Chapter 4

Computation, Complexity, Conic Programming

Source: most of the content of this chapter is described in Ben-Tal and Nemirowski's book on "Modern Convex Opitmization" [4]. An extensive treatment of the linear programming problem is found in Schrijver's book on linear programming [21]. Further reading include Bertsekas's book [5] and Boyd and Vandenberghe [6] (freely available) which content is a bit wider than our interest here. An interesting discussion between complexity theory and mathematics is given by [22]. Physical implications of complexity theory are given in [1]. Oracle complexity is extensively discussed in [16] and a more recent treatment is given in [17].

4.1 Introduction

When considering high dimensional statistics, computation has to be taken into account because the size of the problems to be adressed does not allow to ignore the computational cost of handling them. In particular, one may prefer a less statistically accurate estimator if it can be computed efficiently. Although very intuitive, the mathematical definition of "computation" is highly non trivial and has very strong connections to logics, physics and phylosophy. We start with a brief overview of theoretical computer science concepts which enlights computational properties of the statistical estimators we considered.

The second part of this chapter presents an overview of convex optimization as developped in the 90's. This resulted in classification of families of tractable convex optimization problems for which general purpose numerical solvers were developped. In the context of high dimensional statistics, these elements are mostly interesting for historical purposes as prefered methods for modern data analysis do not fall in the category of methods described in this chapter.

4.2 Models of computation

We first provide an overview of computation formalism. Most of this is borowed from Schrijver's book [21].

4.2.1 Computation over a finite alphabet and complexity over $\mathbb{Q}$

Alphabet, words, size: We consider a finite set Σ (usually $\Sigma = \{0, 1\}$), which is called an alphabet and its elements are called *letters*. An ordered finite sequence of elements in Σ is called a *word*. The set of words is denoted by Σ^*. The *size* of a string is the number of its components. The zero length string is the empty string $\emptyset$.

Strings can be used to represent rational numbers, vectors, matrices, and combinatorial structures such as graphs and trees. There are standard ways to encode these objects over a finite alphabet such as $\{0,1\}$, depending on the chosen way this induces a concept of size for these objects. For example if $\alpha = p/q$ (where p and q are relatively prime integers), $c = (c_1, \ldots, c_n)$ a rational vector and $A = (a_{ij})_{i=1\ldots m, j=1\ldots n}$ a rational matrix, we have

$$\begin{aligned}
\text{size}(\alpha) &= 1 + \lceil \log_2(p) \rceil + \lceil \log_2(q) \rceil \\
\text{size}(c) &= n + \sum_{i=1}^{n} \text{size}(c_i) \\
\text{size}(A) &= nm + \sum_{i=1}^{m} \sum_{j=1}^{n} \text{size}(a_{ij})
\end{aligned}$$

Size of linear inequalities, or equalities are defined in a similar way.

Problems: A *(search) problem* is a subset $\Pi \subset \Sigma^* \times \Sigma^*$, the corresponding meta-mathematical problem read as follows:

Given $z \in \Sigma^*$, find $y \in \Sigma^*$ such that $(z, y) \in \Pi$ or decide that there exists no such y.

An example of a search problem is given a matrix $A \in \mathbb{Q}^{m \times n}$ and a vector $b \in \mathbb{Q}^m$, find $x \in \mathbb{Q}^n$ such that $Ax \leq b$ (where the inequality is understood elementwise). A decision problem is a problem which output is either 0 or 1. For example, given A and b, is there an x such that $Ax \leq b$? A decision problem is often identified with $\mathcal{L} \subset \Sigma^*$, the set of inputs such that the output is 1.

Algorithm and running time: An algorithm is a list of instruction to solve a problem. A *Turing machine* is a thought experiment object which formalizes the notion of algorithm. The *Church-Turing thesis* is a founding hypothesis of computer science stating that functions of natural numbers computable by humans using pen and pencil, following an algorithm are precisely the ones which can be computed by a Turing machine. One can view a Turing machine as a device which performs pen and paper computation automatically and take it as a rigorous formalization of what it means "*to compute*". There exists equivalent formalizations such as recursive functions, lambda calculus, circuits which lead to equivalent notions of computation all of them are called *Turing complete.*

For a given input Σ^*, an algorithm for problem Π determines an output y such that (z, y) is in Π, or stops without delivering an output if there exists no such y. An algorithm can have the shape of a computer program, which is a finite string of symbols from a finite alphabet. Hence, an algorithm can be defined as a finite string A of 0's and 1's. One says that A solves problem Π, if for any instance z of Σ^*, when giving the string (A, z) to a *universal Turing machine* (a Turing machine which could simulate any other Turing machine, in particular, a Turing machine implementing A), the machine stops after a finite number of steps, and delivers y with $(z, y) \in \Pi$, or no string in the case where such a string y does not exist.

The running time of an algorithm is number of elementary operations during the execution of the algorithm. It depends on the precise implementation considered. One way to formalize this is the number of moves of the head of a universal Turing machine before stoping given the input (A, z). Formally, the runing time function of an algorithm $f \colon \mathbb{N} \mapsto \mathbb{N}$ can be given by

$$f(\sigma) = \max_{\text{size}(z) \leq \sigma} (\text{running time of } A \text{ for input } z).$$

Polynomial algorithm and computation over $\mathbb{Q}$ An algorithm is called *polynomial time*, if its time function is upper bounded by a polynomial. A problem is calle *polynomially solvable* if there exists a polynomial time algorithm to solve it.

The elementary operations such as adding, substracting, multiplying, dividing, comparing numbers can be executed in polynomial time. Note that for computation over $\mathbb{Q}$, we use the (polynomial time) Euclidean algorithm to obtain a unique representation of these numbers. Therefore, in order to show that a numerical algorithm is polynomial time, it suffices to show that it applies a number of elementary operations which is polynomial in the size of the input and that the size of the intermediate numbers to which these elementary operations are polynomially bounded by the size of the input.

Note that any numerical software, such as the ones used for statistical estimation, actually perform computation over $\mathbb{Q}$ as they implement finite precision arithmetic. This amounts to consider choose a finite precision $\epsilon \in \mathbb{Q}$, $\epsilon > 0$ and perform all numerical operations by rounding over a discrete grid $\{n\epsilon\}_{n \in \mathbb{Z}} \subset \mathbb{Q}$.

The classes $\mathcal{P}$ and $\mathcal{NP}$ and $\mathrm{co} - \mathcal{NP}$ The class of decision problems solvable in polynomial time is called $\mathcal{P}$. The class $\mathcal{NP}$ is central in complexity analysis and corresponds to decisions problems for which there is an easy to check verification, that is, which have a polynomial size proof. More formaly a decision problem $\mathcal{L} \subset \Sigma^*$ belongs to $\mathcal{NP}$ if there exists a polynomialy solvable decision problem $\mathcal{L}' \subset \Sigma^* \times \Sigma^*$ and a polynomial ϕ such that

$$z \in \mathcal{L} \quad \Leftrightarrow \quad \exists y \in \Sigma^*,\ (z, y) \in \mathcal{L}' \text{ and } \mathrm{size}(y) \leq \phi(\mathrm{size}(z)).$$

The crucial point here is that it is not required that y is found in polynomial time, but if it was given, the proof could be checked in polynomial time. The string y is called a certificate. Brute force search over all possible strings of a given length provides an algorithm showing that for any problem in $\mathcal{NP}$ there exists a polynomial ψ such that the solution for input z can be found in time at most $2^{\psi(\mathrm{size}(z))}$.

Example 4.2.1. *Given a set of cities and distances between cities (in $\mathbb{Q}$), the traveling salesman problem is in $\mathcal{NP}$:*

Given $d \in \mathbb{Q}$, decide if there is a path visiting all the cities of total length at most d.

Indeed, if such a path exists, it has the same length as the total number of cities so that checking that it passes through all cities and that its length is less than d can be done in polynomial time. Therefore, if the decision problem admits a solution, it has a polynomial time certificate.

Example 4.2.2. *Given $A \in \mathbb{Q}^{n \times d}$ and $b \in \mathbb{Q}^n$, consider the problem of deciding if $Ax \leq b$ has a solution over $\mathbb{Q}^n$. It can be shown (See Schiver's book chapter 10) that if such a solution exists, then there should be a solution which size is polynomially bounded by the size of A and b. Hence this decision problem is in $\mathcal{NP}$.*

The class of decision problems $\mathcal{L} \subset \Sigma^*$ which complement in Σ^* is in $\mathcal{NP}$ is denoted by $\mathrm{co} - \mathcal{NP}$. The class $\mathcal{NP} \cap \mathrm{co} - \mathcal{NP}$ consists of those decision problems which answer (positive or negative) have a polynomial length proof. We have $\mathcal{P} \subset \mathcal{NP}$ and $\mathcal{P} \subset \mathrm{co} - \mathcal{NP}$ and it is not known wether these inclusions are strict (there is a million dollars price on these questions).

The term $\mathcal{NP}$ comes from "Non deterministic Polynomial time". This means that a lucky algorithm which has the possibility to "guess" in polynomial time a good certificate over a set with polynomial size can solve the corresponding decision problem.

Karp reduction and $\mathcal{NP}$ completeness A decision problem $\mathcal{L} \in \Sigma^*$ is *Karp* reducible to a decision problem $\mathcal{L}' \subset \Sigma^*$ if there exists a polynomial time algorithm such that, for any input string $z \in \Sigma^*$, A delivers a string x such that

$$z \in \mathcal{L} \quad \Leftrightarrow \quad x \in \mathcal{L}'$$

This can be denoted as $\mathcal{L} \leq \mathcal{L}'$ as an algorithm for solving $\mathcal{L}'$ would provide an algorithm for solving $\mathcal{L}$ with an added computational cost which is at most polynomial.

Example 4.2.3. *For any boolean formula there is an equivalent formula over linearly more variable in conjunctive normal form. The size of the new formula is at most linear in the size of the original formula, using Tseytin transformation for example. We obtain a formula of the form*

$$(a \vee b \vee c \vee d) \wedge (\bar{a} \vee e \vee f \vee \bar{g} \vee d) \dots$$

Then any disjunction can be reduced to a conjunction of disjunctions of size at most 3 *by adding variables. For example*

$$\begin{aligned} & q \vee r \vee s \vee t \vee u \\ \Leftrightarrow \quad & (q \vee r \vee a) \wedge (\bar{a} \vee s \vee b) \wedge (\bar{b} \vee t \vee u). \end{aligned}$$

Thus if $\mathcal{L}$ denotes the boolean formula satisfiability problem (SAT) and $\mathcal{L}'$ denotes the satisfiability problem of boolean formula in 3 conjunctive normal form (3-SAT), we have shown that $\mathcal{L} \leq \mathcal{L}'$.

Similarly, if $\mathcal{L}'$ belongs to $\mathcal{NP}$ and $\mathcal{L} \leq \mathcal{L}'$, then $\mathcal{L}$ also belongs to $\mathcal{NP}$. A problem $\mathcal{L}$ is called *$\mathcal{NP}$-hard*, if each problem in $\mathcal{NP}$ is reducible to $\mathcal{L}$ and if furthermore, $\mathcal{L}$ is in $\mathcal{NP}$, then $\mathcal{L}$ is called *$\mathcal{NP}$-complete*. As we have seen, we have an exponential time algorithm to solve problems in $\mathcal{NP}$, this is a brute force search algorithm. It is widely believed that for a given $\mathcal{NP}$-complete problem, this is the most efficient algorithm to solve all the possible instances. Indeed, a polynomial time algorithm for any $\mathcal{NP}$ complete problem would provide a proof that $\mathcal{P} = \mathcal{NP}$ which is widely believed to be false. This is undelying the $\mathcal{P} \neq \mathcal{NP}$ conjecture. It is important to note that the notion of $\mathcal{NP}$-hardness is a *worst case* notion.

- $\mathcal{NP}$-complete problems are considered to be hard as there si no known polynomial time algorithm to solve them and it is believed that no such algorithm exists.
- This concept relies on Karp reduction which only underlines that some instances are hard, not necessarily all of them.
- There is no notion of constant or exponent in these concepts so that an algorithm in $\mathcal{P}$ may still be intractable in practice. The notion is mostly used to prove computational difficulty of certain problems.

From optimization to decision An optimization problem is the minimization of an objective function c over a finite set or over rational numbers. An efficient algorithm to solve an optimization problem provide an algorithm to decide if there exists a sequence of input with cost less or equal to α, for any α. For example given $A \in \mathbb{Q}^{n \times d}$, $b \in \mathbb{Q}^n$, $c \in \mathbb{Q}^d$, computing

$$\rho = \inf_{Ax \leq B} c^T x$$

provides an algorithm to decide whether $Ax \leq b$ and $c^T x \leq \alpha$ has a solution for any $\alpha \in \mathbb{Q}$. As a result, optimization objectives involving $\mathcal{NP}$-complete problems are considered as hard.

Examples:

Example 4.2.4 (Cook's Theorem). *The boolean satisfiability problem (SAT) consists of decision problem over boolean variables involving boolean formulas in conjunctive normal form: the variables are augmented with their negations, and the formula consists of a conjunction of disjunctions (all clauses made using "or" and are aggregated with an "and"). Example*

$$(x_1 \textit{ and } x_2 \textit{ and } x_6) \textit{ or } (\bar{x_2} \textit{ and } x_3 \textit{ and } \bar{x_7}) \textit{ or } \dots$$

This is the first problem proved to be $\mathcal{NP}$-complete by Cook in 1971.

The idea of the proof is as follows. First the problem is clearly in $\mathcal{NP}$ as it suffices to exhibit a an instance of boolean values which satisfy the formula. The problem is $\mathcal{NP}$-hard because because *a polynomial time verifier implemented on a Turing machine can be shown to be equivalent to a boolean formula (this is the technical bulk of the proof). Finaly there is a polynomial time reduction from any boolean formula to a formula of the above form where each disjunction involves at most* 3 *variables.*

This problem remains $\mathcal{NP}$-complete if we restrict the disjunctions to involve at most 3 *variables (as in the example) by the* 3*-SAT reduction argument. This shows that* 3*-SAT is $\mathcal{NP}$-complete.*

Example 4.2.5. *An important list of $\mathcal{NP}$-complete problems can be found in the classic book, Computers and Intractability: A Guide to the Theory of NP-Completeness.*

Theorem 4.2.1. *Consider the decision problem with input $A \in \mathbb{Q}^{m\times n}$, $b \in \mathbb{Q}^m$, does there exist $x \in \mathbb{Q}^n$ such that $Ax = b$ and $\|x\|_0 \leq m/3$. This problem is $\mathcal{NP}$-hard.*

Proof. We reproduce the proof given in [15]. First, the problem is clearly in $\mathcal{NP}$. Completness is shown by reduction is to "cover by 3 sets" which is an $\mathcal{NP}$-complete problem:

Given a set S and a set C which elements consists of subsets of S of size 3. Decide if there is $\hat{C} \subset C$ such that each elements of S occurs exactly once in $\hat{C}$.

Assume that $S = \{s_1, \dots, s_m\}$ and $C = \{c_1, \dots, c_n\}$ and assume that m is a multiple of 3. Set $b = (1, \dots, 1)^T \in \mathbb{Q}^m$ and $A \in \mathbb{R}^{m\times n}$ which column i is zero except at the j, k, l where $(s_j, s_k, s_l) = c_i$, $i = 1, \dots, n$. We show that there exists $x \in \mathbb{Q}^n$ such that $Ax = b$ with $\|x\|_0 \leq m/3$ if an only if the answer to the "cover by 3 set problem" is positive.

On the one hand given $\hat{C}$, choosing $x_i = 1$ if $c_i \in \hat{C}$ and zero otherwise. We have $Ax = b$ and x has $m/3$ non zero entries. On the other hand if one finds $x \in \mathbb{Q}^n$ with $Ax = b$ and $\|x\|_0 \leq m/3$. The entries of Ax must be in 1. Since x has at most $m/3$ non zero entries and each column has at most 3 nonzero entries, it means that x has exactly $m/3$ nonzero entries. The nonzero entries of x solves the "cover by 3 set" problem. □

The theorem generalizes to real inputs and real variables in the computation model of infinite precision RAM model or computation over the reals and approximate solutions $\|Ax - b\|_2^2 \leq \epsilon$, see [15]. The implication of these results is that solving problems involving the $\|\cdot\|_0$ pseudonorm is hard. For example computing $\hat{\theta}^{\ell_0}$ can be done by solving 2^d unconstrained least squares problems, and, unless $\mathcal{P} = \mathcal{NP}$, no algorithm can do significantly better on all possible instances for all values of d. This underlines the value of the question $\mathcal{P} = \mathcal{NP}$? This kind of statement is very common in computer science: if there is a reduction from a given problem to another problem which is proved (or largely believed) to be hard, then the original problem must be hard.

4.2.2 Computation over the reals

The statistical estimators which are defined in previous chapters, are given over the real field and we only mentioned computation over the rationals so far. The difference may look innocuous at first sight but it actually has tremendous implications. Furthermore most of the optimization theory which we are going to describe is given for algorithms over the reals, and therefore, it is worth mentioning models of computation over the reals. The content of this section is mostly theoretical since real arithmetic is not realisable in the physical word [1] (it would break well accepted physical impossibility principles).

Computable number: Computer Algebra Systems use symbolic programing to perform operations on algebraic objects. However the set of real numbers which can be described by such systems is only denumerable and therefore, miss most of the reals. Another definition of computable number concerns the possibility to approximate it up to an arbitrary precision.

Definition 4.2.1. *A number $a \in \mathbb{R}$ is called computable if there is a terminating algorithm A such that for any $\epsilon \in \mathbb{Q}$, $\epsilon > 0$, $|A(\epsilon) - a| \leq \epsilon$.*

Intuitively, there are only countably many terminating algorithms and the set of computable numbers is therefore only countable. Hence, most real numbers are not computable in this sense.

Real machines: In 1989 Blum, Shub and Smale described a theoretical machine for real computation. This is refered to as *BSD* machine or *real RAM* machine. Roughly speaking such a machine manipulates real numbers instead of element of a finite alphabet and is able to perform addition, multiplication, division and comparison over real numbers.

This is a canonical model for computation over the reals. Although not realisable in the physiscal word, this constitutes an interesting thought experiment. For example, we will see that the linear programming (LP) problem (4.7) is polynomially solvable over the rationals, but it is not known if it is polynomialy solvable over the reals [22]. The main difference between computation over $\mathbb{Q}$ and over $\mathbb{R}$ is that in the first case, the size of the input (number of bits required to decribe it using standard encoding) provides a bound on the accuracy level required to obtain provably correct rounding schemes. In the real case, the size of the input is only the number of entries. For example the condition number of a matrix A depends on its size over $\mathbb{Q}$ while it does not depend on its size over $\mathbb{R}$.

Oracle complexity: The computational model underlying continuous optimization mixes real computation and unknown primitives which are provided by an oracle. For example, if one wishes to minimize a differentiable function f over $\mathbb{R}^d$, one can construct an algorithm which is allowed to querry sequentially the value of f and its gradient ∇f at different points in $\mathbb{R}^d$. The running time of an algorithm is given by the number of call to the oracle and the number of real arithmetic operations performed by the algorithm. Complexity is then given by worst case bounds on the number of operations required solve a specific problem and it usually depends on properties of f such as conditioning. Depending on the oracle of choice, one may define different notions of running time and complexity for optimization algorithms. Note that although this model is quite intuitive, it is actually very far from what is performed in practice when using physical computers. Nemirovski Yudin [16] introduced this notion of complexity as a systematic way to study continous optimization algorithms, further comments and a more recent exposition can be found in the book of Nesterov [17].

4.3 Recap on convexity

We limit ourselves to the finite dimensional setting which is sufficient for our purpose. Most notions given here generalize to infinite dimensions [14, 3]. The content of this section is mostly related to [4, 6].

Convex sets and functions

A subset of a vector space $\mathcal{X}$ is convex if it is closed under convex combinations.

Definition 4.3.1. *Let $\mathcal{X} \subset \mathbb{R}^d$, we say that $\mathcal{X}$ is convex if for any $x, y \in \mathcal{X}$, $\alpha \in [0,1]$, $\alpha x + (1-\alpha)y \in \mathcal{X}$. A function $f\colon \mathbb{R}^d \to \mathbb{R}$ is convex if its epigraph is convex in $\mathbb{R}^{d+1}$. Recall that $\mathrm{epi}(f) = \left\{(x,z) \in \mathbb{R}^{d+1},\, z \geq f(x)\right\}$. Equivalently, for any $x, y \in \mathbb{R}^d$, and any $\alpha \in [0,1]$, $f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y)$.*

Convex sets are closed under many set operations including, interior, closure, intersection, (Minkowski) addition, affine mapping and inverse affine mapping. There is a well defined notion of dimenision for convex set $\mathcal{X}$, it is simply the dimension of the smallest affine set containing $\mathcal{X}$.

Lemma 4.3.1. *For any convex set $\mathcal{X} \subset \mathbb{R}^d$ we have*

- *The closure of $\mathcal{X}$ is convex.*
- *The interior of $\mathcal{X}$ is convex.*
- *For any $u \in \mathrm{int}(\mathcal{X})$ and $v \in \mathrm{cl}(\mathcal{X})$, $[u, v) \subset \mathrm{int}(\mathcal{X})$.*
- *If the interior of $\mathcal{X}$ is non empty, then $\mathrm{cl}(\mathcal{X}) = \mathrm{cl}(\mathrm{int}(\mathcal{X}))$.*

- *The interior of $\mathcal{X}$ is empty if and only if it is contained in a lower dimensional affine subspace.*

Characterization of convex functions

We have the following characterizations of convexity

Theorem 4.3.1. *Let $f\colon \mathbb{R}^d \to \mathbb{R}$:*

1. *If f is continuously differentiable, then f is convex if and only if or any $x, y \in \mathbb{R}^d$, $f(y) \geq f(x) + \nabla f(x)^T(y-x)$.*
2. *If f is continuously differentiable, then f is convex if and only if or any $x, y \in \mathbb{R}^d$, $(\nabla f(x) - \nabla f(y))^T(y-x) \geq 0$.*
3. *If f is twice continuously differentiable, then f is convex if and only if or any $x \in \mathbb{R}^d$, $\nabla^2 f(x)$ is positive semidefinite.*

One has the following consequence which is a central motivation for studying convex optimization problems

Corollary 4.3.1. *Let $f\colon \mathbb{R}^d \to \mathbb{R}$ be a convex continuously differentiable function, then the following are equivalent*

- *x is a global minimizer of f.*
- *$\nabla f(x) = 0$.*

Example 4.3.1. *Consider the least squares linear regression estimate $\hat{\theta}^{LS} \in \arg\min_{\theta \in \mathbb{R}^d} \|\mathbb{X}\theta - y\|_2^2$. The hessian matrix of the objective is $\mathbb{X}^T\mathbb{X}$ which is positive semidefinite so that the objective is convex and first order conditions are sufficient for optimality.*

Separating hyperplane and supporting hyperplane

Theorem 4.3.2 (Separating hyperplane)**.** *Let $\mathcal{X}, \mathcal{Y} \subset \mathbb{R}^d$ be two disjoint closed convex sets, then there exists a vector $v \in \mathbb{R}^d$, $v \neq 0$ and a number $c \in \mathbb{R}$ such that $x^T v > c$ for all $x \in \mathcal{X}$ and $y^T v < c$ for all $y \in \mathcal{Y}$.*

Proof. Set $S = \mathcal{X} - \mathcal{Y} = \{s = x - y,\ x \in \mathcal{X},\ y \in \mathcal{Y}\}$, S is convex and closed. Since $\mathcal{X}$ and $\mathcal{Y}$ are disjoint, $0 \notin S$. Let $\bar{s}$ denote any minimal norm element of s. For any $s \in S$, and $t \in [0,1]$,

$$0 < \|\bar{s}\|_2^2 \leq \|\bar{s} + t(s - \bar{s})\|_2^2 = \|\bar{s}\|_2^2 + 2t\bar{s}^T(s - \bar{s}) + t^2\|(s - \bar{s})\|_2^2.$$

The right hand side is differentiable for $t \in \mathbb{R}$ and the derivative at 0 must be non negative. Hence, for any $s \in S$, $s^T\bar{s} \geq \|\bar{s}\|_2^2 > 0$. We deduce that

$$\inf_{x \in \mathcal{X}} \bar{s}^T x = \|\bar{s}\|_2^2 + \sup_{y \in \mathcal{Y}} \bar{s}^T y$$

which shows that we can choose $v = \bar{s}$ and any $c \in \left(\inf_{x \in \mathcal{X}} \bar{s}^T x, \sup_{y \in \mathcal{Y}} \bar{s}^T y\right)$ where the interval is non empty. □

We deduce the following which is a weak finite dimensional form of the Hahn Banach theorem.

Theorem 4.3.3 (Supporting hyperplane)**.** *Let $\mathcal{X} \subset \mathbb{R}^d$ be a convex sets such that $0 \notin \mathcal{X}$, then there exists a vector $v \in \mathbb{R}^d$, $v \neq 0$ such that $v^T x \geq 0$, for all $x \in \mathcal{X}$.*

Proof. If $0 \notin \mathrm{cl}(\mathcal{X})$ then the result follows immediately from the separating hyperplane theorem. If $0 \in \mathrm{cl}(\mathcal{X})$, since $0 \notin \mathcal{X}$, $0 \notin \mathrm{int}(\mathcal{X})$, and 0 is on the boundary of $\mathcal{X}$. Hence 0 is in the closure of the complement of $\mathrm{cl}(\mathcal{X})$ and there exists a sequence $\{z_k\}_{k\in\mathbb{N}}$ not in $\mathrm{cl}(\mathcal{X})$. Which converges to 0. Applying the separating hyperplane theorem to each element of the sequence ensures that there exists a sequences $\{v_k\}_{k\in\mathbb{N}}$ non zero in $\mathbb{R}^d$ and $\{c_k\}_{k\in\mathbb{N}}$ in $\mathbb{R}$, such that for all $k \in \mathbb{N}$ and all $x \in \mathcal{X}$,

$$\frac{v_k^T z_k}{\|v_k\|} < \frac{c_k}{\|v_k\|} < \frac{v_k^T x}{\|v_k\|}.$$

Let v be any accumulation point of $\frac{v_k}{\|v_k\|}$, the left hand side of tends to 0 hence $\liminf_{k\to\infty} \frac{c_k}{\|v_k\|} \geq 0$ and $v^T x \geq 0$ for all $x \in \mathcal{X}$. □

If $0 \in \mathrm{cl}(\mathcal{X})$, the vector v defines a supporting hyperplane which provides a notion of tangent to a set convex set.

Theorem 4.3.4 (Supporting hyperplane). *Let $\mathcal{X} \subset \mathbb{R}^d$ be a convex set such that 0 is on the boundary of $\mathcal{X}$, then there exists a vector $v \in \mathbb{R}^d$, $v \neq 0$ such that $v^T x \geq 0$, for all $x \in \mathcal{X}$.*

Proof. If $0 \notin \mathcal{X}$ then the result follows from Theorem 4.3.3, we assume that $0 \in \mathcal{X}$.

If the interior of $\mathcal{X}$ is empty, then, by Lemma 4.3.1, $\mathcal{X}$ is contained in a lower dimensional affine space which turns out to be a linear subspace since $0 \in \mathrm{cl}(\mathcal{X})$ any vector orthogonal to this subspace will work.

If $\mathrm{int}(\mathcal{X})$ is not empty since it is convex by Lemma 4.3.1, we may apply Theorem 4.3.3 to 0 and $\mathrm{int}(\mathcal{X})$ and obtain $v \in \mathbb{R}^p$ such that $v^T x \geq 0$ for all $x \in \mathrm{int}(\mathcal{X})$. Lemma 4.3.1 ensures that $\mathrm{cl}(\mathrm{int}(\mathcal{X})) = \mathrm{cl}(\mathcal{X})$ so that $v^T x \geq 0$ for all $x \in \mathrm{cl}(\mathcal{X}) \supset \mathcal{X}$ and the result follows. □

More generally, a supporting hyperplane of $\mathcal{X}$ at x is a closed half space which contains $\mathcal{X}$ and x on its boundary. There is a partial converse.

Theorem 4.3.5. *Let $\mathcal{X}$ be a closed set with nonempty interior, such that for every point x on the boundary of $\mathcal{X}$ admits a supporting hyperplane. Then $\mathcal{X}$ is convex.*

Proof. $\mathcal{X}$ is contained in the set S consisting of intersection of all the half spaces given by all the supporting hyperplanes at each point of the boundary of $\mathcal{X}$. S is convex and closed as the intersection of closed convex sets. Fix any $s \in S$, and assume that $s \notin \mathcal{X}$, choose x in the interior of $\mathcal{X}$, the line segment between s an x crosses the boundary of $\mathcal{X}$ at $y \in \mathcal{X}$. The supporting hyperplane at y provides an affine function A which is positive on $\mathcal{X}$. Restriction of this affine function to the line segment $[x, s]$ is still affine with $A(x) > 0$, $A(y) = 0$ and $y \in (x, s)$, and hence $A(s) < 0$ which contradicts the fact that $s \in S$. Therefore, $S = \mathcal{X}$. □

There is a stronger notion of separating hyperplane.

Theorem 4.3.6 (Separating hyperplane). *Let $\mathcal{X}, \mathcal{Y} \subset \mathbb{R}^d$ be two disjoint convex sets, then there exists a vector $v \in \mathbb{R}^d$, $v \neq 0$ and a number $c \in \mathbb{R}$ such that $x^T v \geq c$ for all $x \in \mathcal{X}$ and $y^T v \leq c$ for all $y \in \mathcal{Y}$.*

Extreme points, polyhedra and polytopes

Definition 4.3.2. *Let $\mathcal{X} \subset \mathbb{R}^d$ be a convex set and $x \in \mathcal{X}$. x is an extreme point of $\mathcal{X}$ if for any $x_1, x_2 \in \mathcal{X}$, $x = (x_1 + x_2)/2$ implies that $x_1 = x_2 = x$.*

Any nonempty compact convex subset of $\mathbb{R}^d$ contains at least one extreme point (any point of maximal norm). The convex hull of a set S is the set of all convex combinations of elements of S, denoted by

$$\mathrm{conv}(S) = \left\{ x,\, \exists n \in \mathbb{N}^*, (x_i)_{i=1}^n \in S^n,\, (\lambda_i)_{i=1}^n \in \mathbb{R}_+^n,\, \sum_{i=1}^n \lambda_i = 1,\, x = \sum_{i=1}^n \lambda_i x_i \right\}.$$

It is seen from the definition that the extreme points of $\operatorname{conv}(S)$ are contained in S. The interest of extreme points is that linear optimization attains its optima at extreme points.

Lemma 4.3.2. *Let $\mathcal{X}$ be a closed convex set, $x \in \mathcal{X}$ such that there exists $c \neq 0$ and $c^T x = \inf_{y \in \mathcal{X}} c^T y$. Then setting $A = \left\{ y \in \mathcal{X},\ y^T c = x^T c \right\}$, any extreme point of A is an extreme point of $\mathcal{X}$.*

Proof. Take $\tilde{p}$ to be one extreme point of A, and suppose that we have $x_1, x_2 \in \mathcal{X}$ such that $(x_1 + x_2)/2 = \tilde{p}$. We have $x_1^T c \geq x^T c$, $x_2^T c \geq x^T c$ and $\frac{1}{2}(x_1 + x_2)^T c = \tilde{p}^T c = x^T c$, the average of two non negative numbers is 0 if and only if both are null and hence $x_1^T c = x_2^T c = \tilde{p}^T c$ and $x_1 \in A$ and $x_2 \in A$. Hence $x_1 = x_2 = \tilde{p}$. □

Lemma 4.3.3. *Let $c \in \mathbb{R}^d$, $c \neq 0$ and $\mathcal{X}$ be a convex and compact set. Then $\min x \in \mathcal{X} c^T x$ is attained then the optimum is attained at an extreme point $\bar{x} \in \mathcal{X}$.*

Proof. If $c = 0$, any extreme point of $\mathcal{X}$ is a solution. If $c \neq 0$, by the compactness of $\mathcal{X}$, the optimum of the problem is attained. Take x^* to be one solution. The set $\{x \in \mathbb{R}^d,\ x^T c = (x^*)^T c\}$ is compact and convex, it contains an extreme point which by Lemma 4.3.2 is an extreme point of $\mathcal{X}$. □

Theorem 4.3.7 (Krein Millman)**.** *Let $\mathcal{X}$ be a compact convex set, then $\mathcal{X} \subset \mathbb{R}^d$ is the convex hull of its extreme points.*

Proof. Let S denote the set of extreme points of $\mathcal{X}$, we have $\operatorname{conv}(S) \subset \mathcal{X}$. Let $x \in \mathcal{X}$, we show that x is in $\operatorname{conv}(S)$. First, we may assume that $\mathcal{X}$ has non empty interior, by reducing the ambiant space to the smallest affine subspace containing $\mathcal{X}$. The proof is now by recursion on d. For $d = 0$, the result is obvious, assume that the result holds for $\mathbb{R}^{d-1}$, $d \geq 1$. Consider any line passing through x, the restriction of $\mathcal{X}$ to this line is compact convex set of dimension 1, that is a segment of the form $[a, b]$ where $a \in \mathcal{X}$, $b \in \mathcal{X}$. Both a and b are on the boundary of $\mathcal{X}$. There is a supporting hyperplane H_a at a and H_b at b. Both sets $\mathcal{X} \cap H_a$ and $\mathcal{X} \cap H_b$ are compact convex sets of dimension $d - 1$. The induction hypothesis ensures that both a and b are convex combinations extreme points of H_a and H_b which are extreme points of $\mathcal{X}$ by Lemma 4.3.2 and the result follows because m is a convex combination of a and b. □

Definition 4.3.3. *A polyhedra is a set $\mathcal{X} \subset \mathbb{R}^d$ which can be described by linear equalities: there exists $A \in \mathbb{R}^{m \times d}$, $b \in \mathbb{R}^m$ such that $\mathcal{X} = x \in \mathbb{R}^d$, $Ax \leq b$, where the inequality is understood entry-wise. This representation is called canonical form.*

Adding slack variables $s \in \mathbb{R}^m$ and considereing x_+ and x_- the entry-wise positive and negative part of x, one may equivalently describe $\mathcal{X} = \{(x_+, x_-, s) \in \mathbb{R}^{2n+m},\ s = b - A(x_+ - x_-),\ s \geq 0, x_+ \geq 0, x_- \geq 0\}$. Hence, one may equivalently consider polyhedra expressed as $\mathcal{X} = \{x \in \mathbb{R}^d, Ax = b, x \geq 0\}$ for a matix A and a vector b which is called standard form.

Lemma 4.3.4. *Let $\mathcal{X} = \left\{ x \in \mathbb{R}^d, Ax = b, x \geq 0 \right\}$ be non empty. Then $\mathcal{X}$ has at least one extreme point and we have the following equivalence*

- *x is an extreme point of $\mathcal{X}$*
- *the columns of A corresponding to non zero entries of x are independent.*

Proof. The existence of extreme points follow from the characterization. If $A = 0$, then $x = 0$ is an extreme point. Suppose that $A \neq 0$, for any $x \in \mathbb{R}^d$, denote by A_x the matrix which columns correspond to the non zero entries of x. For any $x, x_1, x_2 \in \mathcal{X}$, if $x = \frac{x_1 + x_2}{2}$, then $\operatorname{supp}(x_1) \subset \operatorname{supp}(x)$ and $\operatorname{supp}(x_2) \subset \operatorname{supp}(x)$ and $A_x(x_1 - x_2) = 0$ hence, if the columns of A_x are independent, $x_1 = x_2$ and x is an extreme point. On the other hand, if the columns of A_x are not independant, choosing $d \in \mathbb{R}^d$ such that $\operatorname{supp}(d) = \operatorname{supp}(x)$ and $Ad = 0$, one has, for sufficiently small alpha that $x + \alpha d \in \mathcal{X}$ and $x - \alpha d \in \mathcal{X}$ so that x is not an extreme point of $\mathcal{X}$. □

As a result, polyhedra have only finitely many extreme points. A polytope is a compact polyhedra. Krein-Millman theorem ensures that $\mathcal{X}$ is a polytope if and only if it is the convex hull of finitely many points.

Example 4.3.2. *The ℓ_1 ball used to define the $\ell - 1$ constrained least squares estimator:*

$$\hat{\theta}_K^{LS} \in \arg \min_{\|\theta\|_1 \le 1} \|\mathbb{X}\theta - y\|_2^2$$

is a polytope which has $2d$ extreme points corresponding to plus or minus the elements of the canonical basis. Linear fuction over the ℓ_1 ball attains their optimum at one of these extreme points which have a support of size 1. This illustrates the sprasity promoting role of this constraint.

4.4 Conic programming

4.4.1 Conic hierarchy

Definition 4.4.1. *$\mathcal{K} \subset \mathbb{R}^d$ is a cone if it satisfies for any $x \in \mathcal{K}$ and $\alpha \ge 0$, $\alpha x \in \mathcal{K}$.*

Given a closed convex cone $\mathcal{K}$, one can define the corresponding conic program, for any $A \in \mathbb{R}^{m\times d}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^d$,

$$p^* = \inf_{x \in \mathbb{R}^d} \quad c^T x \qquad \text{s.t.} \qquad Ax = b,\, x \in \mathcal{K}. \tag{P}$$

This gives rise to the following classes of optimization problems.

Linear programs: Choosing $\mathcal{K} = \mathbb{R}_+^d$, we obtain a linear program in standard form. The problem of computing $\hat{\theta}_{CS}^{\ell_1}$ can be expressed as a linear program as

$$\begin{aligned} &\min_{\theta \in \mathbb{R}^d} \|\theta\|_1 \quad \text{s.t.} \quad \mathbb{X}\theta = y \\ = &\min_{\theta_+ \in \mathbb{R}^d, \theta_- \in \mathbb{R}^d} 1^T(\theta_+ + \theta_-) \quad \text{s.t.} \quad \mathbb{X}(\theta_+ - \theta_-) = y,\, \theta_+ \in \mathcal{K},\, \theta_- \in \mathcal{K}. \end{aligned}$$

which is a linear program (LP).

Second order cone: The second order cone in $\mathbb{R}^{d+1}$ is given by $\mathcal{K} = \{(x,t) \in \mathbb{R}^{d+1},\, \|x\|_2 \le t\}$. This allows to express linear optimization over convex quadratic constraints such as balls or ellipses and their intersection. Such a problem is called a second order cone program (SOCP).

Semidefinite cone: The set of symmetric positive semidefinite is called the semidefinite cone. Given a symmetric matrix $C \in \mathbb{R}^{d\times d}$, a linear function $\mathcal{A}\colon \mathbb{R}^{d\times d} \to \mathbb{R}^m$ and $b \in \mathbb{R}^m$ a semidefinite program has the form

$$\min_{X \in \mathbb{R}^{d\times d}} \operatorname{tr}(C^T X) \quad \text{s.t.} \quad \mathcal{A}(X) = b,\, X^T = X,\, X \succcurlyeq 0.$$

Such programs are called semidefinite programs (SDP).

Hierarchy of conic programs These conic programs are standard optimization problems for which there exists efficient algorithms allowing to solve numerically efficiently moderate size programs of this type. The term hierarchy refers to the fact that linear programs can be expressed as second order cone programs and second order cone programs can be expressed as semidefinite programs.

4.4.2 Conic duality

Definition 4.4.2. *Let $\mathcal{K} \subset \mathbb{R}^d$ be a convex cone, the dual cone of $\mathcal{K}$ is denoted by*

$$\mathcal{K}^* = \left\{ y \in \mathbb{R}^d,\, x^T y \geq 0,\, \forall x \in \mathcal{K} \right\}$$

If $\mathcal{K} = \mathcal{K}^*$, we say that $\mathcal{K}$ is self dual

All the cones given in the previous section are self-dual. The Lagrangian of problem (P) is given for any $x \in \mathbb{R}^d$, $\mu \in \mathbb{R}^d$, $\nu \in \mathbb{R}^m$, by

$$\mathcal{L}(x, \mu) = c^T x + \mu^T (b - Ax) \tag{4.1}$$

The dual problem to (P) is obtained by minimizing the Lagrangian over $x \in \mathcal{K}$. If $c^T - A^T \mu \notin \mathcal{K}^*$, the infimum of the Lagrangian over $x \in \mathcal{K}$ is $-\infty$. On the other hand, if $c - A^T \mu \in \mathcal{K}^*$, then the minimizer of the Lagrangian is $\mu^T b$. Hence the dual problem has the form

$$d^* = \sup b^T \mu \qquad \text{s.t.} \qquad c - A^T \mu \in \mathcal{K}^*. \tag{D}$$

We have the following relation between primal (P) and dual problems (D).

Theorem 4.4.1. *It holds that $d^* \leq p^*$. Furthermore, if $\operatorname{rank}(A) = m$, and there exists $\bar{x}$ such that $A\bar{x} = b$ and $\bar{x}$ is in the interior of $\mathcal{K}$ and $p^* > -\infty$, then $p^* = d^*$ and the dual problem has a solution. In this case, x is primal optimal if and only if it is primal feasible and there exists a dual feasible μ such that*

$$x^T (c - A^T \mu) = 0 \qquad \text{or} \qquad x^T c = b^T \mu.$$

Proof. If either the primal (P) or the dual problem (D) are not feasible, then the result is obvious as $p^* = +\infty$ or $d^* = -\infty$.

Assuming that both are feasible, for any x feasible for (P) and μ feasible for (D), we have

$$c^T x = c^T x + \mu^T (b - Ax) = \mathcal{L}(x, \mu) = \mu^T b + (c - A^T \mu)^T x \geq \mu^T b \tag{4.2}$$

where the first equality is from primal feasibility, and the last inequality is because $x \in \mathcal{K}$ and $c - A^T \mu \in \mathcal{K}^*$ so that the dot product is nonnegative. This implies that $p^* \geq d^*$

To obtain strong duality (not assuming dual feasibility), consider the sets

$$S_1 = \left\{ (u - x), b - Ax, c^T x + t) \in \mathbb{R}^{d+m+1},\, x \in \mathbb{R}^d,\, u \in \mathcal{K},\, t \geq 0 \right\} \qquad S_2 = \{(0, 0, s),\, s < p^*\}$$

It holds that both S_1 and S_2 are convex. Furthermore, they are disjoint since an element of the intersection would provide a primal feasible x with $c^T x < p^*$. Theorem 4.3.6 ensures that there exists $\alpha_1 \in \mathbb{R}^d$, $\alpha_2 \in \mathbb{R}^m$, $\alpha_3 \in \mathbb{R}$, not all equal to 0, and $\alpha_4 \in \mathbb{R}$ such that for all $x \in \mathbb{R}^d$, $u \in \mathcal{K}$, $t \geq 0$, $s < p^*$

$$\begin{aligned} \alpha_1^T (u - x) + \alpha_2^T (b - Ax) + \alpha_3 (c^T x + t) &\geq \alpha_4 \\ \alpha_3 s &\leq \alpha_4 \end{aligned} \tag{4.3}$$

It must hold that $\alpha_3 \geq 0$ and $\alpha_1 \in \mathcal{K}^*$, otherwise, the left hand side of the first inequality is unbounded from below. From the second inequality, we obtain $\alpha_3 p^* \leq \alpha_4$. We are going to show that the strict feasibility condition ensures that $\alpha_3 > 0$. First note that if $x \in \operatorname{int}(\mathcal{K})$, then for any nonzero $\mu \in \mathcal{K}^*$, we have $x^T \mu > 0$. Second assuming that $\alpha_3 = 0$, choosing $\bar{x}$ as given in the hypothesis and $u = 0$, one has

$$-\alpha_1^T \bar{x} \geq \alpha_3 p^* = 0,$$

and since $\alpha_1^T \neq 0$ implies $\alpha_1^T \bar{x} > 0$ we have $\alpha_1 = 0$. Furthermore, we have $\alpha_2 \neq 0$ and $\alpha_2^T(b - Ax) \geq 0$, for all $x \in \mathbb{R}^d$. This is impossible as it would imply $\alpha_2^T A = 0$ with $\alpha_2 \neq 0$ which contradicts the rank assumption on A.

Finally, $\alpha_3 > 0$ and we obtain from (4.3), for any $x \in \mathcal{K}$

$$\frac{\alpha_2^T}{\alpha_3}(b - Ax) + c^T x \geq p^* + \frac{\alpha_1^T}{\alpha_3} x \geq p^*,$$

where the last inequality follows because $\alpha_1 \in \mathcal{K}^*$. This implies that $c - A^T \alpha_2 / \alpha 3 \in \mathcal{K}^*$ as otherwise, the right hand side would be unbounded from below. Hence $\mu = \alpha_2/\alpha_3$ is dual feasible. Minimizing over x, we obtain that $b^T \mu \geq p^*$. Hence $d^* \geq p^*$ and by weak duality, $d^* = p^*$ and μ is dual optimal.

The last statement follows from the existence of a dual optimal μ and inequality (4.2). □

4.4.3 Interior point methods

Interior point methods were discovered in the 80's, Karmarkar polynomail time (and empirically efficient) algorithm for linear programing was based on interior point methods. There has been an important activity around interior point methods in the 90's. We refer to [4] for a detailed presentation. In this section, we will only briefly touch the topic and describe the main ideas on a simple problem.

4.4.4 Strong convexity

This notion will be important to develop algorithmic ideas to solve the optimization problems which we have seen.

Definition 4.4.3. *A function $f \colon \mathbb{R}^d \mapsto \mathbb{R}$ is μ strongly convex, if $f - \frac{\mu}{2}\|\cdot\|$ is convex. The following provide sufficient conditions:*

- *If f is differentiable, $f(y) \geq f(x) + (y-x)^T \nabla f(x) + \frac{\mu}{2}\|y - x\|_2^2$, for all x, y.*
- *If f is differentiable, $(\nabla f(x) - \nabla f(y))^T (y - x) \geq \mu \|y - x\|_2^2$ for all x, y.*
- *If f is twice differentiable, the matrix $\nabla^2 f(x) - \mu I$ is positive semidefinite for all x.*

Exercise 4.4.1. *Prove that the function $f \colon x \mapsto -\log(1 - \|x\|^2)$ is strongly convex (when restricted to the unit Euclidean ball).*

Newton's method

Newton's method is famously used to solve equations of the form $g(x) = 0$. In the context of convex optimization, one actually solves $f'(x) = 0$. Application of this method to find a zero of the gradient operator of a strongly convex function $f \colon \mathbb{R}^d \mapsto \mathbb{R}$, can be implemented as follows: choose x_0 and iterate for $k \in \mathbb{N}$,

$$x_{k+1} = x_k - \alpha \left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k). \tag{4.4}$$

Where α is a positive stepsize, determined algorithmically. Note that this equation is well defined since by strong convexity, the Hessian is always positive definite and invertible. One intuition about this method is that it minimizes the second order Taylor expansion: $f(y) \simeq f(x) + \nabla f(x)^T (y - x) + (y - x)^T \nabla^2 f(x)(y - x)$.

A detailed convergence rate analysis of Newton's method can be found in [4, 5, 6]. We prove a local quadratic convergence result which illustrate the fast asymptotic convergence of the method. A more refined analysis is more involved and requires to analyse backtracking line search procedures. We limit ourselves here to a local result stating that when initialized close to the optimum, Newton's method with unit step sizes is extremely fast.

Theorem 4.4.2. *Let f be μ-strongly convex, twice continuoulsy differentiable, with L-Lipschitz Hessian (operator norm) and $\bar{x}$ be the (unique) minimum of f. Newton's method with unit step size satisfy, for all $k \in \mathbb{N}$,*

$$\frac{L}{2\mu^2}\|\nabla f(x_k)\|_2 \leq \left(\frac{L}{2\mu^2}\|\nabla f(x_0)\|_2\right)^{2^k},$$

In particular, if $\|\nabla f(x_0)\|_2 < \frac{L}{2\mu^2}$, we obtain extremely fast convergence for Newton's method with unit step size.

Proof. Fix $k \in \mathbb{N}$. From the Newton iterate, we have $\nabla^2 f(x_k)(x_{k+1} - x_k) = -\nabla f(x_k)$. Hence integrating along the segment $[x_{k+1}, x_k]$, we have

$$\begin{aligned}\nabla f(x_{k+1}) &= \nabla f(x_{k+1}) - \nabla f(x_k) - \nabla^2 f(x_k)(x_{k+1} - x_k) \\ &= \int_{t=0}^{1} \left(\nabla^2 f(x_k + t(x_{k+1} - x_k)) - \nabla^2 f(x_k)\right)(x_{k+1} - x_k)dt\end{aligned}$$

Using the Lipschitz assumption, we obtain

$$\|\nabla f(x_{k+1})\|_2 \leq \frac{L}{2}\|x_{k+1} - x_k\|_2^2 = \frac{L}{2}\|\nabla^2 f(x_k)^{-1}\nabla f(x_k)\|_2^2 \leq \frac{L}{2\mu^2}\|\nabla f(x_k)\|_2^2$$

By a simple recursion, we have

$$\frac{L}{2\mu^2}\|\nabla f(x_{k+1})\|_2 \leq \left(\frac{L}{2\mu^2}\|\nabla f(x_k)\|_2\right)^2 \leq \left(\frac{L}{2\mu^2}\|\nabla f(x_0)\|_2\right)^{2^k}$$

□

Interior point method

We refer to [4] for a more detailed overview. We illustrate the idea of interior point methods for the follwing toy problem, for a given $a \in \mathbb{R}^d$, $b \in \mathbb{R}$, and $f \colon \mathbb{R}^d \mapsto \mathbb{R}$, convex differentiable

$$f^* = \min_{x \in \mathbb{R}^d} f(x) \qquad \text{s.t.} \qquad \|x\|_2 \leq 1,\ a^T x \leq b \tag{4.5}$$

We only use this problem to illustrate the main idea of interior point methods. The main idea of interior points methods is to replace this problem by an unconstrained problem using a barier function, for any $t \geq 0$,

$$\min_{x \in \mathbb{R}^d} tf(x) - \log(1 - \|x\|_2^2) - \log(b - a^T x) \tag{4.6}$$

Note that we need to restrict the domain of definition of the objetctive, since the logarithms explode on the boundary of the feasible set. By example 4.4.1, the objective in (4.6) is 2 strongly convex. Denoting by x_t the minimal value of (4.6) for a given $t \geq 0$, this defines the notion of *central path*, a quick argument shows that

$$f(x_t) \underset{t \to \infty}{\to} f^*$$

and furthermore for each t, x_t can be computed efficiently using Newton's method. This provides an algorithm to solve problem (4.6). A detailled complexity analysis of these types of methods is found for example in [18]. Let us mention that the optimality conditions for (4.6), ensure that

$$t\nabla f(x_t) + 2x_t\frac{1}{1 - \|x_t\|_2} + a\frac{1}{b - a^T x_t}$$

so that x_t minimizes also

$$x \mapsto tf(x) + \frac{1}{1 - \|x_t\|_2}\left(\|x\|_2^2 - 1\right) + \frac{1}{b - a^T x_t}\left(a^T x - b\right)$$

This entails that for any feasible x, we have

$$tf(x_t) - 2 \leq tf(x) + \frac{1}{1 - \|x_t\|_2}\left(\|x\|_2^2 - 1\right) + \frac{1}{b - a^T x_t}\left(a^T x - b\right) \leq f(x),$$

so that $f(x_t) \leq f^* + \frac{2}{t}$ and an ϵ suboptimal solution for (4.5) can be found by choosing $t = 2/\epsilon$.

General purpose solvers

One of the most important topics in Optimization during the 90's was interior point methods. These developments led to theoretical and practical results which materialize in the existence of efficient numerical solvers for the classes of conic problems which were discussed in this section.

4.4.5 Polynomial time LP solvers over $\mathbb{Q}$

Algorithms to solve the LP problem date back to Fourier, Kantorovitch and Dantzig who proposed the simplex method still used in many numerical solvers.

Theorem 4.4.3 (Khachiyan,Karmarkar). *Given inputs $A \in \mathbb{Q}^{n \times d}$, $b \in \mathbb{Q}^n$ and $c \in \mathbb{Q}^d$ consider the problem of computing*

$$\rho = \inf_{x \in \mathbb{Q}^d} c^T x \quad \text{s.t. } Ax \leq b. \tag{4.7}$$

This problem is in $\mathcal{P}$.

Proof sketch. We only sketch the main ideas, a full detailed proof is very tedious. We refer to Schiver's book [21] for more details.

- First if the infimum is not attained, either the original problem or its dual are unfeasible and there polynomial time certificates for this can be found in polynomial time.
- If the problem attains its optimum, then it must attain its optimum at one of the vertices of the polyhedra described by the linear inequalities. There are only finitely many of them.
- There are only polynomially many candidate optimal values for ρ. This is because we have finitely many candidate solutions and the size of the input allows to estimate size of largest common denominators and condition numbers of A.
- Local search methods such as ellipsoid method (for Khachiyan's algorithm) or interior point methods (for Karmarkar's algorithm) converge exponentially fast to ρ (see interior point methods).
- Carefully controling the magnitude of accumulated errors allong the local search path and the degree of approximation required to dicriminate between any two candidate optimal values allow to conclude.

□

Historically, the ellipsoid method was the first polynomial time algorithm for linear programming, it has been studied by various authors in the 70's including Shor, Yudin and Nemirovski. It was proved to be polynomial time by Khachiyan [13] but is quite inefficient in practice. Karmarkar proposed the first polynomial time algorithm which was efficient empirically, based on interior point methods [12].

Corollary 4.4.1. *Assuming the model 3.10 holds and $\theta^* \in \mathbb{Q}^d$, $\hat{\theta}^{\ell_1}_{CS}$ in (3.12) is computable exactly using a number of operations which is at most polynomial in n, d and the number of bits required to encode $\mathbb{X}$ and $\mathbb{X}\theta^*$.*

Remark 4.4.1. *Such a result cannot hold for second order cone programs and semidefinite programs. This is because the solution of such programs may not be in $\mathbb{Q}$ eventhough the data is in $\mathbb{Q}$. For example*

$$\begin{aligned} & \min_x \|x\|_2 \quad \text{s.t.} \quad x_1 \geq 1, x_2 \geq 2 \\ = \quad & \min_{x,t} t \quad \text{s.t.} \quad x_1 \geq 1, x_2 \geq 2, t \geq \|x\|_2 \end{aligned}$$

is a second order cone program which value is attained only for $x_1 = x_2 = 1$ and $t = \sqrt{2}$. Hence the solution of this program cannot be found over $\mathbb{Q}$ and one must switch to computation over $\mathbb{R}$, in particular, the program cannot be solved exactly by finite precision numerical methods. As we have seen, computation over $\mathbb{R}$ has different formulations and connections with practice on physical computers is sometimes a bit far fetched. Hence when one talks about polynomial time solvability of general convex program, this is not in the classical Church-Turing thesis sense but in a different sense such as: polynomial time approximation to a any fixed precision, or polynomial time computation over real machines (which do not exist in the physical world).

Another remark of the same kind goes as follows, the matrix

$$\begin{pmatrix} 1 & y \\ y & x \end{pmatrix}$$

being semidefinite positive implies that $x \geq y^2$ and pilling up k such equalities allows to express numbers of the order 2^{2^k} which bit representation size is exponential in k. Hence such a number cannot be approximated in time polynomial in k using standard numerical integer encoding.

Remark 4.4.2. *In the context of linear programing (LP), since the number of candidate solution is finite (extreme points of the undelying polyhedra), and we have explicit description of these points (lemma 4.3.4), one could try to build an algorithm for finding an optimal extreme point. This is the basis for the Simplex method proposed by Dantzig in 1947 and still used in many numerical softwares. We do not describe it here, but mention that it is an efficient method in practice. However there do not exist polynomial time worst case bounds for these types of algorithm. There exist polynomial time bounds for average instances of linear programs and the simplex method is one of the candidate polynomial time algorithm to solve linear programing over the reals. It also motivate many questions about the geometry of polyhedra such as the Hirsh conjecture.*

Exercises

Exercise 4.4.2. *Prove that Lemma 4.3.1, for any convex set $\mathcal{X} \subset \mathbb{R}^d$ we have*

- *The closure of $\mathcal{X}$ is convex.*
- *The interior of $\mathcal{X}$ is convex.*
- *For any $u \in \mathrm{int}(\mathcal{X})$ and $v \in \mathrm{cl}(\mathcal{X})$, $[u, v) \subset \mathrm{int}(\mathcal{X})$.*
- *If the interior of $\mathcal{X}$ is non empty, then $\mathrm{cl}(\mathcal{X}) = \mathrm{cl}(\mathrm{int}(\mathcal{X}))$.*
- *The interior of $\mathcal{X}$ is empty if and only if it is contained in a lower dimensional affine subspace.*

Exercise 4.4.3. *Prove Theorem 4.3.1, et $f \colon \mathbb{R}^d \to \mathbb{R}$:*

1. *If f is continuously differentiable, then f is convex if and only if or any $x, y \in \mathbb{R}^d$, $f(y) \geq f(x) + \nabla f(x)^T(y - x)$.*
2. *If f is continuously differentiable, then f is convex if and only if or any $x, y \in \mathbb{R}^d$, $(\nabla f(x) - \nabla f(y))^T(y - x) \geq 0$.*
3. *If f is twice continuously differentiable, then f is convex if and only if or any $x \in \mathbb{R}^d$, $\nabla^2 f(x)$ is positive semidefinite.*

Exercise 4.4.4. *Prove Theroem 4.3.6, let $\mathcal{X}, \mathcal{Y} \subset \mathbb{R}^d$ be two disjoint convex sets, then there exists a vector $v \in \mathbb{R}^d$, $v \neq 0$ and a number $c \in \mathbb{R}$ such that $x^T v \geq c$ for all $x \in \mathcal{X}$ and $y^T v \leq c$ for all $y \in \mathcal{Y}$.*

Exercise 4.4.5. *Prove that the different conditions for strong convexity are indeed equivalent to $f - \mu\|\cdot\|_2^2$:*

- *If f is differentiable, $f(y) \geq f(x) + (y - x)^T \nabla f(x) + \frac{\mu}{2}\|y - x\|_2^2$, for all x, y.*
- *If f is differentiable, $\left(\nabla f(x) - \nabla f(y)\right)^T (y - x) \geq \mu\|y - x\|_2^2$ for all x, y.*
- *If f is twice differentiable, the matrix $\nabla^2 f(x) - \mu I$ is positive semidefinite for all x.*

Exercise 4.4.6. *Prove that the function $f \colon x \mapsto -\log(1 - \|x\|^2)$ is strongly convex (when restricted to the unit Euclidean ball).*

Exercise 4.4.7. *Let $\mathcal{S}_d^+$ denote the cone of positive semidefinite matrices in $\mathbb{R}^{d\times d}$. We consider the function $h \colon S \mapsto \log\left(\det\left(S\right)\right)$ over $\mathcal{S}_d^{++}$ the cone of positive definite matrices.*

- *Compute the gradient of $\det$ over $\mathcal{S}_d^{++}$ (Hint: use the relation between S^{-1}, $\det(S)$ and C the adjugate matrix of S).*
- *Compute the gradient of h.*
- *Show that h is convex.*
- *Explain how h could be used as a barrier function for interior point methods in semi-definite programming.*