Chapter 5

Constrained Optimization

In this chapter we will focus on constrained programming problems. We will describe in depth some classic methods and also a few tailored algorithms for solving large scale constrained problems in data science (we will put again more emphasis on practical algorithms with a machine learning flavor) together with some conditions that allow us to characterize (and identify) the solutions of a given problem. Then, we will give some examples of real unconstrained problems in data science.

5.1 Optimality conditions for constrained problems

In this section, we consider a problem of the form

$$\begin{array}{l} \min f(x) \\ x \in C \end{array} \tag{5.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a convex function and $C \subseteq \mathbb{R}^n$ is a convex set. We start by giving some definition that can help us to characterize the solutions of the problem.

Definition 5.1 *Let $C \subseteq \mathbb{R}^n$ be a nonempty set. We define the set of* feasible directions *for C in $\bar{x} \in C$ the following set $F(\bar{x})$:*

$$F(\bar{x}) = \{d \in \mathbb{R}^n,\ d \neq 0 :\ \exists\ \delta > 0\ s.t.\ \bar{x} + \alpha d \in C,\ \forall\ \alpha \in (0, \delta)\ \}.$$

Using sets $D(x)$ (defined in Chapter 4) and $F(x)$ it is possible to characterize the local minima of our problem.

Theorem 5.2 *Let $f \in C(\mathbb{R}^n)$. If $x^* \in \mathcal{F}$ is a local (global) minimum of Problem* (5.1) *then*

$$D(x^*) \cap F(x^*) = \emptyset \tag{5.2}$$

Proof. Let us assume by contradiction that (5.2) does not hold. Then it would exist a vector $\bar{d} \in D(x^*) \cap F(x^*)$. Hence, taking into account the way we defined

the two sets D and F, there would exist two positive constants δ_1 and δ_2 such that:

$$f(x^* + \alpha\bar{d}) < f(x^*) \qquad \text{for all} \qquad \alpha \in (0, \delta_1) \tag{5.3}$$
$$x^* + \alpha\bar{d} \in C \qquad \text{for all} \qquad \alpha \in (0, \delta_2). \tag{5.4}$$

Thus by choosing scalar $\alpha \in (0, \min\{\delta_1, \delta_2\})$ we would have that x^* is not a local minimum for f over C. □

The result reported above enable us to give some preliminary characterization of local minimum for a constrained problem. Now we formally describe the set of feasible directions for convex feasible sets.

Proposition 5.3 *Let $C \subseteq \mathbb{R}^n$ be a convex set and $\bar{x} \in C$ a feasible point for C. If $C \neq \{\bar{x}\}$, for all $x \in C$ with $x \neq \bar{x}$, direction*

$$d = x - \bar{x}$$

is feasible for C in $\bar{x}$.

Proof. Exercise. □

We can thus give a new definition of feasible directions for convex sets:

$$F(\bar{x}) = \{d \in \mathbb{R}^n,\ d = x - \bar{x},\ x \in C,\ x \neq \bar{x}\}.$$

Now we give necessary conditions for identifying local minima.

Proposition 5.4 *Let $x^* \in C$ be local minimum for problem*

$$\begin{aligned} &\min f(x) \\ &x \in C \end{aligned}$$

with $C \subseteq \mathbb{R}^n$ convex and $f \in C^1(\mathbb{R}^n)$. Then

$$\nabla f(x^*)^\top (x - x^*) \geq 0 \quad \forall\ x \in C.$$

Proof. If $C = \{x^*\}$ we are done. Let us hence assume that there exists a point $x \in C$ such that $x \neq x^*$. By definition, $d = x - x^*$ is a feasible direction for C in x^*. Using Proposition 5.2, we know that there is no feasible descent direction in x^* (since x^* is a local minimum). Let us then assume, by contradiction, that a direction $d = x - x^*$ exists such that $\nabla f(x^*)^\top d < 0$, we would have that d is a feasible descent direction in x^*, thus contradicting the fact that x^* is a minimum. Hence (i) is proved.

□

A geometrical representation of first order optimality condition is reported in Figure 5.1.

When considering convex problems, (i) is a necessary and sufficient condition to get a global minimum.

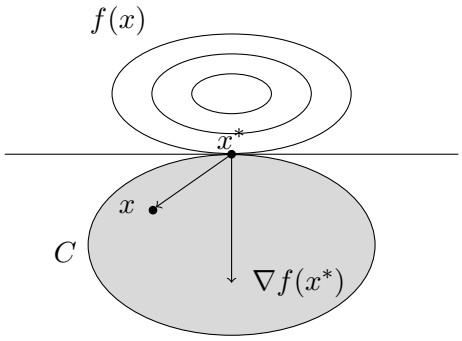

Figure 5.1: Geometrical representation of first order optimality condition.

Proposition 5.5 *Let $C \subseteq \mathbb{R}^n$ be a convex set and $f \in C^1(\mathbb{R}^n)$ be a convex function. Point $x^* \in C$ is a global minimum of problem*

$$\begin{aligned} &\min f(x) \\ &x \in C, \end{aligned}$$

if and only if

$$\nabla f(x^*)^\top (x - x^*) \geq 0 \quad \forall\ x \in C.$$

Proof. Proof of necessary part follows from Proposition 5.4. Now we prove the sufficient part. We hence have

$$\nabla f(x^*)^\top (x - x^*) \geq 0 \quad \forall\ x \in C.$$

From convexity of f we have that for all $x \in C$ we can write

$$f(x) \geq f(x^*) + \nabla f(x^*)^\top (x - x^*).$$

Thus we can write

$$f(x) - f(x^*) \geq \nabla f(x^*)^\top (x - x^*) \geq 0,$$

for all $x \in C$. Therefore,

$$f(x) \geq f(x^*) \quad \forall\ x \in C.$$

We then proved that x^* is a global minimum. □

Keep in mind that strict convexity ensures uniqueness of the global minimum.

5.2 Projected gradient method

It is not possible to directly use the gradient method when dealing with (5.1), since the iterates

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k)$$

might be such that

$$x_{k+1} \notin C.$$

In order to overcome this issue, we simply choose the point in C nearest to $x_k - \alpha_k \nabla f(x_k)$ as the new iterate, that is we project the given point over the set C. This is basically the idea behind the projected gradient method. Before we describe in depth the method and the theoretical issues related to it, we formally introduce the projection and give its basic properties.

Definition 5.6 *Let us consider:*

- $\|\cdot\|$ *euclidean norm;*
- $C \subset \mathbb{R}^n$ *a closed convex set;*
- $\bar{x} \in \mathbb{R}^n$ *a given point.*

We define projection of $\bar{x}$ over C the solution $\rho_C(\bar{x})$ of the following problem:

$$\begin{array}{l} \min\limits_{x \in \mathbb{R}^n} \dfrac{1}{2}\|x - \bar{x}\|^2 \\ x \in C. \end{array} \tag{5.5}$$

We recall in the next proposition some properties of the projection:

Proposition 5.7 *Let us consider:*

- $\|\cdot\|$ *euclidean norm;*
- $C \subset \mathbb{R}^n$ *a closed convex set;*
- $\bar{x} \in \mathbb{R}^n$ *a given point.*

Then,

- $\bar{x} = \rho_C(\bar{x}), \quad \forall\ \bar{x} \in C;$
- $x^* \in C$ *is projection of $\bar{x}$ over C, that is $x^* = \rho_C(\bar{x})$, if and only if*

$$(\bar{x} - x^*)^\top (x - x^*) \leq 0 \quad \forall\ x \in C;$$

- *projection operator is continuous and non-expansive:*

$$\|\rho_C(y) - \rho_C(z)\| \leq \|y - z\| \quad \forall\ y, z \in \mathbb{R}^n.$$

Proof. We now prove the three points:

(*i*) It straightforwardly comes from definition of projection[1].

(*ii*) Projection $x^* = \rho_C(\bar{x})$ is the unique solution[2] of problem (5.5). By taking into account (5.5), we have that $x^* = \rho_C(\bar{x})$ if and only if

$$\nabla f(x^*)^\top (x - x^*) \geq 0$$

for all $x \in C$. By taking into account that, in our case, $\nabla f(x^*) = x^* - \bar{x}$, we get the result.

[1] Notice that $\frac{1}{2}\|x - \bar{x}\|^2 \geq 0$ for all $x \in C$.

[2] Notice that we are minimizing a strictly convex function ($\nabla^2 f(x) = I$) over a convex set.

(*iii*) For two arbitrarily chosen points $y, z \in \mathbb{R}^n$, we can write (by taking into account point (*ii*))

$$(z - \rho_C(z))^\top (\rho_C(y) - \rho_C(z)) \leq 0$$

and

$$(y - \rho_C(y))^\top (\rho_C(z) - \rho_C(y)) \leq 0.$$

Summing up the two inequalities we get

$$(z - \rho_C(z) - y + \rho_C(y))^\top (\rho_C(y) - \rho_C(z)) \leq 0.$$

Thus we can write

$$\|\rho_C(y) - \rho_C(z)\|^2 - (\rho_C(y) - \rho_C(z))^\top (y - z) \leq 0.$$

Using Cauchy-Schwarz inequality we have

$$\|\rho_C(y) - \rho_C(z)\|^2 \leq (\rho_C(y) - \rho_C(z))^\top (y - z) \leq \|\rho_C(y) - \rho_C(z)\| \|y - z\|.$$

Thus we get our result.

□

A geometrical representation of projection operator is reported in Figure 5.2.

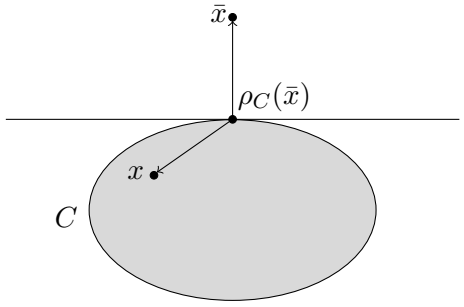

Figure 5.2: Geometrical representation of projection operator.

The detailed scheme of the projected gradient algorithm is given below.

Algorithm 10 `Projected gradient method`

1 Choose a point $x_1 \in C$
2 For $k = 1, \ldots$
3 Set $\hat{x}_k = \rho_C(x_k - s_k \nabla f(x_k))$, with $s_k > 0$
4 If $\hat{x}_k$ satisfies some specific condition, then STOP
5 Set $x_{k+1} = x_k + \alpha_k(\hat{x}_k - x_k)$, with $\alpha_k \in (0, 1]$ suitably chosen stepsize
6 End for

It is easy to see that the method, at iteration k, generates a feasible descent direction for f in x_k. From properties of the projection, we have:

$$(x_k - s_k \nabla f(x_k) - \hat{x}_k)^\top (x - \hat{x}_k) \leq 0, \quad \forall\ x \in C.$$

By setting $x = x_k$, we can write

$$(x_k - s_k \nabla f(x_k) - \hat{x}_k)^\top (x_k - \hat{x}_k) \leq 0$$

and by properly rewriting, we get:

$$\nabla f(x_k)^\top (\hat{x}_k - x_k) \leq -\frac{1}{s_k} \|x_k - \hat{x}_k\|^2. \tag{5.6}$$

Thus d_k is a descent direction if $\|x_k - \hat{x}_k\| \neq 0$.
In Figure 5.3, we report an iteration of the projected gradient method.

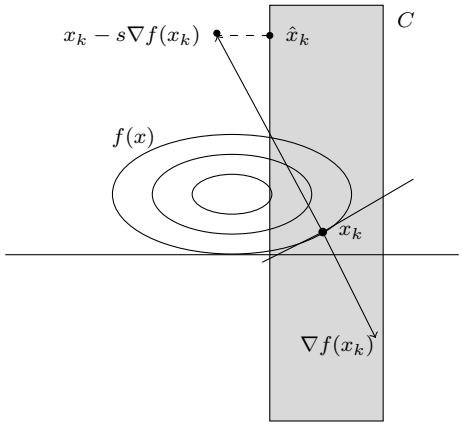

Figure 5.3: Iteration of the projected gradient.

The method can be implemented either by fixing s_k to constant value and using some line search[3] to get $\alpha_k \in (0, 1]$, or by fixing α_k and implementing a search over s_k (in this case we have a curvilinear search over C). Here we only consider the case $s_k = s > 0$ constant.

By taking into account previous results we can also obtain a condition that can be used to stop the algorithm at Step 3.

Proposition 5.8 *Let $x^* \in C$ be local minimum of the problem:*

$$\begin{aligned} &\min f(x) \\ &x \in C \end{aligned}$$

with $C \subseteq \mathbb{R}^n$ convex and $f \in C^1(\mathbb{R}^n)$. Then

$$x^* = \rho_C \left(x^* - s \nabla f(x^*) \right),$$

with $s > 0$.

[3]Line searches are similar to the ones described in the unconstrained case. Main difference is that feasibility needs to be ensured, thus we get a stepsize $\alpha_k \in (0, 1]$.

Proof. We first recall that

$$\hat{x}^* = \rho_C \left(x^* - s\nabla f(x^*) \right).$$

Taking into account previous results we get that

$$\nabla f(x^*)^\top (x - x^*) \geq 0,$$

for all $x \in C$. From the properties of projection operator, we get

$$(x^* - s\nabla f(x^*) - \hat{x}^*)^\top (x^* - \hat{x}^*) \leq 0.$$

Combining the two inequalities, we get

$$(x^* - \hat{x}^*)^\top (x^* - \hat{x}^*) \leq -s\nabla f(x^*)(\hat{x}^* - x^*) \leq 0.$$

That is

$$\|x^* - \hat{x}^*\| = 0,$$

and we have

$$x^* = \hat{x}^* = \rho_C \left(x^* - s\nabla f(x^*) \right).$$

□

When f is convex we have a necessary and sufficient optimality condition.

Proposition 5.9 *Let $C \subseteq \mathbb{R}^n$ be a closed convex set and $f \in C^1(\mathbb{R}^n)$ be a convex function. Point $x^* \in C$ is a global minimum of the problem*

$$\begin{array}{l} \min f(x) \\ x \in C \end{array}$$

if and only if

$$x^* = \rho_C \left(x^* - s\nabla f(x^*) \right),$$

with $s > 0$.

We now fix $s = 1/L$ and $\alpha_k = 1/L$. We then have that a generic iterate of the projected gradient method is

$$x_{k+1} = x_k - \frac{1}{L} g_C(x_k),$$

where $g_C(x_k) = x_k - \hat{x}_k$ is commonly called the *gradient mapping*. It is possible to prove the following result for the method

Proposition 5.10 *Let us consider a convex function with Lipschitz continuous gradient having constant $L > 0$. Then, we have*

$$f(\hat{x}_k) - f(x_k) \leq -\frac{\|g_C(x_k)\|^2}{2L}.$$

If we further have that f is σ-strongly convex, we have

$$g_C(x)^\top (x - x^*) \geq \frac{\sigma}{2}\|x - x^*\|^2 + \frac{1}{2L}\|g_C(x)\|^2.$$

Using these results we can analyze the theoretical properties of the method by adapting the theoretical analysis we already carried out for the gradient method.

Theorem 5.11 *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function with Lipschitz continuous gradient having Lipschitz constant $L > 0$. Projected gradient method with fixed stepsize $\alpha_k = 1/L$ satisfies:*

$$f(x_{k+1}) - f(x^*) \leq \frac{2L}{k}\|x_1 - x^*\|^2.$$

Furthermore, if f is σ-strongly convex, projected gradient method with fixed stepsize $\alpha_k = 1/L$ satisfies:

$$\|x_{k+1} - x^*\|^2 \leq \left(1 - \frac{\sigma}{L}\right)^k \|x_1 - x^*\|^2.$$

Proof. The proof of the first part directly follows from the analysis of the gradient method. We consider now the case when f is σ-strongly convex. Taking into account the iteration k we can write

$$\begin{aligned}
\|x_{k+1} - x^*\|^2 &= \|x_k - \frac{1}{L}g_C(x_k) - x^*\|^2 \\
&= \|x_k - x^*\|^2 - \frac{2}{L}g_C(x_k)^\top(x_k - x^*) + \frac{1}{L^2}\|g_C(x_k)\|^2 \\
&\leq \|x_k - x^*\|^2 - \frac{2}{L}\left[\frac{\sigma}{2}\|x_k - x^*\|^2 + \frac{1}{2L}\|g_C(x_k)\|^2\right] + \frac{1}{L^2}\|g_C(x_k)\|^2 \\
&= (1 - \frac{\sigma}{L})\|x_k - x^*\|^2.
\end{aligned}$$

By induction, we get

$$\|x_{k+1} - x^*\|^2 \leq \left(1 - \frac{\sigma}{L}\right)^k \|x_1 - x^*\|^2.$$

□

Projected gradient guarantees similar convergence rates as the gradient method in the unconstrained case. Main issue here is that for huge-scale problems computing the projection might be an expensive task. We see now a few examples where projection can be performed at a reasonable cost.

Example 5.12 *We now give two example of projections over a set:*

- *ℓ_2 ball: $C = \{x \in \mathbb{R}^n : \|x\|_2 \leq 1\}$. In this case we have*

$$\rho_C(\bar{x}) = \begin{cases} \bar{x} & \text{if } \|\bar{x}\|_2 \leq 1 \\ \dfrac{\bar{x}}{\|\bar{x}\|_2} & \text{if } \|\bar{x}\|_2 > 1 \end{cases};$$

- *box constraints:* $C = \{x \in \mathbb{R}^n : \ l \leq x \leq u\}$. *In this case we describe the projection component-wise*

$$\rho_C(\bar{x})_i = \begin{cases} l_i & \text{if } \bar{x}_i < l_i \\ \bar{x}_i & \text{if } l_i \leq \bar{x}_i \leq u_i \\ u_i & \text{if } \bar{x}_i > u_i \end{cases},$$

  for $i = 1, \ldots, n$.

In both cases we have that projection costs $\mathcal{O}(n)$.

5.3 Frank-Wolfe method

The Frank-Wolfe method (aka conditional gradient method or reduced gradient method) is an iterative first-order optimization algorithm originally proposed by Marguerite Frank and Philip Wolfe in 1956 to solve quadratic programming problems with linear constraints. It has seen an impressive revival recently due to its nice properties compared to projected gradient methods, in particular for machine learning applications. In this section, we will describe in depth the method and its theoretical properties. Furthermore, we will try to explain why people in data science use this method in practice. Here, we consider a problem of the form

$$\begin{aligned} &\min f(x) \\ &x \in C \end{aligned} \tag{5.7}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a convex function with Lipschitz continuous gradient having constant $L > 0$ and $C \subseteq \mathbb{R}^n$ is a convex compact[4] set. We start by giving some definition that can help in the theoretical analysis. In particular, we define the *diameter* of the set C as

$$D = \max_{x,y \in C} \|x - y\|_2.$$

From compactness of C, we have that D is a finite value.

Now, we give a detailed description of the algorithm. We start with a feasible solution and, at each iteration, we define a descent direction in the current iterate x_k by solving the problem:

$$\min_{x \in C} \nabla f(x_k)^\top (x - x_k)$$

We notice that this is equivalent to minimize the linear approximation of f in x_k:

$$\min_{x \in C} f(x_k) + \nabla f(x_k)^\top (x - x_k)$$

From compactness[5] of C, we have that there exists a solution $\hat{x}_k \in C$ for the linearized problem. If

$$\nabla f(x_k)^\top (\hat{x}_k - x_k) = 0,$$

[4] A compact set is both closed and bounded.

[5] We use the Weierstrass theorem here. If we minimize a continuous function over a compact set, we can always be sure there exists a global minimum for the problem.

then we have

$$0 = \nabla f(x_k)^\top (\hat{x}_k - x_k) \leq \nabla f(x_k)^\top (x - x_k) \quad \forall\ x \in C$$

and x_k satisfies first order optimality conditions. if

$$\nabla f(x_k)^\top (\hat{x}_k - x_k) < 0$$

we have a new descent direction in x_k:

$$d_k = \hat{x}_k - x_k.$$

Thus we can have a new iterate

$$x_{k+1} = x_k + \alpha_k d_k$$

with $\alpha_k \in (0, 1]$ calculated by means of a line search. We report here the scheme of the method:

Algorithm 11 `Frank-Wolfe method`

1 Choose a point $x_1 \in C$
2 For $k = 1, \ldots$
3 Set $\hat{x}_k = \arg\min_{x \in C} \nabla f(x_k)^\top (x - x_k)$
4 If $\hat{x}_k$ satisfies some specific condition, then STOP
5 Set $x_{k+1} = x_k + \alpha_k(\hat{x}_k - x_k)$, with $\alpha_k \in (0, 1]$ suitably chosen stepsize
6 End for

We prove convergence of the method with diminishing stepsize $\alpha = \dfrac{2}{k+1}$ in the next theorem:

Theorem 5.13 *Let f be a convex function with Lipschitz continuous gradient having constant $L > 0$. The Frank-Wolfe method with stepsize $\alpha_k = \frac{2}{k+1}$ satisfies the following inequality:*

$$f(x_{k+1}) - f(x^*) \leq \frac{2LD^2}{k+1}, \tag{5.8}$$

for all $k \geq 1$.

Proof. By using first order convexity properties, we have

$$f(x) \geq f(x_k) + \nabla f(x_k)^\top (x - x_k), \quad \forall\ x \in C.$$

Minimizing on both sides of the inequality over C, we get

$$f(x^*) \geq f(x_k) + \nabla f(x_k)^\top (\hat{x}_k - x_k),$$

that can be rewritten as

$$f(x^*) - f(x_k) \geq \nabla f(x_k)^\top (\hat{x}_k - x_k),$$

or equivalently as follows

$$-(f(x_k) - f(x^*)) \geq \nabla f(x_k)^\top (\hat{x}_k - x_k). \tag{5.9}$$

We consider the point

$$x_{k+1} = x_k + \alpha_k d_k = x_k + \alpha_k(\hat{x}_k - x_k)$$

and, using Lipschitz continuity of the gradient an inequality (5.9), we write

$$\begin{aligned} f(x_{k+1}) - f(x^*) &\leq f(x_k) + \alpha_k \nabla f(x_k)^\top (\hat{x}_k - x_k) + \frac{\alpha_k^2 L}{2}\|x_k - \hat{x}_k\|^2 - f(x^*) \\ &\leq f(x_k) - f(x^*) - \alpha_k(f(x_k) - f(x^*)) + \frac{\alpha_k^2 L}{2}\|x_k - \hat{x}_k\|^2 \\ &\leq (1 - \alpha_k)(f(x_k) - f(x^*)) + \frac{\alpha_k^2 L}{2}\|x_k - \hat{x}_k\|^2 \\ &\leq (1 - \alpha_k)(f(x_k) - f(x^*)) + \frac{\alpha_k^2 L D^2}{2}. \end{aligned}$$

We set $r_{k+1} = f(x_{k+1}) - f(x^*)$ and rewrite

$$r_{k+1} \leq (1 - \alpha_k) r_k + \frac{\alpha_k^2 L D^2}{2}. \tag{5.10}$$

By induction we can show

$$r_{k+1} \leq \frac{2LD^2}{k+1}.$$

We first prove that the inequality holds for $k = 1$. By means of inequality (5.10), we get

$$r_2 \leq (1 - \alpha_1) r_1 + \frac{\alpha_1^2 L D^2}{2}.$$

Since $\alpha_1 = 1$, we have

$$r_2 \leq \frac{LD^2}{2} \leq LD^2.$$

Now we assume that inequality

$$r_{k+1} \leq \frac{2LD^2}{k+1}$$

holds for any $k \geq 1$, we want to show that it holds also for $k + 1$. By using inequality (5.10) again, we get

$$\begin{aligned} r_{k+2} &\leq (1 - \alpha_{k+1}) r_{k+1} + \frac{\alpha_{k+1}^2 L D^2}{2} \\ &\leq \left(1 - \frac{2}{k+2}\right) \frac{2LD^2}{k+1} + \frac{LD^2}{2}\left(\frac{2}{k+2}\right)^2 \\ &= 2LD^2 \left(\frac{k}{(k+1)(k+2)} + \frac{1}{(k+2)^2}\right) \\ &\quad \text{(using the fact that } k+1 \leq k+2) \\ &\leq 2LD^2 \left(\frac{k}{(k+1)(k+2)} + \frac{1}{(k+1)(k+2)}\right) \\ &= \frac{2LD^2}{k+2}. \end{aligned}$$

Thus concluding the proof. □

This result implies that the convergence rate is $\mathcal{O}(\frac{1}{k})$. It is possible to improve the rate (i.e., getting a linear rate) if we make stronger assumptions on the problem, that is:

- feasible sets with special structure (like, e.g., polytopes);
- function is σ-strongly convex;
- optimal solution in the interior.

In Figure 5.4, we report an iteration of the Frank-Wolfe method.

For a given point $x \in C$ we can define a simple dual function:

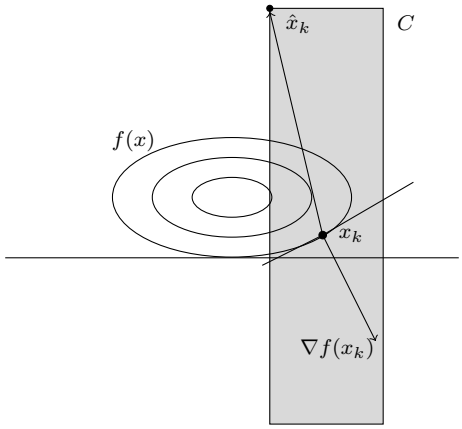

Figure 5.4: Iteration of the Frank-Wolfe method.

$$w(x) = \min_{z \in C} f(x) + \nabla f(x)^\top (z - x).$$

This minimum is always attained since C is compact and the linear function is continuous in z. A weak duality result holds in this case

Proposition 5.14 *For all pairs $x, y \in C$ it holds that*

$$w(x) \leq f(y).$$

Proof. We have:

$$\begin{aligned} w(x) &= \min_{z \in C} f(x) + \nabla f(x)^\top (z - x) \\ &\leq f(x) + \nabla f(x)^\top (y - x) \\ &\leq f(y). \end{aligned}$$

□

For a given point $x \in C$ we can define the duality gap:

$$g(x) = f(x) - w(x) = \max_{z \in C} \nabla f(x)^\top (x - z) = -\min_{z \in C} \nabla f(x)^\top (z - x).$$

By the weak duality result we have

$$g(x) \geq f(x) - f(x^*) \geq 0, \quad \forall\ x \in C.$$

Since primal error (i.e., $f(x) - f(x^\star)$) is not computable (x^* unknown), duality gap represents a good optimality measure, e.g. as a stopping criterion. Keep in mind that, since we solve problem at Step 3 of the algorithm, we have $g(x)$ for free at each iteration. So, if we want a primal gap $0 \leq f(x) - f(x^*) \leq \epsilon$, we need

$$f(x) - f(x^*) \leq g(x) \leq \epsilon.$$

Thus we can stop the method when

$$\nabla f(x_k)^\top (\hat{x}_k - x_k) \geq -\epsilon.$$

The Frank-Wolfe method is really appealing in the machine learning context for two main reasons:

- the cost per iteration is much smaller than the one we have for projected gradient method (Frank-Wolfe method is a projection-free algorithm);
- the iterates keep desirable structure (like, e.g., sparsity).

We will discuss in depth these two facts. First of all, it is easy to see that solving problem at Step 2 of the algorithm costs less than projecting over C in general. Indeed, if we think about a polyhedral feasible set, at each iteration we need to solve a linear program, while projection is equivalent to solve a quadratic programming problem.

5.3.1 Frank-Wolfe method for structured feasible sets

At each iteration of the algorithm we solve the problem:

$$\hat{x}_k = \arg\min_{x \in C} \nabla f(x_k)^\top (x - x_k)$$

When C is a polytope, we know, by means of the fundamental theorem of linear programming that one of the vertices is solution of the linear program. Hence, we get that the Frank-Wolfe iteration in some cases has linear cost:

Unit simplex. feasible set is

$$C = \{x \in R^n :\ e^\top x = 1,\ x \geq 0\} = conv(\{e_i,\ i = 1, \ldots, n\});$$

solution in this case is

$$\hat{x}_k = e_{i_k},$$

with $i_k = \arg\min_i \nabla_i f(x_k)$. It is easy to see that cost per iteration is $\mathcal{O}(n)$. A 3D simplex can be seen in Figure (5.5). The Frank-Wolfe scheme is

Algorithm 12 `Frank-Wolfe method for unit simplex`

1 Set $x_1 = e_i$, with $i = 1, \ldots, n$
2 For $k = 1, \ldots$
3 Set $\hat{x}_k = e_{i_k}$, with $i_k = \arg\min_i \nabla_i f(x_k)$.
4 If $\hat{x}_k$ satisfies some specific condition, then STOP
5 Set $x_{k+1} = x_k + \alpha_k(\hat{x}_k - x_k)$, with $\alpha_k = \frac{2}{k+1}$
6 End for

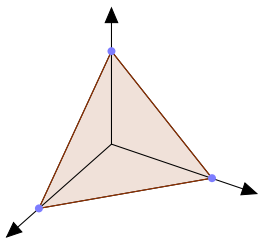

Figure 5.5: Unit simplex.

ℓ_1-ball. feasible set is

$$C = \{x \in R^n : \ \|x\|_1 \leq 1\} = conv(\{\pm e_i, \ i = 1, \ldots, n\});$$

solution in this case is

$$\hat{x}_k = sign(-\nabla_{i_k} f(x_k)) \cdot e_{i_k},$$

with $i_k = \arg\max_i |\nabla_i f(x_k)|$. It is easy to see that cost per iteration is $\mathcal{O}(n)$. A 3D ℓ_1 ball can be seen in Figure (5.6). The Frank-Wolfe scheme is

Algorithm 13 `Frank-Wolfe method for` ℓ_1 `ball`

1 Set $x_1 = \pm e_i$, with $i = 1, \ldots, n$
2 For $k = 1, \ldots$
3 Set $\hat{x}_k = sign(-\nabla_{i_k} f(x_k)) \cdot e_{i_k}$, with $i_k = \arg\max_i |\nabla_i f(x_k)|$.
4 If $\hat{x}_k$ satisfies some specific condition, then STOP
5 Set $x_{k+1} = x_k + \alpha_k(\hat{x}_k - x_k)$, with $\alpha_k = \frac{2}{k+1}$
6 End for

It is further easy to see that at most one new nonzero is included at each iteration in both cases. Support of the solution (i.e., number of nonzero component of x_k) is upper bounded by k. Frank-Wolfe can be seen in this case as a variant of coordinate descent (we use coordinate directions at each step).

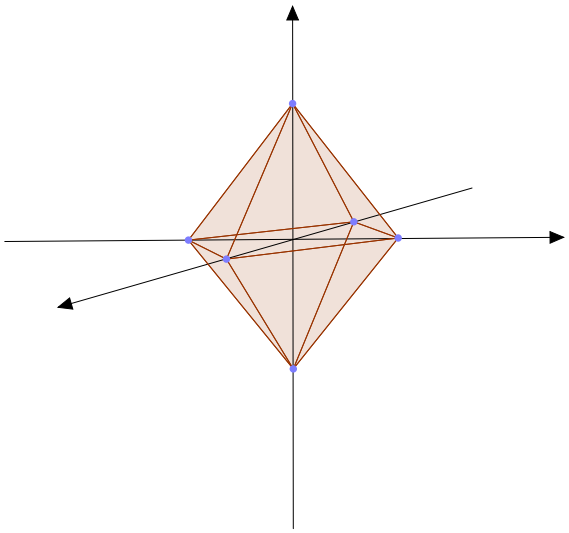

Figure 5.6: ℓ_1 ball.

If we choose a re-parameterization of C by a surjective linear or affine map $M : \hat{C} \to C$ then

$$\min_{x \in C} f(x) \equiv \min_{y \in \hat{C}} \hat{f}(y)$$

with $\hat{f}(y) = f(My)$. In this case is easy to see that every iteration of FW Algorithm remains the same (thanks to $\nabla \hat{f}(y) = M^\top \nabla f(My)$). Hence we have that Frank-Wolfe is invariant under "distortion" (existing approaches in optimization strongly depend on distortion of the domain). In order to better understand this concept, we consider problem

$$\begin{array}{l} \min f(x) \\ x \in C = conv\{v_1, \ldots, v_p\}, \end{array} \tag{5.11}$$

where C is a polytope that can be described in terms of its vertices. We use *bary-centric coordinates* to reparameterize the problem. In this case M contains the vertices as columns

$$M = [v_1 \ldots v_p]$$

and $\hat{C}$ is just the unit simplex. It is easy to see that solving the original problem using the Frank-Wolfe method is equivalent to solve (by means of Frank-Wolfe) the reparameterized problem

$$\begin{array}{ll} \min & f(My) \\ s.t. & e^\top y = 1 \\ & y \geq 0 \end{array} \tag{5.12}$$

Indeed, at each iteration we have

$$\begin{array}{ll} \min & \nabla \hat{f}(y_k)^\top (y - y_k) \\ s.t. & e^\top y = 1 \\ & y \geq 0 \end{array} \tag{5.13}$$

and by taking into account the expression of $\hat{f}$, we get

$$\nabla \hat{f}(y_k)^\top (\hat{y}_k - y_k) = \left(M^\top \nabla f(My_k)\right)^\top (\hat{y}_k - y_k) = \nabla f(My_k)^\top (M\hat{y}_k - My_k).$$

By further considering that $x_k = My_k$ and $\hat{x}_k = M\hat{y}_k$, we get

$$\nabla \hat{f}(y_k)^\top (\hat{y}_k - y_k) = \nabla f(x_k)^\top (\hat{x}_k - x_k).$$

5.3.2 Sublinear rate in Frank-Wolfe method

The sublinear rate in the Frank-Wolfe algorithm is due to the use a *linear minimization oracle* over C that is defined as follows

$$LMO(y) = \arg\min_{x \in C} y^\top x.$$

A classic example of sublinear rate for the Frank Wolfe algorithm is obtained when C is a polytope and x^* is on the boundary of the feasible set. This is because the iterates of the algorithm start to zig-zag between the vertices defining the face containing the solution x^*. We report an example of the zig-zagging phenomenon in Figure 5.7.

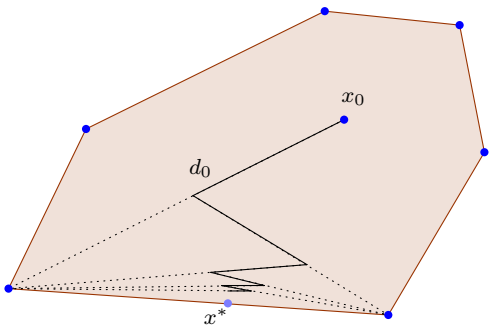

Figure 5.7: zig-zagging phenomenon.

While in general is not possible to improve the sublinear rate of the Frank-Wolfe algorithm, in the polyhedral case there exist some variants[6] of the Frank-Wolfe algorithm that guarantee (under suitable assumptions like, e.g. σ-strong convexity) convergence at a linear rate. We report here two well known variants.

[6]Those variants, as we will see, are still based on linear minimization oracles.

5.3.3 Away-step Frank-Wolfe method

This modification of the Frank-Wolfe method was proposed by Wolfe (1970). As we have previously seen Frank-Wolfe directions are always directed towards extreme points. When we are close to the optimum (and the optimum is on the boundary) directions get more and more orthogonal to the gradient thus getting the so-called zig-zagging phenomenon. In order to avoid this, Wolfe suggested to include directions pointing away from extreme points. Linear convergence can be obtained in case f is σ-strongly convex and C is polyhedral. Here, we consider a problem of the form

$$\begin{aligned} &\min f(x) \\ &x \in C = conv\{v_1, \dots, v_p\} \end{aligned} \tag{5.14}$$

If we call $V = \{v_1, \dots, v_p\}$, we know that, at step k, iterate is represented as a sparse convex combination of at most k vertices $S_k \subseteq V$. We report the scheme of the method below:

Algorithm 14 `Away-step Frank-Wolfe method`

1 Choose a point $x_1 \in C$
2 For $k = 1, \dots$
3 Set $\hat{x}_k^{FW} = \arg\min_{x \in C} \nabla f(x_k)^\top (x - x_k)$
4 If $\hat{x}_k^{FW}$ satisfies some specific condition, then STOP
5 Set $\hat{x}_k^{AS} = \arg\max_{x \in S_k} \nabla f(x_k)^\top (x - x_k)$
6 Set $d_k^{FW} = \hat{x}_k^{FW} - x_k$ and $d_k^{AS} = x_k - \hat{x}_k^{AS}$
7 If $\nabla f(x_k)^\top d_k^{FW} \leq \nabla f(x_k)^\top d_k^{AS}$
Then set $d_k = d_k^{FW}$ and $\bar{\alpha} = 1$
Else set $d_k = d_k^{AS}$ and $\bar{\alpha} = \max_\beta \{x_k + \beta d_k^{AS} \in C\}$
8 End If
9 Set $x_{k+1} = x_k + \alpha_k d_k$, with $\alpha_k \in (0, \bar{\alpha}]$ suitably chosen stepsize
10 Calculate S_{k+1} set of currently used vertices.
11 End for

Hence, at each iteration we calculate the classic Frank-Wolfe direction and the so-called away-step direction, that is a direction pointing away from the worst vertex (i.e., the one with highest value of the linearized function) describing the current iterate. Then we choose the best between the two and perform a line search along that direction (See Step 9). Finally, we update S_k. In Figure 5.8, we show how the away-step Frank-Wolfe works in practice. We notice that storing and updating S_k might be costly in practice. Furthermore, there might be multiple ways to represent iterate k as combination of vertices.

5.3.4 Pairwise Frank-Wolfe method

The Pairwise Frank-Wolfe method was first described by Mitchel et al. for the polytope distance problem. This method is strongly related to classic SMO

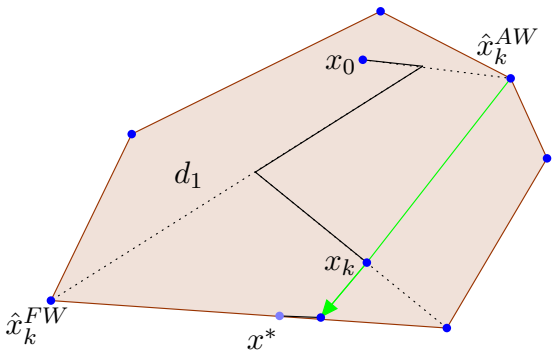

Figure 5.8: Behavior of the away-step Frank-Wolfe method.

algorithms in machine learning. The main idea is moving weight from the away vertex to the Frank-Wolfe vertex. In practice, at each iteration we use the search direction $d_k = d_k^{FW} + d_k^{AS}$. Linear convergence can be obtained under similar assumptions as the away-step Frank-Wolfe method. What we actually we get is that the linear rate is more loose than away-step variant. Anyway, the method is very efficient in practice. We report the scheme of the method below:

Algorithm 15 `Pairwise Frank-Wolfe method`

1 Choose a point $x_1 \in C$
2 For $k = 1, \ldots$
3 Set $d_k = d_k^{FW} + d_k^{AS}$ and $\bar{\alpha} = \max_\beta \{x_k + \beta d_k \in C\}$
4 Set $x_{k+1} = x_k + \alpha_k d_k$, with $\alpha_k \in (0, \bar{\alpha}]$ suitably chosen stepsize
5 Calculate S_{k+1} set of currently used vertices.
6 End for

In Figure 5.9, we show how the pairwise Frank-Wolfe works in practice.

5.4 Fully corrective Frank-Wolfe method

In this section, we consider the more general problem:

$$\begin{aligned} &\min f(x) \\ &x \in C \end{aligned} \tag{5.15}$$

where f is continuously differentiable and C is a compact convex set. The fully corrective Frank Wolfe method (aka Simplicial Decomposition) represents a class of methods used for dealing with convex problems. It was first introduced by Holloway (1970) and then further studied in other papers. The method basically uses an iterative *inner approximation* of the feasible set C. In practice, the feasible set is approximated with the convex hull of an ever expanding finite

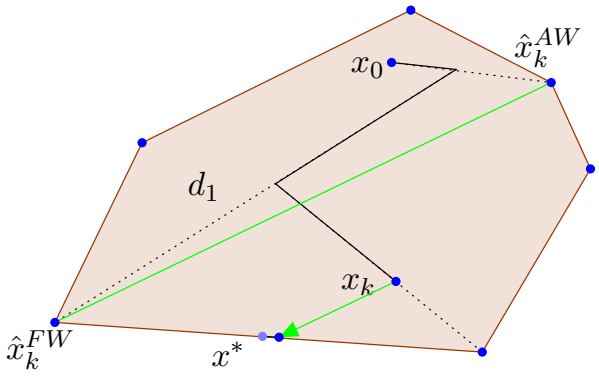

Figure 5.9: Behavior of the pairwise Frank-Wolfe method.

set $C_k = \{\hat{x}_0, \hat{x}_2, \ldots, \hat{x}_k\}$ where $\hat{x}_i$, $i = 0, \ldots,$ k are extreme points of C. We denote this set with $conv(C_k)$:

$$conv(C_k) = \{x \mid x = \sum_{i=0}^{k} \lambda_i \hat{x}_i, \sum_{i=0}^{k} \lambda_i = 1, \lambda_i \geq 0\} \tag{5.16}$$

At each iteration, it is possible to add new extreme points to C_k in such a way that a function reduction is guaranteed when minimizing the objective function over the convex hull of the new (enlarged) set of extreme points. If the algorithm does not find at least one new point, the solution is optimal and the algorithm terminates.

The use of the proposed method is particularly indicated when the following two conditions are satisfied:

1. Minimizing a linear function over C is much simpler than solving the original nonlinear problem;

2. Minimizing the original objective function over the convex hull of a relatively small set of extreme points is much simpler than solving the original nonlinear problem (i.e., tailored algorithms can be used for tackling the specific problem in our case).

The first condition is needed due to the way a new extreme point is generated. Indeed, this new point is the solution of the following linear programming problem

$$\begin{array}{ll} \min & \nabla f(x_k)^\top (x - x_k) \\ \text{s.t.} & x \in C \end{array} \tag{5.17}$$

where a linear approximation calculated at the last iterate x_k (i.e., the solution obtained by minimizing f over $conv(C_{k-1})$) is minimized over the original feasible set C.

Below, we report the detailed scheme related to the algorithm.

Algorithm 16 `Fully corrective Frank-Wolfe method`

1 Choose an extreme point $\hat{x}_0$ of C, then set $C_0 = \{\hat{x}_0\}$ and $x_1 = \hat{x}_0$
2 For $k = 1, \ldots$
3 Set $\hat{x}_k = \arg\min_{x \in C} \nabla f(x_k)^\top (x - x_k)$
4 If $\hat{x}_k$ satisfies some specific condition, then STOP
5 Set $C_k = C_{k-1} \cup \{\hat{x}_k\}$
6 Set $x_{k+1} = \arg\min_{x \in conv(C_k)} f(x)$
7 End for

We first generate an extreme point $\hat{x}_k$ by solving the linear program at Step 3. Then, we update C_k. Finally, we minimize f over the set $conv(C_k)$, thus obtaining the new iterate x_{k+1}.

Finite convergence of the method is stated in the following Proposition:

Proposition 5.15 *Fully corrective Frank-Wolfe algorithm obtains a solution of Problem* (5.15) *(when C is a polytope) in a finite number of iterations.*

Proof. Extreme point $\hat{x}_k$, obtained by approximately solving linear problem at Step 3, can only satisfy one of the following conditions

1. $\nabla f(x_k)^\top (\hat{x}_k - x_k) \geq 0$. Hence we get

$$\min_{x \in C} \nabla f(x_k)^\top (x - x_k) = \nabla f(x_k)^\top (\hat{x}_k - x_k) \geq 0,$$

that is necessary and sufficient optimality conditions are satisfied and x_k minimizes f over the feasible set C;

2. $\nabla f(x_k)^\top (\hat{x}_k - x_k) < 0$, hence direction $d_k = \hat{x}_k - x_k$ is descent direction and

$$\hat{x}_k \notin conv(C_{k-1}). \tag{5.18}$$

Indeed, since x_k minimizes f over $conv(C_{k-1})$ it satisfies necessary and sufficient optimality conditions, that is $\nabla f(x_k)^\top (x - x_k) \geq 0$ for all $x \in conv(C_{k-1})$.

From (5.18) we thus have $\hat{x}_k \notin C_{k-1}$. Since our feasible set C has a finite number of extreme points, case 2 occurs only a finite number of times, and case 1) will eventually occur. □

In practice, the method makes more progress per iteration and results in iterates that are combination of even fewer vertices (better sparsity) with respect to other Frank-Wolfe variants. Anyway, solving the inner problem, in some cases, is as hard as solving the original one. In Figure 5.9, we show how the fully corrective Frank-Wolfe works in practice.

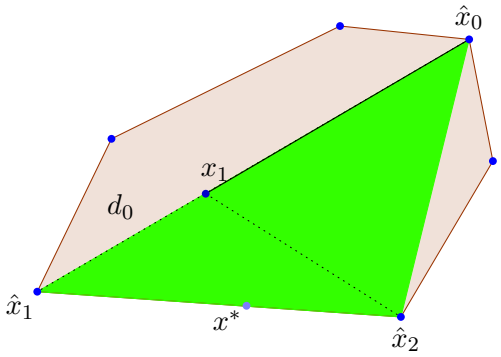

Figure 5.10: Behavior of the fully corrective Frank-Wolfe method.

5.5 Interior point methods

Interior point methods represent a class of approaches for solving linear and nonlinear programming problems. The first algorithm of this type was proposed by Karmarkar (1984). In his seminal paper the author described a polynomial time algorithm for solving linear programs and made some strong claims about its performance in practice. The algorithm was controversial at the time of its introduction, but there have been many improvements both in theory and practice since then. Interior point method is now considered to be better than simplex, especially on large LPs. The method, as we will see, can be used to solve general convex programs. This is the reason why, in this section, we consider problems having the following form:

$$\begin{aligned} &\min c^\top x \\ &x \in C \end{aligned} \tag{5.19}$$

where C is a compact convex set with non-empty interior. We notice that any convex problem

$$\begin{aligned} &\min f(x) \\ &x \in C \end{aligned} \tag{5.20}$$

can be reformulated like Problem (5.19) by minimizing a linear function over the epigraph of the original objective f:

$$\begin{aligned} &\min_{x,y} y \\ &x \in C \\ &f(x) \leq y. \end{aligned} \tag{5.21}$$

In interior point methods, we add to the original objective function a term $B(x)$ defined in the interior of C. This function, usually called *barrier function*, is continuous and tends to $+\infty$ as the point approaches the boundary of the feasible set. The following assumptions are made on this term:

- the barrier should map the interior of the feasible space to the real space, that is $B(x) : int(C) \to \mathbb{R}$;

- since we include the barrier into the objective function, we need it to be smooth, i.e., continuously differentiable;
- it should be such that
$$B(x) \xrightarrow[x \to \partial C]{} +\infty;$$
- Hessian of B should be positive definite for each x.

When considering a general feasible set of the form

$$C = \{x \in \mathbb{R}^n : \ g_i(x) \leq 0, \ i = 1, \ldots, p\},$$

with $g_i(x), \ i = 1, \ldots, p$ convex functions, we can use the so-called *logarithmic* barrier:

$$B(x) = -\sum_{i=1}^{p} \ln(-g_i(x)).$$

Another example of term that can be used in this case is the *inverse* barrier:

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

It is easy to see that all functions are convex and satisfy the assumptions made before. The *barrier method* basically solves a sequence of problems of the following form

$$\min_{x \in \mathbb{R}^n} \ p \ c^\top x + B(x), \tag{5.22}$$

with $p \in \mathbb{R}^+$. The solution of this problem is usually indicated with $x^*(p)$. It is possible to prove that when $p \to +\infty$, we get

$$x^*(p) \to x^*,$$

with x^* optimal solution of the original problem. The sequence of $\{x^*(p)\}$ is usually called *central path*. The idea of the barrier method is to move along the central path by "boosting" a fast locally convergent algorithm, which we denote for the moment by $\mathcal{A}$, using the following scheme: at each iteration k, one chooses a value p_k and uses $\mathcal{A}$ initialized at $x^*(p_{k-1})$ to compute $x^*(p_k)$. The choice of p_k is crucial. Indeed, p_k should be large in order to make as much progress as possible on the central path, but on the other hand the new point needs to be close enough to previous point in the central path so that it is in the basin of fast convergence for $\mathcal{A}$ when solving the problem with respect to p_k. A general scheme for a barrier method is described in Algorithm 17.

Since the barrier is defined only in the interior, the central path must be in the interior of the feasible set. In order to solve Problem (5.22), we can use Newton method. A basic interior-point approach is thus obtained by means of the *path-following scheme*, which alternates between an updating of p_k and the calculation of an approximate solution of problem (5.22) by means of a single Newton step:

$$x_{k+1} = x_k - \nabla^2 B(x_k)^{-1}[p_k c + \nabla B(x_k)].$$

Algorithm 17 `General barrier method`

1 Choose a point $x_0 \in int(C)$ and $p_0 > 0$
2 For $k = 0, \ldots$
3 Use $\mathcal{A}$ with starting point x_k to get

$$x^*(p_k) \in \arg\min_{x \in \mathbb{R}^n} \; p_k \; c^\top x + B(x)$$

4 If $x^*(p_k)$ satisfies some specific condition, then STOP
5 Set $x_{k+1} = x^*(p_k)$
6 Update p_{k+1}
7 End for

A classic updating rule for p_k (guaranteeing $p_k \to +\infty$) is

$$p_{k+1} = \left(1 + \frac{1}{13\sqrt{\nu}}\right) p_k,$$

with ν parameter depending on the barrier used. For this approach is possible to prove the following result.

Theorem 5.16 *The path-following scheme satisfies*

$$c^\top x_k - c^\top x^* \leq \frac{2\nu}{p_0} e^{-\frac{k}{1+13\sqrt{\nu}}}.$$

It is easy to see, by simple calculations, that computational complexity is $\mathcal{O}(M\sqrt{\nu}\ln(\nu/\varepsilon))$, where M is the cost of one Newton step (usually around $\mathcal{O}(n^2)$, with n dimension of the problem) and ν is some data-dependent scale factor. The ratio ν/ε is usually considered as the relative accuracy and the term $\ln(\nu/\varepsilon)$ can be taught of as the number of accuracy digits in the ε-solution. With this interpretation, the polynomiality of a method means that for this method the arithmetic cost of an accuracy digit is bounded from above by a polynomial of the problem size, and this polynomial can be thought of as the characteristic of the complexity of the method. In case C is polyhedral, it is possible to show that $\nu = n$. We thus have a polynomial time algorithm for linear programs (notice that interior-point methods guarantee better computational complexity than the the simplex).

It makes sense to compare this approach with the information-based approach we introduced in the previous chapters. In the information-based complexity theory the problem is represented by an oracle (a black box) so that a method, starting its job, has no information on the instance; this information is then collected via sequential calls to the oracle, and the number of these calls sufficient to find an ε-solution basically represents the complexity of the method. It is important to highlight that this complexity does include neither the computational effort of the oracle, nor the arithmetic cost of processing the answers of the oracle by the method. On the other side, when dealing with interior point methods the data specifying the problem instance represent the input to the method, so that the method from the very beginning possesses

complete global information on the problem instance. What the method does is transforming the input information into ε-solution to the problem, and the complexity of the method is defined by the arithmetic cost of this transformation. hence, this approach is not as general as the information-based one, since now we can speak only on families of problems of a reasonable analytic structure (otherwise the notion of the data dependent parameters becomes senseless). As a compensation, the complexity is much more adequate measure of the actual computational effort than the information-based complexity.

Summarizing, when dealing with huge scale data, we need to keep in mind that, since n is quite large, operations like Hessian evaluation cannot be performed. Furthermore, the rate of the method described here depends on ν (which is related to the dimension of the problem). If we want to solve huge scale convex problems in data science, we better choose *dimension-free* methods (i.e., methods whose rate does not depend on the data dimension) that only use first order information.

5.6 Constrained problems in data science

In this section, we describe in depth a few examples of constrained problems in data science. In particular, we focus on training of linear Support Vector Machines (SVMs), the PageRank problem (aka *Google problem*), the Markowitz Portfolio problem and the Least Absolute Shrinkage and Selection Operator (LASSO) problem.

5.6.1 Training of linear SVMs for classification problems

Support Vector Machines, first described by Vapnik, represent a very important class of machine learning tools. The basic idea, when dealing with binary classification problems is to build a hyperplane that minimizes the *separation margin* between the elments of the two classes. We then consider two sets A and B and assume that are linearly separable, i.e., there exists a hyperplane

$$H = \{x \in R^n : \ w^\top x + \theta = 0\}$$

such that

$$\begin{cases} w^\top x^i + \theta \geq 1 & x^i \in A \\ w^\top x^i + \theta \leq -1 & x^i \in B. \end{cases} \tag{5.23}$$

For a given hyperplane H, pair (w, θ) satisfying (5.23), we define as separation margin for H the minimum distance[7] ρ between samples in $A \cup B$ and H:

$$\rho(w, \theta) = \min_{x^i \in A \cup B} \left\{ \frac{|w^\top x^i + \theta|}{\|w\|} \right\}.$$

[7]Projection of a point x_0 over a hyperplane $H = \{x \in \mathbb{R}^n : \ a^\top x = b\}$ is pretty simple to calculate. Indeed, we have

$$\rho_H(x_0) = x_0 + \frac{b - a^\top x_0}{a^\top a} a$$

and

$$\|\rho_H(x_0) - x_0\| = \frac{|b - a^\top x_0|}{\|a\|}.$$

The optimal hyperplane $H(w^*, \theta^*)$ is the one with maximum separation margin (See Figure 5.11 for an example of optimal hyperplane).

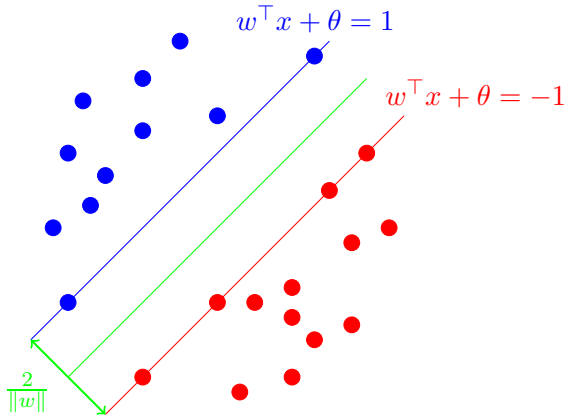

Figure 5.11: Optimal hyperplane for a classification problem.

It is possible to prove existence and uniqueness of the optimal hyperplane. Determining the optimal hyperplane is equivalent to solve the following problem:

$$\begin{array}{ll} \max\limits_{w,\ \theta} \rho(w, \theta) & \\ w^\top x^i + \theta \geq 1 & x^i \in A \\ w^\top x^i + \theta \leq -1 & x^i \in B. \end{array}$$

It is possible to prove that this problem is equivalent to the following convex quadratic programming problem:

$$\begin{array}{ll} \min\limits_{w,\ \theta} \dfrac{1}{2}\|w\|^2 & \\ y^i(w^\top x^i + \theta) - 1 \geq 0 & i = 1, \ldots, P. \end{array}$$

with $y^i = 1$ if $x^i \in A$, $y^i = -1$ if $x^i \in B$ and $P = |A \cup B|$. When A and B are not linearly separable, the system (5.23) has no solutions. We thus introduce new variables $\xi^i \geq 0,\ i = 1, \ldots, P$ and the system becomes:

$$\left\{ \begin{array}{ll} w^\top x^i + \theta \geq 1 - \xi^i & x^i \in A \\ w^\top x^i + \theta \leq -1 + \xi^i & x^i \in B \\ \xi^i \geq 0, \quad i = 1, \ldots, P. & \end{array} \right. \tag{5.24}$$

Notice that when x^i is not correctly classified, variable ξ^i is greater than 1, thus the term

$$\sum_{i=1}^{P} \xi^i$$

is an upper bound over the training errors. Problem hence becomes:

$$\begin{array}{ll} \min\limits_{w,\ \theta,\ \xi} \dfrac{1}{2}\|w\|^2 + C\sum\limits_{i=1}^{P} \xi^i & \\ y^i(w^\top x^i + \theta) - 1 \geq -\xi^i & i = 1, \ldots, P \\ \xi^i \geq 0 & i = 1, \ldots, P. \end{array}$$

5.6.2 Support Vector Machines for regression problems

In case we deal with regression problems, it is possible to consider the linear model

$$f(x; w, \theta) = w^\top x + \theta$$

and a specific precision level ϵ, used to approximate the unknown function. Estimate is considered correct if the following condition is satisfied:

$$|y^i - w^\top x^i - \theta| \leq \epsilon.$$

We use the loss function

$$|y - f(x; w, \theta)|_\epsilon = \max\{0, |y - f(x; w, \theta) - \epsilon|\}$$

and define the training error as follows (keep in mind that in this case $y^i \in \mathbb{R}$ for $i = 1, \ldots, P$):

$$E = \sum_{i=1}^{P} |y^i - f(x^i; w, \theta)|_\epsilon.$$

Training error is zero if the following system is satisfied:

$$\begin{cases} w^\top x^i + \theta - y^i \leq \epsilon & i = 1, \ldots, P \\ y^i - w^\top x^i - \theta \leq \epsilon & i = 1, \ldots, P. \end{cases} \tag{5.25}$$

Also in this case it is possible to introduce new variables ξ_i and $\hat{\xi}_i$ and consider the following convex quadratic problem:

$$\begin{array}{ll} \min\limits_{w,\ \theta,\ \xi,\ \hat{\xi}} \dfrac{1}{2}\|w\|^2 + C\sum\limits_{i=1}^{P}(\xi_i + \hat{\xi}_i) & \\ w^\top x^i + \theta - y^i \leq \epsilon + \xi_i & i = 1, \ldots, P \\ y^i - w^\top x^i - \theta \leq \epsilon + \hat{\xi}_i & i = 1, \ldots, P \\ \xi_i, \hat{\xi}_i \geq 0 & i = 1, \ldots, P. \end{array}$$

5.6.3 LP problems with random costs

Consider the linear program

$$\begin{array}{ll} \min\limits_{x \in \mathbb{R}^n} & c^\top x \\ \text{s.t.} & Gx \leq h \\ & Ax = b, \end{array}$$

assume that the cost vector c is a random vector with mean $\bar{c}$ and covariance

$$\mathbf{E}\left[(c - \bar{c})(c - \bar{c})^\top\right] = \Sigma,$$

and that all the other parameters are deterministic. For a vector $x \in R^n$, cost $c^\top x$ is a random variable with mean

$$\mathbf{E}[c^\top x] = \bar{c}^\top x$$

and variance

$$\mathbf{var}(c^\top x) = \mathbf{E}[c^\top x - \mathbf{E}[c^\top x]]^2 = x^\top \Sigma x.$$

Usually, we try to balance between expected value and variance. A classic model is given by combining the two terms:

$$\mathbf{E}[c^\top x] + \gamma \mathbf{var}(c^\top x),$$

with $\gamma \geq 0$ *risk-aversion* parameter. Keeping in mind that Σ is a positive semidefinite matrix, we get the following convex quadratic program:

$$\begin{array}{ll} \min\limits_{x \in \mathbb{R}^n} & \bar{c}^\top x + \gamma x^\top \Sigma x \\ \text{s.t.} & Gx \leq h \\ & Ax = b. \end{array}$$

5.6.4 Portfolio optimization

We have n available assets. We call x_i the quantity of money invested on the i-th asset during the considered period and with r_i the returns on the i-th asset. We have two different constraints. The first one is non-negativity for the variables (i.e., $x_i \geq 0$). It basically means that short selling (selling asset that we still don't own) is not allowed. We then have the budget constraint:

$$\sum_{i=1}^{n} x_i = B,$$

the total amount of money invested needs to be equal to the budget B (B can be simply set to 1).
Consider a stochastic model for the returns: $r \in \mathbb{R}^n$ is a randomly generated vector with mean $\bar{r}$ and covariance Σ. Thus expected return will be

$$\bar{r}^\top x$$

and variance

$$x^\top \Sigma x.$$

Classic portfolio problem, described by Markowitz (1952), is a convex quadratic programming problem:

$$\begin{array}{ll} \min\limits_{x \in \mathbb{R}^n} & \gamma x^\top \Sigma x - \bar{r}^\top x \\ s.t. & e^\top x = 1 \\ & x \geq 0, \end{array}$$

with $\gamma > 0$ risk-aversion parameter. Goal is thus finding the set of assets that minimizes the variance (risk connected to the given portfolio) while maximizing the expected return (we obviously need to satisfy budget and non-negativity constraints).

5.6.5 PageRank problem

The effectiveness of Google search engine largely relies on its PageRank (named after Google's founder Larry Page) algorithm, which quantitatively ranks the importance of each page on the web, allowing Google to thereby present to the user the more important (and typically most relevant and helpful) pages first. If we consider the web of interest composed of n pages, each labelled with $k = 1, \ldots, n$, we can model this web as a directed graph, where pages are the nodes of the graph, and a directed edge exists pointing from node k_i to node k_j if the web page k_i contains a link to k_j. We call x_k, with $k = 1, \ldots, n$, the importance score of page k. A simple initial idea would be to assign the score to any page k according to the number of other web pages that link to the considered page (the so-called backlinks). In Figure 5.12, we report a small example of web. In this case, the scores would be $x_1 = 2$, $x_2 = 1$, $x_3 = 3$, $x_4 = 2$, so that page $k = 3$ appears to be the most relevant page, whereas page $k = 2$ is the least important. Using this approach, a page score can be interpreted as the number of "votes" that a page receives from other pages, where each incoming link is a vote.

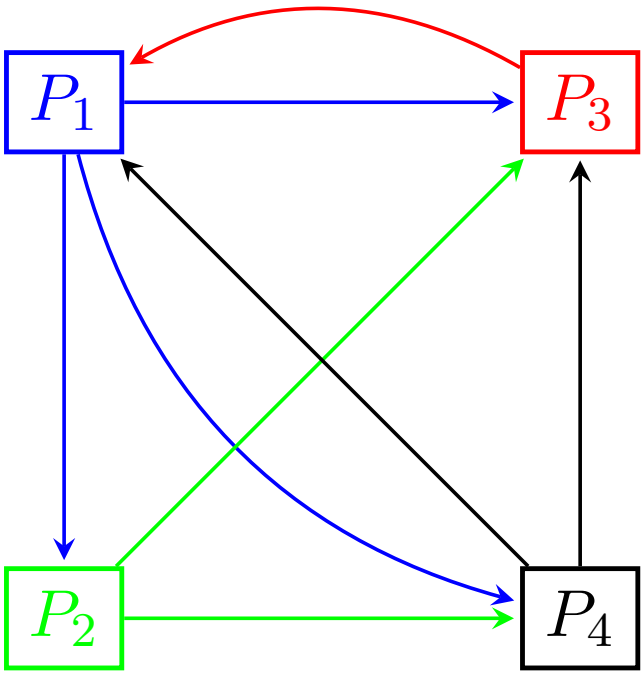

Figure 5.12: Small web example.

However, the web is not as simple, since the relevance of a page typically depends on the relevance of the pages pointing to it. In other words, your page relevance should be higher if your page is pointed directly by Google.com rather than by Nobodycares.com. Votes should thus be weighted (not simply counted), and the weight should be related to the score of the pointing page itself. The actual scoring count goes then as follows: each page j has a score x_j and n_j outgoing links; as an assumption, we do not allow links from a page to itself,

and we do not allow pages without outgoing links, therefore $n_j > 0$ for all j. The score x_j represents the total power of node j, which we need to split among the n_j outgoing links; each outgoing link thus carries x_j/n_j units of vote. Let L_k denote the indices related to pages that point to page k (backlinks). Then, the score of page k is computed as follows:

$$\sum_{j \in L_k} \frac{x_j}{n_j}, \quad j = 1, \ldots, n.$$

If we write this equations for our small web, we have

$$\begin{cases} x_1 = & & & & & x_3 & + & \frac{1}{2}x_4 \\ x_2 = & \frac{1}{3}x_1 & & & & & & \\ x_3 = & \frac{1}{3}x_1 & + & \frac{1}{2}x_2 & + & & & \frac{1}{2}x_4 \\ x_4 = & \frac{1}{3}x_1 & + & \frac{1}{3}x_2 & & & & \end{cases}$$

that we can be rewritten as follows

$$x = Ax,$$

with

$$A = \begin{pmatrix} 0 & 0 & 1 & \frac{1}{2} \\ \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{3} & \frac{1}{2} & 0 & 0 \end{pmatrix}.$$

Matrix A, which is a *left stochastic matrix*[8], is the *link matrix* for the given web. It is actually possible to prove that 1 is eigenvalue for A and there exists an eigenvector with all components greater or equal than zero. So, solving the PageRank problem corresponds to find the x that satisfies $x = Ax$, i.e., the eigenvector related to the eigenvalue 1. In our example a possible solution x is

$$x = \begin{pmatrix} 0.3871 \\ 0.1290 \\ 0.2903 \\ 0.1935 \end{pmatrix}.$$

Page 1 is then the most relevant in our web. A problem we have when using this approach is that there might be multiple eigenvectors related to 1. Hence we consider a modified matrix

$$\hat{A} = (1 - \alpha)A + \alpha C,$$

where $\alpha \in (0, 1)$[9] and C is an $n \times n$ matrix with all entries equal to $1/n$. Notice that C basically gives a surfer probability to jump randomly on any page in the web. This modified matrix (usually called *Google matrix*) has only one eigenvector with corresponding eigenvalue equal to 1. It is actually possible to prove that the eigenvector x related to eigenvalue 1 satisfies the following conditions:

[8] All components are non-negative and each column sums up to 1.

[9] Classic choice is $\alpha = 0.15$.

- x is unique;
- $x \geq 0$;
- $e^\top x = 1$.

Finally, we can model PageRank as a constrained optimization problem:

$$\begin{array}{ll} \min\limits_{x \in \mathbb{R}^n} & \|Ax - x\|^2 \\ \text{s.t.} & e^\top x = 1 \\ & x \geq 0. \end{array}$$

An alternative unconstrained formulation, often used in practice, is the following:

$$\min_{x \in \mathbb{R}^n} \|Ax - x\|^2 + \gamma [e^\top x - 1]^2$$

with $\gamma > 0$ penalty parameter. In practice we add a term to the objective function that measures the violation of the equality constraint. Non-negativity constraints are simply dropped out in this case.

5.6.6 LASSO problem

The LASSO, proposed by Tibshirani in 1996, is a popular tool for sparse linear regression. Given the training set

$$T = \{(a^i, b^i),\ a^i \in \mathbb{R}^n,\ b^i \in \mathbb{R},\ i = 1, \ldots, m\}$$

the goal is finding a sparse linear model[10] describing the data. This problem is strictly connected with the Basis Pursuit Denoising (BPDN) Problem in signal analysis (see Chapter 3 for further details). In this case, given a discrete-time input signal b, and a *dictionary*

$$D = \{a_j \in \mathbb{R}^m \ :\ j = 1, \ldots, n\}$$

of elementary discrete-time signals, usually called atoms, the goal is finding a sparse linear combination of the atoms that *approximate* the real signal[11].

Example 5.17 *We describe here a classic image compression example. We have a real signal $\tilde{x}$ that has a sparse representation for a given $n \times n$ basis Ψ, that is*

$$\tilde{x} = \Psi x$$

and x sparse. We call $b \in \mathbb{R}^m$ the measurement vector (compressed signal). This compressed signal is simply obtained by means of Φ $m \times n$ the so-called sampling matrix:

$$b = \Phi \tilde{x}.$$

We then define a reconstruction matrix *by multiplying Φ and Ψ, that is $A = \Phi\Psi$, As we will see, the available reconstruction tools use b and A to get the sparse representation x (See Figure 5.13).*

[10] A sparse linear model is a model with a small number of non-zero parameters.

[11] Notice that this is just the approximated version of the problem described in the section related to compressive sensing (see Chapter 3 for further details).

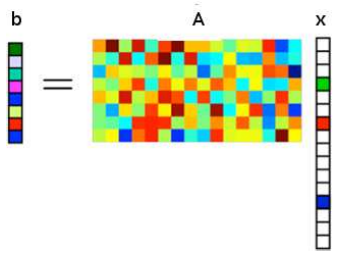

Figure 5.13: Image compression example.

LASSO/BPD problem can be formulated as follows:

$$\begin{array}{ll} \min\limits_{x\in\mathbb{R}^n} & \|Ax-b\|_2^2 \\ s.t. & |supp(x)| \leq \tau \end{array} \tag{5.26}$$

The parameter τ controls the amount of shrinkage that is applied to the model (number of nonzero components in x). Constraint is represented by means of a nonconvex and discontinuous function. We then need to use an approximation to make problem tractable. Using the same idea already seen for the compressive sensing problem, we formulate LASSO/BPD problem as follows:

$$\begin{array}{ll} \min\limits_{x\in\mathbb{R}^n} & \|Ax-b\|_2^2 \\ s.t. & \|x\|_1 \leq \tau. \end{array} \tag{5.27}$$

Thus we get a convex problem that can be easily handled.

5.7 Duality in nonlinear programming

Duality theory gives us very important results both from a theoretical and an algorithmic point of view. In particular, it enables us to relate every constrained problem with a dual problem (similar to what we have seen in Chapter 3 for linear programs). Under some specific assumptions the dual has a strong connection with the primal and, usually, a structure that can be better exploited from a computational point of view.

In this section, we quickly review some basic results related to duality theory in nonlinear programming. Then we apply those results on some data science problems we previously described.

5.7.1 Lagrangian dual

Let us consider the following problem:

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & h(x) = 0 \\ & g(x) \leq 0 \\ & x \in X \end{array} \tag{5.28}$$

with $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^m$, $h : \mathbb{R}^n \to \mathbb{R}^p$ and $X \subseteq \mathbb{R}^n$.

Hence we have:

$$C = \{x \in X : \quad h(x) = 0, \quad g(x) \leq 0\}. \tag{5.29}$$

In connection with the previous problem (5.28) we can build the *Lagrangian function* $L : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ given as:

$$L(x, \lambda, \mu) = f(x) + \lambda^\top g(x) + \mu^\top h(x). \tag{5.30}$$

We can thus define the *Lagrangian dual problem* related to the primal problem described in (5.28):

$$\begin{array}{ll} \max & \varphi(\lambda, \mu) \\ \text{s.t.} & \lambda \geq 0 \end{array} \tag{5.31}$$

where $\varphi : \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ is given as follows:

$$\varphi(\lambda, \mu) = \inf_{x \in X} \left\{ f(x) + \lambda^\top g(x) + \mu^\top h(x) \right\} = \inf_{x \in X} L(x, \lambda, \mu). \tag{5.32}$$

Notice that for some (λ, μ) function $\varphi(\lambda, \mu)$ might be $-\infty$. Hence, in some cases, it might be useful to consider the set:

$$\Delta = \{(\lambda, \mu) \in \mathbb{R}^m \times \mathbb{R}^p : \varphi(\lambda, \mu) > -\infty\}. \tag{5.33}$$

We can now give a first result that connects primal and dual problem.

Theorem 5.18 (Weak duality) *Let $f \in C(\mathbb{R}^n)$, $g_i \in C(\mathbb{R}^n)$, for all $i = 1, \ldots, m$, and $h_j \in C(\mathbb{R}^n)$, for all $j = 1, \ldots, p$. For any feasible point $x \in \mathbb{R}^n$ of the primal problem (5.28), that is $x \in X$, $g(x) \leq 0$ and $h(x) = 0$, and for any feasible point (λ, μ) of the dual problem (5.31), that is $(\lambda, \mu) \in \mathbb{R}^m \times \mathbb{R}^p$ and $\lambda \geq 0$, we have:*

$$\varphi(\lambda, \mu) \leq f(x). \tag{5.34}$$

Proof. From definition of φ and from $x \in X$, $g(x) \leq 0$ e $h(x) = 0$, $\lambda \geq 0$, we have

$$\varphi(\lambda, \mu) = \inf_{x \in X} \left\{ f(x) + \lambda^\top g(x) + \mu^\top h(x) \right\} \leq f(x) + \lambda^\top g(x) + \mu^\top h(x) \leq f(x).$$

Thus we proved our result. □

From the previous result immediately descends the following corollary.

Corollary 5.19 *Let $f \in C(\mathbb{R}^n)$ and $g_i \in C(\mathbb{R}^n)$, for all $i = 1, \ldots, m$, and $h_j \in C(R^n)$, for all $j = 1, \ldots, p$. The following properties hold:*

i)

$$\max_{\lambda \geq 0} \varphi(\lambda, \mu) \leq \min_{x \in C} f(x);$$

ii) if a $(\bar{\lambda}, \bar{\mu}) \in \mathbb{R}^m \times \mathbb{R}^p$ *with* $\bar{\lambda} \geq 0$ *and a point* $\bar{x} \in X$ *with* $g(\bar{x}) \leq 0$ *and* $h(\bar{x}) = 0$*, are such that*

$$\varphi(\bar{\lambda}, \bar{\mu}) = f(\bar{x}),$$

then $(\bar{\lambda}, \bar{\mu})$ *is an optimal solution for the dual and* $\bar{x}$ *is an optimal solution for the primal;*

iii) if the primal is unbounded, then

$$\varphi(\lambda, \mu) = -\infty,$$

for all $(\lambda, \mu) \in R^m \times R^p$ *with* $\lambda \geq 0$*;*

iv) if the dual is unbounded, then the primal is unfeasible.

From i) we have that for an x^* optimal solution of the primal and a pair (λ^*, μ^*) optimal solution of the dual, the following inequality holds:

$$\varphi(\lambda^*, \mu^*) \leq f(x^*).$$

If

$$\varphi(\lambda^*, \mu^*) < f(x^*),$$

we have a *duality gap*.

In case

$$\varphi(\lambda^*, \mu^*) = f(x^*),$$

we have a zero duality gap. We finally give a definition that helps identifying optimal solutions in the general constrained case.

Definition 5.20 *We say that* (x^*, λ^*, μ^*) *satisfy optimality conditions for the primal if the following conditions are satisfied:*

- *Dual feasibility:*

$$x^* \in \arg\min_{x \in X} L(x, \lambda^*, \mu^*),$$ [12]

and $\lambda^* \geq 0$*;*

- *Primal feasibility:*

$$g(x^*) \leq 0, \quad h(x^*) = 0, \quad x^* \in X;$$

- *Complementary slackness:*

$$\lambda^{*\top} g(x^*) = 0.$$

[12] When functions are continuously differentiable and some other convexity assumptions are satisfied, we can equivalently write

$$\nabla f(x^*) + \nabla g(x^*)^\top \lambda^* + \nabla h(x^*)^\top \mu^* = 0.$$

5.7.2 The quadratic case

Now we focus on convex quadratic problems of the form:

$$\begin{array}{ll} \min & \frac{1}{2}x^\top Qx + c^\top x \\ \text{s.t.} & Ax \leq b, \end{array}$$

with $x \in \mathbb{R}^n$, $Q \in \mathbb{R}^{n\times n}$, $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$. We consider the Lagrangian dual related to the above problem:

$$\begin{array}{ll} \max & \varphi(\lambda) \\ \text{s.t.} & \lambda \geq 0, \end{array} \tag{5.35}$$

with

$$\varphi(\lambda) = \inf_{x\in\mathbb{R}^n} \left\{ L(x,\lambda) = \frac{1}{2}x^\top Qx + c^\top x + \lambda^\top (Ax - b) \right\}.$$

We assume here that the primal admits an optimal solution. The function $L(x,\lambda)$, for any fixed value $\lambda \geq 0$, is a convex quadratic function. It is thus bounded from below if and only if its minimum is achieved, which in turn can be true if and only if the gradient of L with respect to x vanishes at some point (see optimality conditions in Chapter 4). Thus, if $\varphi(\lambda) = -\infty$, there is no x satisfying

$$\nabla_x L(x,\lambda) = Qx + c + A^\top \lambda = 0.$$

Otherwise we can rewrite

$$\begin{aligned} L(x,\lambda) &= -\frac{1}{2}x^\top Qx + x^\top (Qx + c + A^\top \lambda) - \lambda^\top b \\ &= -\frac{1}{2}x^\top Qx - \lambda^\top b, \end{aligned}$$

and our dual becomes

$$\begin{array}{ll} \max & -\frac{1}{2}x^\top Qx - \lambda^\top b \\ \text{s.t.} & Qx + c + A^\top \lambda = 0 \\ & \lambda \geq 0. \end{array} \tag{5.36}$$

It is possible to prove the following result:

Theorem 5.21 (Strong duality for quadratic problems) *Let x^* be optimal for the primal, then there exists a vector $\lambda^* \geq 0$ such that (x^*, λ^*) is optimal for the dual and the two extremal values are equal. Furthermore, if $(\tilde{x}, \tilde{\lambda})$ is optimal for the dual, then some x^* satisfying*

$$Q(x^* - \tilde{x}) = 0, \tag{5.37}$$

$$\tilde{\lambda}^\top (Ax^* - b) = 0 \tag{5.38}$$

and

$$Ax^* \leq b$$

is optimal for the primal and the two extremal values are equal.

We now can apply this result to the SVM training problems analyzed in Subsection 5.6.1. Let us start with the linearly separable case. The Lagrangian function for the problem is

$$L(w,\theta,\lambda) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{P} \lambda_i \left[y^i(w^\top x^i + \theta) - 1 \right].$$

We then write dual problem as follows:

$$\begin{array}{ll} \max & \frac{1}{2}\|w\|^2 - \sum_{i=1}^{P} \lambda_i \left[y^i(w^\top x^i + \theta) - 1 \right] \\ \text{s.t.} & \nabla_w L(w,\theta,\lambda) = w - \sum_{i=1}^{P} \lambda_i y^i x^i = 0 \\ & \nabla_\theta L(w,\theta,\lambda) = \sum_{i=1}^{P} \lambda_i y^i = 0 \\ & \lambda \geq 0. \end{array} \tag{5.39}$$

Using the first equality constraint we can rewrite the problem as follows

$$\begin{array}{ll} \min & \frac{1}{2}\sum_{i=1}^{P}\sum_{j=1}^{P} y^i y^j (x^i)^\top x^j \lambda_i \lambda_j - \sum_{i=1}^{P} \lambda_i \\ \text{s.t.} & \sum_{i=1}^{P} \lambda_i y^i = 0 \\ & \lambda \geq 0. \end{array} \tag{5.40}$$

Or equivalently

$$\begin{array}{ll} \min & \frac{1}{2}\lambda^\top X^\top X \lambda - e^\top \lambda \\ \text{s.t.} & \sum_{i=1}^{P} \lambda_i y^i = 0 \\ & \lambda \geq 0, \end{array} \tag{5.41}$$

with $X = [y^1 x^1 \dots y^P x^P]$. Thus we get a convex quadratic problem with simple constraints[13]. Using the result reported in Theorem 5.21, we have that the primal optimal solution (w^*, θ^*) can be built starting from the dual solution $(\tilde{w}, \tilde{\theta}, \tilde{\lambda})$, where

$$\tilde{w} = \sum_{i=1}^{P} \tilde{\lambda}_i y^i x^i \quad \text{and} \quad \tilde{\theta} \in \mathbb{R}.$$

Indeed, since equality (5.37) holds, we have

$$w^* = \tilde{w} = \sum_{i=1}^{P} \tilde{\lambda}_i y^i x^i.$$

[13]Notice that we get a minimization problem in the end. In this case, we just used the equivalence

$$\max_{x \in C} f(x) \equiv -\min_{x \in C} -f(x).$$

Those vector that have a $\tilde{\lambda}_i > 0$ are called *support vectors*. Furthermore, since (5.38) holds, we can write

$$\tilde{\lambda}_i[y^i({w^*}^\top x^i + \theta^*) - 1] = 0, \quad i = 1, \dots, P$$

and for all $\tilde{\lambda}_i > 0$, we have $y^i({w^*}^\top x^i + \theta^*) = 1$. Hence, we can calculate θ^* by using any of those equations. The nonlinearly separable case can be handled equivalently and we get a dual of the following form:

$$\begin{aligned}
\max \quad & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{P}\xi_i - \sum_{i=1}^{P}\lambda_i\left[y^i(w^\top x^i + \theta) - 1 + \xi_i\right] - \sum_{i=1}^{P}\mu_i\xi_i \\
\text{s.t.} \quad & \nabla_w L(w, \theta, \lambda, \xi) = w - \sum_{i=1}^{P}\lambda_i y^i x^i = 0 \\
& \nabla_\theta L(w, \theta, \xi, \lambda, \mu) = \sum_{i=1}^{P}\lambda_i y^i = 0 \\
& \nabla_\xi L(w, \theta, \lambda, \xi, \mu) = C - \lambda_i - \mu_i = 0 \\
& \lambda \geq 0 \\
& \mu \geq 0.
\end{aligned} \tag{5.42}$$

Following the same reasoning as before, we get

$$\begin{aligned}
\min \quad & \frac{1}{2}\lambda^\top X^\top X\lambda - e^\top\lambda \\
\text{s.t.} \quad & \sum_{i=1}^{P}\lambda_i y^i = 0 \\
& 0 \leq \lambda \leq C,
\end{aligned} \tag{5.43}$$

with $X = [y^1x^1 \dots y^Px^P]$. Thus we have a convex quadratic problem with simple constraints. Using again the result reported in Theorem 5.21, we have that the primal optimal solution (w^*, θ^*, ξ^*) can be built starting from the dual solution $(\tilde{w}, \tilde{\theta}, \tilde{\xi}, \tilde{\lambda}, \tilde{\mu})$, where

$$\tilde{w} = \sum_{i=1}^{P}\tilde{\lambda}_i y^i x^i, \quad \tilde{\theta} \in \mathbb{R} \quad \text{and} \quad \tilde{\xi} \in \mathbb{R}^P.$$

Indeed, since equality (5.37) holds, we have

$$w^* = \tilde{w} = \sum_{i=1}^{P}\tilde{\lambda}_i y^i x^i, \quad i = 1, \dots, P.$$

Furthermore, since (5.38) holds, we can write

$$\tilde{\lambda}_i[y^i({w^*}^\top x^i + \theta^*) - 1 + \xi_i^*] = 0, \quad \tilde{\mu}_i\xi_i^* = 0, \quad i = 1, \dots, P$$

and for all $\tilde{\lambda}_i > 0$, we have $y^i({w^*}^\top x^i + \theta^*) = 1 - \xi_i^*$. Hence, we can calculate θ^* by using any of those equations.

In both cases (linearly/nonlinearly separable) the classifier is defined as follows:

$$f(x) = sign({w^*}^\top x + \theta^*).$$

Following the same path, we can also define the dual related to SVMs for regression problems.

5.8 Distributed optimization and learning using the Alternating Direction Method of Multipliers

As we have already seen in this and previous chapters, many problems of recent interest in statistics and machine learning can be posed in the framework of convex optimization. Due to the high dimension and complexity of modern datasets, it is really important to be able to solve problems with a very huge number of features or training examples. As a result, both the decentralized storage of these datasets as well as the development of distributed methods are desirable. In this section, we describe the Alternating Direction Method of Multipliers (ADMM), first introduced by Douglas and Rachford (1956). This approach is well suited to distributed convex optimization and, in particular, to huge-scale problems arising in data science. We first describe the method and its theoretical properties. Then, we discuss applications to a few data science problems.

5.8.1 Augmented Lagrangian and the method of multipliers

We focus now on the problem:

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & Ax = b, \end{array} \tag{5.44}$$

with $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $f : \mathbb{R}^n \to \mathbb{R}$ continuously differentiable convex function. We rewrite the problem in the equivalent[14] form

$$\begin{array}{ll} \min & f(x) + \dfrac{\rho}{2}\|Ax - b\|^2 \\ \text{s.t.} & Ax = b, \end{array} \tag{5.45}$$

where $\rho > 0$ is a *penalty parameter*, and consider the Lagrangian function related to this equivalent reformulation:

$$L_\rho(x, \mu) = f(x) + \mu^\top (Ax - b) + \frac{\rho}{2}\|Ax - b\|^2.$$

This is the *augmented Lagrangian* function related to Problem (5.44). The reason why we introduce the new term in the objective function is that now the dual function

$$\varphi(\mu) = \inf_{x\in\mathbb{R}^n} L_\rho(x, \mu)$$

has nice properties (like, e.g., continuous differentiability under mild conditions). The gradient of the augmented dual function can be obtained very easily, by minimizing over x, and then evaluating the resulting equality constraint residual. In practice, for a given $\bar{\mu}$, if we want to calculate the gradient, we first find

$$\bar{x} \in \arg\min_{x\in\mathbb{R}^n} L_\rho(x, \bar{\mu}).$$

[14] Equivalence is easy to get, since for any feasible x the term added to the objective is zero.

Then we set

$$g(\bar{\mu}) = A\bar{x} - b.$$

Applying a gradient like approach (*dual ascent method*) to the dual problem[15]

$$\max_{\mu \in \mathbb{R}^m} \varphi(\mu)$$

yields the algorithm that everybody knows as *method of multipliers*. The main steps are:

$$x_{k+1} = \arg\min_{x \in \mathbb{R}^n} L_\rho(x, \mu_k)$$

and

$$\mu_{k+1} = \mu_k + \rho(Ax_{k+1} - b).$$

It is possible to prove that the choice of ρ as a stepsize guarantees dual feasibility[16]. Furthermore, as the method of multipliers proceeds, the primal residual $Ax_{k+1} - b$ converges to zero, thus giving optimality (see Definition 5.20).

5.8.2 Alternating direction method of multipliers

Let us consider a different problem:

$$\begin{array}{ll} \min & f(x) + g(z) \\ \text{s.t.} & Ax + Cz = b, \end{array} \tag{5.46}$$

With $A \in \mathbb{R}^{m \times n_1}$, $C \in \mathbb{R}^{m \times n_2}$, $b \in \mathbb{R}^m$, $f : \mathbb{R}_1^n \to \mathbb{R}$ and $g : \mathbb{R}_2^n \to \mathbb{R}$ continuously differentiable convex functions. Again, we consider the augmented Lagrangian function related to the problem, thus getting

$$L_\rho(x, z, \mu) = f(x) + g(z) + \mu^\top (Ax + Cz - b) + \frac{\rho}{2}\|Ax + Cz - b\|^2,$$

with $\rho > 0$. ADMM consists of three different steps:

$$x_{k+1} = \arg\min_{x \in \mathbb{R}_1^n} L_\rho(x, z_k, \mu_k),$$

$$z_{k+1} = \arg\min_{z \in \mathbb{R}_2^n} L_\rho(x_{k+1}, z, \mu_k),$$

and

$$\mu_{k+1} = \mu_k + \rho(Ax_{k+1} + Cz_{k+1} - b).$$

In ADMM, x and z are updated in an alternating or sequential fashion, which accounts for the term alternating direction. ADMM can be hence viewed as a version of the method of multipliers where a single Gauss-Seidel step over x and z is used instead of the usual joint minimization. Splitting the minimization over x and z into two steps is precisely what allows for decomposition when f or g are separable. We report the scheme of the method below.

[15] Keep in mind that we are maximizing here, then we want to get a ascent direction. Easy to check that the best ascent direction is the gradient.

[16] Checking dual feasibility of (x_{k+1}, μ_{k+1}) is very easy. Indeed, from minimization on x, we get

$$0 = \nabla_x L_\rho(x_{k+1}, \mu_k) = \nabla_x f(x_{k+1}) + A^\top(\mu_k + \rho(Ax_{k+1} - b)) = \nabla_x f(x_{k+1}) + A^\top \mu_{k+1}.$$

Algorithm 18 `Alternating Direction Method of Multipliers`

1 Choose points $x_1 \in \mathbb{R}^{n_1}$, $z_1 \in \mathbb{R}^{n_2}$, $\mu_1 \in \mathbb{R}^m$ and $\rho > 0$
2 For $k = 1, \ldots$
3 Set
$$x_{k+1} = \arg\min_{x \in \mathbb{R}_1^n} L_\rho(x, z_k, \mu_k)$$
4 Set
$$z_{k+1} = \arg\min_{z \in \mathbb{R}_2^n} L_\rho(x_{k+1}, z, \mu_k)$$
5 Set
$$\mu_{k+1} = \mu_k + \rho(Ax_{k+1} + Cz_{k+1} - b)$$
7 End for

Convergence of the method can be proved under standard assumptions. The rate is in general sublinear. A standard extension is to use possibly different penalty parameters ρ_k for each iteration, with the goal of improving the convergence in practice. ADMM will converge even when the minimization with respect to x and z are not carried out exactly, provided certain conditions are satisfied. This result is due to Eckstein and Bertsekas. This modification is important when iterative methods are needed to get the x or z updates. Indeed, this strategy allows us to solve the minimizations only approximately at first, and then more accurately as the iterations go on.

5.8.3 Consensus optimization

There has recently been interest in coordination of networks consisting of multiple agents, where the goal is to collectively optimize a global objective. This is motivated mainly by the emergence of large-scale networks (e.g., mobile ad hoc networks and wireless-sensor networks) characterized by the lack of centralized access to information and time-varying connectivity. Control and optimization algorithms deployed in such networks should be

- completely distributed, relying only on local observations and information;
- robust against unexpected changes in topology, such as link or node failures;
- scalable in the size of the network.

Here we describe two variants of the *consensus problem* and distributed ADMM-based methods for solving them.

In consensus, we consider a multiagent network model, where P agents exchange information over a connected network. Each agent i has a "local function" $f_i(x)$. The vector $x \in \mathbb{R}^n$ is a global decision vector that the agents need to collectively determine. The goal of the agents is to cooperatively optimize a global-objective

function, that is solving the following problem:

$$\min_{x\in\mathbb{R}^n} \quad \sum_{i=1}^{P} f_i(x), \tag{5.47}$$

with $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, P$ continuously differentiable convex functions. This problem can be rewritten as follows:

$$\begin{aligned} \min \quad & \sum_{i=1}^{P} f_i(x^i), \\ \text{s.t.} \quad & x^i = z, \quad i = 1, \ldots, P \end{aligned} \tag{5.48}$$

with the auxiliary variables $z \in \mathbb{R}^n$, $x^i \in \mathbb{R}^n$, $i = 1, \ldots, P$. This is called the *global consensus problem*[17]. The augmented Lagrangian function is in this case

$$L_\rho(x^1, \ldots, x^P, z, \mu) = \sum_{i=1}^{P} \left[f_i(x^i) + {\mu^i}^\top (x^i - z) + \frac{\rho}{2}\|x^i - z\|^2 \right],$$

with $\rho > 0$.

We indicate with x^{-i} the set of all x^j, such that $j \neq i$. We thus can describe the generic iteration of ADMM for the problem:

$$x^i_{k+1} = L_\rho(x^i, x^{-i}_k, z_k, \mu_k) = \arg\min_{x^i\in\mathbb{R}^n} f_i(x^i) + {\mu^i_k}^\top (x^i - z_k) + \frac{\rho}{2}\|x^i - z_k\|^2, \quad i = 1, \ldots, P,$$

$$z_{k+1} = \arg\min_{z\in\mathbb{R}^n} L_\rho(x^1_{k+1}, \ldots, x^P_{k+1}, z, \mu_k) = \frac{1}{P}\sum_{i=1}^{P} \left(x^i_{k+1} + \frac{\mu^i_k}{\rho} \right),$$

and

$$\mu^i_{k+1} = \mu^i_k + \rho(x^i_{k+1} - z_{k+1}), \quad i = 1, \ldots, P.$$

The x^i and μ^i calculations are carried out independently for each $i = 1, \ldots, P$. In the literature, the processing element that handles the global variable z is usually called *central collector* or *fusion center*. In Figure 5.14, we report the global consensus problem. C is the central collector, and $1, \ldots, P$ are the other nodes in the network. With **I**, **O** and **OBJECTIVE**, we indicate the inputs, outputs and objective, respectively.

It is possible to consider a more general form of the consensus problem, in which each agent i has a "local function" $f_i(x^i)$. The vector $x^i \in \mathbb{R}^{n_i}$, is a selection of the components of the global vector $z \in \mathbb{R}^n$ that the agents need to collectively determine. This problem can be written as follows:

$$\begin{aligned} \min \quad & \sum_{i=1}^{P} f_i(x^i), \\ \text{s.t.} \quad & x^i = z^i, \quad i = 1, \ldots, P, \end{aligned} \tag{5.49}$$

[17] Notice that the constraints are such that all the local variables should agree, i.e., be equal.

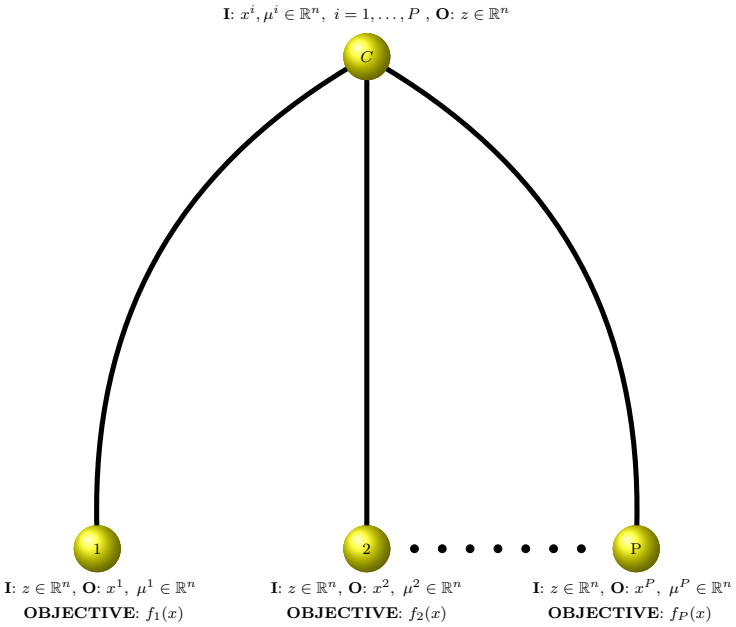

Figure 5.14: Global consensus problem.

where $z^i \in \mathbb{R}^{n_i}$ is a subvector of the global vector z. The augmented Lagrangian function is in this case

$$L_\rho(x^1, \ldots, x^P, z, \mu^1, \ldots, \mu^P) = \sum_{i=1}^{P} \left[f_i(x^i) + {\mu^i}^\top (x^i - z^i) + \frac{\rho}{2} \|x^i - z^i\|^2 \right],$$

with $\rho > 0$.

We indicate with x^{-i} the set of all x^j, such that $j \neq i$. We thus can describe the generic iteration of ADMM for the problem:

$$x_{k+1}^i = L_\rho(x^i, x_k^{-i}, z_k, \mu_k^1, \ldots, \mu_k^P) = \underset{x^i \in \mathbb{R}^n}{\arg\min}\, f_i(x^i) + {\mu_k^i}^\top (x^i - z_k^i) + \frac{\rho}{2} \|x^i - z_k^i\|^2, \quad i = 1, \ldots, P,$$

$$z_{k+1} = \underset{z \in \mathbb{R}^n}{\arg\min}\, L_\rho(x_{k+1}^1, \ldots, x_{k+1}^P, z, \mu_k^1, \ldots, \mu_k^P)$$

and

$$\mu_{k+1}^i = \mu_k^i + \rho(x_{k+1}^i - z_{k+1}^i), \quad i = 1, \ldots, P.$$

The calculation of z_{k+1} for each component $(z_{k+1})_j$ is given as follows:

$$(z_{k+1})_j = \frac{1}{|S_j|} \sum_{i \in S_j} \left((x_{k+1}^i)_{j_i} + \frac{(\mu_k^i)_{j_i}}{\rho} \right), \ j = 1, \ldots, \ n$$

where we indicate with S_j the set of indices i related to the vectors x^i whose component $(x^i)_{j_i}$ corresponds to $(z)_j$. In Figure 5.15, we report the general global consensus problem.

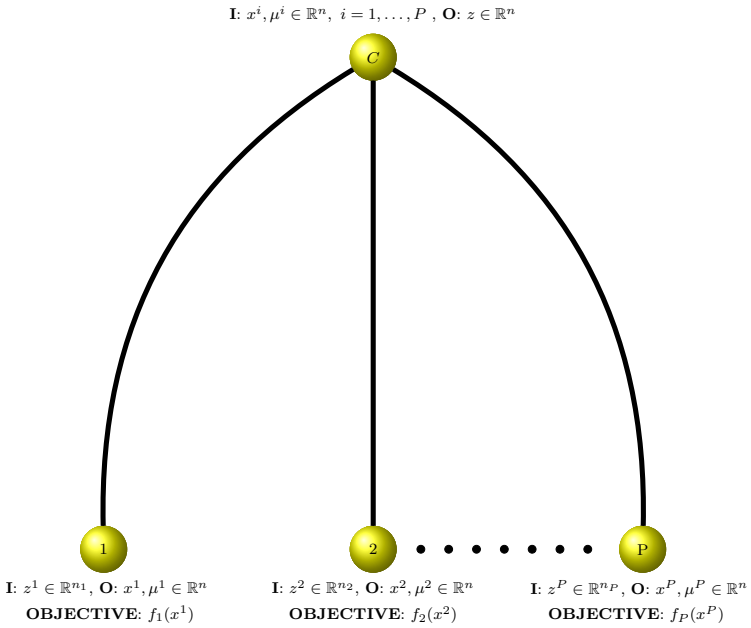

Figure 5.15: General global consensus problem.