 0$ (that represent μ_1, μ_2, used to normalize the system) such that

$\tilde{x} = \bar{x}/u$ and $\tilde{y} = \bar{y}/v$ solve the following system:

$$\begin{cases} \tilde{x} \geq 0 \\ C_1\tilde{y} + e \geq 0 \\ \tilde{x}_i(C_1\tilde{y} + e)_i = 0 \quad \forall i = 1, ..., m \\ \tilde{y} \geq 0 \\ C_2^T\tilde{x} + e \geq 0 \\ \tilde{y}_j(C_2^T\tilde{x} + e)_j = 0 \quad \forall j = 1, ..., n \end{cases} \tag{S}$$

[**Sufficient Condition**] If $(\tilde{x}, \tilde{y})$ solves the system S with $\tilde{x} \neq 0$ and $\tilde{y} \neq 0$, then

$$\left(\frac{\tilde{x}}{\sum_{i=1}^m \tilde{x}_i}, \frac{\tilde{y}}{\sum_{j=1}^n \tilde{y}_j} \right)$$

is a Nash equilibrium.

Define the polyhedrons

$$P = \left\{ (x_1, ..., x_m) : \begin{cases} x_i \geq 0 & \forall i = 1, ..., m \\ (C_2^T x + e)_j \geq 0 & \forall j = m+1, ..., m+n \end{cases} \right\}$$

for Player 1 and

$$Q = \left\{ (y_{m+1}, ..., y_{m+n}) : \begin{cases} (C_1 y + e)_i \geq 0 & \forall i = 1, ..., m \\ y_j \geq 0 & \forall j = m+1, ..., m+n \end{cases} \right\}$$

for Player 2:

- $(\tilde{x}, \tilde{y})$ solves the system S if and only if $\tilde{x} \in P$, $\tilde{y} \in Q$ and $\forall k = 1, ..., m+n$ either the k-th constraint of P is active in $\tilde{x}$ or the k-th constraint of Q is active in $\tilde{y}$.
- if the vertices of P and Q are non-degenerate (each vertex of P or Q has exactly m or n active constraints) and $(\tilde{x}, \tilde{y})$ solves the system S, then $\tilde{x}$ is a vertex of P and $\tilde{y}$ is a vertex of Q.

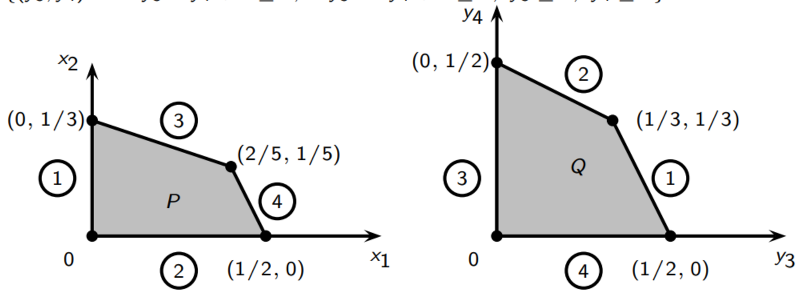
Therefore if $C_1 < 0$, $C_2 < 0$ and vertices of P and Q are non-degenerate, then we can find *all* the Nash equilibria analyzing all the pairs (x, y) of vertices of P and Q, checking if for any $k = 1, ..., m+n$ either the k-th constraint of P is active in x or the k-th constraint of Q is active in y. [Example on iPad].

Lemke-Howson Algorithm When $C_1 < 0$ and $C_2 < 0$ and vertices of P and Q are non-degenerate, a Nash equilibrium (only one, not all) can be found using the Lemke-Howson algorithm:

1. Set $x = 0$, $y = 0$. Define $I_x = \{1, \dots, m\}$, $I_y = \{m+1, \dots, m+n\}$. Choose an exiting index $h \in \{1, \dots, m\}$.
2. In the polyhedron P, move from x along the edge given by constraints in $I_x \setminus \{h\}$ to the adjacent vertex x'.
3. Find the entering index k corresponding to the new active constraint at x'. Set $x = x'$ and $I_x = (I_x \setminus \{h\}) \cup \{k\}$.
4. If $k \notin I_y$ then STOP: (x, y) solves (S) $\rightarrow$ find a Nash equilibrium. else set $h = k$ and in the polyhedron Q move from y along the edge given by constraints in $I_y \setminus \{h\}$ to the adjacent vertex y'.
5. Find the entering index k corresponding to the new active constraint at y'. Set $y = y'$ and $I_y = (I_y \setminus \{h\}) \cup \{k\}$.
6. If $k \notin I_x$ then STOP: (x, y) solves (S) $\rightarrow$ find a Nash equilibrium. else set $h = k$ and go to step 2.

where I_x is the set of active constraints at the point x. The idea of this algorithm is to move along the edges and change vertex until we find a Nash equilibrium. Here's an example:

Example. Consider the following bimatrix game: $C_1 = \begin{pmatrix} -2 & -1 \\ -1 & -2 \end{pmatrix}$ $C_2 = \begin{pmatrix} -1 & -2 \\ -3 & -1 \end{pmatrix}$
$P = \{(x_1, x_2) : \ x_1 \geq 0, \ x_2 \geq 0, \ -x_1 - 3x_2 + 1 \geq 0, \ -2x_1 - x_2 + 1 \geq 0\}$
$Q = \{(y_3, y_4) : \ -2y_3 - y_4 + 1 \geq 0, \ -y_3 - 2y_4 + 1 \geq 0, \ y_3 \geq 0, \ y_4 \geq 0\}$



Iter	h	k	x	I_x	y	I_y
1			$(0,0)$	$\{1,2\}$	$(0,0)$	$\{3,4\}$
2	1	4	$(1/2,0)$	$\{2,4\}$	$(0,0)$	$\{3,4\}$
3	4	2	$(1/2,0)$	$\{2,4\}$	$(0,1/2)$	$\{2,3\}$
4	2	3	$(2/5,1/5)$	$\{3,4\}$	$(0,1/2)$	$\{2,3\}$
5	3	1	$(2/5,1/5)$	$\{3,4\}$	$(1/3,1/3)$	$\{1,2\}$
	Nash	equil.	$(2/3,1/3)$		$(1/2,1/2)$	

10.3 Convex Games

Now we consider the following two-person non-cooperative game

$$\text{Player 1: }\begin{cases}\min\limits_{x} f_1(x,y) \\ g_i^1(x) \le 0 \qquad \forall i = 1,...,p\end{cases} \qquad \text{Player 2: }\begin{cases}\min\limits_{y} f_2(x,y) \\ g_j^2(y) \le 0 \qquad \forall j = 1,...,q\end{cases}$$

where f_1, g^1, f_2 and g^2 are continuously differentiable. The game is said convex if the optimization problem of each player is convex.

[**Sufficient condition for the existence of Nash equilibrium**] If

- the feasible regions $X = \{x \in \mathbb{R}^m : g_i^1(x) \le 0 \; i = 1,...,p\}$ and $Y = \{y \in \mathbb{R}^n : g_j^2(y) \le 0 \; j = 1,...,q\}$ are closed, convex and bounded
- the cost function $f_1(\cdot, y)$ is quasi-convex for any $y \in Y$ and $f_2(x, \cdot)$ is quasi-convex for any $x \in X$

then there exists at least a Nash equilibrium.

[**Necessary Optimality Condition**] If $(\bar{x}, \bar{y})$ is a Nash equilibrium and the Abadie constraints qualification holds both in $\bar{x}$ and $\bar{y}$, then are $\lambda^1 \in \mathbb{R}^p$ and $\lambda^2 \in \mathbb{R}^q$ such that

$$\begin{cases}\nabla_x f_1(\bar{x},\bar{y}) + \sum_{i=1}^p \lambda_i^1 \nabla g_i^1(\bar{x}) = 0 \\ \lambda^1 \ge 0 \\ g^1(\bar{x}) \le 0 \\ \lambda_i^1 g_i^1(\bar{x}) = 0 & i = 1,...,p \\ \nabla_y f_2(\bar{x},\bar{y}) + \sum_{j=1}^q \lambda_j^2 \nabla g_j^2(\bar{y}) = 0 \\ \lambda^2 \ge 0 \\ g^2(\bar{y}) \le 0 \\ \lambda_j^2 g_j^2(\bar{y}) = 0 & j = 1,...,q\end{cases}$$

[**Sufficient Optimality Condition**] If $(\bar{x}, \bar{y}, \lambda^1, \lambda^2)$ solves the above system and the game is convex, then $(\bar{x}, \bar{y})$ is a Nash equilibrium.

10.3.1 Merit Functions

The aim of the Merit functions is to reformulate the Nash equilibrium problem into an equivalent optimization problem but non-convex: this means that the global optimum is difficult to find. Assuming that the game is convex, we can define the *Nikaido-Isoda function*

$$f(x, y, u, v) = f_1(u, y) - f_1(x, y) + f_2(x, v) - f_2(x, y)$$

where $x, u \in \mathbb{R}^m$ (variables of Player 1) and $y, v \in \mathbb{R}^n$ (variables of Player 2). In practice the left part of the equation represent the advantage for Player 1 to change the current strategy x to another strategy u, assuming that Player 2 plays strategy y; the right part of the equation similarly represent the advantage for Player 2 to change the current strategy y to another strategy v, assuming that Player 1 plays strategy x.

Below we will discuss about three Merit functions.

Gap Function From the Nikaido-Isoda function we can define the gap function as

$$\psi(x, y) = \max_{u \in X, v \in Y} [-f(x, y, u, v)]$$

that has the following properties

- the problem defining ψ is convex: we're maximizing a concave function with convex constraints;
- $\psi(x, y) \geq 0$ for any $(x, y) \in X \times Y$;
- $(\bar{x}, \bar{y})$ is a Nash equilibrium if and only if $(\bar{x}, \bar{y}) \in X \times Y$ and $\psi(\bar{x}, \bar{y}) = 0$ [Proof on iPad].

This function gives us a gap between (x, y) and the Nash equilibrium $(\bar{x}, \bar{y})$. Therefore, finding Nash equilibria is equivalent to solve the (*non-convex*) constrained *optimization problem*

$$\begin{cases} \min \ \psi(x, y) \\ (x, y) \in X \times Y \end{cases}$$

In the particular case of a bi-matrix game, which $f_1(x, y) = x^T C_1 y$, $f_2(x, y) = x^T C_2 y$ is easy to find check if (x, y) is a Nash equilibrium

$$\begin{aligned}
\psi(x, y) &= \max_{u \in X, v \in Y} [-f(x, y, u, v)] \\
&= \max_{u \in X, v \in Y} [x^\mathsf{T} C_1 y - u^\mathsf{T} C_1 y + x^\mathsf{T} C_2 y - x^\mathsf{T} C_2 v] \\
&= x^\mathsf{T}(C_1 + C_2)y + \max_{u \in X}[-u^\mathsf{T} C_1 y] + \max_{v \in Y}[-x^\mathsf{T} C_2 v] \\
&= x^\mathsf{T}(C_1 + C_2)y - \min_{u \in X}[u^\mathsf{T} C_1 y] - \min_{v \in Y}[v^\mathsf{T} C_2^\mathsf{T} x] \\
&= x^\mathsf{T}(C_1 + C_2)y - \min_{1 \leq i \leq m} (C_1 y)_i - \min_{1 \leq j \leq n} (C_2^\mathsf{T} x)_j
\end{aligned}$$

Regularized Gap Function In general the gap function ψ is not differentiable, but is possible to regularize it using a parameter $\alpha > 0$. The regularized gap function is defined as

$$\psi_\alpha(x,y) = \max_{u\in X, v\in Y} \left[-f(x,y,u,v) \right] - \frac{\alpha}{2}\|(x,y)-(u,v)\|^2 \Big]$$

that has the following properties

- the problem defining ψ_α is convex and has an unique optimal solution
- ψ_α is continuously differentiable
- $\psi_\alpha(x,y) \geq 0$ for any $(x,y) \in X \times Y$
- $(\bar{x},\bar{y})$ is a Nash equilibrium if and only if $(\bar{x},\bar{y}) \in X \times Y$ and $\psi_\alpha(\bar{x},\bar{y}) = 0$

Therefore, finding Nash equilibria is equivalent to solve the (non-convex) *smooth* constrained optimization problem

$$\begin{cases} \min \psi_\alpha(x,y) \\ (x,y) \in X \times Y \end{cases}$$

note that now ψ_α is smooth and we can solve the above problem using derivatives.

In the particular case of a bi-matrix game, which $f_1(x,y) = x^T C_1 y$, $f_2(x,y) = x^T C_2 y$ is easy to find check if (x,y) is a Nash equilibrium

$$\begin{aligned}
\psi_\alpha(x,y) &= \max_{u\in X, v\in Y}[-f(x,y,u,v) - \frac{\alpha}{2}\|(x,y)-(u,v)\|^2] \\
&= x^{\mathsf{T}}(C_1+C_2)y + \max_{u\in X}[-u^{\mathsf{T}}C_1 y - \frac{\alpha}{2}\|x-u\|^2] + \max_{v\in Y}[-x^{\mathsf{T}}C_2 v - \frac{\alpha}{2}\|y-v\|^2] \\
&= x^{\mathsf{T}}(C_1+C_2)y - \frac{\alpha}{2}(\|x\|^2+\|y\|^2) + \max_{u\in X}\left[-\frac{\alpha}{2}\|u\|^2 + \alpha x^{\mathsf{T}}u - u^{\mathsf{T}}C_1 y\right] \\
&\quad + \max_{v\in Y}\left[-\frac{\alpha}{2}\|v\|^2 + \alpha y^{\mathsf{T}}v - v^{\mathsf{T}}C_2^{\mathsf{T}}x\right] \\
&= x^{\mathsf{T}}(C_1+C_2)y - \frac{\alpha}{2}(\|x\|^2+\|y\|^2) - \min_{u\in X}\left[\frac{\alpha}{2}\|u\|^2 + u^{\mathsf{T}}(C_1 y - \alpha x)\right] \\
&\quad - \min_{v\in Y}\left[\frac{\alpha}{2}\|v\|^2 + v^{\mathsf{T}}(C_2^{\mathsf{T}}x - \alpha y)\right]
\end{aligned}$$

note that we have to solve two quadratic convex problem.

D-Gap Function The D-Gap function allow us to reformulate the problem of finding Nash equilibria as an unconstrained optimization problem. The D-Gap function is defined as

$$\psi_{\alpha,\beta}(x,y) = \psi_\alpha(x,y) - \psi_\beta(x,y)$$

that has the following properties

- $\psi_{\alpha,\beta}$ is continuously differentiable: is the subtraction of two continuously differentiable functions;
- $\psi_{\alpha,\beta}(x,y) \geq 0$ for any $(x,y) \in \mathbb{R}^m \times \mathbb{R}^n$ (even if x and y are not feasible);
- $(\bar{x},\bar{y})$ is a Nash equilibrium if and only if $\psi_{\alpha,\beta}(\bar{x},\bar{y}) = 0$.

Therefore, finding Nash equilibria is equivalent to solve the (non-convex) smooth *unconstrained* optimization problem

$$\begin{cases} \min\ \psi_{\alpha,\beta}(x,y) \\ (x,y) \in \mathbb{R}^m \times \mathbb{R}^n \end{cases}$$

In the particular case of a bi-matrix games the Nash equilibrium can be found solving $\psi_\alpha(x,y) - \psi_\beta(x,y)$ in the same way as the regularized gap function: so is good to have an unconstrained problem, but is computationally expensive because we have to apply four time the algorithm to find the minimum value of a quadratic function.